# SUPPORT OF GULF OF MEXICO HYDRATE RESEARCH CONSORTIUM: ACTIVITIES TO SUPPORT ESTABLISHMENT OF A SEA FLOOR MONITORING STATION PROJECT

SEMIANNUAL TECHNICAL REPORT
1 OCTOBER, 2005 THROUGH 31 MARCH, 2006

PREPARED BY THE MANAGEMENT TEAM,
CENTER FOR MARINE RESOURCES AND ENVIRONMENTAL TECHNOLOGY
(J. Robert Woolsey, Tom McGee, Carol Lutken, Elizabeth Stidham)
(Contact: Carol Lutken)

220 OLD CHEMISTRY BUILDING, UNIVERSITY, MS 38677

JUNE 2006

DOE Award Number DE-FC26-02NT41628

This report was prepared with the support of the United States Department of Energy, under award No. DE-FC26-02NT41628. However, any opinions, findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the DOE. DOE Award Number DE-FC26-02NT41628 is managed by the U.S. Department of Energy's National Energy Technology Laboratory.

FY03 Subcontractors:

Paul Higley, Specialty Devices, Inc., 2905 Capital Street, Wylie, TX 75098
Task 1: Continuation of Work on the Vertical Line Array

J. Robert Woolsey, Mississippi Mineral Resources Institute (MMRI) and Center for Marine Resources and Environmental Technology (CMRET), 220 Old Chemistry Building, University of Mississippi, University, Mississippi, 38677
Task 2: Construction of the Prototype Sea Floor Probe

Ralph Goodman, Department of Marine Sciences, University of Southern Mississippi, 1020 Balch Blvd., Stennis Space Center, MS 39529
Task 3: Acoustic System for Monitoring Gas Hydrates

Vernon Asper, Department of Marine Sciences, University of Southern Mississippi, 1020 Balch Blvd., Stennis Space Center, MS 39529
Task 4: Construction and Testing of an Electromagnetic Bubble Detector and Counter

Boris Mizaikoff, School of Chemistry and Biochemistry, Georgia Institute of Technology, Applied Sensors Laboratory, 770 State St, Atlanta, GA 30332
Task 5: Mid-Infrared Sensor Systems for Continuous Methane Monitoring in Seawater

Angela Davis, AUGER Geophysical Services, School of Ocean Sciences, University of Wales, Bangor, Menai Bridge, Anglesey LL59 5EY, Bangor, Wales, UK
Task 6: Seismo-Acoustic Characterization of Sea Floor Properties and Processes at the Hydrate Monitoring Station

FY04 Subcontractors:

Barrodale Computing Services, Ltd. Hut R, McKenzie Avenue, University of Victoria, Victoria, BC V8W 3W2 Canada
Task 1: Data Management and Processing Software for the Sea-floor Monitoring Station

Bob A. Hardage, Bureau of Economic Geology, John A. and Katherine G. Jackson School of Geosciences, University of Texas at Austin, University Station, Box X, Austin, TX 78713
Task 2: Applications of VSP Technology for Evaluation of Deep-Water Gas Hydrate Systems*

Jeffrey Chanton, Department of Oceanography, Florida State University, Tallahassee, FL 32306
Task 3: Coupling of Continuous Geochemical and Sea-floor Acoustic Measurements

Rudy Rogers, Swalm School of Chemical Engineering, P.O. Box 9595, Mississippi State, MS 39762
Task 4: Microbial Activity Related to Gas Hydrate Formation and Sea-floor Instabilities

FY05 Subcontractors:

Barrodale Computing Services, Ltd. Hut R, McKenzie Avenue, University of Victoria, Victoria, BC V8W 3W2 Canada
Task 1: Data Management and Archiving System and Matched Field Inversion Software Development for the Sea-floor Monitoring Station

Paul Higley, Specialty Devices, Inc., 2905 Capital Street, Wylie, TX 75098
Task 2: Experiment to generate Shear Waves in the Sea-floor and Record them with a Horizontal Line Array

Jeffrey Chanton, Department of Oceanography, Florida State University, Tallahassee, FL 32306
Task 3: Coupling of Continuous Geochemical and Sea-floor Acoustic Measurements

* includes seismo-acoustic characterization of sea-floor properties and processes at the hydrate monitoring station until VSP data can be collected

**DISCLAIMER**

## ABSTRACT

The Gulf of Mexico Hydrates Research Consortium (GOM-HRC) was established in 1999 to assemble leaders in gas hydrates research. The Consortium is administered by the Center for Marine Resources and Environmental Technology, CMRET, at the University of Mississippi. The primary objective of the group is to design and emplace a remote monitoring station or sea floor observatory (MS/SFO) on the sea floor in the northern Gulf of Mexico by the year 2007, in an area where gas hydrates are known to be present at, or just below, the sea floor. This mission, although unavoidably delayed by hurricanes and other disturbances, necessitates assembling a station that will monitor physical and chemical parameters of the marine environment, including sea water and sea-floor sediments, on a more-or-less continuous basis over an extended period of time. In 2005, biological monitoring, as a means of assessing environmental health was added to the mission of the MS/SFO

Establishment of the Consortium has succeeded in fulfilling the critical need to coordinate activities, avoid redundancies and communicate effectively among researchers in the arena of gas hydrates research. Complementary expertise, both scientific and technical, has been assembled to promote innovative research methods and construct necessary instrumentation. The observatory has now achieved a microbial dimension in addition to the geophysical and geochemical components it had already included.

Initial components of the observatory, a probe that collects pore-fluid samples and another that records sea floor temperatures, were deployed in Mississippi Canyon 118 in May of 2005. Follow-up deployments, planned for fall 2005, had to be postponed due to the catastrophic effects of Hurricane Katrina (and later, Rita) on the Gulf Coast. Every effort was made to locate and retain the services of a suitable vessel and submersibles or Remotely Operated Vehicles (ROVs) following the storms and the loss of the contracted vessel, the *M/V Ocean Quest* and its two submersibles, but these efforts have been fruitless due to the demand for these resources in the tremendous recovery effort being made in the Gulf area. Station/observatory completion, anticipated for 2007, will likely be delayed by at least one year.

The seafloor monitoring station/observatory is funded approximately equally by three federal Agencies: Minerals Management Services (MMS) of the Department of the Interior (DOI), National Energy Technology Laboratory (NETL) of the Department of Energy (DOE), and the National Institute for Undersea Science and Technology (NIUST), an agency of the National Oceanographic and Atmospheric Administration (NOAA).

Subcontractors with FY03 funding fulfilled their technical reporting requirements in a previously submitted report (41628R10). Only unresolved matching funds issues remain and will be addressed in the report of the University of Mississippi's Office of Research and Sponsored Programs.

Noteworthy accomplishments of Consortium researchers during this six month cycle funded with DOE's contributions to this multiagency effort include:

- Data Management and Processing Software for the Sea-floor Monitoring Station (Barrodale Computing Services Ltd. (BCS)):
    - The question of which object relational database management system to choose – PostgreSQL or IBM Informix – is addressed, and a comparison of the two options is presented. PostgreSQL is recommended for the initial database development but this decision is to be reconsidered once initial data have been loaded and user access requirements are better understood.
    - The data management and archive system (DMAS) design is presented in a series of diagrams, and each then described in further detail. Final decisions about how the sensors are to be configured and the format of the data to be produced have not yet been made by the Consortium, so database design has been, necessarily flexible. This design reflects the basic architecture decided on previously, with placeholders to house the details that will be determined later.
    - A design is outlined for modifications to the software system for simulation and matched field inversion (MFI) that will be required to allow its use with the proposed 3D array configurations of the monitoring station arrays. Experiments are designed that allow investigation of the benefits (and possible drawbacks) of using the data from the 3D arrays on matched field inversion.
    - An updated, extended version of the previous report, *Software for simulation and matched-field inversion for the Gulf of Mexico Hydrates Seafloor Observatory,* that includes monitoring of acoustic array data is given. Newly-developed software for monitoring using matched field techniques, and a description of the modifications to previous software components to support this monitoring are given.
    - Methods for simulating acoustic array data from shots or ship noise were developed, and were then used to support the development and validation of matched field methods under controlled conditions. The result was that these methods are now in place and ready to be applied when real data from the array become available (now expected in the fall of 2006).
    - A prototype database which can now be extended and used to house the Observatory data when they become available is complete.
    - An enhanced acoustic simulation and matched field inversion (and monitoring) software system suitable for use with 3D arrays and range-dependent bathymetry, that can be applied to estimate geoacoustic models from the Observatory data is complete.

- Progress on the Applications of Vertical Seismic Profiling (VSP) Technology for Evaluation of Deep-Water Gas Hydrate Systems* at the University of Texas Bureau of Economic Geology's Exploration Geophysics Laboratory, EGL

(seismo-acoustic characterization of sea-floor properties and processes at the hydrate monitoring station until VSP data can be collected):
    - EGL scientists have developed software that implements a new theory to create higher-resolution P-SV (P-wave to SV-shear wave conversion) images of near-seafloor geology from large volumes of 4-C (4-component) ocean-bottom cable (OBC) seismic data.
    - The quality of the P-SV image that can be generated is identical to that achieved with the initial experimental code.

- Progress on the Coupling of Continuous Geochemical and Sea-floor Acoustic Measurements:
    - Acoustic wipe-out zones at Mississippi Canyon Lease Block 118 (MC118) may be indicative of active methane venting from sediments containing gas hydrate. Cores both outside and within the wipe-out zones were collected and measured for dissolved methane concentrations.
    - Methane source and microbial controls on methane venting to the overlying water were established by determining the characteristic biogeochemical depth zonation of sulfate, methane and bicarbonate concentrations and stable carbon isotopes.
    - Geochemical data outside the wipe-out zones were consistent with no venting activity.
    - Within the wipe-outs, three distinct vent types could be characterized that suggest an episodic venting history at MC 118.
    - This data set shows that acoustic wipe-out zones at MC 118 are not indicative of homogenous active methane venting at the sediment water interface, but define 3 distinct vent types at MC118.
    - A second long PFA (Pore-Fluid Array) (8 m) has been constructed.
    - A third short PFA (0.5m, for high resolution sampling near a hard ground) has been constructed and will be deployed in the fall of 2006.
    - 2 designs for pressurized pore-water “peepers” have been completed: One design is undergoing laboratory testing and the second is under construction.

- Progress on the Microbial Activity Related to Gas Hydrate Formation and Sea-floor Instabilities:
    - Eight shallow (< 8m) sediment cores from two locations in the Mississippi Canyon, MC118 and MC798, were analyzed for propensity to form gas hydrates and therefore to constitute possible production level deposits.
    - Hydrate formation rates and crystal initiation times were measured in the laboratory as a function of depth below sea-floor and as a function of lateral displacement. Effects of six parameters were analyzed with regard to induction times and formation rates and lateral variability documented.
    - Depth of the sulfate zone, pore-water salinity, and bioactivity appear to be important in determining ease of hydrate formation in shallow sediments.

- Progress on the Experiment to Generate Shear Waves in the Sea-Floor and Record them with a Horizontal Line Array
    - The communications systems for 3-C accelerometer (land seismic) sensors have been modified for a deep-ocean environment. Specialty Devices, Inc. (SDI) is addressing a software and circuitry problem, working with Input-Output (I-O) to develop a solution and proceed with testing.

- Administration of the Monitoring Station/Sea-floor Observatory project this reporting period has consisted of
    - organizing and hosting two meetings: a November semiannual meeting addressing primarily geochemical projects, and a February annual meeting focused on geophysical components of the monitoring station.
    - organizing and carrying out three cruises: an October cruise to install and test the Ultrashort base-line navigation system and to collect core samples in support of several MS/SFO projects, a March cruise to establish a surface-source/deep-receiver (SS/DR) survey and a second March cruise to test the drift camera developed by SDI. The latter is a direct result of the loss to the program of the *M/V Ocean Quest* and its two manned submersibles in August of 2005.
    - reporting to and interacting with sponsoring agencies and their officers as well as with Consortium members. Two semiannual progress reports were completed and submitted to DOE during this reporting period: 41628R08 submitted in November, 2005 and 41628R10, submitted in March, 2006.

**TABLE OF CONTENTS** **PAGE**



**LIST OF GRAPHICAL MATERIALS**

Graphical materials used to illustrate reports can be found in the individual reports submitted by the subcontractors.

## INTRODUCTION / PROJECT SUMMARY

The Gulf of Mexico-Hydrate Research Consortium (GOM-HRC) is in its sixth year of developing a sea-floor station to monitor a mound where hydrates outcrop on the sea floor. The plan for the Monitoring Station/Sea Floor Observatory (MS/SFO) is that it be a multi-sensor station that provides more-or-less continuous monitoring of the near-seabed hydrocarbon system, within the hydrate stability zone (HSZ) of the northern Gulf of Mexico (GOM). The goal of the GOM-HRC, is to oversee the development and emplacement of such a facility to provide a better understanding of this complex hydrocarbon system, particularly hydrate formation and dissociation, fluid venting to the water column, and associated microbial and/or chemosynthetic communities. Models developed from these studies should provide a better understanding of gas hydrates and associated free gas as: 1) a geo-hazard to conventional deep oil and gas activities; 2) a future energy resource of considerable significance; and 3) a source of hydrocarbon gases, venting to the water column and eventually the atmosphere, with global climate implications.

Initial funding for the MS/SFO was received from the Department of Interior (DOI) Minerals Management Service (MMS) in FY1998. Funding from the Department of Energy (DOE) National Energy Technology Laboratory (NETL) began in FY2000 and from the Department of Commerce National Oceanographic and Atmospheric Administration's National Undersea Research Program (DOC NOAA-NURP) in 2002. Some ten industries and fifteen universities, the United States Geological Survey (USGS), the US Navy, Naval Meteorology and Oceanography Command, Naval Research Laboratory and NOAA's National Data Buoy Center are involved at various levels of participation. Funded investigations include a range of physical, chemical, and, more recently, microbiological studies.

## EXECUTIVE SUMMARY

A consortium has been assembled for the purpose of consolidating both the laboratory and field efforts of leaders in gas hydrates research. The Consortium, established at and administered by the University of Mississippi's Center for Marine Resources and Environmental Technology (CMRET), has, as its primary objective, the design and emplacement of a remote monitoring station on the sea floor in the northern Gulf of Mexico by the year 2007. The primary purpose of the station is to monitor activity in an area where gas hydrates are known to be present at, or just below, the sea-floor. In order to meet this goal, the Consortium has begun assembling a station that will monitor physical and chemical parameters of the sea water, sea-floor sediments, and shallow subsea-floor sediments on a more-or-less continuous basis over an extended period of time. Central to the establishment of the Consortium is the need to coordinate activities, avoid redundancies and promote effective and efficient communication among researchers in this growing area of research. Complementary expertise, both scientific and technical, has been assembled; collaborative research and coordinated research methods have grown out of the Consortium and design and construction of instrumentation for the sea-floor station is nearing completion.

The MS/SFO was designed to accommodate the possibility of expanding its capabilities to include biological monitoring. A portion of FY04 funding from the MMS was directed toward this effort to support the study of chemosynthetic communities and their interactions with geologic processes. In addition, results will provide an assessment of environmental health in the area of the station. NOAA National Undersea Research Program (NURP) has, as a focal point, investigations of the effects of deep sea activities on world atmosphere and therefore, weather. In July of 2005, the Director of the National Institute for Undersea Science and Technology (NIUST) of NOAA-NURP made a portion of that agency's budget available *via* competitive grants to researchers with proven expertise in microbial research. A sea-floor microbial observatory is an objective of that agency and these sponsored projects sited at the MS/SFO are designed to fulfill that directive.

The centerpiece of the monitoring station, as originally conceived, is a series of vertical line arrays of sensors (VLAs), to be moored to the sea floor. Each VLA was to have extended approximately 200 meters from the sea-floor. Sensors in the VLAs include hydrophones to record water-borne acoustic energy (and measure sound speed in the lower water column), thermistors to measure water temperature, tilt meters to sense deviations from the vertical induced by water currents, and compasses to indicate the directions in which the deviations occur. During discussions among the members of the geophysical subgroup of the Consortium, it was discovered that the project may be better served if some vertical arrays are converted to horizontal line arrays (HLAs). The prospective horizontal water-bottom arrays, will consist of hydrophones and 3-component accelerometers and will be laid upon, and pressed into, the soft sediment of the sea-floor. They will be arranged into a cross so that they simulate two perpendicular arrays. Their deployment will be accomplished by means of a sea-floor sled designed to lay cable and deploy probes into shallow, unconsolidated sediments. This sled will also be used as a seismic source of compressional and shear waves for calibrating the subsurface seismo-acoustic array commissioned by the Joint Industries Program (JIP).

The prototype DOE-funded VLA has been completed and tested together with the associated data-logging and processing systems. An Oceanographic Line Array (OLA) is ready to be equipped with any of a variety of geochemical sensors - thermistors, fluorometers, transmissometers, mass spectrometers, conductivity and current flow meters – and deployed at the observatory site, Mississippi Canyon 118 (MC118). Processing techniques continue to be developed for vertical array data by Consortium participants who are currently funded by the MMS.

A Remotely Operated Vehicle (ROV) mateable connector system was designed and installed in the VLA Data Acquisition and Telemetry System (DATS) deployed in 2005. This improved design has been incorporated into the VLA and the OLA components of the observatory. Positioning sensors – including compass and tilt sensors – have been completed and tested. Pressure housings rated twice that of any anticipated deployment have been built and pressure tested.

In May, 2005, the Sea-Floor Probe (SFP) was used to retrieve core samples from MC118 as part of the effort to select sites appropriate for deployment of the geophysical and geochemical probes. The northwestern portion of the mound area defined on images recovered during a C&C autonomous underwater vehicle (AUV) survey April 30-May 2, 2005, was selected for probe deployments based on information from these cores. Both the pore-fluid array and the geophysical line array were deployed via SFP at MC118 in May, 2005.

Additional MS/SFO deployments, scheduled for September and October, 2005, were delayed due to the devastation of the Mississippi Gulf Coast and environs by Hurricane Katrina and, to a lesser extent, the Louisiana Gulf Coast by Hurricane Rita. The immediate cause for delay was the removal of the *Ocean Quest*, the vessel that, with its two submersibles, was to have provided the platform from which many of the bottom-founded sensors would have been deployed and cable connections made. It would also have provided the visual survey needed to make optimal choices of deployment sites for station components. Every effort has been made to find a replacement vessel and/or vessels but most vessels in the Gulf are being used in the massive recovery effort that is underway in the region. In addition, damage to ship yards and various forms of infrastructure has been extensive. Work scheduling is largely guesswork at this point and prospects of securing a vessel with submersibles or even ROV capability in the near future seem remote. It is now apparent that the station deployment date will have to be pushed forward by at least one year and perhaps more. September 2006 is the soonest we have been able to reserve the type of ship/vessel we will need to complete the deployments phase of the project. The CMRET has secured seven days of ship time aboard the Seward Johnson with use of its manned-submersible, the Johnson SeaLink. This vessel will be used to retrieve the osmopump packages and data-loggers deployed in 2005, and to deploy sensors and experiments.

In spite of the delays, project development continues:

Barrodale Computing Services Ltd. (BCS) designed and developed a software system for simulation and matched-field inversion (MFI) of acoustic array data (previous report, 41628R12). The system, termed BCOMFI (**B**arrodale **CO**mputing **M**atched **F**ield **I**nversion), provides a comprehensive and validated environment for investigating the application of MFI techniques to detect changes in the sub-bottom gas hydrate deposits under the sea-floor in the region of the MS/SFO. This approach is based on the expectation that MFI analysis of acoustic array data originating from nearby sources of opportunity (passing ships) can be used to detect such changes. The basic principle is to derive geoacoustic models for sub-bottom regions of the station by applying MFI to data from calibration measurements, and then use these models to match with future data obtained from passing sources. The presence of a large mismatch would be taken as evidence of a change.

This software has been updated and extended to support the use of full 3D arrays. A design is outlined for modifications to the software system for simulation and matched field inversion (MFI) that will be required to allow its use with the proposed 3D array configurations of the monitoring station arrays. Experiments are designed that

allow investigation of the benefits (and possible drawbacks) of using the data from the 3D arrays on matched field inversion.

The question of which object relational database management system to choose – PostgreSQL or IBM Informix – is addressed, and a comparison of the two options is presented. PostgreSQL is recommended for the initial database development but BCS recommends revisiting this decision once initial data have been loaded and user access requirements are better understood.

The DMAS design is presented in a series of diagrams, and each then described in further detail. Final decisions about how many of the sensors are to be configured and the format of the data to be produced have not yet been made by the Consortium, so database design has been, necessarily flexible. This design reflects the basic architecture decided on previously, with placeholders to house the details that will be determined later.

An updated, extended version of the previous BCS report, *Software for simulation and matched-field inversion for the Gulf of Mexico Hydrates Seafloor Observatory* that includes monitoring of acoustic array data is given. Newly-developed software for monitoring using matched field techniques, and a description of the modifications to previous software components to support this monitoring are given.

Methods for simulating acoustic array data from shots or ship noise were developed, and were then used to support the development and validation of matched field methods under controlled conditions. The result was that these methods are now in place and ready to be applied when real data from the array become available (now expected in the fall of 2006).

A prototype database which can now be extended and used to house the Observatory data when they become available is complete. An enhanced acoustic simulation and matched field inversion (and monitoring) software system suitable for use with 3D arrays and range-dependent bathymetry, that can be applied to estimate geoacoustic models from the Observatory data is complete.

EGL scientists have developed software that implements a new theory to create higher-resolution P-SV images of near-seafloor geology from large volumes of 4-C OBC seismic data. The approach takes advantage of the fact that shear-wave velocity in near sea-floor sediments is much slower than compressional wave velocity while their frequencies are the same. Therefore, the wavelengths must be shorter for the shear wave data from near the sea-floor. This feature has enabled researchers to process data of short wavelengths in a manner similar to that for high resolution data. The quality of the P-SV image that can be generated is identical to that achieved with the initial experimental code developed by EGL researchers in the previous reporting period.

The presence of abundant acoustic wipe-out zones in chirp sonar data acquired over MC118 in May of 2005 by C&C prompted Consortium researchers to attempt to

determine a geochemical link/factor to sea-floor acoustic measurements. Acoustic wipe-out zones may be indicative of active methane venting from sediments containing gas hydrate. Cores both outside and within the wipe-out zones were collected and measured for dissolved methane concentrations.

Methane source and microbial controls on methane venting to the overlying water were established by determining the characteristic biogeochemical depth zonation of sulfate, methane and bicarbonate concentrations and stable carbon isotopes. Geochemical data outside the wipe-out zones were consistent with no venting activity; Within the wipe-outs, three distinct vent types could be characterized that suggest an episodic venting history at MC 118; a “mud-prone” vent associated with sharply defined wipe-outs and low concentrations of purely thermogenic methane suggesting a new vent whose microbial community is not yet well-established, a “transitional” vent with sharply defined acoustic wipe-out but high concentrations of both thermogenic and biogenic methane suggesting active microbial communities at shallow depths and a “mineral-prone” vent with roughly defined sea-floor and interference features overlain by thicker units than the first two vent types whose sediments include carbonate nodules and biogenic methane and no down-core variation in methane or sulfate suggesting previous microbial activity that is now exhausted. This data set shows that acoustic wipe-out zones at MC 118 are not indicative of homogenous active methane venting at the sediment water interface, but define 3 distinct vent types at MC118.

A second long PFA (8 m) and a third short PFA (0.5m, for high resolution sampling near a hard ground) have been constructed and will be deployed in the fall of 2006. In addition, two designs for pressurized pore water “peepers” have been completed: One design is undergoing laboratory testing and the second is under construction.

Evaluation of induction times and rates of formation from eight sediment cores from two different blocks in the Mississippi Canyon, 118 and 798, reveal the effects of six different parameters on hydrate formation, findings that could provide guides to locating economic deposits of hydrates. Variations in ease of formation have now been documented to occur both laterally and with depth beneath the sea-floor. Carbonate nodules appear to denote the bottom of the sulfate zone, and hydrogen sulfide (sulfur smell) the transition region of the sulfate zone. Where these occurred in the sediments, a minimum and/or maximum occurred in the hydrate formation and induction-time curves, possibly suggesting a significant change in hydrate-forming ease at the locale in question. Wherever bioactivity had been indicated, formation rates increased and induction times decreased, suggesting an increase in hydrate formation. Wherever high salinity of the pore waters was documented, hydrate formation was impaired.

Progress on the experiment to generate shear waves in the sea-floor and record them with an HLA has been delayed both by storm activity and by the necessity to modify sensor communications systems for a deep ocean environment. However, the communications systems for 3-C accelerometer (land seismic) sensors have been modified for a deep-ocean environment and SDI is addressing a software and circuitry

problem, working with Input-Output to develop a solution and proceed with testing.

Administration of the Monitoring Station/Sea-floor Observatory project this reporting period included organizing and hosting two meetings of the Consortium, organizing and carrying out three cruises to MC118, publishing two semiannual technical reports and establishing a monthly reporting protocol. Much research and rescheduling went into these efforts but interactions of Consortium members, including funding agency representatives, has been especially productive during this period.

A November semiannual meeting was held to address primarily geochemical projects associated with the monitoring station. Related scheduling of tests and deployments resulted in a tentative schedule of tasks and the CMRET assumed the task of coordinating the cruise efforts. At the February annual meeting, the focus was geophysical components of the monitoring station. Efforts to secure ship time resulted in scheduling LUMCON's *R/V Pelican* for several blocks of time in 2006 to complete the SS/DR survey and to deploy station support systems. The Seward Johnson and its manned submersible, the Johnson SeaLink are reserved for seven days in September, primarily to retrieve the data-loggers and osmopump package deployed in May and to facilitate deployment of geochemical sensors and geochemical and microbial experiments that cannot be deployed from the sea's surface. It will also provide visual information as deployments are made.

Three cruises were organized and executed during the October – March time period. In October, the *Pelican* was used to install and test the Ultrashort base-line (USBL) navigation system and to collect core samples in support of several MS/SFO projects, including those funded by DOE entitled "Coupling of Continuous Geochemical and Sea-floor Acoustic Measurements" and "Microbial Activity Related to Gas Hydrate Formation and Sea-floor Instabilities." A cruise in early March established parameters for and a beginning of a surface-source/deep-receiver (SS/DR) survey and a second March cruise was made to test the drift camera developed by SDI. The latter is a direct result of the loss to the program of the *Ocean Quest* and its two manned submersibles in August of 2005. The new instrument, the DeepSee, can be deployed from a non-dynamic positioning vessel to water depths of 2000m, well beyond the maximum water depth of 920m at MC118. The development of the DeepSee is funded by NOAA but its capabilities benefit all aspects of the MS/SFO that involve the sea-floor.

Reporting to and interacting with sponsoring agencies and their officers as well as with Consortium members is a primary administrative function of CMRET. Two semiannual progress reports were completed and submitted to DOE during this reporting period: 41628R08 submitted in November, 2005 and 41628R10, submitted in March, 2006. Several monthly reports documenting progress of subcontractors as well as the Consortium in general were also submitted.

## EXPERIMENTAL

Experiments are described in the individual reports submitted by the subcontractors and included in the "Results and Discussion" section, which follows.

## RESULTS AND DISCUSSION

Results and discussion of those results are described in the individual reports submitted by the subcontractors. Reports from the subcontractors follow.

## CONCLUSIONS

This report covers the accomplishments of the seventh six-month period funding of Cooperative agreement Project #DE-FC26-02NT41628, between the Department of Energy and the Center for Marine Resources and Environmental Technology, University of Mississippi. The efforts of the Hydrates Research Consortium are reviewed and plans for the deployments of station components presented. Conclusions of various projects are happening and every effort is being made to coordinate site surveys and sensor emplacements in a sequence that allows all participants maximum access to and benefit from the cruises being scheduled for summer and fall, 2006.

Project summaries of the subcontractors' efforts appear in their reports contained within this document. All FY03 subcontractors have completed their technical reporting although financial reports are not yet complete. The CMRET continues to work with the sponsored programs officials at several institutions to resolve these delays. The VLA and the SFP are complete and have been proven. The "bubble counter" is complete but awaits testing in deep water, something that the Consortium is arranging. The "SphereIR" is essentially complete though it has never been field tested. Both it and the acoustic device have depleted funds prior to completion of their sensor packages. Laboratory studies of gas hydrates have expanded to new areas and new depths and include analyses of additional parameters impacting hydrate formation and stability.

Software development and innovative processing techniques are complete until additional sensor details become available. The pore-fluid studies and laboratory experiments continue to break new ground in hydrates research and to provide valuable information regarding where, in what, and to what extents – vertically and laterally – hydrates may be expected to be found in the marine environment. Initial components of the station, a pore-fluid sampling probe and a thermistor geophysical probe, were emplaced on the sea floor in May of 2005. The data-logger and sample-collecting box of these components contain the first date produced at the MS/SFO. Their retrieval is a priority for the upcoming field season. Additional components will be added during 2006 visits to the station site with completion of the station anticipated in 2007.

An Appendix is included for informational/historical purposes. Revision to it since the previous progress report has been minimal but it provides a useful reference when reviewing current projects.

## REFERENCES

Relevant references appear following contributions by the individual subcontractors.

Designs for DMAS and MFI Software Enhancements for the Gulf of Mexico Hydrates Seafloor Observatory

Version 1.1

**Barrodale Computing Services Ltd. (BCS)**
**Hut R, McKenzie Ave., University of Victoria**
**Victoria BC V8W 3W2 Canada**
**www.barrodale.com**

**December 2, 2005**

# Table of Contents

# Executive Summary

This Report by Barrodale Computing Services Ltd. (BCS) for the University of Mississippi Gulf of Mexico Hydrate Research project describes designs for the data management and archive system (DMAS) for the Gulf of Mexico Hydrates Seafloor Observatory, and for the enhancements to the simulation and matched field software that will be required to support the use of full 3D arrays.

The question of which object relational database management system to choose – PostgreSQL or IBM Informix – is first considered. It is proposed that either would be suitable for implementing the DMAS, and a comparison of the two options is presented. It is noted that a decision can be deferred, without cost, until the time comes to develop database extensions. It is recommended that PostgreSQL be used for the initial database development and that this decision be revisited once initial data has been loaded and user access requirements are better understood.

The DMAS design is then presented, in the form of entity-relationship diagrams describing the basic conceptual components of the database and the types of relations between them. Each of these diagrams is then described in detail, and additional issues such as calibration, data products, data annotation, and user access are discussed. It is noted that final decisions about how the sensors are to be configured and the format of the data to be produced have not yet been made by the Consortium, and so it has been necessary to take a flexible approach to database design. This design reflects the basic architecture decided on previously, with placeholders to house the details that will be determined later.

A design is then outlined for the modifications to the software system for simulation and matched field inversion that will be required to allow its use with the proposed 3D array configurations of the monitoring station arrays. In addition, experiments are designed that would allow investigation of the benefits (and possible drawbacks) of using the data from the 3D arrays on matched field inversion.

# Introduction

The Gulf of Mexico Hydrates Research Consortium and the Center for Marine Resources and Environmental Technologies are developing a multi-sensor seafloor observatory (monitoring station) to be installed on the continental slope of the northern Gulf of Mexico. The aim of this observatory is to monitor and investigate the hydrocarbon system within the hydrate stability zone of the northern Gulf of Mexico, and to remotely observe changes in the physical and chemical parameters of gas hydrates. The intention has been to equip the observatory with a variety of sensors that would enable the determination of a steady-state description of physical, chemical and thermal conditions in its local environment, as well as to detect temporal changes of those conditions. Major components of this Seafloor Observatory are geochemical instruments, temperature sensors, accelerometers, and an array of hydrophones that will collect acoustic data.

All this data will need to be archived in an appropriately structured data management and archive system (DMAS). In two previous reports for the University of Mississippi[1], Barrodale Computing Services Ltd. (BCS) laid the framework for this DMAS development by characterizing the sensors and data for the monitoring station, and by designing the basic architecture of such a system. In the present report, we extend this effort by first presenting a conceptual design for the DMAS, outlining the required tables and defining their relationships. We then present a design for the enhancements to the matched field software system that will be required to implement the use of 3D hydrophone arrays.

In formulating the database design, it was first necessary to update the information on which sensors were currently targeted as array components and to characterize these sensors and their data. Based on the information provided in response to inquiries that we initiated, a description and list of sensor components for each of the various arrays was finalized; this list is presented in Appendix 1. Several sensors that were not included in the list in our original report, and also those for which additional information is now available, were also identified and characterized; this supplementary information is presented in Appendix 2. Additional advice about the sensor and array configurations was also obtained in two phone interviews with sensor experts; and synopses of these interviews are contained in Appendix 3. Finally, a detailed design or the DMAS (an expansion of the diagrammed design, in tabular form) is given in Appendix 4.

We note that many decisions about how the sensors are to be configured and the format of the data to be produced have not been made. This state of affairs has required a flexible approach to database design, and so the design presented here falls between a simple conceptual model and a detailed physical design. This design reflects the basic architecture decided on to date, with placeholders to house the details

[1] *Sensor and Data Characterization for the Gulf of Mexico Hydrates Seafloor Observatory – Version 1.2* (January 31, 2005) and *Data Management Architecture Design for the Gulf of Mexico Hydrates Seafloor Observatory – Version 1.1* (February 28, 2005)

that will be determined later.

## Design for DMAS

This section presents a design for the Gulf of Mexico Hydrates Seafloor Observatory data management and archive system (DMAS).

Database designs range from simple conceptual models, which lay out the basic entities and relationships of a database, to complex physical designs that define exactly what is stored, how it is stored, and where it is stored. The design presented here falls midway between these two extremes. It is premature at this time to present a detailed physical design, since the details of such a design would depend on factors not yet decided, for example, the configuration options to use on a particular instrument. Instead, this design reflects the basic architecture decided on to date, with placeholders to house the detail that will be determined later. It will likely be the case that some of these details will not be filled in until the data from the instruments is actually retrieved.

### Choice of ORDBMS

While the design presented in this document could be implemented using a variety of conventional relational database management systems (RDBMS), future enhancements will benefit from a database management system with object relational support[2].

In the short term, data from the instruments will be retrieved on a discrete schedule. The data will be stored in files, these files can easily be loaded into conventional database binary large objects (blobs), and metadata for these files can be precomputed and stored in conventional simple integer, float, and text columns. The files (and hence blobs) will be relatively small, so there will be no real penalty in fetching entire blobs and processing them exclusively by the client applications.

In the longer term, however, data will be reaching the database, via fiber cable, as a stream rather than as a prepackaged set of files. While the data will ultimately have to be broken into discrete chunks, object relational extensions can be used to preserve the "illusion" of a continuous stream. Using a single SQL statement it will be easy, for example, to extract all the data that falls between two specific time values, regardless of whether this data is stored in a single chunk or in parts of many chunks. These object relational extensions will require that the data be stored in "smart" chunks, not simple blobs, and this "smartness" cannot be designed until the communication and control mechanisms of the continuously-feeding-data observatory are finalized.

As a result of these considerations, candidate database management systems were restricted to ones that enable user-defined object-relational extensions. There are two of these: PostgreSQL (http://www.postgresql.org/) and IBM Informix (http://www-

[2] The benefits of using an object relational database approach were outlined in *Data Management Architecture Design for the Gulf of Mexico Hydrates Seafloor Observatory – Version 1.1* (February 28, 2005)

306.ibm.com/software/data/informix). Either product would be suitable for the Gulf of Mexico Hydrates Seafloor Observatory DMAS; the following is a brief comparison of the two products, with the ***P*** or ***I*** symbol used to indicate which product has an advantage in a particular area:

1) PostgreSQL is free; Informix has a (high) up-front cost (***P***).
2) PostgreSQL is open-source; Informix is proprietary. This makes maintaining the object-relational extensions easier in PostgreSQL (***P***).
3) PostgreSQL has a more cumbersome garbage collection mechanism than Informix ***(I)***.
4) PostgreSQL is single-threaded, while Informix is multi-threaded. This can make Informix more efficient in situations with many concurrent users ***(I)***.
5) The smart blob implementation in Informix is more efficient than that used in PostgreSQL ***(I)***.

The initial database will use conventional features common to both Informix and PostgreSQL, so a decision can be deferred, without cost, until the time comes to develop database extensions. It is therefore recommended that PostgreSQL be used for the initial database and that this decision be revisited once initial data has been loaded and user access requirements are better understood.

# Proposed DMAS Design

## Design Diagrams

The following five diagrams illustrate various aspects of the DMAS design. The notation used in these diagrams is explained immediately following the last diagram (Figure 5).

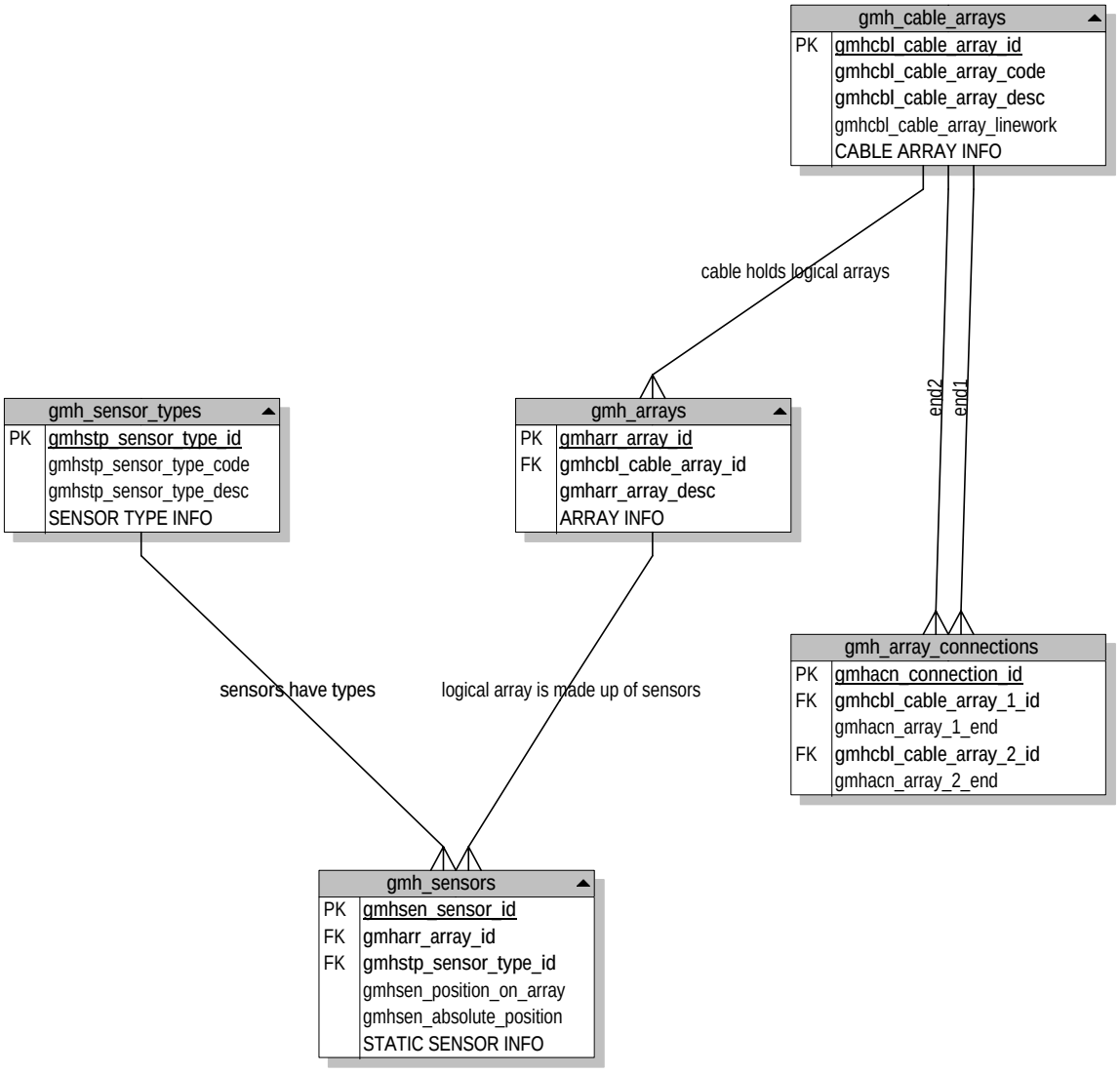



**Figure 1: DMAS Sensor and Array Configuration**

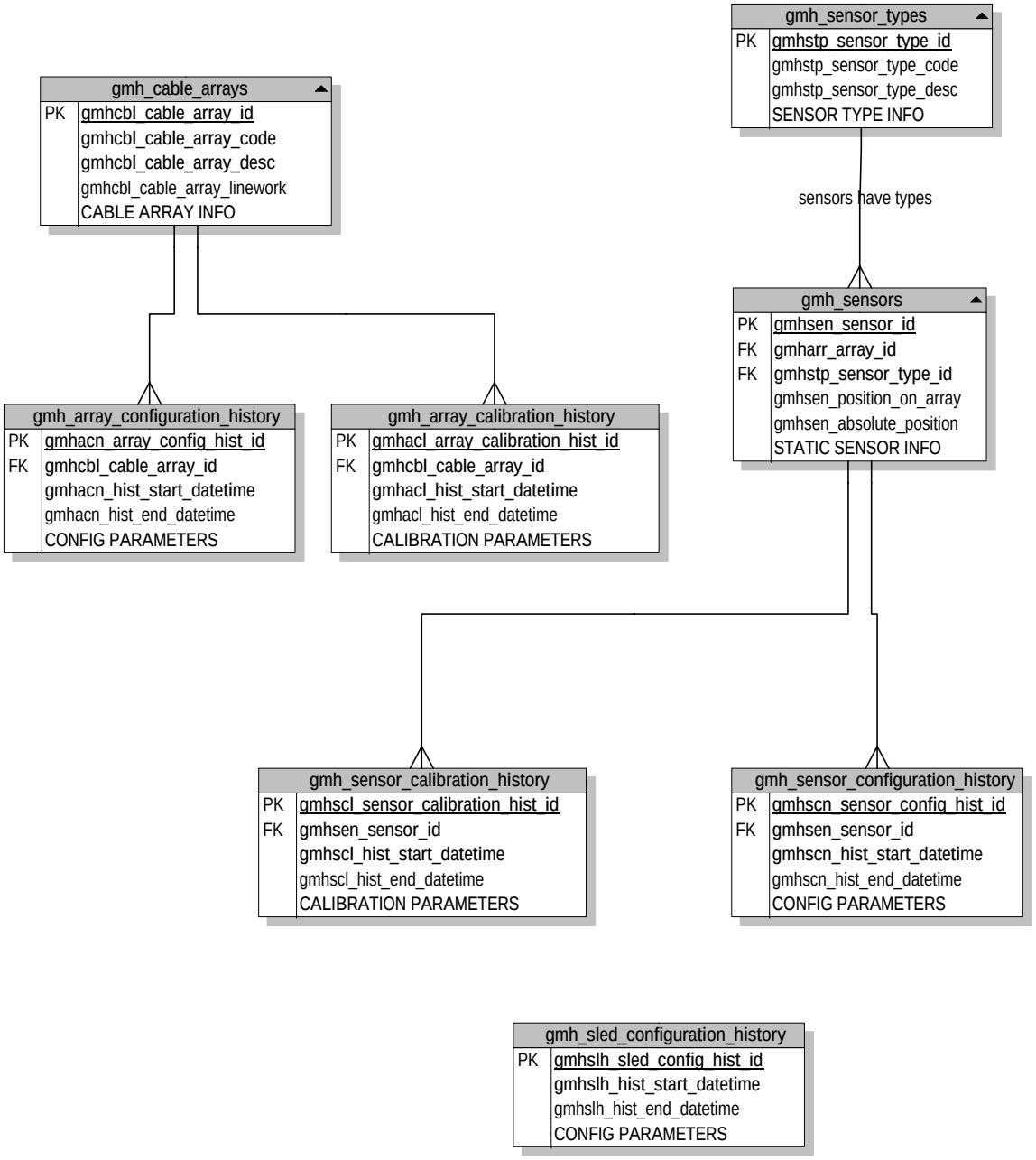



**Figure 2: DMAS Calibration and Configuration History**

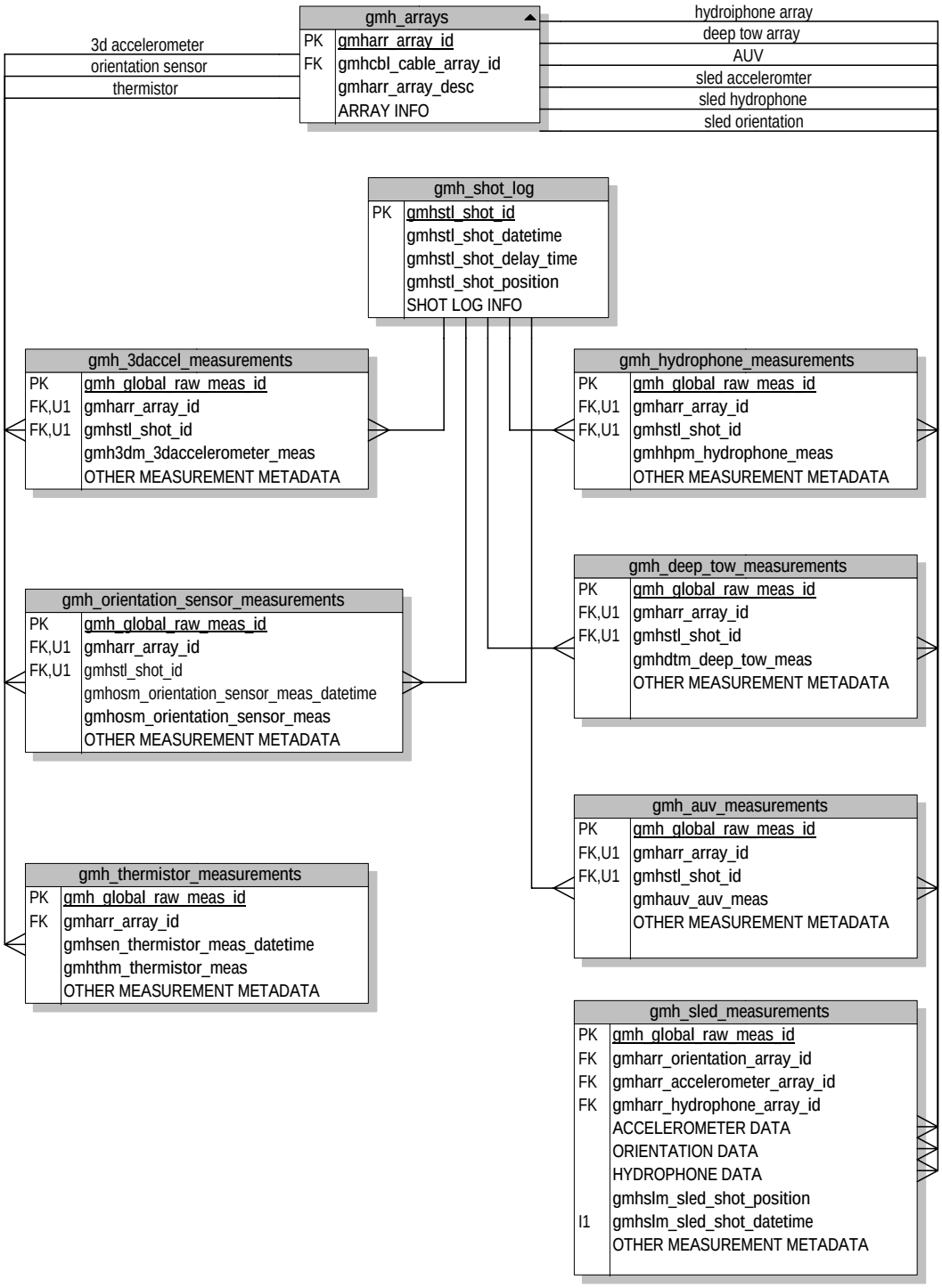



**Figure 3: DMAS Seismic Sensor Measurements**

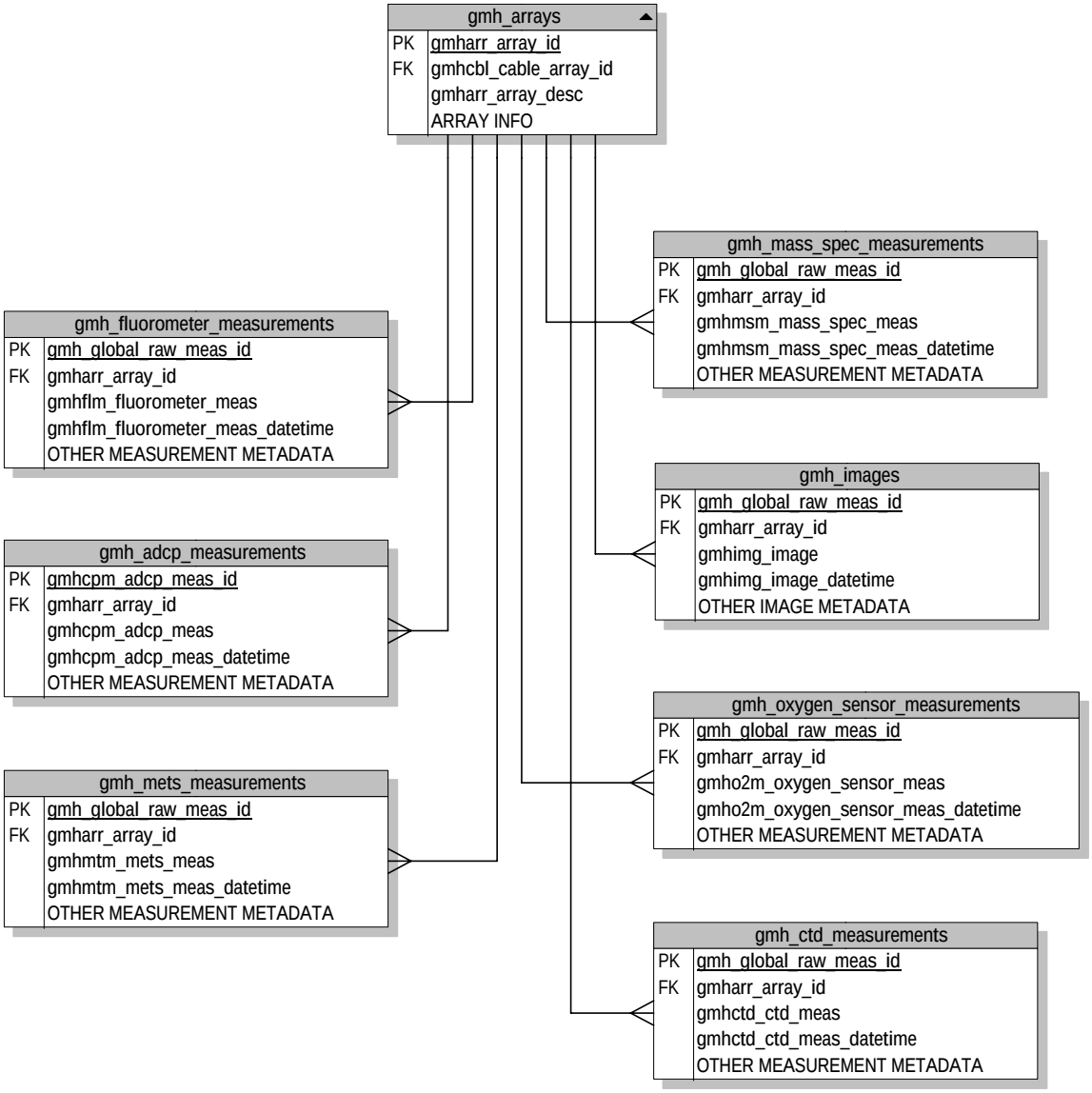



**Figure 4: DMAS Environmental Sensor Measurements**

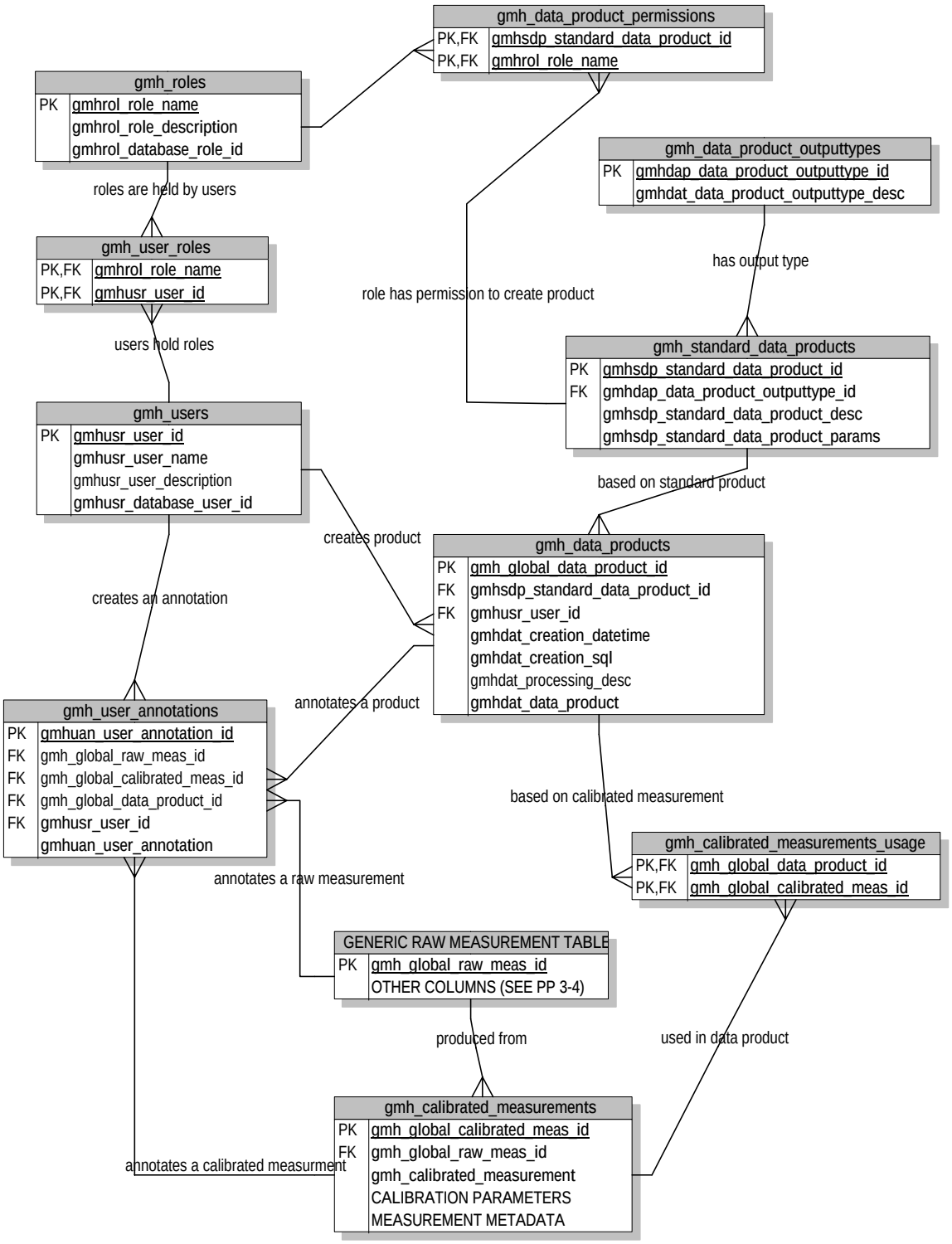



**Figure 5: DMAS Users, Roles, Measurements and Data Products**

## Diagram Notation

1) Each box represents a table; text in the gray area at the top of the box is the table name.
2) Table columns are listed in the white part of the box.
3) The primary key[3] column(s) of each table are underlined and have PK written next to them on the left.
4) Foreign key[4] columns have an FK written next to them on the left.
5) Columns with the same U*n* value written next to them together form a unique key; together their values uniquely identify a row in the table.
6) All PK, FK, and U*n* columns are indexed; other indexed columns are identified by an I*n* designator appearing next to them.
7) Column names written in **bold** lower-case represent columns that must always have a value (cannot be left NULL); column names written in non-bold lower-case are nullible columns.
8) Column names written in UPPER CASE represent placeholders; they will be replaced by real columns in a later design.
9) Lines between boxes represent 1-to-many relationships between tables. The line end with a just a single part is the "1" end of the relationship; the line end with the three parts ("crows feet") is the "many" end of the relationship. For example, in Figure 1, the line between the table gmh_sensor_types and gmh_sensors can be interpreted as follows: For each sensor type there are 0 or more ("many") sensors with that type and for each sensor there is just a single ("1") sensor type. Specifically, the relationship ties the PK column in the "1" table to the FK column having the same name in the "many" table.
10) Tables can be related in more than one way. In Figure 1 there are two lines between gmh_cable_arrays and gmh_array_connections. A cable can be connected to one set of other cables at one end and a different set of cables at the other end.
11) Some lines have "verb phrases" written across them; these help to explain the nature of the relationship.

The tables and relationships shown in Figures 1 through 5 are described extensively in the database dictionary report reproduced in Appendix 4. In addition, some features of the five views of the database shown in Figures 1 through 5 are highlighted in the following sections.

[3] The primary key for a table is a column or a combination of columns whose value(s) are used to uniquely identify a single row in the table.
[4] A foreign key in a table is a column or a combination of columns whose value(s) are used to uniquely identify a single related row in another table.

## Sensor and Array Configuration

Figure 1 illustrates the notions of ***sensor***, ***sensor array***, and ***cable array***. The observatory houses a number of sensors of various types (e.g., hydrophones, thermistors, orientation sensors). These sensors may provide data independently of other sensors, or they may act as a group with other sensors to produce combined measurements. The group of one or more sensors producing a measurement is referred to as a *logical* array and the sensor(s) forming a logical array are placed on a *physical* array. These physical arrays can be connected, at either end, to other physical arrays.

The assignment of these conceptual objects to tables is as follows:

| | |
|---|---|
| Individual sensors: | gmh_sensors |
| Sensor types: | gmh_sensor_types |
| Logical, or sensor, arrays: | gmh_arrays |
| Physical, or cable, arrays: | ghm_cable_arrays |
| Connections between physical arrays: | gmh_array_connections |

## Calibration and Configuration History

Figure 2 illustrates the relationships between sensors / physical arrays and their configuration and calibration histories.

Throughout the life of the Gulf of Mexico Hydrates Seafloor Observatory, the observatory's sensors and physical arrays will undergo changes in configuration and calibration. In order to reprocess historical data it may be necessary to access the calibration and configuration data that was in effect at that time. The *gmh…history* tables will store the information needed for this.

## Seismic Sensor Measurements

The HLA, VLA, and BLA physical arrays described in Appendix 1 will house the instruments and sensors needed to conduct matched field inversion (MFI)[5]. Figure 3 illustrates how the tables associated with these sensor measurements are related to one another.

There are seven measurement tables. Each includes the foreign key values necessary to represent the relationship of a measurement to the array from which it is taken and the shot number it represents, if any. The tables also include the measurements themselves, in addition to any other metadata useful in identifying or finding

[5] MFI will require knowledge of the locations of the source and hydrophone sensors, and the measurements received from the array of sensors.

measurements needed for a particular application.

Five of these tables (gmh_3daccel_measurements, gmh_auv_measurements, gmh_orientation_sensor_measurements, gmh_hydrophone_measurements, and gmh_deep_tow_measurements) are related to both the shot log (stored in gmh_shot_log) and to logical arrays (information about which is stored in gmh_arrays). Each of these measurements relates to a single shot (identified by the gmhstl_shot_id value) and to a single logical array (identified by the gmharr_array_id column). No time value is explicitly stored with the measurement, since the appropriate time value can be accessed via the shot log.

Thermistor measurements, on the other hand, are not tied to a specific shot and so the table for thermistor measurements (gmh_thermistor_measurements) includes a specific datetime value.

The sled system includes its own acoustic source, so again there is no need to relate the sled system table (gmh_sled_measurements) to the shot log.

## Environmental Sensor Measurements

The Benthic Boundary Layer Array (BBLA) and the Chimney Array described in Appendix 1 will house a variety of instruments and sensors. The measurements (or images in the case of the cameras) will be stored in the seven tables shown in the lower portion of Figure 4. Each row of each of these tables will include a measurement (or image), information identifying the array to which the measurement / image was taken, the datetime of the measurement or image, and any other metadata useful in identifying or finding measurements/images needed for a particular application.

## Users, Roles, Measurements, and Data Products

Figure 5 illustrates many aspects of the processes involved in accessing and managing data in the Gulf of Mexico Hydrates Seafloor Observatory DMAS. Processes reflected by Figure 5 include:

1) calibration of raw data to produce calibrated data,
2) creation of data products from calibrated data,
3) annotation of raw data, calibrated data, and data products, and
4) user access to raw data, calibrated data, and data products.

## Calibration of Raw Data

As mentioned previously, raw measurement data will be stored in the tables shown in Figures 3 and 4. In Figure 5, these individual tables are referred to collectively by the

“GENERIC RAW MEASUREMENT TABLE” box. When viewing Figure 5, it should be kept in mind that this one box actually represents 14 measurement tables, which will all have the same relationships with the gmh_user_annotations and gmh_calibrated_measurements tables shown in Figure 5.

Each raw measurement may be calibrated one or more times in order to produce a set of calibrated measurements. These calibrated measurements are stored in the gmh_calibrated_measurement table.

## Creation of Data Products

Once raw measurements have been calibrated they may be used, perhaps in concert with other calibrated measurements, to produce a data product. Each data product is based on a particular pre-defined “standard” data product, although the design does accommodate the need to sometimes create products in an *ad hoc* fashion. Each of these standard data product types has a particular output type (e.g., image, spreadsheet, or textual report), and these output types are defined by the gmh_data_product_outputtypes table.

## Annotation of Data

Each user of the DMAS can produce and store annotations for raw measurements, calibrated measurements, and/or data products. These annotations are stored in the gmh_user_annotations table.

## User Access

The Gulf of Mexico Hydrates Seafloor Observatory DMAS may have many different types of users, with these types distinguished by the sorts of things they can do with the data. These user types, called “roles”, are stored in the gmh_roles table, while the DMAS users are enumerated in the gmh_users table. Each user might play multiple roles, so the gmh_user_roles table keeps track of the roles that each user can play. As mentioned previously, users can create annotations and data products; those two relationships are illustrated by the relationships appearing on the center left side of Figure 5.

It may be the case that not every user is allowed to create every type of data product. The relationship between user roles and the types of data products that can be created is represented by the gmh_data_product_permissions table.

# Design for MFI Software Enhancements

There are two main objectives in this design stage:

- To identify the modifications to both the simulation and the matched field components of the BCOMFI[6] system that would be required to allow its use with the proposed 3D array configuration for the monitoring station arrays.
- To design experiments that would allow investigation of the benefits (and possible drawbacks) of using the data from the 3D arrays on matched field inversion.

We note that, since real data from the vertical array will not be available until February 2006, it has not been possible to experiment with MFI techniques applied to calibration data obtained from this array. Because such experimentation is essential to guide the development of the actual MFI-based monitoring software, it is clear that this development must await the results of such testing with real data. The above objectives provide a basis for the analysis of the real data to be obtained from the present and future arrays,
and for the effective processing of these data when they become available.

## Design of BCOMFI Modifications

The changes to BCOMFI to accommodate 3D arrays involve the setup of the propagation modeling programs ORCA and RAM. ORCA is used in both simulation and matched field components, and it already has an option to make use of an array of arbitrary geometry. RAM is used only in the simulation component of BCOMFI and does not have such an option.

The approach that will be taken to implement and test 3D arrays will involve the following steps:

- Modifying the IDL routines involved in reading the array, setting up the options file, and reading in the computed field, both for the simulation and matched field components.
- Modifying the IDL routines that set up and read in the RAM file for the simulation component.
- Using RAM to generate simulated fields for 3D arrays and cross-checking these fields by applying matched field techniques using ORCA.

These three stages are described in more detail in the following sections.

### Modifications to ORCA-Based Modeling

[6] **B**arrodale **CO**mputing **M**atched **F**ield **I**nversion, the software system developed by BCS, Version 1.0 of which was delivered on August 22, 2005.

The following steps will be required to implement the use of 3D arrays with ORCA:

- As a first step, a simple 3D array will be defined, and two other arrays representing partial (vertical/horizontal-cross) and full (vertical/horizontal-cross/borehole) 3D configurations will be specified.
- It will be then verified that the standalone version of ORCA can be run using both the simple and the realistic 3D arrays.
- The BCOMFI simulation routines that read in the array, set up the ORCA options file, and read in the output complex field file will be modified as required to handle 3D arrays.
- The BCOMFI matched field routines that read in the arrays, set up the ORCA options file, and read in the field file will be modified as required to handle 3D arrays.
- The modified system will be tested internally by simulating data for a 3D array and analyzing these data using matched field techniques.

No new routines will be required as a result of these changes, and the existing interfaces will remain unchanged.

## Modifications to RAM-Based Modeling

Unlike ORCA, RAM does not have an explicit provision for using 3D arrays. The output of RAM is simply a 2D grid of complex field values along a vertical plane intersecting the source and the receiver positions. If the array being used is vertical or if the array has all its sensors in this plane, then using RAM is most efficient, since it is a simple matter of interpolating the grid at the appropriate ranges to obtain the field values at the array locations. However, if the array is truly 3D, many of the sensors will no longer lie in the plane, and the corresponding fields will vary with bearing as well as range.

One option for implementing 3D arrays using RAM is to perform a separate run for each sensor with a different source-receiver path in the horizontal plane. However, this option would be very time-consuming, since numerous paths would in general be involved for a 3D array, and RAM runs themselves are time-intensive compared to ORCA.

An alternative approach is proposed for investigation and possible adoption. This approach assumes that the horizontal extent of the array is small enough that the source-receiver paths are effectively independent of bearing. In this case, a single RAM run from the source to a suitable reference point of the array (e.g., the geometric center) could be performed. For each sensor, the source-sensor range could then be computed and used to interpolate the grid in a similar way to the approach taken for planar arrays.

If the environment is range-independent, the two approaches will produce identical results. If it is not, then the results will differ by amounts which increase with the difference between the respective range-dependent paths. It is anticipated that for

relatively small discrepancies in the paths (say, 100 m in a range of 2 km), a mildly range-dependent environment, and suitable frequencies, the agreement could be sufficiently close to allow the application of the more rapid approach. This would allow the generation of fields for a 3D array in a range-dependent environment in a reasonable time frame. (This approach would also facilitate the eventual use of RAM in matched field applications.)

The implementation of this approach (which involves only the simulation component of BCOMFI) will involve the following steps:

- A convention for defining the array reference point will be defined.
- The BCOMFI simulation routines that read in the array and set up the RAM input file will be modified as required to handle 3D arrays.
- The routine that performs the interpolation of the output grid of field values will be modified to compute the source-receiver distances for all sensors and perform the interpolation using those values.

As with ORCA, no new routines will be required as a result of these changes, and the existing interfaces will remain unchanged.

### Verification of Fields for 3D Array

The field values computed for 3D arrays using ORCA and the RAM 3D grid will be compared using the matched field techniques in BCOMFI. Fields for a 3D array in a range-independent environment will be computed using RAM, at ranges from 1 to 5 km. These will be matched using fields computed by ORCA to verify that the correspondence is sufficiently close.

To test the assumption of low sensitivity to bearing for the array configurations of interest, another set of simulations will be performed. This will involve running RAM for each of the individual paths corresponding to the sensors and saving the computed fields to build up vectors of true field values. This will be matched with the vectors computed using the array reference point approach and single RAM runs. These tests will be run for several different spatial conditions and frequencies, with the aim of establishing guidelines as to where the proposed approach could be applied and achieve a satisfactorily high degree of matching.

## Design of Experimentation Using 3D Arrays

The ability to model 3D arrays will allow investigation of the basic question of how this additional information contributes to the effectiveness of MFI relative to simply using a vertical array. A series of experiments will be performed that will generate fields and perform MFI for particular sets of conditions using only the 16-element vertical array, and the results will be compared with those obtained when the two above variants of 3D arrays are used. The experimentation that will be performed using this capability will

include:

- ***Source localization***. The use of a 3D array with horizontal components will allow the estimation/inversion of source bearing. The effectiveness of this estimation will be examined using the two 3D array variants. The effect of the horizontal components on the resolution and the ambiguity of estimation of source range and depth will also be examined.

- ***Inversion of geoacoustic parameters***. Several different geoacoustic models will be defined, and fields will be computed using the vertical array and the two 3D array variants. These will be examined using ambiguity functions and MFI analysis to determine the extent to which the parameters may be more effectively estimated using 3D arrays.

- ***Effect of range***. In previous experiments, the effectiveness of sub-bottom parameter estimation was found to depend strongly on range, with the parameter estimates at larger ranges being more accurate and less ambiguous than those for closer ranges. The effects of using 3D arrays on this range effect for geoacoustic parameter estimation will be investigated.

# Appendix 1: Array Configurations and Numbers of Sensors

### Acoustic Vertical Line Array – VLA

| Sensor | No. of sensors |
|---|---|
| Hydrophone | 16 |
| Orientation Sensor | 16 |
| Thermistor | 8 |

### Horizontal Line Array – HLA (4 Arrays)

| Sensor | No. of sensors per array |
|---|---|
| Hydrophone | 16 |
| 3D Accelerometer | 8 |
| Orientation Sensor | 8 |

### Borehole Vertical Line Array – BLA

| Sensor | No. of sensors |
|---|---|
| Hydrophone | 16 |
| 3D Accelerometer | 8 |
| Thermistor | 16 |

### Benthic Boundary Layer Array – BBLA

| Sensor | No. of sensors |
|---|---|
| ADCP | 1-2 |
| CTD | 1-2 |
| Chlorophyll fluorometer | 1-2 |
| CDOM fluorometer | 1-2 |
| PMT fluorometer | 1-2 |
| Methane sensor (METS) | 1-2 |
| Oxygen sensor | 1-2 |

### Chimney Array – CA

| Sensor | No. of sensors |
|---|---|
| Methane sensor (METS) | 1-2 |
| Conductivity, depth, oxygen sensor | 1 |

| | |
|---|---|
| Monochrome camera | 2 |
| Color camera | 1 |

# Appendix 2: Characterization of "New" Sensors

In BCS's report of January 31, 2005, entitled *Sensor and Data Characterization for the Gulf of Mexico Hydrates Monitoring Station – Version 1.2*, a number of sensors and instruments were identified that were targeted to be included as components of the monitoring station. Since that time, several sensors that were not originally identified have been proposed for inclusion. In this appendix, these "new" sensors are characterized. In addition, some of the previously identified sensors are discussed and/or characterized in greater detail.

## RDI Workhorse Current Meter (ADCP)

| Sensor | Information |
|---|---|
| Consortium contact(s) | Vernon Asper / Jean Whelan / Paul Higley |
| Product name/number | Deep Water Workhorse 300 kHz |
| Manufacturer | RD Instruments |
| Physical quantity measured | a) Velocity<br>b) Direction<br>c) Tilt<br>d) Temperature |
| Dimensionality of measurements | a) Vector array (units cm/s)<br>b) Vector array (units degrees magnetic)<br>c) Scalar (units degrees)<br>d) Scalar (units degrees Celsius) |
| Range | a) 0 to 500<br>b) 0 to 360<br>c) 0 to 15<br>d) –5 to 45 |
| Resolution | a) 0.1<br>b) 0.01<br>c) 0.01<br>d) 0.01 |
| Type of sampling (continuous or sporadic, variability) | Periodic (specified sample duration every hour)<br>Continuous |
| Typical sampling rate | 1 Hz sampling averaged to mean values for a specified sampling duration (5-60 minutes per hour) |
| Peak sampling rate | 2 Hz |
| Typical data transfer rate (per sensor) | 1 record (a few kilobytes) per sampling event |
| Peak data transfer rate (per sensor) | 50,720 bps |
| Data format(s) (proprietary or public standard) | Variable; formats available in product manuals |
| Required metadata and how | The RDI family of ADCPs has an extensive |

| | |
|---|---|
| they are created | command set for instrument configuration and operation. For metadata, operational modes, dynamic configuration and data formats, please refer to the document “Workhorse Commands and Output Data Format”. |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled<br>Streaming<br>Internally logging |
| Calibration requirements | Calibrations are not typically required for the transducers, although the compass may require swinging. The sensing elements are not particularly susceptible to biofouling. |
| Brochure/description available | Yes |
| Notes | a) Data is typically stored on the instrument in a single binary file. The data acquisition frequency, averaging intervals (if any) and other sampling characteristics are determined in advance by the investigating (scientific) authority and programmed into the instrument’s configuration. Available storage in the instrument (typically on the order of 40 MB) usually constrains the sampling parameters for a given deployment.<br>b) Data formats for post-processing are typically determined by the user after recovery of the information. After an instrument’s data file has been downloaded to a PC, software provided by RDI processes the raw binary data into the data format (typically one of many ASCII formats) as instructed by the user at this point.<br>c) Configuration files are stored in the instrument, but it is important to maintain pre- and post-deployment configuration information for later stages of data processing.<br>d) Compass calibration data may be required to correct directional current information. This calibration (“compass swinging”) is usually done on-site to reflect the local magnetic environment.<br>e) Timestamps are stored along with the data in the instrument. The instrument |

| | |
|---|---|
| | maintains an on-board real-time clock, but it is useful to record the instrument's clock values pre- and post-deployment in order to correct for clock drift.<br>f) Instrument configuration (model, frequency, serial number, and sampling parameters) can be polled interactively and dumped to a configuration file. Note that there is a fairly large set of configurable options that can be set and should be recorded. This metadata should be considered an integral and essential component of the instrument data. A sample of this metadata is provided at the end of this Appendix.<br>g) The standard post-processing methods available via the provided RDI software can be replaced with or augmented by an investigator's own processing methods. Examples of non-standard post-processing include compensation for mooring pull-down in strong currents; compensation for real-time clock drift; correction for beam bias (intensity); correction for temperature variation in the upper water column; and filtering for data quality (e.g., examining error velocity and vertical velocity components). |

## Seabird SBE-37 SI CTD Sensor (and optionally SBE-5 pump)

The SBE 37-SI (MicroCAT) is a high-accuracy conductivity and temperature sensor (pressure optional) without internal batteries or memory. It is designed for moorings and other long-term fixed-site deployments at depths up to 7,000 m (23,000 ft), and is easily integrated with current meters, moored instruments, remotely operated vehicles, AUVs, or Mini-subs. It includes a standard serial interface. Titanium and other non-corroding materials are used to minimize maintenance and ensure long-life.

The SBE 37-SIP combines most of the features of the 37-SI with an integral, internal pump. The pump provides improved conductivity response and anti-foul protection.

The SBE 37-SI series of CTDs does not have its own internal memory for data logging.

The MicroCAT is supplied with SEATERM, a powerful Win 95/98/NT/2000/XP terminal program for easy communication and data retrieval. SEATERM can send commands to the MicroCAT to provide status display, data acquisition setup, data display and capture, and diagnostic tests.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Jean Whelan / Rich Camilli / Paul Higley |
| Product name/number | SBE-37 SI |
| Manufacturer | Seabird Electronics Inc. |
| Physical quantity measured | a) Temperature<br>b) Conductivity<br>c) Pressure |
| Dimensionality of measurements | a) Scalar (units deg C)<br>b) Scalar (units S/m)<br>c) Scalar (units dbar) |
| Range | a) –5 to +35<br>b) 0 to 7<br>c) 0 to 7,000 |
| Resolution | a) 0.0001 (accuracy 0.002)<br>b) 0.00001 (accuracy 0.0003)<br>c) 0.002% (accuracy 0.1%) |
| Type of sampling (continuous or sporadic, variability) | **Autonomous Sampling** – At pre-programmed intervals, the MicroCAT samples. There are two types of autonomous sampling:<br>• *Continuous sampling* at the fastest rate possible (0.66 second minimum), or<br>• *Interval sampling* at intervals of 10 seconds to 9.1 hours. Jumper positioning determines whether the MicroCAT goes to sleep between samples.<br>**Polled Sampling** – On command from a computer or satellite, radio, or wire telemetry equipment, the MicroCAT takes a sample and transmits the data.<br>**Serial Line Sync** – In response to a pulse on the serial line, the MicroCAT wakes up, samples, transmits the data, and goes to sleep. |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 1 Hz |
| Typical data transfer rate (per sensor) | 768 bps (RS-232 9600,N,8,1) |
| Peak data transfer rate (per sensor) | Approx. 768 bps |
| Data format(s) (proprietary or public standard) | Variable; formats available in product. The SBE 37SI can be configured to output ASCII data in hexadecimal format or in calibrated engineering units.<br>**FORMAT=1** (default)<br>ttt.tttt,cc.ccccc, pppp.ppp, dddd.ddd, sss.ssss, vvvv.vvv, |

| | |
|---|---|
| | rrr.rrrr, dd mmm yyyy, hh:mm:ss<br><br>Leading zeros are suppressed, except for one zero to the left of the decimal point. |
| Required metadata and how they are created | A comprehensive list of instrument-specific metadata is provided in the instrument manual. |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled<br>Streaming |
| Calibration requirements | To resolve fine structure and detect very small changes in property measurements, the sensors must be calibrated periodically. In particular, the conductivity cell is susceptible to biofouling and needs both cleaning and calibration at an interval determined by the rate of algal growth on the sensing elements. |
| Brochure/description available | Yes |
| Other relevant information | Rich Camilli noted that "the CTD will require its own RS232/485 line, I think we should plan on using a SeaBird. I am working on some circuitry to multiplex all of the fluorometers and METS type sensors together. The data from each sensor will be output to a RS232/485, where a single polling command string will yield a data string that will need to be parsed into the appropriate sensor channels. Each of the sensor data channels will be a 12 bit voltage value ranging from 0 to 5 volts (appearing on the serial line as a low and high byte). The conversion from raw voltages to concentrations can be done on the topside. For each sensor node on the mooring we should expect to parse and store at least 14 channels (this could be expanded later to 25 channels with an additional serial line) at up to 1Hz." |

## Seabird SBE-5 Miniature submersible pump

**SBE 5M**

The SBE 5M pump module consists of a centrifugal pump head and a long-life, DC, ball-bearing motor contained in a compact titanium pressure housing useable to 10,500 m (34,400 ft) deep. The pump impeller and electric drive motor are coupled magnetically through the housing, providing high reliability by eliminating moving seals. Motor drive electronics is intrinsically protected against accidental reversed polarity.

The SBE 5M is standard on the 19plus SEACAT Profiler CTD. It is optional on the SBE 16plus and 16plus-IM SEACAT C-T Recorder, and in custom applications. The pump flushes water through the conductivity cell at a constant rate, independent of the CTD's motion, improving dynamic performance. For applications requiring pumping through additional sensors (for example, a dissolved oxygen sensor), the SNE 5T pump should be used instead.

**SBE 5T**

The SBE 5T pump module is a compact unit consisting of a centrifugal pump head and a long-life, brushless, DC, ball-bearing motor contained in a titanium pressure housing useable to 10,500 m (34,400 ft). The pump impeller and electric drive motor are coupled magnetically through the housing, providing high reliability by eliminating moving seals.

The SBE 5T is a primary component in the SBE 9plus CTD Underwater Unit and SBE 25 SEALOGGER CTD. It is also used as optional equipment on the SBE 16plus, 16plus-IM, and 19plus SEACAT CTDs. The pump flushes water through the conductivity cell at a constant rate, independent of the CTD's motion, improving dynamic performance. The pump may also be suitable for custom applications, where pressure heads are less than 300 cm of water and flow rates are less than 100 ml/s.

## Ocean Marine METS Methane Sensor

The CAPSUMS`s METS is a unique underwater methane sensor solving the problems of the in-situ measurements of $CH_4$. Applications range from the treatment of sewage waters and landfill leachates to water quality monitoring, from biogas production to climate change studies and also includes many aspects of the offshore oil, gas and hydrate exploration activities (e.g., installation safety, environmental protection, exploration surveys).

| Sensor | Information |
|---|---|
| Consortium contact(s) | Jean Whelan / Rich Camilli |
| Product name/number | METS |
| Manufacturer | Capsum Technologie GmbH |
| Physical quantity measured | Methane concentration. The hydrocarbon molecules diffuse out of the liquid through a special silicone membrane into the detector room. The adsorption of hydrocarbon on the active layer leads to electron exchange with oxygen and thus to modification of the resistance, which the electronic transduces into a voltage. |
| Dimensionality of measurements | nmol/l |
| Range | 50 nmol/l to 10 µmol/l |
| Resolution | Up to 24 bit (SDI A/D converter) 0-5V |
| Type of sampling (continuous or sporadic, variability) | Periodic (Polled)<br>Continuous (Polled) |

| | |
|---|---|
| Typical sampling rate | 1 second (sensor has 1-3 sec response time and a t90 time of up to 3 minutes) |
| Peak sampling rate | 1 Hz |
| Typical data transfer rate (per sensor) | 24 bps |
| Peak data transfer rate (per sensor) | 24 bps |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled |
| Brochure/description available | Yes |
| Other relevant information | Transmission protocol: 9600 baud, 8 data bits, 1 stop bit, no parity<br><br>Rich Camilli noted that “the methane and temperature voltages are used to calculate the actual methane concentration using a unique algorithm specific to each sensor. That algorithm is supplied in the documentation that comes with each instrument.”<br><br>Rich Camilli noted with respect to CTDs (assumed that this may be appropriate for METS): “ The data from each sensor will be output to a RS232/485, where a single polling command string will yield a data string that will need to be parsed into the appropriate sensor channels. Each of the sensor data channels will be a 12 bit voltage value ranging from 0 to 5 volts (appearing on the serial line as a low and high byte). The conversion from raw voltages to concentrations can be done on the topside. For each sensor node on the mooring we should expect to parse and store at least 14 channels (this could be expanded later to 25 channels with an additional serial line) at up to 1Hz.” |

## Chlorophyll Fluorometer – Turner Designs SCUFA

The SCUFA (Self-Contained Underwater Fluorescence Apparatus) is an accurate, simple-to-use and versatile submersible fluorometer for chlorophyll and dye tracing applications. The SCUFA has been designed to operate in a wide range of concentrations and environmental conditions. The capability to

program sampling intervals via the Windows Interface Software and the automatic range control enable the user to configure the SCUFA for any type of profiling or moored deployment.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Jean Whelan / Rich Camilli |
| Product name/number | SCUFA (Self Contained Underwater Fluorescence Apparatus) |
| Manufacturer | Turner Designs |
| Physical quantity measured | a) Fluorescence<br>b) Turbidity |
| Dimensionality of measurements | a) µg/l (chlorophyll) or cells/ml (cyanobacteria)<br>b) NTU |
| Range | a) 0.02 µg/l detection limit; 4 orders of magnitude dynamic range<br>b) 150 cells/ml detection limit; 4 orders of magnitude dynamic range |
| Resolution | 12 bit 0-5V signal (1.22 mV) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 5 Hz |
| Typical data transfer rate (per sensor) | 528 bps (528 bit data string) |
| Peak data transfer rate (per sensor) | 2640 bps |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled |
| Brochure/description available | Yes |

## Chlorophyll Fluorometer – Wetlabs ECO FL Series (Open Path)

Chlorophyll-*a* fluorescence serves as a valuable indicator of active phytoplankton biomass and chlorophyll concentrations in waters. This measurement is used for tracking biological variability and abundance in the water column.

The *Environmental Characterization Optics*, or *ECO* miniature fluorometer allows the user to measure relative chlorophyll, CDOM, uranine, phycocyanin, or phycoerythrin concentrations by directly measuring the amount of fluorescence emission in a sample volume of water. The *ECO* uses an LED to provide the excitation source. An interference filter is used to reject the small amount of out-of-band light emitted by the LED. The light from the source enters the water

volume at an angle of approximately 55–60 degrees with respect to the end face of the unit. Fluoresced light is received by a detector positioned where the acceptance angle forms a 140-degree intersection with the source beam. An interference filter is used to discriminate against the scattered excitation light.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Jean Whelan / Rich Camilli |
| Product name/number | ECO FL series |
| Manufacturer | Wetlabs |
| Physical quantity measured | Fluorescence (470/695 nm) |
| Dimensionality of measurements | µg/l (chlorophyll) |
| Range | 0.01 µg/l detection limit up to 125 µg/l |
| Resolution | 14 bit 0-5V signal (0.3 mV or 0.01 µg/l) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 8 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |
| Peak data transfer rate (per sensor) | 2560 bps (est.) (19,200 baud communication rate) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## CDOM Fluorometer – Wetlabs ECO FL Series (Open Path)

The CDOM ECO allows measurement of CDOM fluorescence across a wide range of environments, from mangrove swamps to oligotrophic blue water.

The *Environmental Characterization Optics*, or *ECO* miniature fluorometer allows the user to measure relative chlorophyll, CDOM, uranine, phycocyanin, or phycoerythrin concentrations by directly measuring the amount of fluorescence emission in a sample volume of water. The *ECO* uses an LED to provide the excitation source. An interference filter is used to reject the small amount of out-of-band light emitted by the LED. The light from the source enters the water volume at an angle of approximately 55–60 degrees with respect to the end face of the unit. Fluoresced light is received by a detector positioned where the acceptance angle forms a 140-

degree intersection with the source beam. An interference filter is used to discriminate against the scattered excitation light.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Jean Whelan / Rich Camilli |
| Product name/number | ECO FL series |
| Manufacturer | Wetlabs |
| Physical quantity measured | Fluorescence (370/460 nm) |
| Dimensionality of measurements | ppb (Concentration) |
| Range | 0.25 ppb to 300 ppb |
| Resolution | 14 bit 0-5V signal (0.3 mV or 0.25 ppb) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 8 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |
| Peak data transfer rate (per sensor) | 2560 bps (est.) (19200 baud communication rate) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## Chlorophyll Fluorometer – Wetlabs WETStar Series (Flow-through)

The WETStar fluorometer provides a high level of performance for detecting various types of fluorescence. It uses a central axial optical tube to provide efficient signal output.

The meter has a unique flow tube design that lends itself to both pump-through and flow-through operation. Its power input of 7–15 VDC and 0–5 VDC analog output allow it to easily mate with existing CTD packages.

The WETStar series is compatible with external pumps, such as the Seabird SBE-5.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Jean Whelan / Rich Camilli |

| | |
|---|---|
| Product name/number | WETStar |
| Manufacturer | Wetlabs |
| Physical quantity measured | Fluorescence (460/695 nm) |
| Dimensionality of measurements | µg/l (chlorophyll) |
| Range | 0.03 µg/l detection limit up to 75 µg/l |
| Resolution | 12 bit 0-5V signal (1.22 mV or 0.03 µg/l) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 8 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |
| Peak data transfer rate (per sensor) | 2560 bps (est.) (9600,N,8,1 communications) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## CDOM Fluorometer – Wetlabs WETStar Series (Flow-through)

The WETStar fluorometer provides a high level of performance for detecting various types of fluorescence. It uses a central axial optical tube to provide efficient signal output.

The meter has a unique flow tube design that lends itself to both pump-through and flow-through operation. Its power input of 7–15 VDC and 0–5 VDC analog output allow it to easily mate with existing CTD packages.

The WETStar series is compatible with external pumps, such as the Seabird SBE-5.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Jean Whelan / Rich Camilli |
| Product name/number | WETStar series |
| Manufacturer | Wetlabs |
| Physical quantity measured | Fluorescence (370/460 nm) |
| Dimensionality of measurements | ppb (Concentration) |
| Range | 0.100 ppb to 100, 250 or 1000 ppb |
| Resolution | 14 bit 0-5V signal (0.3 mV or 0.1 ppb) |
| Type of sampling (continuous or sporadic, | Periodic (polled)<br>Continuous (polled) |

| | |
|---|---|
| variability) | |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 8 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |
| Peak data transfer rate (per sensor) | 2560 bps (est.) (9600,N,8,1 communications) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## Seapoint Sensors UV CDOM Fluorometer

**A new sensor for measuring chromophoric dissolved organic matter (CDOM), the Seapoint Ultraviolet Fluorometer, is now available. This compact sensor features high performance, low power requirements, selectable ranges, and 6,000 meter depth rating. Applications include ocean color research, wastewater discharge quality monitoring, and DOM cycle studies.**

The Seapoint Ultraviolet Fluorometer (SUVF) is a high-performance, low power instrument for *in situ* measurement of chromophoric dissolved organic matter (CDOM). Its small size, very low power consumption, high sensitivity, wide dynamic range, 6,000 meter depth capability, and open or pump-through sample volume options provide the power and flexibility to measure CDOM in a wide variety of conditions. The SUVF uses modulated ultraviolet LED lamps and excitation filter to excite CDOM. The fluorescent light emitted by the CDOM passes through a blue emission filter and is detected by a silicon photodiode. The low level signal is then processed using synchronous demodulation circuitry which generates an output voltage proportional to CDOM concentration. The SUVF may be operated with or without a pump. The sensing volume may be left open to the surrounding water, or, with the use of the supplied cap, can have water pumped through it. Two control lines allow the user to set the range to one of four options. These lines may be hardwired or microprocessor-controlled to provide a suitable range and resolution for a given application. The sensor is easily interfaced with data acquisition packages; a 5 ft pigtail is supplied. Custom configurations are available.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Jean Whelan / Rich Camilli |
| Product name/number | Seapoint Ultraviolet Fluorometer |
| Manufacturer | Seapoint Sensors |
| Physical quantity measured | Fluorescence (370/440 nm) |
| Dimensionality of | µg/l |

| | |
|---|---|
| measurements | |
| Range | 0.1 µg/l to 50, 150, 500 or 1500 µg/l |
| Resolution | 14 bit 0-5V signal (0.3 mV or 0.1 µg/l) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 1 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |
| Peak data transfer rate (per sensor) | 2560 bps (est.) (9600,N,8,1 communications) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## Chlorophyll Fluorometer – Chelsea Technologies Group UV AQUA-Tracka

The UV AQUA-Tracka is a submersible fluorometer to monitor the concentration of hydrocarbons (360nm) or Gelbstoff (440nm) in a wide range of oceanographic applications. In support of this, it has been designed to be deployed from undulating vehicles, moored or profiling systems. This robust, compact, lightweight instrument has built-in test (BITE) circuitry which ensures high stability. The instrument is easy to use and gives accurate and repeatable measurements.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Jean Whelan / Rich Camilli |
| Product name/number | UV AQUA-Tracka |
| Manufacturer | Chelsea Technologies Group |
| Physical quantity measured | Hydrocarbon Fluorescence (239/360 nm) |
| Dimensionality of measurements | µg/l (Carbazole) |
| Range | 0.001 µg/l detection limit up to 10 µg/l |
| Resolution | 12 bit 0-4V or 0-8V signal (0.001 µg/l) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 8 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |

| | |
|---|---|
| Peak data transfer rate (per sensor) | 2560 bps (est.) (9600,N,8,1 communications) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## PixeLINK PL-A741 Monochrome Camera

The PL-A741 is a popular machine vision camera. It is a monochrome 1.3 megapixel digital FireWire camera with global shutter, external trigger and a rich set of on-board features that provides high-quality digital video streams to the computer.

**Color/Mono** : Mono
**Resolution** : 1280 x 1024
**Frame Rate** : 8000 max, 105 @ 640 x 480, 27 @ 1280 x 1027
**Bit Depth** : 8 or 10
**Pixel Pitch** : 6.7 µm
**Lens Mount** : C 2/3
**Interface** : FireWire
**Trigger Type** : H/W or S/W
**Shutter Type** : Global
**Gen Purpose Outputs** : 2 Multi-function

## PiexLINK PL-A742 Color Camera

The PL-A742 is a color 1.3 megapixel digital FireWire camera with global shutter, external trigger and a rich set of on-board features that provides high-quality digital video streams to the computer.

**Color/Mono** : Color
**Resolution** : 1280 x 1024
**Frame Rate** : 8000 max, 105 @ 640 x 480, 27 @ 1280 x 1027
**Bit Depth** : 8 or 10
**Pixel Pitch** : 6.7 µm
**Lens Mount** : C 2/3
**Interface** : FireWire
**Trigger Type** : H/W or S/W
**Shutter Type** : Global
**Gen Purpose Outputs** : 2 Multi-function

## Sample RDI ADCP Configuration File

(commented by user, instrument set up prior to deployment)

```
;WorkHorse-300 s/n 1234
;1-year deployment, starting in mid September 200x
;  operating at 100 m depth in 200 m of water.
;  Memory:  40 Mb
;  Battery: 1x400 watt-hours at 0 celsius

;Reset WH300 to factory settings
CR1
;Flow Control - automatic ensemble, ping, com/ascii if enabled, com disabled, rcdr enabled
CF11001
;Power level
;CQ Power-level not selectable on WH300

;Heading alignment
EA00000
;Heading bias
EB00000
;Instrument depth specified as 1100 dm
;Target is 110 m, but could be as shallow as 100 m or as deep as 120 m
;Mooring pull-down estimated as 2 m
;WH-0272 has no pressure sensor for depth determination
ED01100
;Salinity specified as 32
ES32
;Coordinate transform, Earth coordinates, use pitch/roll, allow 3-beam, allow bin mapping
EX11111
;Data source: calculate SV, depth, hdg, pitch, roll, specify salinity, measure T
EZ1011101

;ENSEMBLE INTERVAL FOR DEPLOYMENT IS 30 minutes
TE00:30:00.00
;ENSEMBLE INTERVAL CAN BE REDUCED BY 10X FOR BENCH TEST
;TE00:03:00.00

;PING INTERVAL FOR DEPLOYMENT IS 1 second
;Allow sufficient time for both water-column (WPnn) and bottom-track (BPnn) pings
;  30 pings for WC plus 0 pings for BT, at 59 sec ---> 29.5 minutes
TP00:59.00
;PING INTERVAL CAN BE REDUCED BY 10X FOR BENCH TEST
;TP00:05.90

;SELECT START TIME BEFORE DEPLOYMENT (yy/mm/dd)
TF02/09/15,17:00:00
```

```
;Use reduced bandwidth to extend range
WB1
;Record velocity, correlation, intensity, percent good (default)
WD111100000
;Lockout is 176 cm (default for 300 kHz)
WF0176
;Cell size is 800 cm
WS0800
;Number of depth cells is 17
WN017
;Pings per ensemble is 30
WP00030
;Ambiguity velocity change to 050 cm/s radial, 150 cm/s horizontal (default is 170 cm/s radial)
WV050

;Fish rejection threshold set to 30 dB
WA064
;Error velocity rejection threshold set to 200 mm/s as 3 SD's
WE0200
;Correlation rejection threshold at default value (64)
WC64


;Delay 1 ensemble before trying to re-acquire the bottom
;BD command is not available for WH bottom-tracking (BD001, although documented in the
    manual
;No BT <<< Error velocity rejection threshold set to 200 mm/s as 3 SD's
;BE0200
;No BT <<< Set minimum expected range to bottom (ice) at 500 dm
;BF00500  Need to think about this one (conflicting advice)
;No BT <<< Bottom-tracking pings per ensemble is 10 (if BP000000, no bottom tracking)
;BP010  Need to think about this one (needs a lot of power)
;No BT <<< Bottom track mode 4
;BM4
;No BT <<< Limit bottom tracking to 2000 decimeters
;BX2000
;No BT <<<Limit BT pulses to 30% of water depth (default: use smaller values only in very
    shallow water)
;&R30

;Keep parameters as user defaults
CK
;Start pinging now or after delay
CS

;Deployment hours = 17520.00
```

;Temperature = -1.50
;Frequency = 307200

# Appendix 3: Phone Interviews with Sensor Experts

## Interview with Paul Higley (Specialty Devices Inc.)

Location: Barrodale Computing Services Ltd. (BCS), Victoria, BC, Canada
Time: 12:00-13:30, Thursday, November 10, 2005
Present: Cedric Zala, Mike Dunham-Wilkie, Kent Berger-North (BCS)
On Phone: Paul Higley, Larry Higley (Specialty Devices Inc.)

1. Introductions were made; Larry is a software developer/integrator at SDI.
2. Paul indicated that SDI was responsible for the seismic arrays and an oceanographic array data collection platform less sensors. There are others responsible for the benthic boundary array and geochemical experiments.
3. The seismic arrays and oceanographic arrays will operate independently, probably until 2007-2008, and will not be networked and integrated initially.
4. At one point in time there had been a specification for the seafloor observatory which indicated that sensor packages intended to interface to the observatory should have data loggers. SDI issued an interface specification that defined serial communications, Ethernet, and available power.
5. Data access will be direct access to the logger hard drives via Ethernet.
6. Specific formats and headers will be used for SDI's data; some are readily available (e.g., the P16 formats for the hydrophones) while others will have to wait until the system is integrated.
7. With respect to data volume, the seismic arrays will account for the majority of the data, followed by camera imagery. The remaining data (primarily oceanographic) will be on the order of kilobytes per week or month.
8. Typical data events may consist of a four-day near-continuous operation of the hydrophone arrays (6 second bursts at 12 second intervals) generating approximately 20GB.
9. Due to high data volumes, the intention is to recover the data from the data loggers on the sea floor to the buoy and physically recover (swap) the hard drives in the surface buoy.
10. The duration and frequency of data sets are expected to be 3-4 days every couple of months (seasonally).
11. Norm Farr at WHOI is responsible for the oceanographic array. Access to the seafloor nodes will initially be via submersible (SSD/ROV).
12. Integration of the seismic and oceanographic arrays will not take place until 2007-2008.
13. The geometry of the oceanographic array will be a single mooring with multiple connected pods that are placed in position by manned vehicle, SSD/ROV.
14. The SDI formats for the accelerometers and orientation sensors are TBD.
15. Examples of the orientation output may be available from a data set collected in Italy (2004). **Paul** will send Cedric an example of this data**.**
16. The thermistor, accelerometer, orientation sensor and hydrophone components of the seismic array will most likely be stored in separate files due to differing sampling regimes (acquisition intervals, sample frequency, and sample duration).

17. All seismic data will be referenced to a shot number (from the acoustic sources), a.k.a. “pop”.
18. Every pop will be stored in a separate file.
19. A log file will maintain the shot times. As timing is critical for the seismic data processing, all time information is relative to the shot “T-zero” and requires <<millisecond or better resolution. T-zero references are generated on the ship and sent down to the instrumentation package. While real-time clock (RTC) information will be recorded, it is not used for reference purposes. Real-time clocks in the logging computers, instruments, and sensors are subject to drift.
20. The temperature data is stored with RTC information.
21. There are a number of important metadata files that will be required for analysis of the acoustic/seismic data sets:
    a. The “Ship Log” data file. This contains navigation data for the ship (GPS positions and time) and gun array, and for the buoy over the gun array. It contains seismic shot file metadata, ship motion data, etc. USBL data are maintained by the vessel. (To be confirmed). Scott Sharpe will have the ship log formats. **Paul** will send Cedric a sample file.
    b. The “Ship Pop” data file. This records the shot number, the GPS position, the time, the delay from the gun, etc. This is maintained by the ship-board acquisition and control system. Larry will have the ship “pop” formats. **Paul** will send Cedric a sample file.
    c. The “Deep Tow” file. This contains data related to the hydrophone far field from guns close to the seafloor. The seismic data file is stored on the ship in real time. The format has yet to be determined, though it will likely be SEG-Y if 24-bit or P16 if 16-bit.
    d. The Sled system. This contains the seismic shear wave source, two orthogonal sources on the seafloor. It will contain 3-axis accelerometer data, orientation (of sled) data, hydrophones, USBL data and shot number. It is placed on the sea floor, triggered, and pulled 1 m at a time across the seafloor. It is also a good way to test the data requirements for the downhole array as the configuration will be similar.
    e. The AUV data. The AUV will likely be on a streamer. A hydrophone is located on the nose and the vehicle is placed close to the seafloor. Deployment is likely in 2006-2007. Data will probably be internally stored, single channel hydrophone. Position and time information is not necessary as the wave will already be well developed; the hydrophone will pick up the signal and subsequent reflection.
22. Graphics and descriptions of these systems may be on the Ole Miss web site:
    http://www.olemiss.edu/depts/mmri/programs/gulf_res.html
    http://www.olemiss.edu/depts/mmri/programs/station_schem_02.jpg
23. The seismic array data sampling regime will likely be as follows:
    a. Hydrophones: 16 channels for 1 to 4 seconds (assume 4) at 10 kHz of 16-bit data continuously for 48-96 hours.
    b. Accelerometers: 8 channels x 3 sensors (3 axes) for 1-4 seconds of 24-bit data at 1 kHz continuously for 48-96 hours.

c. Thermistors: 8 channels for a 12-second record at 4 Hz of 24-bit data every 6 hours for a year.
d. Two sets of 2 orientation sensors (heading, pitch, roll) at 4 Hz for 1-4 seconds.
e. The orientation sensor works as a current meter and may be activated once per hour throughout the year.

24. The deep tow is similar to the accelerometer array; 16-bits at 10 kHz. There is some interest in going faster, up to 100 kHz. The duration will be 1-4 seconds per pop continuously for 48-96 hours.
25. The sled is the same as the other hydroacoustic / seismometer arrays, with only 50 pops (50 m drag, 1 pop per m).
26. The AUV will be the same as the deep tow.
27. The number of arrays will be:
    a. One vertical array with 16 hydrophones.
    b. One downhole array with 8 sets of accelerometers, 16 hydrophones, and thermistors.
    c. Four horizontal arrays (like the downhole), with maybe twice as many accelerometers as the vertical arrays.
28. Jeff Chanton and Laura Lapham data sets representing pore fluids from the sediment will be tabular data.
29. The bottom sediment thermistors (x3) on the bottom hole will collect temperature flux of the near bottom at 30 minute intervals.
30. Two arrays have been installed (already) in the sub-bottom as 10 m gravity entrenchments. One is a pore fluid array and the other is a temperature array.

## Interview with Norm Farr (Woods Hole Oceanographic Institute)

Location: Barrodale Computing Services Ltd., Victoria, BC, Canada
Date: 16:00-16:30, Wednesday, November 16, 2005
Present: Cedric Zala, Mike Dunham-Wilkie, Kent Berger-North (BCS)
On Phone: Norm Farr (Woods Hole Oceanographic Institute - WHOI)

1. Norm is responsible for two oceanographic arrays: the chimney array and the BBLA (Benthic Boundary Layer Array).
2. The chimney array will be situated over a hydrate mass to measure property flux.
3. The BBLA will maintain flotation at the top of the mooring and a cage on the bottom containing the data logger. For the BBLA:
    a. The array will be vertical in orientation.
    b. It is planned to be deployed in Jan/Feb 2006 for 48 hrs and recovered, then redeployed for a week.
    c. Most of the decisions regarding sensor sampling regimes will be determined after inspecting the data collected.
    d. The cage/node at the bottom will have the ability to transport the data over IP.
    e. The data logger may be turned on by event triggers (e.g., METS sensor thresholds).

   f. There will be 3, potentially 6, fluorometers on the BBLA. One will be at 50 m and one at 1.5-2.0 m above the bottom.
   g. The ADCP will be on top. The orientation (upward-looking, downward-looking) has not been determined. [Note: if downward-looking, the instruments and the mooring may contaminate the acoustic field.]
   h. There will be a CTD on the BBLA.
   i. There will be an oxygen sensor on the BBLA (possibly Aanderaa?).
   j. There may be duplicate sensors of everything on the BBLA (TBD).
   k. Data storage, computer, HDD, and batteries will be in the node.
   l. The METS is a vulnerable sensor; it will be on both arrays.
4. The chimney array will contain:
   a. The vision package (still being designed): 2 monochrome and 1 color camera simultaneously collecting data, with data reduction schemes where possible.
   b. An Aanderaa sensor for conductivity, depth, and oxygen.
   c. A METS sensor.
   d. All sensors will be connected to the vision package.
5. MBARI has done a lot of work with the MARS observatory. We should look at their database design. Contact: Mark Chaffey. The mooring system will be similar to their MOOS.
6. All data in the arrays will be centrally logged in their respective nodes, with a common timestamp, if possible.
7. The controller in the node will drive the duty cycles of the sensors.
8. How the event triggering will be realized has yet to be addressed.
9. In late January, Norm will be able to commit to some form of data structure.
10. The AUV is completely independent of the other arrays. The data products will likely be benthic, chemical and photographic maps (lat/long).
11. WHOI is currently logging meteorological and optical data from a buoy, with both raw data and calibration information available to users. Norm has a web-enabled database containing this data and some optical data. **Norm** will send BCS the link to this database.
12. MBARI is running a similar site, without calibration info.
13. The buoy(s) use a “Big S” mooring configuration. See: http://www.mbari.org/news/homepage/2005/mtm3.html
14. Norm is hoping that the ADCP stores data in real units.
15. The mass spectrometer (MS) will go on the AUV, and perhaps on the chimney. Rich Camilli is the contact for the MS, which is possibly manufactured by http://www.monitorinstruments.com.
16. **Norm** will provide BCS with sample data from the arrays when they are available. **BCS** will contact Norm following the Jan/Feb cruise to make arrangements to receive this data, which is proprietary to the Consortium.

# Appendix 4: DMAS Database Design Report

Target DBMS: ODBC Generic Driver
Number of tables: 37
Number of columns: 169
Number of indexes: 6
Number of foreign keys: 44

| Tables | Column | Indexes | Foreign keys | Notes |
|---|---|---|---|---|
| gmh_data_product_permissions | 2 | 0 | 2 | This table relates user roles to the data products that can be requested by users with that role. |
| gmh_user_roles | 2 | 0 | 2 | This table relates GMH users to the access role(s) they have. |
| gmh_standard_data_products | 4 | 0 | 1 | Table of standard data products. Each data product corresponds to a single standard data product type (e.g., "time series of N-minute averages of calibrated data at each of M ordinates"). To accommodate ad-hoc requests, one "standard data product" will be "custom". |
| gmh_calibrated_measurements_usage | 2 | 0 | 2 | Tracks which (calibrated) measurements were used to produce each data product, since a data product can be formed from multiple measurements, and a measurement can be used in more than one data product. |
| gmh_data_product_outputtypes | 2 | 0 | 0 | This table defines the various classes of data products that can be produced from GMH. Examples include images, Excel spreadsheets, Textual reports, etc. |
| gmh_data_products | 7 | 0 | 2 | This table is a catalog of the various data products that have been produced. |
| gmh_roles | 3 | 0 | 0 | This table describes the various database roles that GMH users can have. |
| gmh_users | 4 | 0 | 0 | This table defines the GMH users. |
| gmh_user_annotations | 6 | 0 | 4 | Table of annotations that GMH users have attached to raw and/or calibrated measurements and/or data products. |
| GENERIC RAW MEASUREMENT TABLE | 2 | 0 | 0 | This is just a placeholder representing one of the gmh_*_measurements or the gmh_images table. |
| gmh_calibrated_measurements | 5 | 0 | 1 | This table is a companion to the various raw measurement tables. Post-recovery, the raw measurements can be post-processed by applying calibration parameters to the data. The calibrated data and the calibration parameters are stored in this table.<br><br>In this design, all post-calibration data |

| | | | | |
|---|---|---|---|---|
| | | | | is stored in this single table, since the "blob" data type is sufficiently general to store any type of measurement. If in the future instrument-specific opaque data types are used, then this table can be replaced by a table hierarchy, with a subtable for each different type of measurement. |
| gmh_fluorometer_measurements | 5 | 0 | 1 | Fluorometers will be stored on the BBLA (Farr-1-3). |
| gmh_sled_configuration_history | 4 | 0 | 0 | This table contains information on how the sled system has been configured, either at present (gmhslh_hist_end_datetime is NULL) or in the past (gmhslh_hist_end_datetime is not NULL). The calibration and configuration history of the sensors on the sled are stored in the gmh_sensor_calibration_history and gmh_sensor_configuration_history tables, respectively. |
| gmh_auv_measurements | 5 | 1 | 2 | Table of AUV data measurements, taken from hydrophone in nose of AUV.<br><br>Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?<br>Reference: Higley-1-23e.<br><br>"Position and time information is not necessary as the wave will already be well-developed. The hydrophone will pick up the signal and subsequent reflection." |
| gmh_sled_measurements | 10 | 1 | 3 | Table of "Sled System" data measurements. Sled contains its own shear wave source; consequently no foreign key to a shot number is needed. Measures accelerometer data, orientation data, hydrophone data.<br>Reference: Higley-1-23d. |
| gmh_deep_tow_measurements | 5 | 1 | 2 | Table of "Deep Tow" measurements.<br>Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?<br>Reference: Higley-1-23c. |
| gmh_sensor_configuration_history | 5 | 0 | 1 | This table contains information on how a particular sensor has been configured, either at present (gmhscn_hist_end_datetime is NULL) or in the past (gmhscn_hist_end_datetime is not NULL). |
| gmh_sensor_calibration_history | 5 | 0 | 1 | This table contains information on how a particular sensor has been |

| | | | | |
|---|---|---|---|---|
| | | | | calibrated, either at present (gmhscl_hist_end_datetime is NULL) or in the past (gmhscl_hist_end_datetime is not NULL). |
| gmh_3daccel_measurements | 5 | 1 | 2 | Table of 3D accelerometer measurements.<br>Reference: Higley-1-23b.<br><br>Sampling regime is 8 channels x 3 sensors for 1-4 seconds at 1 kHz of 24 bit data continuously for 48-96 hours. |
| gmh_thermistor_measurements | 5 | 0 | 1 | Table of thermistor measurements. Not tied to shot number; thermistor has own RTC (Higley-1-20). Records every 6 hours for a year.<br>Reference: Higley-1-23c.<br><br>Sampling regime is 8 channels for 12 seconds at 4 Hz of 24 bit data every 6 hours for a year. |
| gmh_orientation_sensor_measurement | 6 | 1 | 2 | Table of orientation sensor measurements.<br>Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?<br>Reference: Higley-1-23d<br><br>There are 2 sets of 2 orientation sensors measuring heading, pitch, and roll.<br><br>Sampling regime is 4 Hz for 1-4 seconds.<br><br>Higley-1-23e says the orientation sensor works as a current meter and may be activated once per hour throughout the year. |
| gmh_hydrophone_measurements | 5 | 1 | 2 | Table of hydrophone measurements.<br>Reference: Higley-1-23a.<br><br>Sampling regime is 16 channels for 1-4 seconds (assume 4) at 10 kHz of 16 bit data continuously for 48-96 hours. |
| gmh_sensor_types | 4 | 0 | 0 | This table contains all the relevant information related to specific sensor types, e.g., name, type of sensor (hydrophone, ADCP, thermistor, etc), generic properties, etc. |
| gmh_shot_log | 5 | 0 | 0 | Database realization of the "shot log". Also called the "Ship Pop" file. See Higley-1-19 and Higley-1-21b. |
| gmh_sensors | 6 | 0 | 2 | In this context a sensor is either an instrument or an individual sensor on a multi-sensor instrument.<br><br>This table contains all the relevant information related to a specific sensor, e.g., what logical array is it |

| | | | | |
|---|---|---|---|---|
| | | | | part of? what type of sensor is it? where is it? etc. Note that sensor measurement values are stored in other tables. |
| gmh_arrays | 4 | 0 | 1 | This table contains information describing the individual acoustic and environmental (logical) arrays of sensors. Note that in this context an "array" is a set of one or more instruments or sensors that together produce a measurement stored in one of the measurement tables. For example, an array of hydrophones produces a multi-channel output file.<br><br>An array can also be a single instrument, if that instrument gives rise to a measurement that is stored. |
| gmh_array_calibration_history | 5 | 0 | 1 | This table contains information on how a particular physical array has been calibrated, either at present (gmhacl_hist_end_datetime is NULL) or in the past (gmhacl_hist_end_datetime is not NULL). |
| gmh_array_configuration_history | 5 | 0 | 1 | This table contains information on how a particular physical array has been configured, either at present (gmhacn_hist_end_datetime is NULL) or in the past (gmhacn_hist_end_datetime is not NULL). |
| gmh_oxygen_sensor_measurements | 5 | 0 | 1 | The BBLA will have oxygen sensors (Farr-1-3). |
| gmh_adcp_measurements | 5 | 0 | 1 | The BBLA will have an ADCP at the top (Farr-1-3). |
| gmh_mets_measurements | 5 | 0 | 1 | The chimney array and the BBLA will both have methane sensors (Farr-1-3 and Farr-1-4). |
| gmh_ctd_measurements | 5 | 0 | 1 | The BBLA will have CTD instruments (Farr-1-3). |
| gmh_mass_spec_measurements | 5 | 0 | 1 | The chimney array may have a mass spectrometer (Farr-1-15). |
| gmh_images | 5 | 0 | 1 | The chimney array will have 3 cameras (Farr-1-4). |
| gmh_cable_arrays | 5 | 0 | 0 | This table contains information describing the individual acoustic and environmental physical arrays (cables) in the observatory. A physical array can contain multiple types of sensors and logical arrays of sensors (e.g., A physical array can contain a logical array of hydrophones + a logical array of thermistors + a selection of individual sensors). |
| gmh_array_connections | 5 | 0 | 2 | This table contains information on which arrays are connected together. From this table and the gmh_sensors and gmh_arrrays tables it will be possible to determine the current topology and 3D positioning of the |

| | | | | |
|---|---|---|---|---|
| | | | | arrays and sensors. NB There is no provision for storing the connection history. |

| Conceptual name: | GENERIC RAW MEASUREMENT TABLE |
|---|---|
| Notes: | This is just a placeholder representing one of the gmh_*_measurements or the gmh_images table. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 2 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| OTHER COLUMNS (SEE PP 3-4) | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| GENERIC RAW MEASUREMENT TABLE_gmh_calibrated_measurements_FK1 | gmh_calibrated_measurements.gmh_global_raw_meas_id | gmh_global_raw_meas_id |
| GENERIC RAW MEASUREMENT TABLE_gmh_user_annotations_FK1 | gmh_user_annotations.gmh_global_raw_meas_id | gmh_global_raw_meas_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. OTHER COLUMNS (SEE PP 3-4)

| | |
|---|---|
| Conceptual name: | OTHER COLUMNS (SEE PP 3-4) |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | See the column descriptions in the appropriate table on diagrams page 3 (Seismic measurements) or page 4 (Environmental measurements). |

| Conceptual name: | gmh_3daccel_measurements |
|---|---|
| Notes: | Table of 3D accelerometer measurements.<br>Reference: Higley-1-23b.<br><br>Sampling regime is 8 channels x 3 sensors for 1-4 seconds at 1 kHz of 24 bit data continuously for 48-96 hours. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 1 |
| Number of foreign keys: | 2 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK,U1) | INTEGER | Not allowed | |
| gmhstl_shot_id (FK,U1) | INTEGER | Not allowed | |
| gmh3dm_3daccelerometer_meas | LONGVARBINARY | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_3daccel_measurements_AK1 (U1) | gmhstl_shot_id | Ascending |
| | gmharr_array_id | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_3daccel_measurements_FK1 | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |
| gmh_arrays_gmh_3daccel_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK,U1)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of sensor performing the measurement.<br><br>Higley-1-23b says that the sampling regime for accelerometers is 8 channels x 3 sensors (3 axes) for 1-4 seconds of 24-bit data at 1kHz continuously for 48-96 hours. THIS IMPLIES THAT THE MEASUREMENT IS A FUNCTION OF ARRAY (COLLECTION OF SENSORS), NOT A SINGLE SENSOR, SO PERHAPS GMHSTL_SENSOR_ID SHOULD BE REPLACED BY GMHARR_ARRAY_ID ? |

3. gmhstl_shot_id (FK,U1)

| | |
|---|---|
| Conceptual name: | gmhstl_shot_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of shot (pop) associated with this measurement. WILL ALL MEASUREMENTS BE TIED TO A SHOT? |

4. gmh3dm_3daccelerometer_meas

| | |
|---|---|
| Conceptual name: | gmh3dm_3daccelerometer_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | Blob of measurement data. Higley-1-17 and Higley-1-18 imply that there is one file per shot (pop). |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

Index details

gmh_3daccel_measurements_AK1

| | |
|---|---|
| Column(s): | gmhstl_shot_id (Asc)<br>gmharr_array_id (Asc) |
| Unique: | Yes |

Foreign key details (child)

gmh_sensors_gmh_3daccel_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | --Empty-- | gmh_sensors.gmhsen_sensor_id |
| Relationship type: | Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_shot_log_gmh_3daccel_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each 3d accelerometer measurement is for a single shot. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_arrays_gmh_3daccel_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmharr_array_id | gmh_arrays.gmharr_array_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each logical array of 3d accelerometers produces measurements. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

| Conceptual name: | gmh_adcp_measurements |
|---|---|
| Notes: | The BBLA will have an ADCP at the top (Farr-1-3). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmhcpm_adcp_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhcpm_adcp_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK) | INTEGER | Not allowed | |
| gmhcpm_adcp_meas | LONGVARBINARY | Not allowed | |
| gmhcpm_adcp_meas_datetime | DATETIME | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_arrays_gmh_adcp_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmhcpm_adcp_meas_id

| | |
|---|---|
| Conceptual name: | gmhcpm_adcp_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the logical array to which the instruments generating this measurement belong. |

3. gmhcpm_adcp_meas

| | |
|---|---|
| Conceptual name: | gmhcpm_adcp_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

4. gmhcpm_adcp_meas_datetime

| | |
|---|---|
| Conceptual name: | gmhcpm_adcp_meas_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

Foreign key details (child)

gmh_arrays_gmh_adcp_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_array_id | gmh_arrays.gmharr_array_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each ADCP produces measurements. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

| Conceptual name: | gmh_array_calibration_history |
|---|---|
| Notes: | This table contains information on how a particular physical array has been calibrated, either at present (gmhacl_hist_end_datetime is NULL) or in the past (gmhacl_hist_end_datetime is not NULL). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmhacl_array_calibration_hist_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhacl_array_calibration_hist_id | INTEGER | Not allowed | |
| gmhcbl_cable_array_id (FK) | INTEGER | Not allowed | |
| gmhacl_hist_start_datetime | DATETIME | Not allowed | |
| gmhacl_hist_end_datetime | DATETIME | Allowed | |
| CALIBRATION PARAMETERS | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_cable_arrays_gmh_array_calibration_history_FK1 | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |

Column details

1. gmhacl_array_calibration_hist_id

| | |
|---|---|
| Conceptual name: | gmhacl_array_calibration_hist_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the physical array calibration history record. Probably a generated key. |

2. gmhcbl_cable_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmhcbl_cable_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of the physical (cable) array to which this history pertains. |

3. gmhacl_hist_start_datetime

| | |
|---|---|
| Conceptual name: | gmhacl_hist_start_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when this record became effective. |

4. gmhacl_hist_end_datetime

| | |
|---|---|
| Conceptual name: | gmhacl_hist_end_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Allowed |
| Notes: | Date and time when this record ceased to be effective. |

5. CALIBRATION PARAMETERS

| | |
|---|---|
| Conceptual name: | CALIBRATION PARAMETERS |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This will be replaced by a set of columns, one column for each calibration parameter, holding the value of that parameter.<br><br>Note that each type of sensor will have different calibration parameters, so either:<br><br>1. The parameters will be stored in subtables, with one subtable for each type of sensor, or<br><br>2. The parameters will be stored in a single table, with the parameters that are not applicable to a particular sensor simply being left blank.<br><br>Option 1 is cleaner, but harder to maintain. It is much easier with Option 2 to add parameters |

for new types of sensors, or add additional parameters for existing types of sensors.

Foreign key details (child)

gmh_cable_arrays_gmh_array_calibration_history_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each physical array has a calibration history. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_array_configuration_history |
|---|---|
| Notes: | This table contains information on how a particular physical array has been configured, either at present (gmhacn_hist_end_datetime is NULL) or in the past (gmhacn_hist_end_datetime is not NULL). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmhacn_array_config_hist_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhacn_array_config_hist_id | INTEGER | Not allowed | |
| gmhcbl_cable_array_id (FK) | INTEGER | Not allowed | |
| gmhacn_hist_start_datetime | DATETIME | Not allowed | |
| gmhacn_hist_end_datetime | DATETIME | Allowed | |
| CONFIG PARAMETERS | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_cable_arrays_gmh_array_configuration_history_FK1 | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |

Column details

1. gmhacn_array_config_hist_id

| | |
|---|---|
| Conceptual name: | gmhacn_array_config_hist_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the physical array configuration history record. Probably a generated key. |

2. gmhcbl_cable_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmhcbl_cable_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of the physical (cable) array to which this history pertains. |

3. gmhacn_hist_start_datetime

| | |
|---|---|
| Conceptual name: | gmhacn_hist_start_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when this record became effective. |

4. gmhacn_hist_end_datetime

| | |
|---|---|
| Conceptual name: | gmhacn_hist_end_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Allowed |
| Notes: | Date and time when this record ceased to be effective. |

5. CONFIG PARAMETERS

| | |
|---|---|
| Conceptual name: | CONFIG PARAMETERS |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This will be replaced by a set of columns, one column for each configuration parameter, holding the value of that parameter.<br><br>Note that each type of sensor will have different configuration parameters, so either:<br><br>1. The parameters will be stored in subtables, with one subtable for each type of sensor, or<br><br>2. The parameters will be stored in a single table, with the parameters that are not applicable to a particular sensor simply being left blank. |

Option 1 is cleaner, but harder to maintain. It is much easier with Option 2 to add parameters for new types of sensors, or add additional parameters for existing types of sensors.

Foreign key details (child)

gmh_cable_arrays_gmh_array_configuration_history_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each physical array has a configuration history. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_array_connections |
|---|---|
| Notes: | This table contains information on which arrays are connected together. From this table and the gmh_sensors and gmh_arrrays tables it will be possible to determine the current topology and 3D positioning of the arrays and sensors. NB There is no provision for storing the connection history. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | gmhacn_connection_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhacn_connection_id | INTEGER | Not allowed | |
| gmhcbl_cable_array_1_id (FK) | INTEGER | Not allowed | |
| gmhacn_array_1_end | CHAR(1) | Allowed | |
| gmhcbl_cable_array_2_id (FK) | INTEGER | Not allowed | |
| gmhacn_array_2_end | CHAR(1) | Allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_cable_arrays_gmh_array_connections_FK1 | gmhcbl_cable_array_1_id | gmh_cable_arrays.gmhcbl_cable_array_id |
| gmh_cable_arrays_gmh_array_connections_FK2 | gmhcbl_cable_array_2_id | gmh_cable_arrays.gmhcbl_cable_array_id |

**Column details**

1. gmhacn_connection_id

| | |
|---|---|
| Conceptual name: | gmhacn_connection_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying an individual connection between arrays. Probably a generated key. |

2. gmhcbl_cable_array_1_id (FK)

| | |
|---|---|
| Conceptual name: | gmhcbl_cable_array_1_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of one of the arrays participating in the connection. |

3. gmhacn_array_1_end

| | |
|---|---|
| Conceptual name: | gmhacn_array_1_end |
| Physical data type: | CHAR(1) |
| Allow NULLs: | Allowed |
| Notes: | Which end of the array is connected to this connection? |

4. gmhcbl_cable_array_2_id (FK)

| | |
|---|---|
| Conceptual name: | gmhcbl_cable_array_2_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of another one of the arrays participating in the connection. |

5. gmhacn_array_2_end

| | |
|---|---|
| Conceptual name: | gmhacn_array_2_end |
| Physical data type: | CHAR(1) |
| Allow NULLs: | Allowed |
| Notes: | Which end of the array is connected to this connection? |

**Foreign key details (child)**

gmh_cable_arrays_gmh_array_connections_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhcbl_cable_array_1_id | gmh_cable_arrays.gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Identifies one of the physical arrays (cable) involved in the connection. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_cable_arrays_gmh_array_connections_FK2

| Definition: | Child | Parent |
|---|---|---|
| | gmhcbl_cable_array_2_id | gmh_cable_arrays.gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Identifies one of the physical arrays (cables) involved in the connection. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_arrays |
|---|---|
| Notes: | This table contains information describing the individual acoustic and environmental (logical) arrays of sensors. Note that in this context an "array" is a set of one or more instruments or sensors that together produce a measurement stored in one of the measurement tables. For example, an array of hydrophones produces a multi-channel output file.<br><br>An array can also be a single instrument, if that instrument gives rise to a measurement that is stored. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 4 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmharr_array_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmharr_array_id | INTEGER | Not allowed | |
| gmhcbl_cable_array_id (FK) | INTEGER | Not allowed | |
| gmharr_array_desc | VARCHAR(255) | Not allowed | |
| ARRAY INFO | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_cable_arrays_gmh_arrays_FK1 | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |
| gmh_arrays_gmh_hydrophone_measurements_FK1 | gmh_hydrophone_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_3daccel_measurements_FK1 | gmh_3daccel_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_orientation_sensor_measurements_FK1 | gmh_orientation_sensor_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_thermistor_measurements_FK1 | gmh_thermistor_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_deep_tow_measurements_FK1 | gmh_deep_tow_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_auv_measurements_FK1 | gmh_auv_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_sled_measurements_FK1 | gmh_sled_measurements.gmharr_hydrophone_array_id | gmharr_array_id |
| gmh_arrays_gmh_sled_measurements_FK2 | gmh_sled_measurements.gmharr_orientation_array_id | gmharr_array_id |
| gmh_arrays_gmh_sled_measurements_FK3 | gmh_sled_measurements.gmharr_accelerometer_array_id | gmharr_array_id |
| gmh_arrays_gmh_fluorometer_measurements_FK1 | gmh_fluorometer_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_mass_spec_measurements_FK1 | gmh_mass_spec_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_images_FK1 | gmh_images.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_oxygen_sensor_measurements_FK1 | gmh_oxygen_sensor_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_ctd_measurements_FK1 | gmh_ctd_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_adcp_measurements_FK1 | gmh_adcp_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_mets_measurements_FK1 | gmh_mets_measurements.gmharr_array_id | gmharr_array_id |
| gmh_arrays_gmh_sensors_FK2 | gmh_sensors.gmharr_array_id | gmharr_array_id |

Column details

1. gmharr_array_id

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the array. Probably a generated key. |

2. gmhcbl_cable_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmhcbl_cable_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID identifying the physical (cable) array upon which the instruments in this logical array are mounted. |

3. gmharr_array_desc

| | |
|---|---|
| Conceptual name: | gmharr_array_desc |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Not allowed |
| Notes: | A one line description of the array. |

4. ARRAY INFO

| | |
|---|---|
| Conceptual name: | ARRAY INFO |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of one or more columns holding other parameters needed to describe this array of sensors and any other relevant metadata. |

Foreign key details (child)

gmh_cable_arrays_gmh_arrays_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each logical array is physically located on a physical (cable) array. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_auv_measurements |
|---|---|
| Notes: | Table of AUV data measurements, taken from hydrophone in nose of AUV.<br><br>Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?<br>Reference: Higley-1-23e.<br><br>"Position and time information is not necessary as the wave will already be well-developed. The hydrophone will pick up the signal and subsequent reflection." |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 1 |
| Number of foreign keys: | 2 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK,U1) | INTEGER | Not allowed | |
| gmhstl_shot_id (FK,U1) | INTEGER | Not allowed | |
| gmhauv_auv_meas | LONGVARBINARY | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_auv_measurements_AK1 (U1) | gmhstl_shot_id | Ascending |
| | gmharr_array_id | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_auv_measurements_FK1 | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |
| gmh_arrays_gmh_auv_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK,U1)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of logical array containing the single AUV hydrophone. |

3. gmhstl_shot_id (FK,U1)

| | |
|---|---|
| Conceptual name: | gmhstl_shot_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | gmh_auv_measurements has gmhstl_shot_id. |

4. gmhauv_auv_meas

| | |
|---|---|
| Conceptual name: | gmhauv_auv_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | Blob of measurement data. Higley-1-17 and Higley-1-18 imply that there is one file per shot (pop). |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |

| | |
|---|---|
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

**Index details**

gmh_auv_measurements_AK1

| | |
|---|---|
| Column(s): | gmhstl_shot_id (Asc)<br>gmharr_array_id (Asc) |
| Unique: | Yes |

**Foreign key details (child)**

gmh_shot_log_gmh_auv_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each AUV measurement is for a single shot. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_arrays_gmh_auv_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmharr_array_id | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | The AUV produces measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_cable_arrays |
|---|---|
| Notes: | This table contains information describing the individual acoustic and environmental physical arrays (cables) in the observatory. A physical array can contain multiple types of sensors and logical arrays of sensors (e.g., A physical array can contain a logical array of hydrophones + a logical array of thermistors + a selection of individual sensors). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmhcbl_cable_array_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhcbl_cable_array_id | INTEGER | Not allowed | |
| gmhcbl_cable_array_code | CHAR(10) | Not allowed | |
| gmhcbl_cable_array_desc | VARCHAR(255) | Not allowed | |
| gmhcbl_cable_array_linework | LONGVARBINARY | Allowed | |
| CABLE ARRAY INFO | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_cable_arrays_gmh_array_connections_FK1 | gmh_array_connections.gmhcbl_cable_array_1_id | gmhcbl_cable_array_id |
| gmh_cable_arrays_gmh_array_connections_FK2 | gmh_array_connections.gmhcbl_cable_array_2_id | gmhcbl_cable_array_id |
| gmh_cable_arrays_gmh_arrays_FK1 | gmh_arrays.gmhcbl_cable_array_id | gmhcbl_cable_array_id |
| gmh_cable_arrays_gmh_array_configuration_history_FK1 | gmh_array_configuration_history.gmhcbl_cable_array_id | gmhcbl_cable_array_id |
| gmh_cable_arrays_gmh_array_calibration_history_FK1 | gmh_array_calibration_history.gmhcbl_cable_array_id | gmhcbl_cable_array_id |

Column details

1. gmhcbl_cable_array_id

| | |
|---|---|
| Conceptual name: | gmhcbl_cable_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the cable. Probably a generated key. |

2. gmhcbl_cable_array_code

| | |
|---|---|
| Conceptual name: | gmhcbl_cable_array_code |
| Physical data type: | CHAR(10) |
| Allow NULLs: | Not allowed |
| Notes: | A more mnemonic code identifying the cable array. e.g. HLA, VBA, etc. |

3. gmhcbl_cable_array_desc

| | |
|---|---|
| Conceptual name: | gmhcbl_cable_array_desc |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Not allowed |
| Notes: | A one line description of the array. |

4. gmhcbl_cable_array_linework

| | |
|---|---|
| Conceptual name: | gmhcbl_cable_array_linework |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Allowed |
| Notes: | Geometric coordinates (lat,long,depth) describing the shape of the cable array. |

5. CABLE ARRAY INFO

| | |
|---|---|
| Conceptual name: | CABLE ARRAY INFO |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of one or more columns, holding other parameters |

needed to describe this cable array, and any other relevant metadata.

| Conceptual name: | gmh_calibrated_measurements |
|---|---|
| Notes: | This table is a companion to the various raw measurement tables. Post-recovery, the raw measurements can be post-processed by applying calibration parameters to the data. The calibrated data and the calibration parameters are stored in this table.<br><br>In this design, all post-calibration data is stored in this single table, since the "blob" data type is sufficiently general to store any type of measurement. If in the future instrument-specific opaque data types are used, then this table can be replaced by a table hierarchy, with a subtable for each different type of measurement. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_calibrated_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_calibrated_meas_id | INTEGER | Not allowed | |
| gmh_global_raw_meas_id (FK) | INTEGER | Not allowed | |
| gmh_calibrated_measurement | LONGVARBINARY | Not allowed | |
| CALIBRATION PARAMETERS | INTEGER | Not allowed | |
| MEASUREMENT METADATA | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| GENERIC RAW MEASUREMENT TABLE_gmh_calibrated_measurements_FK1 | gmh_global_raw_meas_id | GENERIC RAW MEASUREMENT TABLE.gmh_global_raw_meas_id |
| gmh_calibrated_measurements_gmh_user_annotations_FK1 | gmh_user_annotations.gmh_global_calibrated_meas_id | gmh_global_calibrated_meas_id |
| gmh_calibrated_measurements_gmh_calibrated_measurements_usage_FK1 | gmh_calibrated_measurements_usage.gmh_global_calibrated_meas_id | gmh_global_calibrated_meas_id |

Column details

1. gmh_global_calibrated_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_calibrated_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the calibrated measurement. Probably a generated key. |

2. gmh_global_raw_meas_id (FK)

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

3. gmh_calibrated_measurement

| | |
|---|---|
| Conceptual name: | gmh_calibrated_measurement |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | The calibrated measurement itself. |

4. CALIBRATION PARAMETERS

| | |
|---|---|
| Conceptual name: | CALIBRATION PARAMETERS |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | These columns will define the specific calibration parameters used in producing the calibrated measurement / image from the raw measurement / image.<br><br>This design assumes that the combinations of calibration parameter values will be quite |

dynamic (i.e., the values are quite specific to individual measurements). If instead it is the case that the calibration parameter values are quite static (i.e., a few specific combinations are used for a wide range of measurements) then it might be better to factor these columns out into a separate table and just include a foreign key to that table in this table.

5. MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the calibrated measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was produced, and the size of the measurement. Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

Foreign key details (child)

GENERIC RAW MEASUREMENT TABLE_gmh_calibrated_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_raw_meas_id | GENERIC RAW MEASUREMENT |
| | TABLE.gmh_global_raw_meas_id | |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each raw measurement or image (recorded in one of the gmh_*_measurements or the gmh_images table) can have 0 or more corresponding post-deployment calibrated measurements / images. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

| Conceptual name: | gmh_calibrated_measurements_usage |
|---|---|
| Notes: | Tracks which (calibrated) measurements were used to produce each data product, since a data product can be formed from multiple measurements, and a measurement can be used in more than one data product. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 2 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | 1. gmh_global_data_product_id<br>2. gmh_global_calibrated_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_data_product_id (FK) | INTEGER | Not allowed | |
| gmh_global_calibrated_meas_id (FK) | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_calibrated_measurements_gmh_calibrated_measurements_usage_FK1 | gmh_global_calibrated_meas_id | gmh_calibrated_measurements.gmh_global_calibrated_meas_id |
| gmh_data_products_gmh_calibrated_measurements_usage_FK1 | gmh_global_data_product_id | gmh_data_products.gmh_global_data_product_id |

**Column details**

1. gmh_global_data_product_id (FK)

| | |
|---|---|
| Conceptual name: | gmh_global_data_product_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the data product. |

2. gmh_global_calibrated_meas_id (FK)

| | |
|---|---|
| Conceptual name: | gmh_global_calibrated_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the calibrated measurement. Probably a generated key. |

**Foreign key details (child)**

gmh_calibrated_measurements_gmh_calibrated_measurements_usage_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_calibrated_meas_id | gmh_calibrated_measurements.gmh_global_calibrated_meas_id |
| Relationship type: | Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | A single calibrated measurement can be used in zero or more data products. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_data_products_gmh_calibrated_measurements_usage_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_data_product_id | gmh_data_products.gmh_global_data_product_id |
| Relationship type: | Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |

| | |
|---|---|
| Notes: | A data product is formed from one or more calibrated measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_ctd_measurements |
|---|---|
| Notes: | The BBLA will have CTD instruments (Farr-1-3). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK) | INTEGER | Not allowed | |
| gmhctd_ctd_meas | LONGVARBINARY | Not allowed | |
| gmhctd_ctd_meas_datetime | DATETIME | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_arrays_gmh_ctd_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the logical array to which the instruments generating this measurement belong. |

3. gmhctd_ctd_meas

| | |
|---|---|
| Conceptual name: | gmhctd_ctd_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

4. gmhctd_ctd_meas_datetime

| | |
|---|---|
| Conceptual name: | gmhctd_ctd_meas_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

Foreign key details (child)

gmh_arrays_gmh_ctd_measurements_FK1

| | Child | Parent |
|---|---|---|
| Definition: | gmharr_array_id | gmh_arrays.gmharr_array_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | The CTD sensors produce measurements. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

| Conceptual name: | gmh_data_product_outputtypes |
|---|---|
| Notes: | This table defines the various classes of data products that can be produced from GMH. Examples include images, Excel spreadsheets, Textual reports, etc. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 2 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmhdap_data_product_outputtype_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhdap_data_product_outputtype_id | INTEGER | Not allowed | |
| gmhdat_data_product_outputtype_desc | VARCHAR(100) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_data_product_outputtypes_gmh_standard_data_products_FK1 | gmh_standard_data_products.gmhdap_data_product_outputtype_id | gmhdap_data_product_outputtype_id |

Column details

1. gmhdap_data_product_outputtype_id

| | |
|---|---|
| Conceptual name: | gmhdap_data_product_outputtype_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID - probably autogenerated. |

2. gmhdat_data_product_outputtype_desc

| | |
|---|---|
| Conceptual name: | gmhdat_data_product_outputtype_desc |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | Description of the class of data product - e.g., "Excel spreadsheet" |

| Conceptual name: | gmh_data_product_permissions |
|---|---|
| Notes: | This table relates user roles to the data products that can be requested by users with that role. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 2 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | 1. gmhsdp_standard_data_product_id<br>2. gmhrol_role_name |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhsdp_standard_data_product_id (FK) | INTEGER | Not allowed | |
| gmhrol_role_name (FK) | VARCHAR(32) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_roles_gmh_data_product_permissions_FK1 | gmhrol_role_name | gmh_roles.gmhrol_role_name |
| gmh_standard_data_products_gmh_data_product_permissions_FK1 | gmhsdp_standard_data_product_id | gmh_standard_data_products.gmhsdp_standard_data_product_id |

**Column details**

1. gmhsdp_standard_data_product_id (FK)

| | |
|---|---|
| Conceptual name: | gmhsdp_standard_data_product_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the standard data product description - probably autogenerated |

2. gmhrol_role_name (FK)

| | |
|---|---|
| Conceptual name: | gmhrol_role_name |
| Physical data type: | VARCHAR(32) |
| Allow NULLs: | Not allowed |
| Notes: | Alphanumeric identifier of the role. |

**Foreign key details (child)**

gmh_roles_gmh_data_product_permissions_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhrol_role_name | gmh_roles.gmhrol_role_name |
| Relationship type: | Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each role has an associated list of standard data products that users with that role can produce. In order to produce a particular data product, that data product must appear in the list associated with one of the user's roles. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_standard_data_products_gmh_data_product_permissions_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhsdp_standard_data_product_id | gmh_standard_data_products.gmhsdp_standard_data_product_id |
| Relationship type: | Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each data product has an associated list of user roles. In order to produce a particular data product, a user must have one of the roles in that data product's list. | |

| | |
|---|---|
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

[redacted]

| Conceptual name: | gmh_data_products |
|---|---|
| Notes: | This table is a catalog of the various data products that have been produced. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 7 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | gmh_global_data_product_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_data_product_id | INTEGER | Not allowed | |
| gmhsdp_standard_data_product_id (FK) | INTEGER | Not allowed | |
| gmhusr_user_id (FK) | VARCHAR(32) | Not allowed | |
| gmhdat_creation_datetime | DATETIME | Not allowed | |
| gmhdat_creation_sql | LONGVARCHAR | Not allowed | |
| gmhdat_processing_desc | VARCHAR(255) | Allowed | |
| gmhdat_data_product | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_standard_data_products_gmh_data_products_FK1 | gmhsdp_standard_data_product_id | gmh_standard_data_products.gmhsdp_standard_data_product_id |
| gmh_users_gmh_data_products_FK1 | gmhusr_user_id | gmh_users.gmhusr_user_id |
| gmh_data_products_gmh_calibrated_measurements_usage_FK1 | gmh_calibrated_measurements_usage.gmh_global_data_product_id | gmh_global_data_product_id |
| gmh_data_products_gmh_user_annotations_FK1 | gmh_user_annotations.gmh_global_data_product_id | gmh_global_data_product_id |

Column details

1. gmh_global_data_product_id

| | |
|---|---|
| Conceptual name: | gmh_global_data_product_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the data product. |

2. gmhsdp_standard_data_product_id (FK)

| | |
|---|---|
| Conceptual name: | gmhsdp_standard_data_product_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |

3. gmhusr_user_id (FK)

| | |
|---|---|
| Conceptual name: | gmhusr_user_id |
| Physical data type: | VARCHAR(32) |
| Allow NULLs: | Not allowed |
| Notes: | Unique alphanumeric ID for the GMH user who requested this data product. Not necessarily the same as the PostgreSQL userid that gets used. |

4. gmhdat_creation_datetime

| | |
|---|---|
| Conceptual name: | gmhdat_creation_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | When was the data product created? |

5. gmhdat_creation_sql

| | |
|---|---|
| Conceptual name: | gmhdat_creation_sql |
| Physical data type: | LONGVARCHAR |
| Allow NULLs: | Not allowed |
| Notes: | What SQL was used to create this data product? |

6. gmhdat_processing_desc

| | |
|---|---|
| Conceptual name: | gmhdat_processing_desc |

| | |
|---|---|
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Allowed |
| Notes: | Other information relevant to the processing (e.g., name of external program used). |

7. gmhdat_data_product

| | |
|---|---|
| Conceptual name: | gmhdat_data_product |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | The data product itself. |

Foreign key details (child)

gmh_standard_data_products_gmh_data_products_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsdp_standard_data_product_id | gmh_standard_data_products.gmhsdp_standard_data_product_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each data product corresponds to a single standard data product type (e.g., "time series of N-minute averages of calibrated data at each of N ordinates") | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_users_gmh_data_products_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhusr_user_id | gmh_users.gmhusr_user_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each data product was requested / produced by one user. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

| Conceptual name: | gmh_deep_tow_measurements |
|---|---|
| Notes: | Table of "Deep Tow" measurements.<br>Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?<br>Reference: Higley-1-23c. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 1 |
| Number of foreign keys: | 2 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK,U1) | INTEGER | Not allowed | |
| gmhstl_shot_id (FK,U1) | INTEGER | Not allowed | |
| gmhdtm_deep_tow_meas | LONGVARBINARY | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_deep_tow_measurements_AK1 (U1) | gmhstl_shot_id | Ascending |
| | gmharr_array_id | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_deep_tow_measurements_FK1 | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |
| gmh_arrays_gmh_deep_tow_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK,U1)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of logical array representing the deep tow hydrophone(s). |

3. gmhstl_shot_id (FK,U1)

| | |
|---|---|
| Conceptual name: | gmhstl_shot_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of shot (pop) associated with this measurement. |

4. gmhdtm_deep_tow_meas

| | |
|---|---|
| Conceptual name: | gmhdtm_deep_tow_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | Blob of measurement data. Higley-1-17 and Higley-1-18 imply that there is one file per shot (pop).<br><br>Format yet to be determined, but likely SEG-Y if 24 bit or P16 if 16 bit (Higley 1-21c). |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |

Notes: This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

## Index details

gmh_deep_tow_measurements_AK1

Column(s): gmhstl_shot_id (Asc)
gmharr_array_id (Asc)
Unique: Yes

## Foreign key details (child)

gmh_shot_log_gmh_deep_tow_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Notes: Each deep tow measurement is for a single shot.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_arrays_gmh_deep_tow_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_array_id | gmh_arrays.gmharr_array_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Notes: The deep tow array produces measurements.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

| Conceptual name: | gmh_fluorometer_measurements |
|---|---|
| Notes: | Fluorometers will be stored on the BBLA (Farr-1-3). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK) | INTEGER | Not allowed | |
| gmhflm_fluorometer_meas | LONGVARBINARY | Not allowed | |
| gmhflm_fluorometer_meas_datetime | DATETIME | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_arrays_gmh_fluorometer_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the logical array to which the instruments generating this measurement belong. |

3. gmhflm_fluorometer_meas

| | |
|---|---|
| Conceptual name: | gmhflm_fluorometer_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

4. gmhflm_fluorometer_meas_datetime

| | |
|---|---|
| Conceptual name: | gmhflm_fluorometer_meas_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

## Foreign key details (child)

gmh_arrays_gmh_fluorometer_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_array_id | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each fluorometer produces measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_hydrophone_measurements |
|---|---|
| Notes: | Table of hydrophone measurements.<br>Reference: Higley-1-23a.<br><br>Sampling regime is 16 channels for 1-4 seconds (assume 4) at 10 kHz of 16 bit data continuously for 48-96 hours. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 1 |
| Number of foreign keys: | 2 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK,U1) | INTEGER | Not allowed | |
| gmhstl_shot_id (FK,U1) | INTEGER | Not allowed | |
| gmhhpm_hydrophone_meas | LONGVARBINARY | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_hydrophone_measurements_AK1 (U1) | gmhstl_shot_id | Ascending |
| | gmharr_array_id | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_arrays_gmh_hydrophone_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |
| gmh_shot_log_gmh_hydrophone_measurements_FK1 | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK,U1)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of logical array to which the hydrophone sensors producing this measurement belong. |

3. gmhstl_shot_id (FK,U1)

| | |
|---|---|
| Conceptual name: | gmhstl_shot_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of shot (pop) associated with this measurement. |

4. gmhhpm_hydrophone_meas

| | |
|---|---|
| Conceptual name: | gmhhpm_hydrophone_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | Blob of measurement data. Higley-1-17 and Higley-1-18 imply that there is one file per shot (pop). |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the |

stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

**Index details**

gmh_hydrophone_measurements_AK1

| | |
|---|---|
| Column(s): | gmhstl_shot_id (Asc)<br>gmharr_array_id (Asc) |
| Unique: | Yes |

**Foreign key details (child)**

gmh_arrays_gmh_hydrophone_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_array_id | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each logical array of hydrophones produces measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_shot_log_gmh_hydrophone_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each hydrophone measurement is for a single shot. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_images |
|---|---|
| Notes: | The chimney array will have 3 cameras (Farr-1-4). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK) | INTEGER | Not allowed | |
| gmhimg_image | LONGVARBINARY | Not allowed | |
| gmhimg_image_datetime | DATETIME | Not allowed | |
| OTHER IMAGE METADATA | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_arrays_gmh_images_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the image. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the logical array to which the instruments generating this measurement belong. |

3. gmhimg_image

| | |
|---|---|
| Conceptual name: | gmhimg_image |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | The image or video itself. |

4. gmhimg_image_datetime

| | |
|---|---|
| Conceptual name: | gmhimg_image_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the image/video was taken. |

5. OTHER IMAGE METADATA

| | |
|---|---|
| Conceptual name: | OTHER IMAGE METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the image. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored image, the time when it was taken, the size of the image, and any other parameters captured by the camera (other than the image itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all iamges that ..." and "What is this image?"). |

Foreign key details (child)

gmh_arrays_gmh_images_FK1

| Definition: | Child | Parent |
|---|---|---|

| | | |
|---|---|---|
| | gmharr_array_id | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each camera produces images. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_mass_spec_measurements |
|---|---|
| Notes: | The chimney array may have a mass spectrometer (Farr-1-15). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK) | INTEGER | Not allowed | |
| gmhmsm_mass_spec_meas | LONGVARBINARY | Not allowed | |
| gmhmsm_mass_spec_meas_datetime | DATETIME | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_arrays_gmh_mass_spec_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the logical array to which the instruments generating this measurement belong. |

3. gmhmsm_mass_spec_meas

| | |
|---|---|
| Conceptual name: | gmhmsm_mass_spec_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

4. gmhmsm_mass_spec_meas_datetime

| | |
|---|---|
| Conceptual name: | gmhmsm_mass_spec_meas_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

Foreign key details (child)

gmh_arrays_gmh_mass_spec_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_array_id | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | The mass spectrometer produces measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_mets_measurements |
|---|---|
| Notes: | The chimney array and the BBLA will both have methane sensors (Farr-1-3 and Farr-1-4). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK) | INTEGER | Not allowed | |
| gmhmtm_mets_meas | LONGVARBINARY | Not allowed | |
| gmhmtm_mets_meas_datetime | DATETIME | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_arrays_gmh_mets_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the logical array to which the instruments generating this measurement belong. |

3. gmhmtm_mets_meas

| | |
|---|---|
| Conceptual name: | gmhmtm_mets_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

4. gmhmtm_mets_meas_datetime

| | |
|---|---|
| Conceptual name: | gmhmtm_mets_meas_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

Foreign key details (child)

gmh_arrays_gmh_mets_measurements_FK1

| | Child | Parent |
|---|---|---|
| Definition: | gmharr_array_id | gmh_arrays.gmharr_array_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each methane sensor (METS) produces measurements. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

| Conceptual name: | gmh_orientation_sensor_measurements |
|---|---|
| Notes: | Table of orientation sensor measurements.<br>Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?<br>Reference: Higley-1-23d<br><br>There are 2 sets of 2 orientation sensors measuring heading, pitch, and roll.<br><br>Sampling regime is 4 Hz for 1-4 seconds.<br><br>Higley-1-23e says the orientation sensor works as a current meter and may be activated once per hour throughout the year. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 6 |
| Number of indexes: | 1 |
| Number of foreign keys: | 2 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK,U1) | INTEGER | Not allowed | |
| gmhstl_shot_id (FK,U1) | INTEGER | Allowed | |
| gmhosm_orientation_sensor_meas_datetime | DATETIME | Allowed | |
| gmhosm_orientation_sensor_meas | LONGVARBINARY | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_orientation_sensor_measurements_AK1 (U1) | gmhstl_shot_id | Ascending |
| | gmharr_array_id | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_orientation_sensor_measurements_FK1 | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |
| gmh_arrays_gmh_orientation_sensor_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK,U1)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of logical array to which the orientation sensors producing this measurement belong. |

3. gmhstl_shot_id (FK,U1)

| | |
|---|---|
| Conceptual name: | gmhstl_shot_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Allowed |
| Notes: | ID of shot (pop) associated with this measurement. Some measurements will not be associated with a pop (Higley-1-23e). |

4. gmhosm_orientation_sensor_meas_datetime

| | |
|---|---|
| Conceptual name: | gmhosm_orientation_sensor_meas_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Allowed |

| | |
|---|---|
| Notes: | The date and time of the measurement. Higley-1-23e says that the orientation sensor works as a current meter and may be activated once per hour throughout the year. So some measurements will not be tied to a shot number and will need the measurement date and time stored explicitly. |

5. gmhosm_orientation_sensor_meas

| | |
|---|---|
| Conceptual name: | gmhosm_orientation_sensor_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | Blob of measurement data. Higley-1-17 and Higley-1-18 imply that there is one file per shot (pop). |

6. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

## Index details

gmh_orientation_sensor_measurements_AK1

| | |
|---|---|
| Column(s): | gmhstl_shot_id (Asc)<br>gmharr_array_id (Asc) |
| Unique: | Yes |

## Foreign key details (child)

gmh_sensors_gmh_orientation_sensor_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | --Empty-- | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_shot_log_gmh_orientation_sensor_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each orientation sensor measurement may be for a single shot (but may not correspond to a shot at all). |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_arrays_gmh_orientation_sensor_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_array_id | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each logical array of orientation sensors produces measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_oxygen_sensor_measurements |
|---|---|
| Notes: | The BBLA will have oxygen sensors (Farr-1-3). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK) | INTEGER | Not allowed | |
| gmho2m_oxygen_sensor_meas | LONGVARBINARY | Not allowed | |
| gmho2m_oxygen_sensor_meas_datetime | DATETIME | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_arrays_gmh_oxygen_sensor_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the logical array to which the instruments generating this measurement belong. |

3. gmho2m_oxygen_sensor_meas

| | |
|---|---|
| Conceptual name: | gmho2m_oxygen_sensor_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

4. gmho2m_oxygen_sensor_meas_datetime

| | |
|---|---|
| Conceptual name: | gmho2m_oxygen_sensor_meas_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

## Foreign key details (child)

gmh_arrays_gmh_oxygen_sensor_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_array_id | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each oxygen sensor produces measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_roles |
|---|---|
| Notes: | This table describes the various database roles that GMH users can have. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 3 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmhrol_role_name |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhrol_role_name | VARCHAR(32) | Not allowed | |
| gmhrol_role_description | VARCHAR(255) | Not allowed | |
| gmhrol_database_role_id | VARCHAR(64) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_roles_gmh_user_roles_FK1 | gmh_user_roles.gmhrol_role_name | gmhrol_role_name |
| gmh_roles_gmh_data_product_permissions_FK1 | gmh_data_product_permissions.gmhrol_role_name | gmhrol_role_name |

Column details

1. gmhrol_role_name

| | |
|---|---|
| Conceptual name: | gmhrol_role_name |
| Physical data type: | VARCHAR(32) |
| Allow NULLs: | Not allowed |
| Notes: | Alphanumeric identifier of the role. |

2. gmhrol_role_description

| | |
|---|---|
| Conceptual name: | gmhrol_role_description |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Not allowed |
| Notes: | Description of the role - what basic access rights does it entail? |

3. gmhrol_database_role_id

| | |
|---|---|
| Conceptual name: | gmhrol_database_role_id |
| Physical data type: | VARCHAR(64) |
| Allow NULLs: | Not allowed |
| Notes: | What is the corresponding database-level role? (i.e., the role according to the underlying PostgreSQL database). |

| Conceptual name: | gmh_sensor_calibration_history |
|---|---|
| Notes: | This table contains information on how a particular sensor has been calibrated, either at present (gmhscl_hist_end_datetime is NULL) or in the past (gmhscl_hist_end_datetime is not NULL). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmhscl_sensor_calibration_hist_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhscl_sensor_calibration_hist_id | INTEGER | Not allowed | |
| gmhsen_sensor_id (FK) | INTEGER | Not allowed | |
| gmhscl_hist_start_datetime | DATETIME | Not allowed | |
| gmhscl_hist_end_datetime | DATETIME | Allowed | |
| CALIBRATION PARAMETERS | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_sensor_calibration_history_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmhscl_sensor_calibration_hist_id

| | |
|---|---|
| Conceptual name: | gmhscl_sensor_calibration_hist_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the sensor calibration history record. Probably a generated key. |

2. gmhsen_sensor_id (FK)

| | |
|---|---|
| Conceptual name: | gmssen_sensor_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the individual sensor. |

3. gmhscl_hist_start_datetime

| | |
|---|---|
| Conceptual name: | gmhscl_hist_start_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when this record became effective. |

4. gmhscl_hist_end_datetime

| | |
|---|---|
| Conceptual name: | gmhscl_hist_end_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Allowed |
| Notes: | Date and time when this record ceased to be effective. |

5. CALIBRATION PARAMETERS

| | |
|---|---|
| Conceptual name: | CALIBRATION PARAMETERS |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This will be replaced by a set of columns, one column for each calibration parameter, holding the value of that parameter.<br><br>Note that each type of sensor will have different calibration parameters, so either:<br><br>1. The parameters will be stored in subtables, with one subtable for each type of sensor, or<br><br>2. The parameters will be stored in a single table, with the parameters that are not applicable to a particular sensor simply being left blank.<br><br>Option 1 is cleaner, but harder to maintain. It is much easier with Option 2 to add parameters |

for new types of sensors, or add additional parameters for existing types of sensors.

Foreign key details (child)

gmh_sensors_gmh_sensor_calibration_history_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each sensor has a calibration history. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

| Conceptual name: | gmh_sensor_configuration_history |
|---|---|
| Notes: | This table contains information on how a particular sensor has been configured, either at present (gmhscn_hist_end_datetime is NULL) or in the past (gmhscn_hist_end_datetime is not NULL). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmhscn_sensor_config_hist_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhscn_sensor_config_hist_id | INTEGER | Not allowed | |
| gmhsen_sensor_id (FK) | INTEGER | Not allowed | |
| gmhscn_hist_start_datetime | DATETIME | Not allowed | |
| gmhscn_hist_end_datetime | DATETIME | Allowed | |
| CONFIG PARAMETERS | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_sensor_configuration_history_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmhscn_sensor_config_hist_id

| | |
|---|---|
| Conceptual name: | gmhscn_sensor_config_hist_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the sensor configuration history record. Probably a generated key. |

2. gmhsen_sensor_id (FK)

| | |
|---|---|
| Conceptual name: | gmhsen_sensor_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the individual sensor. |

3. gmhscn_hist_start_datetime

| | |
|---|---|
| Conceptual name: | gmhscn_hist_start_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when this record became effective. |

4. gmhscn_hist_end_datetime

| | |
|---|---|
| Conceptual name: | gmhscn_hist_end_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Allowed |
| Notes: | Date and time when this record ceased to be effective. |

5. CONFIG PARAMETERS

| | |
|---|---|
| Conceptual name: | CONFIG PARAMETERS |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This will be replaced by a set of columns, one column for each configuration parameter, holding the value of that parameter.<br><br>Note that each type of sensor will have different configuration parameters, so either:<br><br>1. The parameters will be stored in subtables, with one subtable for each type of sensor, or<br><br>2. The parameters will be stored in a single table, with the parameters that are not applicable to a particular sensor simply being left blank.<br><br>Option 1 is cleaner, but harder to maintain. It is much easier with Option 2 to add parameters |

for new types of sensors, or add additional parameters for existing types of sensors.

Foreign key details (child)

gmh_sensors_gmh_sensor_configuration_history_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each sensor has a configuration history. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

| Conceptual name: | gmh_sensor_types |
|---|---|
| Notes: | This table contains all the relevant information related to specific sensor types, e.g., name, type of sensor (hydrophone, ADCP, thermistor, etc), generic properties, etc. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 4 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmhstp_sensor_type_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhstp_sensor_type_id | INTEGER | Not allowed | |
| gmhstp_sensor_type_code | CHAR(10) | Allowed | |
| gmhstp_sensor_type_desc | VARCHAR(255) | Allowed | |
| SENSOR TYPE INFO | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensor_types_gmh_sensors_FK1 | gmh_sensors.gmhstp_sensor_type_id | gmhstp_sensor_type_id |

Column details

1. gmhstp_sensor_type_id

| | |
|---|---|
| Conceptual name: | gmhstp_sensor_type_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the sensor type (class). Probably a generated key. |

2. gmhstp_sensor_type_code

| | |
|---|---|
| Conceptual name: | gmhstp_sensor_type_code |
| Physical data type: | CHAR(10) |
| Allow NULLs: | Allowed |
| Notes: | A more mnemonic code identifying the type of sensor. |

3. gmhstp_sensor_type_desc

| | |
|---|---|
| Conceptual name: | gmhstp_sensor_type_desc |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Allowed |
| Notes: | A one line description of the type of sensor - e.g. make, model, manufacturer. |

4. SENSOR TYPE INFO

| | |
|---|---|
| Conceptual name: | SENSOR TYPE INFO |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of one or more columns holding any other parameters needed to describe this type of sensor. |

| Conceptual name: | gmh_sensors |
|---|---|
| Notes: | In this context a sensor is either an instrument or an individual sensor on a multi-sensor instrument.<br><br>This table contains all the relevant information related to a specific sensor, e.g., what logical array is it part of? what type of sensor is it? where is it? etc. Note that sensor measurement values are stored in other tables. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 6 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | gmhsen_sensor_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhsen_sensor_id | INTEGER | Not allowed | |
| gmharr_array_id (FK) | INTEGER | Not allowed | |
| gmhstp_sensor_type_id (FK) | INTEGER | Not allowed | |
| gmhsen_position_on_array | FLOAT | Allowed | |
| gmhsen_absolute_position | VARBINARY(10) | Allowed | |
| STATIC SENSOR INFO | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensor_types_gmh_sensors_FK1 | gmhstp_sensor_type_id | gmh_sensor_types.gmhstp_sensor_type_id |
| gmh_arrays_gmh_sensors_FK2 | gmharr_array_id | gmh_arrays.gmharr_array_id |
| gmh_sensors_gmh_sensor_configuration_history_FK1 | gmh_sensor_configuration_history.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_sensor_calibration_history_FK1 | gmh_sensor_calibration_history.gmhsen_sensor_id | gmhsen_sensor_id |

Column details

1. gmhsen_sensor_id

| | |
|---|---|
| Conceptual name: | gmhsen_sensor_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the individual sensor. Probably a generated key. |

2. gmharr_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | gmh_sensors has gmharr_array_id. |

3. gmhstp_sensor_type_id (FK)

| | |
|---|---|
| Conceptual name: | gmhstp_sensor_type_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID identifying the class of sensor to which this sensor belongs. |

4. gmhsen_position_on_array

| | |
|---|---|
| Conceptual name: | gmhsen_position_on_array |
| Physical data type: | FLOAT |
| Allow NULLs: | Allowed |
| Notes: | Measuring from a designated end of the array, where is this sensor located? |

5. gmhsen_absolute_position

| | |
|---|---|
| Conceptual name: | gmhsen_absolute_position |
| Physical data type: | VARBINARY(10) |
| Allow NULLs: | Allowed |
| Notes: | What is the absolute location of this sensor? (geographic coordinates and depth). |

6. STATIC SENSOR INFO

| | |
|---|---|
| Conceptual name: | STATIC SENSOR INFO |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of one or more columns holding other relevant information about the sensor, other than time-dependent properties like configuration, calibration, and measurements, which are stored in other tables. |

Foreign key details (child)

gmh_sensor_types_gmh_sensors_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhstp_sensor_type_id | gmh_sensor_types.gmhstp_sensor_type_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each sensor has a particular sensor type, or class of sensors. Fields that describe the class as a whole are stored in the gmh_sensor_types table; fields that are specific to an individual sensor are described in the gmh_sensors table. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_arrays_gmh_sensors_FK1

| Definition: | Child | Parent |
|---|---|---|
| | --Empty-- | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each sensor is mounted on an array; an array holds zero or more sensors. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_arrays_gmh_sensors_FK2

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_array_id | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each sensor is part of a logical array (possibly by itself). |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_shot_log |
|---|---|
| Notes: | Database realization of the "shot log". Also called the "Ship Pop" file. See Higley-1-19 and Higley-1-21b. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmhstl_shot_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhstl_shot_id | INTEGER | Not allowed | |
| gmhstl_shot_datetime | DATETIME | Not allowed | |
| gmhstl_shot_delay_time | TIME | Not allowed | |
| gmhstl_shot_position | BINARY(10) | Not allowed | |
| SHOT LOG INFO | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_hydrophone_measurements_FK1 | gmh_hydrophone_measurements.gmhstl_shot_id | gmhstl_shot_id |
| gmh_shot_log_gmh_3daccel_measurements_FK1 | gmh_3daccel_measurements.gmhstl_shot_id | gmhstl_shot_id |
| gmh_shot_log_gmh_orientation_sensor_measurements_FK1 | gmh_orientation_sensor_measurements.gmhstl_shot_id | gmhstl_shot_id |
| gmh_shot_log_gmh_deep_tow_measurements_FK1 | gmh_deep_tow_measurements.gmhstl_shot_id | gmhstl_shot_id |
| gmh_shot_log_gmh_auv_measurements_FK1 | gmh_auv_measurements.gmhstl_shot_id | gmhstl_shot_id |

Column details

1. gmhstl_shot_id

| | |
|---|---|
| Conceptual name: | gmhstl_shot_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the shot. This will likely be the value assigned at the site. |

2. gmhstl_shot_datetime

| | |
|---|---|
| Conceptual name: | gmhstl_shot_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the shot was fired. See Higley-1-21b. |

3. gmhstl_shot_delay_time

| | |
|---|---|
| Conceptual name: | gmhstl_shot_delay_time |
| Physical data type: | TIME |
| Allow NULLs: | Not allowed |
| Notes: | Delay time for the shot. See Higley-1-21b. |

4. gmhstl_shot_position

| | |
|---|---|
| Conceptual name: | gmhstl_shot_position |
| Physical data type: | BINARY(10) |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the shot was fired. See Higley-1-21b. |

5. SHOT LOG INFO

| | |
|---|---|
| Conceptual name: | SHOT LOG INFO |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns each representing one of the other fields in the shot log. |

| Conceptual name: | gmh_sled_configuration_history |
|---|---|
| Notes: | This table contains information on how the sled system has been configured, either at present (gmhslh_hist_end_datetime is NULL) or in the past (gmhslh_hist_end_datetime is not NULL). The calibration and configuration history of the sensors on the sled are stored in the gmh_sensor_calibration_history and gmh_sensor_configuration_history tables, respectively. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 4 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmhslh_sled_config_hist_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhslh_sled_config_hist_id | INTEGER | Not allowed | |
| gmhslh_hist_start_datetime | DATETIME | Not allowed | |
| gmhslh_hist_end_datetime | DATETIME | Allowed | |
| CONFIG PARAMETERS | INTEGER | Not allowed | |

Column details

1. gmhslh_sled_config_hist_id

| | |
|---|---|
| Conceptual name: | gmhslh_sled_config_hist_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the sled configuration history record. Probably a generated key. |

2. gmhslh_hist_start_datetime

| | |
|---|---|
| Conceptual name: | gmhslh_hist_start_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when this record became effective. |

3. gmhslh_hist_end_datetime

| | |
|---|---|
| Conceptual name: | gmhslh_hist_end_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Allowed |
| Notes: | Date and time when this record ceased to be effective. |

4. CONFIG PARAMETERS

| | |
|---|---|
| Conceptual name: | CONFIG PARAMETERS |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | This will be replaced by a set of columns, one column for each configuration parameter, holding the value of that parameter. |

| Conceptual name: | gmh_sled_measurements |
|---|---|
| Notes: | Table of "Sled System" data measurements. Sled contains its own shear wave source; consequently no foreign key to a shot number is needed. Measures accelerometer data, orientation data, hydrophone data.<br>Reference: Higley-1-23d. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 10 |
| Number of indexes: | 1 |
| Number of foreign keys: | 3 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_orientation_array_id (FK) | INTEGER | Not allowed | |
| gmharr_accelerometer_array_id (FK) | INTEGER | Not allowed | |
| gmharr_hydrophone_array_id (FK) | INTEGER | Not allowed | |
| ACCELEROMETER DATA | INTEGER | Not allowed | |
| ORIENTATION DATA | INTEGER | Not allowed | |
| HYDROPHONE DATA | LONGVARBINARY | Not allowed | |
| gmhslm_sled_shot_position | BINARY(10) | Not allowed | |
| gmhslm_sled_shot_datetime (I1) | DATETIME | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_sled_measurements_IDX1 (I1) | gmhslm_sled_shot_datetime | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_arrays_gmh_sled_measurements_FK1 | gmharr_hydrophone_array_id | gmh_arrays.gmharr_array_id |
| gmh_arrays_gmh_sled_measurements_FK2 | gmharr_orientation_array_id | gmh_arrays.gmharr_array_id |
| gmh_arrays_gmh_sled_measurements_FK3 | gmharr_accelerometer_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_orientation_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_orientation_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of the logical array to which the sled's orientation sensors belong. |

3. gmharr_accelerometer_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_accelerometer_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of the logical array to which the sled's accelerometers belong. |

4. gmharr_hydrophone_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_hydrophone_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of the logical array to which the sled's hydrophones belong. |

5. ACCELEROMETER DATA

| | |
|---|---|
| Conceptual name: | ACCELEROMETER DATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |

Notes: Measurement from the inboard accelerometers.

6. ORIENTATION DATA
Conceptual name: ORIENTATION DATA
Physical data type: INTEGER
Allow NULLs: Not allowed
Notes: Measurement from the inboard orientation sensors.

7. HYDROPHONE DATA
Conceptual name: HYDROPHONE DATA
Physical data type: LONGVARBINARY
Allow NULLs: Not allowed
Notes: Measurement from the inboard hydrophones.

8. gmhslm_sled_shot_position
Conceptual name: gmhslm_sled_shot_position
Physical data type: BINARY(10)
Allow NULLs: Not allowed
Notes: Position of the sled (Lat,long,depth) when source fired.

9. gmhslm_sled_shot_datetime (I1)
Conceptual name: gmhslm_sled_shot_datetime
Physical data type: DATETIME
Allow NULLs: Not allowed
Notes: Date and time when source fired.

10. OTHER MEASUREMENT METADATA
Conceptual name: OTHER MEASUREMENT METADATA
Physical data type: INTEGER
Allow NULLs: Not allowed
Notes: This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

## Index details

gmh_sled_measurements_IDX1
Column(s): gmhslm_sled_shot_datetime (Asc)
Unique: No

## Foreign key details (child)

gmh_arrays_gmh_sled_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_hydrophone_array_id | gmh_arrays.gmharr_array_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Notes: The logical array of hydrophones on the sled produces measurements.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_arrays_gmh_sled_measurements_FK2

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_orientation_array_id | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | The logical array of orientation sensors on the sled produces measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_arrays_gmh_sled_measurements_FK3

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_accelerometer_array_id | gmh_arrays.gmharr_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | The logical array of 3d accelerometers on the sled produces measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_standard_data_products |
|---|---|
| Notes: | Table of standard data products. Each data product corresponds to a single standard data product type (e.g., "time series of N-minute averages of calibrated data at each of M ordinates"). To accommodate ad-hoc requiests, one "standard data product" will be "custom". |
| Owner: | |
| Target DB name: | |
| Number of columns: | 4 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmhsdp_standard_data_product_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhsdp_standard_data_product_id | INTEGER | Not allowed | |
| gmhdap_data_product_outputtype_id (FK) | INTEGER | Not allowed | |
| gmhsdp_standard_data_product_desc | VARCHAR(255) | Not allowed | |
| gmhsdp_standard_data_product_params | VARCHAR(255) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_data_product_outputtypes_gmh_standard_data_products_FK1 | gmhdap_data_product_outputtype_id | gmh_data_product_outputtypes.gmhdap_data_product_outputtype_id |
| gmh_standard_data_products_gmh_data_products_FK1 | gmh_data_products.gmhsdp_standard_data_product_id | gmhsdp_standard_data_product_id |
| gmh_standard_data_products_gmh_data_product_permissions_FK1 | gmh_data_product_permissions.gmhsdp_standard_data_product_id | gmhsdp_standard_data_product_id |

**Column details**

1. gmhsdp_standard_data_product_id

| | |
|---|---|
| Conceptual name: | gmhsdp_standard_data_product_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID - probably autogenerated |

2. gmhdap_data_product_outputtype_id (FK)

| | |
|---|---|
| Conceptual name: | gmhdap_data_product_outputtype_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of the output type for this type of standard data product (e.g., image, spreadsheet, text report, etc.) |

3. gmhsdp_standard_data_product_desc

| | |
|---|---|
| Conceptual name: | gmhsdp_standard_data_product_desc |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Not allowed |
| Notes: | Description of the standard data product. Examples:<br><br>"time series of N-minute averages of calibrated data at each of M ordinates"<br><br>"custom ad-hoc request" |

4. gmhsdp_standard_data_product_params

| | |
|---|---|
| Conceptual name: | gmhsdp_standard_data_product_params |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Not allowed |
| Notes: | Description of any parameter values that the user must specify in requesting this data product. |

**Foreign key details (child)**

gmh_data_product_outputtypes_gmh_standard_data_products_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhdap_data_product_outputtype_id | gmh_data_product_outputtypes.gmhdap_data_product_outputtype_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each standard data product has a particular output data type (image, spreadsheet, etc.) | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

| Conceptual name: | gmh_thermistor_measurements |
|---|---|
| Notes: | Table of thermistor measurements.<br>Not tied to shot number; thermistor has own RTC (Higley-1-20).<br>Records every 6 hours for a year.<br>Reference: Higley-1-23c.<br><br>Sampling regime is 8 channels for 12 seconds at 4 Hz of 24 bit data every 6 hours for a year. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INTEGER | Not allowed | |
| gmharr_array_id (FK) | INTEGER | Not allowed | |
| gmhsen_thermistor_meas_datetime | DATETIME | Not allowed | |
| gmhthm_thermistor_meas | LONGVARBINARY | Not allowed | |
| OTHER MEASUREMENT METADATA | INTEGER | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_arrays_gmh_thermistor_measurements_FK1 | gmharr_array_id | gmh_arrays.gmharr_array_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmharr_array_id (FK)

| | |
|---|---|
| Conceptual name: | gmharr_array_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | ID of logical array to which the thermistor sensors producing this measurement belong.<br><br>Higley-1-23c says that the sampling regime for thermistors is 8 channels for 12 second record of 24-bit data at 4Hz every 6 hours for a year. |

3. gmhsen_thermistor_meas_datetime

| | |
|---|---|
| Conceptual name: | gmhsen_thermistor_meas_datetime |
| Physical data type: | DATETIME |
| Allow NULLs: | Not allowed |
| Notes: | The date and time of the measurement. Higley-1-23c says that the sampling regime is 8 channels for a 12-second record at 4 Hz of 24-bit data every 6 hours for a year. Therefore these records do not necessarily correspond to shots.<br><br>IF ANY MEASUREMENTS DO CORRESPOND TO SHOTS, WE NEED TO ADD A GMHSTL_SHOT_ID COLUMN. |

4. gmhthm_thermistor_meas

| | |
|---|---|
| Conceptual name: | gmhthm_thermistor_meas |
| Physical data type: | LONGVARBINARY |
| Allow NULLs: | Not allowed |
| Notes: | Blob of measurement data. |

5. OTHER MEASUREMENT METADATA

| | |
|---|---|
| Conceptual name: | OTHER MEASUREMENT METADATA |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |

| | |
|---|---|
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

Foreign key details (child)

gmh_sensors_gmh_thermistor_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | --Empty-- | gmh_sensors.gmhsen_sensor_id |
| Relationship type: | Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_shot_log_gmh_thermistor_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | --Empty-- | gmh_shot_log.gmhstl_shot_id |
| Relationship type: | Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_arrays_gmh_thermistor_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmharr_array_id | gmh_arrays.gmharr_array_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Notes: | Each logical array of thermistors produces measurements. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

| Conceptual name: | gmh_user_annotations |
|---|---|
| Notes: | Table of annotations that GMH users have attached to raw and/or calibrated measurements and/or data products. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 6 |
| Number of indexes: | 0 |
| Number of foreign keys: | 4 |
| Primary key: | gmhuan_user_annotation_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhuan_user_annotation_id | INTEGER | Not allowed | |
| gmh_global_raw_meas_id (FK) | INTEGER | Allowed | |
| gmh_global_calibrated_meas_id (FK) | INTEGER | Allowed | |
| gmh_global_data_product_id (FK) | INTEGER | Allowed | |
| gmhusr_user_id (FK) | VARCHAR(32) | Not allowed | |
| gmhuan_user_annotation | LONGVARCHAR | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_users_gmh_user_annotations_FK1 | gmhusr_user_id | gmh_users.gmhusr_user_id |
| GENERIC RAW MEASUREMENT TABLE_gmh_user_annotations_FK1 | gmh_global_raw_meas_id | GENERIC RAW MEASUREMENT TABLE.gmh_global_raw_meas_id |
| gmh_calibrated_measurements_gmh_user_annotations_FK1 | gmh_global_calibrated_meas_id | gmh_calibrated_measurements.gmh_global_calibrated_meas_id |
| gmh_data_products_gmh_user_annotations_FK1 | gmh_global_data_product_id | gmh_data_products.gmh_global_data_product_id |

Column details

1. gmhuan_user_annotation_id

| | |
|---|---|
| Conceptual name: | gmhuan_user_annotation_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID - probably autogenerated. |

2. gmh_global_raw_meas_id (FK)

| | |
|---|---|
| Conceptual name: | gmh_global_raw_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Allowed |
| Notes: | The ID of the raw measurement associated with this annotation (if any). At least one of gmh_global_raw_meas_id, gmh_global_calibrated_meas_id, or gmh_global_data_product_id must be specified. |

3. gmh_global_calibrated_meas_id (FK)

| | |
|---|---|
| Conceptual name: | gmh_global_calibrated_meas_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Allowed |
| Notes: | The ID of the calibrated measurement associated with this annotation (if any). At least one of gmh_global_raw_meas_id, gmh_global_calibrated_meas_id, or gmh_global_data_product_id must be specified. |

4. gmh_global_data_product_id (FK)

| | |
|---|---|
| Conceptual name: | gmh_global_data_product_id |
| Physical data type: | INTEGER |
| Allow NULLs: | Allowed |
| Notes: | The ID of the data product associated with this annotation (if any). At least one of gmh_global_raw_meas_id, gmh_global_calibrated_meas_id, or gmh_global_data_product_id must be specified. |

5. gmhusr_user_id (FK)

| | |
|---|---|
| Conceptual name: | gmhusr_user_id |
| Physical data type: | VARCHAR(32) |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the GMH user making this annotation. |

6. gmhuan_user_annotation

| | |
|---|---|
| Conceptual name: | gmhuan_user_annotation |
| Physical data type: | LONGVARCHAR |
| Allow NULLs: | Not allowed |
| Notes: | The annotation text itself. |

**Foreign key details (child)**

gmh_users_gmh_user_annotations_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhusr_user_id | gmh_users.gmhusr_user_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each annotation is written by a GMH user. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

GENERIC RAW MEASUREMENT TABLE_gmh_user_annotations_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_raw_meas_id TABLE.gmh_global_raw_meas_id | GENERIC RAW MEASUREMENT |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each raw measurement may be annotated zero or more times by one or more users. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_calibrated_measurements_gmh_user_annotations_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_calibrated_meas_id | gmh_calibrated_measurements.gmh_global_calibrated_meas_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each calibrated measurement may be annotated zero or more times by one or more users. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_data_products_gmh_user_annotations_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_data_product_id | gmh_data_products.gmh_global_data_product_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |

| | |
|---|---|
| Notes: | Each data product may be annotated zero or more times by one or more users. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

[redacted]

| Conceptual name: | gmh_user_roles |
|---|---|
| Notes: | This table relates GMH users to the access role(s) they have. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 2 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | 1. gmhrol_role_name<br>2. gmhusr_user_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhrol_role_name (FK) | VARCHAR(32) | Not allowed | |
| gmhusr_user_id (FK) | VARCHAR(32) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_users_gmh_user_roles_FK1 | gmhusr_user_id | gmh_users.gmhusr_user_id |
| gmh_roles_gmh_user_roles_FK1 | gmhrol_role_name | gmh_roles.gmhrol_role_name |

Column details

1. gmhrol_role_name (FK)

| | |
|---|---|
| Conceptual name: | gmhrol_role_name |
| Physical data type: | VARCHAR(32) |
| Allow NULLs: | Not allowed |
| Notes: | Alphanumeric identifier of the role. |

2. gmhusr_user_id (FK)

| | |
|---|---|
| Conceptual name: | gmhusr_user_id |
| Physical data type: | VARCHAR(32) |
| Allow NULLs: | Not allowed |
| Notes: | Unique alphanumeric ID for the GMH user. Not necessarily the same as the PostgreSQL userid that gets used. |

Foreign key details (child)

gmh_users_gmh_user_roles_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhusr_user_id | gmh_users.gmhusr_user_id |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each user can take on one or more roles. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_roles_gmh_user_roles_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhrol_role_name | gmh_roles.gmhrol_role_name |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Notes: | Each role can be taken on by zero or more GMH users |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

| Conceptual name: | gmh_users |
|---|---|
| Notes: | This table defines the GMH users. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 4 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmhusr_user_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhusr_user_id | VARCHAR(32) | Not allowed | |
| gmhusr_user_name | VARCHAR(100) | Not allowed | |
| gmhusr_user_description | VARCHAR(255) | Allowed | |
| gmhusr_database_user_id | VARCHAR(64) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_users_gmh_user_annotations_FK1 | gmh_user_annotations.gmhusr_user_id | gmhusr_user_id |
| gmh_users_gmh_user_roles_FK1 | gmh_user_roles.gmhusr_user_id | gmhusr_user_id |
| gmh_users_gmh_data_products_FK1 | gmh_data_products.gmhusr_user_id | gmhusr_user_id |

Column details

1. gmhusr_user_id

| | |
|---|---|
| Conceptual name: | gmhusr_user_id |
| Physical data type: | VARCHAR(32) |
| Allow NULLs: | Not allowed |
| Notes: | Unique alphanumeric ID for the GMH user. Not necessarily the same as the PostgreSQL userid that gets used. |

2. gmhusr_user_name

| | |
|---|---|
| Conceptual name: | gmhusr_user_name |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | Name of the GMH user. |

3. gmhusr_user_description

| | |
|---|---|
| Conceptual name: | gmhusr_user_description |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Allowed |
| Notes: | Other descriptive information about the GMH user. E.g., Organization, office location, phone number, etc. |

4. gmhusr_database_user_id

| | |
|---|---|
| Conceptual name: | gmhusr_database_user_id |
| Physical data type: | VARCHAR(64) |
| Allow NULLs: | Not allowed |
| Notes: | The PostgreSQL userid corresponding to this GMH user. |

Software for Simulation, Matched-Field Inversion and Monitoring for the Gulf of Mexico Hydrates Seafloor Observatory
Version 1.0

**Barrodale Computing Services Ltd. (BCS)**
**Hut R, McKenzie Ave., University of Victoria**
**Victoria BC V8W 3W2 Canada**
**www.barrodale.com**

**February 14, 2006**

# Table of Contents

# Executive Summary

This report describes a software system, developed by Barrodale Computing Services Ltd. (BCS) under contract to the University of Mississippi, for simulation, matched-field inversion (MFI) and monitoring of acoustic array data. It is an updated and extended version of the previous BCS report *Software for Simulation and Matched-Field Inversion for the Gulf of Mexico Hydrates Seafloor Observatory* (Sept. 30, 2005). The extensions in the current report include a section on the newly-developed software for monitoring using matched field techniques, and a description of the modifications to previous software components to support this monitoring.

The purpose of the system, termed BCOMFI[7], is to provide a comprehensive and validated environment for investigating the application of MFI techniques to detect changes in the sub-bottom gas hydrate deposits under the sea floor in the region of the monitoring station of the Gulf of Mexico. This approach is based on the expectation that MFI analysis of acoustic array data originating from nearby sources of opportunity (passing ships) can be used to detect such changes. The basic principle is to derive geoacoustic models for the current sub-bottom regions of the station by applying MFI to data from calibration measurements, and then use these models to match with future data obtained from passing sources. The presence of a large mismatch would be taken as evidence of a change.

The BCOMFI software developed by BCS provides a suite of components for generating synthetic data simulating shots and ship noise, transforming these data into a form where they can be used for matched field techniques, and applying these techniques to analyze the data. This analysis can be done on a single data set, or, for the monitoring case, on a sequence of data sets. The methods for simulating acoustic array data from shots or ship noise were developed first, and were then used to support the development and validation of matched field methods under controlled conditions. The result was that these methods are now in place and ready to be applied when real data from the array become available (now expected in Summer 2006).

The software was developed in IDL[8]. The reasons for this choice were that:

- IDL provides a powerful environment that allows rapid software development, with an extensive mathematical library, as well as built-in visualization and GUI tools.
- BCS has extensive experience, accumulated over several years, in writing MFI software modules in IDL.
- IDL provides a cost-free run-time environment, the IDL Virtual Machine (IDL VM), which allows users to run IDL executables, including the present BCOMFI software, without having to purchase an IDL license.

The software also makes use of two Fortran programs, ORCA[9] and RAM[10], as engines

[7] for **B**arrodale **CO**mputing **M**atched **F**ield **I**nversion
[8] Interactive Data Language, from Research Systems, Inc. (owned by ITT Industries - see www.rsinc.com)

for the acoustic propagation modeling computations. ORCA is a normal-mode code for range-independent environments, and RAM is a parabolic equation program that can be used for range-dependent environments. The use of two acoustic propagation codes provided versatility in the modeling and also furnished a means of checking and verifying the results of the simulations in range-independent environments.

The report is organized as follows:

- The background and context for the software development are outlined.
- Some properties of the environment in the region of the station are summarized, and the selected parameterization of this environment (which involves 23 geometric and geoacoustic parameters, with up to three layers) for matched field analysis is described.
- The methods used in the propagation modeling are presented.
- The installation and use of the software system are described in detail, including the files involved and the functionality of all of the components for simulation, ambiguity function generation, MFI, monitoring and display. The description for each component includes:
  - a flowchart of the computational stages involved and the associated data;
  - a Graphical User Interface (GUI) for setting up and running the component;
  - a pseudocode description of the algorithm used;
  - a guide to using the component.

  The implementation of an option in BCOMFI that supports parallel processing for matched field computations is also described.
- The internal data structures and modules in the BCOMFI software are then described in two Appendices.

The BCOMFI software components described in this report are as follows:

- ***Simulation of shot data***. This component uses the broadband option of ORCA to generate impulse responses at the array for a source at a specified location. These are convolved with a wavelet to generate simulated traces for a shot. This simulation provides synthetic shot data to test MFI procedures.
- ***Simulation of ship data***. This component uses ORCA or RAM to generate synthetic frequency domain array data at selected frequencies for one or two moving sources. It also provides for addition of noise from several noise models, and for cross-spectral matrix estimation.
- ***Conversion of shot to ship data***. This component uses phase randomization to convert shot traces to simulated ship noise time series with the same spectra as the shots.

---

[9] Westwood E.K., *An efficient broadband normal-mode model for acoustoelastic ocean environments*, J. Acoust. Soc. Am., **96**, 3352 (1994); Westwood E.K., Tindle C.T., and Chapman N.R., *A normal mode model for acoustoelastic ocean environments*, J. Acoust. Soc. Am., **100**, 3631-3645 (1996).

[10] M.D. Collins, *Generalization of the split-step Pade solution*, J. Acoust. Soc. Am. **96**, 382-385 (1994); M.D. Collins, R. J. Cederberg, D.B. King, and S.A. Chin-Bing, *Comparison of algorithms for solving parabolic wave equations*, J. Acoust. Soc. Am. **100**, 178-182 (1996).

- ***Conversion of real data to shot or ship data***. This component converts real array data in p16 format to an internal format used for storing shot or ship data.
- ***Preprocessing of shot data***. This component performs a fast Fourier transform (FFT) on the traces for the shots and saves the cross-spectral matrices for selected frequencies for use in MFI.
- ***Preprocessing of ship data***. This component overlaps, windows, and FFTs consecutive data segments, and estimates cross-spectral matrices from this data for selected frequencies for use in MFI.
- ***Ambiguity function generation***. This component generates 1D, 2D or 3D ambiguity functions using the Bartlett power processor as the matching function. This allows investigation of how the function depends on the parameters, and can assist in defining appropriate conditions (e.g., parameter subsets) for performing MFI.
- ***MFI***. This component implements a global search / local optimization approach for MFI, allowing the estimation of parameter values for a range-independent geoacoustic model consisting of up to 23 parameters.
- ***Monitoring***. This component analyzes a sequence of data sets to estimate selected geoacoustic parameters from each set by a two-stage process. In the first stage, a depth-range ambiguity function is generated using precomputed replicas. In the second stage, the maximum in this function is used as input to the search / optimization algorithm to estimate a multi-parameter geoacoustic model. This process is repeated for each data set.
- ***Visualization***. This component allows the shot and ship time series data, the ambiguity functions, and the results of MFI and monitoring to be displayed.

The BCOMFI software provides a comprehensive framework for examining the effectiveness of various MFI procedures for modeling the environment in the region of the monitoring station, and for future development of the actual software that will perform continuous monitoring of the real-time array data to be produced by the station.

# Introduction

The Gulf of Mexico Hydrates Research Consortium and the Center for Marine Resources and Environmental Technologies (CMRET) at the University of Mississippi are currently developing a multi-sensor Seafloor Observatory to be installed on the continental slope of the northern Gulf of Mexico. The aim of this station is to monitor and investigate the hydrocarbon system within the hydrate stability zone of the northern Gulf of Mexico, and to remotely observe changes in the physical and chemical parameters of gas hydrates. A key component of the monitoring is a vertical array of hydrophones which will record data that will be analyzed and interpreted using matched field techniques. The ultimate aim of this analysis is to use acoustic energy emitted by passing ships to monitor the sub-bottom layers in the region of the station, with the goal of detecting the occurrence of large-scale changes in the hydrate structures within these layers.

Barrodale Computing Services Ltd. (BCS) has been contracted by the University of Mississippi to design and develop data management and processing software for this monitoring station. In four reports written during the earlier stages of this project, BCS has characterized the data to be produced by the station[11], proposed a design for a data management and archiving system for these data[12], formulated a design for a software system for simulating the data to be acquired by the vertical acoustic array of the station and analyzing these data using matched-field inversion (MFI) techniques[13], implemented this design[14], and provided a User Guide to the initial version of the software[15]. The present report extends the original implementation report of Sept. 30 to include a section on the newly-developed software for monitoring using matched field techniques, and a description of the modifications to previous software components to support this monitoring.

The software system has been termed BCOMFI, for **B**arrodale **CO**mputing **M**atched-**F**ield **I**nversion. The simulation and conversion software component of BCOMFI provides several types of functionality. First, since the real data will consist of both time-limited shots and continuous data from nearby moving ships, the software has been implemented to simulate both shot traces and ambient noise data streams that would be observed at the sensors of an array. It also provides the ability to preprocess these data into a form where the resulting complex pressure fields can be used for MFI. In addition, to provide for greater flexibility in modeling correlated noise, and to promote efficiency, the software allows direct generation of frequency domain data (i.e., complex pressure fields) at selected frequencies, to serve as input to the MFI algorithms. These complex "measured" data can be generated in the form of either a signal vector **m** or a

[11] "Sensor and Data Characterization for the Gulf of Mexico Hydrates Monitoring Station" (Jan. 31, 2005).
[12] "Data Management Architecture Design for the Gulf of Mexico Hydrates Seafloor Observatory" (Feb. 28, 2005).
[13] "Software Design for Simulation, Matched-Field Inversion, and Monitoring for the Gulf of Mexico Hydrates Seafloor Observatory" (April 26, 2005).
[14] Software for Simulation and Matched-Field Inversion for the Gulf of Mexico Hydrates Seafloor Observatory (September 30, 2005).
[15] "BCOMFI Software User Guide" (August 12, 2005).

cross-spectral matrix $\mathbf{M} = \sum_j \mathbf{m}_j \mathbf{m}_j^*$ , where (*) denotes the conjugate transpose operation. By using appropriate acoustic propagation models, the complex fields can be computed for both range-independent environments and for constant-slope bathymetry environments. A further component of BCOMFI allows real data "P16" files, which will be produced by the hydrophone array, to be read in and converted to a format where the data can then be analyzed by MFI.

The ambiguity function generation, MFI and monitoring software components of BCOMFI allow "measured" frequency domain data to be matched with replica data vectors generated from a test model. This range-independent model includes 23 geometric and geoacoustic parameters, and can provide for up to three separate layers over a halfspace. The Bartlett power processor is used as a quantitative measure of the match between replica vectors generated using the model and the "measured" data. The software provides options for generating 1D, 2D, and 3D ambiguity functions at the points of a regular grid. This functionality allows the investigation of parameter sensitivity and correlation, and can provide an indication of the numbers of optima and other characteristics of the overall parameter space. MFI has been implemented using a global search / local optimization approach.[16] In the search stage, matches given by random choices of selected parameter values are obtained; then the best matches are chosen as starting points and for each of these, the matching function is optimized. This software allows investigation of issues such as the conditions where the model and geometry can be estimated with reasonable confidence, and could also be applied to use data obtained during calibration to generate "standard" range-independent models. The newly-developed monitoring component of BCOMFI analyzes a sequence of data sets to estimate selected geoacoustic parameters from each set by a two-stage process. In the first stage, a depth-range ambiguity function is generated using precomputed replicas. In the second stage, the maximum in this function is used as input to the search / optimization algorithm to estimate a multi-parameter geoacoustic model. This process is repeated for each data set.

An additional software component of BCOMFI provides the ability to display the results of the simulation and processing. It allows visualization of the simulated time domain data, the ambiguity functions, and the results of the MFI and monitoring processing.

The capabilities and use of the software are described in detail below. Very generally, the BCOMFI software consists of two basic types of code:

- ***Two Fortran executable modules for generating the acoustic fields***. These "computational engines" are: ORCA, a normal-mode code for range-independent environments, and RAM, a parabolic equation program that can be used for range-dependent environments. Most of the simulation computations, and all of the current ambiguity function and MFI/monitoring computations, involve using ORCA, which is generally more than an order of magnitude faster than RAM. In the simulation

[16] Zala, C. A. and Ozard, J. M. *Estimation of geoacoustic parameters from narrowband data using a search/optimization technique*, J. Comput. Acoust. **6**, 223–243 (1998).

components, RAM is used for generating data for an environment with constant slope. Note that the availability of two entirely separate propagation models provides a consistency check by allowing comparison of the fields that are produced by the two models.

- ***IDL code for the front end and the back end***. This software provides the environment for setting up GUIs, controlling the simulation and MFI, calling the computational engines, processing the computed fields, and displaying the results. The software has been developed in such a way that it can be run using the IDL Virtual Machine (IDL VM – available at no cost from RSI) and therefore does not require an IDL Development Environment. The actual code has been delivered in the form of a save file (analogous to an executable file) that can run under the IDL VM.

In this report, the geoacoustic environment in the region of the proposed site is first described and the parameterization of this environment is outlined. Then the approaches for propagation modeling are outlined: the normal mode program ORCA is used to generate the fields for a range-independent model, for both simulation and MFI, and the (slower) parabolic equation program RAM (Range-dependent Acoustic Model) is used to simulate data for simple range-dependent non-elastic environments. The design and use of the simulation and MFI components are then described: each description includes a diagram of the overall process, an image and explanation of the GUI, a statement of the algorithm, and a description of how to use the component. Our implementation of a parallel processing option for matched field computations is also described, and the use of the display components is outlined. A computing environment to support this processing is then outlined. Finally, two Appendices describe the data structures and tags used in the software and detail the modular structure of each of the components.

## Geoacoustic Environment

### Description

In order to formulate effective designs for the functionality of the software to be developed, it was necessary to obtain some information about the general geoacoustic environment in which the monitoring station is to be deployed. At a meeting of the Hydrates Research Consortium in November 2004, a desirable site for the station was identified by the attendees, in Mississippi Canyon Block 118. Early in 2005, a final choice of the location within this block was made, and several seismic surveys were performed along four different tracks. These surveys, and associated data, indicated that:

- The depth at the site was about 875 m.
- There was a bottom slope up to about 2 degrees, with depth increasing to the southeast.
- The sloping region immediately surrounding the site was reasonably planar, but there was a canyon running northwest-southeast about 1.5 km to the northeast, and a flatter plateau, which showed high acoustic reflectivity, about 2 km to the south.

- There was a strong sub-bottom reflection about 250 msec below the bottom reflection, with two or three weaker reflections above this region and occasional strong deeper reflections. It was suggested that the reflection at 250 msec could possibly represent the bottom of the hydrate stability zone.
- There was a strong sub-bottom domed feature 1.5 – 2 km to the southwest of the site.
- There were several bright spots in the seismic sections, also at about the 250 msec horizon. These were thought to correspond to pockets of gas.

These data clearly showed that this environment is significantly range-dependent, both with respect to the water depth (approximately planar with a maximum slope of about 2°) and the characteristics of the 250-msec reflection. Because of this range dependence, a key question is the extent to which this environment can be satisfactorily modeled by a range-independent propagation model such as ORCA. The initial approach taken in this project was to assume that there is some region close to the station within which the environment may be taken to be approximately range-independent. This assumption both simplifies the parameterization and reduces the time required for matched-field inversion using optimization.

Based on the above considerations, we have designed and implemented a comprehensive software system that allows generation of synthetic data for range-independent environments and analysis of synthetic and real data using matched-field inversion, assuming a range-independent environment. The implementation includes the ability to generate 1D, 2D and 3D ambiguity functions for specified parameters, and to perform parameter optimization. In addition, to allow further investigation of the effects of simple range dependence, the design also provides for the ability to generate synthetic data for a sloping-bottom range-dependent environment and to analyze these data using a range-independent model.

## Parameterization

It was essential to define a geoacoustic and geometric parameterization that is acceptably realistic for use in range-independent modeling of the environment, yet is sufficiently simple to allow matched field inversion to be performed in a reasonable time frame. Based on the above description of the environment, a set of variables was defined that consisted of source-array geometry parameters, up to three sediment layers that could each have a gradient in their acoustic parameters, and a basement[17] halfspace with constant parameter values. The proposed 23-parameter model to be used and optimized for MFI of the environment is as follows:

- ***Water depth***. Variations in water depth due to tides, and the presence of some degree of range dependence require that water depth be a parameter in the inversion.

[17] The term “basement” is used here in a modeling, rather than geological, context.

- ***Source depth***. This will generally be close to the surface (0.5 – 5 m) for both shot data and sources of opportunity, but should be included for investigative purposes.
- ***Source range***. Matching is known to be strongly dependent on range. As noted above, it is anticipated that for reasonably close source ranges (say, 1 – 3 km) a range-independent propagation model may be used to estimate the source range.
- ***Source bearing***. For a vertical array in a range-independent environment, bearing is not resolvable (the fields are bearing-independent). However for a tilted array, bearing does have an effect and is potentially invertible.
- ***Array tilt angle***. Arrays will be tilted in practice, and it is known that this tilt has a large effect on the field matches. Optimization of this parameter (and the following parameter, i.e., tilt direction) could significantly enhance the matching. For the present, it will be assumed that the tilted array is linear.
- ***Array tilt direction***. As for previous parameter.
- ***Layer parameters***. These consist of five parameters for each of a prespecified number of layers (0 – 3: note that the number of layers is not optimizable):
  - ***Layer thickness***. It is expected that this will be a useful parameter for detecting changes at the base of the hydrate stability zone.
  - ***Compressional sound speed at layer top***. Changes in this parameter (or the following parameter, or both) could be indicative of changes that occur within the hydrate stability zone.
  - ***Compressional sound speed at layer bottom***. As for previous parameter.
  - ***Density at layer top***. Although this parameter is generally quite insensitive with respect to matching, it has been included for investigative purposes.
  - ***Density at layer bottom***. As for previous parameter.
- ***Compressional sound speed in basement halfspace***. Changes in this parameter could also reflect changes that occur at the base of the hydrate stability zone.
- ***Density in basement halfspace***. Although this parameter is generally quite insensitive with respect to matching, it has been included for investigative purposes.

An option was provided in the matched-field applications to force the top and bottom compressional sound speed and density parameters in the layers to be equal (i.e., no gradient), so as to allow investigation of the effect of matching using a constant value when there is in fact a gradient in the layer. Note that shear sound speed and compressional and shear attenuations are not included as parameters at present.

Also, the sound-speed profile in the water column was not included in the parameter set, but rather, a means has been provided of specifying this data in an input file. This file could contain a generic, seasonally appropriate, sound-speed profile to be used in the matched field applications.

The above parameterization was designed to give considerable flexibility in the modeling while still being tractable in terms of matched-field inversion.

It is expected that the most useful optimizations will involve the geometric parameters water depth, source-array range and the thicknesses and sound speeds of the first one or two layers. The densities are not expected to have a substantial effect on the matches, but are provided nonetheless as a research tool.

# Propagation Modeling

The ORCA normal mode acoustic propagation model is the “engine” used to compute the acoustic fields required for simulation and matched-field-inversion. For a given ocean environment, specified by the sound-speed profile in the water column and a geoacoustic profile of the ocean bottom, ORCA finds the normal modes and computes the acoustic field at the sensors of an array. The model includes the effects of sound-speed gradients in the water and the bottom layers, shear waves in the bottom layers, steep-angle propagation represented by leaky modes, and attenuation in the bottom layers. It may be used to predict narrowband or broadband propagation. ORCA is unique among underwater acoustic propagation codes because it is largely automatic: the user does not need to guess at any convergence parameters such as depth- or range-sampling resolutions. It is also computationally efficient, typically requiring a few tenths of a second (on a 3 GHz Windows computer) to compute a propagation model at frequencies of interest in the present application.

ORCA is written in Fortran, requires several input files, and was modified to produce an output file with the field values at the sensors. Since the simulation and MFI code are in IDL, a simple communication protocol and data transfer scheme between IDL and ORCA was defined. In this scheme, the IDL process writes out files with the information required by ORCA, and then spawns an ORCA process that reads in these files and produces an output file containing the complex field values. The IDL process then reads in this file and uses the data in its internal computations.

Despite its advantages and efficiency, ORCA is a range-independent propagation model and is not directly applicable to range-dependent environments. The environment in the array, however, is known to have significant range dependence, and it is desirable to be able to investigate the effects of this at some stage of the project by simulating range-dependent data. While it might be possible to implement an adiabatic mode approximation using ORCA to achieve this, it was deemed preferable to implement a separate range-dependent code for performing such simulations. In addition, such a code provided the opportunity to perform a validation check on the correctness of the ORCA results.

Accordingly, the RAM (Range-dependent Acoustic Model) parabolic equation model was implemented, and modified as required to integrate it into the BCOMFI system, to allow simulation of data in a simple range-dependent environment. This model is also written in Fortran and so the same general communication strategy as used for ORCA (i.e., IDL spawning a RAM process and reading in the output file) was used for RAM.

## Installing the BCOMFI Software

The BCOMFI software that runs under the IDL VM is provided in the form of the zipfile "bcomfi.zip". It sets up an appropriate directory structure and contains various data files as well as the software. To install this software, simply unzip the file into a directory of your choosing. The following directory structure and files will be produced:

- ***\Bin***. This contains the Windows version of the executable files for ORCA, RAM, and the orca server enabling parallel processing of matched field computations, i.e.,
  - orca.exe
  - rampe.exe
  - orcaserver.exe
- ***\Data***. This contains a set of files of the form xxxxnnnnn.ttt, where xxxx is a four-character file prefix, nnnnn is a five-character identification (ID) number, and ttt is either "dat" for a user-editable data file or "sav" for an IDL save file. These files are used to provide input and to store the results. (Certain of them may be overwritten during the processing, if the ID numbers are not changed during GUI setup.) See below for more detailed descriptions of these files, which correspond to IDL data structures used internally by the BCOMFI system.
- ***\Doc***. This contains User Guides for ORCA (PDF file) and RAM (PostScript file), and the present BCOMFI report.
- ***\Inputs***. This contains input files used by ORCA. In particular, the files sim_shot_svp and sim_ship_svp must be present in this directory; these files can be edited to modify the sound speed profile to be used by ORCA and/or RAM during the simulation. The other file types are generated during runs of the BCOMFI simulation system.
- ***\Linux***. This contains the Linux version of the executable files for ORCA and the orca server enabling parallel processing of matched field computations.
- ***\Outputs***. This contains files used by ORCA for output.
- ***\P16_data***. This contains sample ship and shot real data in p16 format in its subdirectories \ship and \shot.
- ***\Ramout***. This is initially empty but will contain files output by RAM during a simulation run where RAM is used.
- ***\Source***. This contains the files "bcomfi.sav", "ramgeo.in", and "servers.dat", and will contain some input files for ORCA (orca_in) and/or RAM (ramgeo.dat) that are required for, or generated during, runs.

## Installing the IDL VM and Launching BCOMFI

The IDL Virtual Machine can be obtained from the RSI web site. To download the IDL VM, go to http://www.rsinc.com/idlvm/, press the "Download IDL VM" option, enter the required registration information, check the "Windows" option, and follow the instructions. When the download is complete, run the install script. This will install the IDL VM on your computer and create an icon on the desktop.

The IDL VM allows you to locate an IDL save (.sav) file – in this case, bcomfi.sav – and

run it. Pressing the IDL VM icon will first display a splash screen; you must click on this screen to proceed. When you have done so, a new file browser screen will appear. Navigate to the directory containing the bcomfi.sav file and select this file by double-clicking or selecting it into the “File name:” field. The file browser will then disappear and the BCOMFI program will be launched.

For convenience, it is a good idea to edit the “Properties” field of the IDL VM screen icon, select the “Shortcut” tab, and set the “Start in:” field to the directory containing the bcomfi.sav file. This will allow fast launching of the BCOMFI system.

The BCOMFI system consists of an initial GUI (the “Main Menu”) that allows selection of a number of processing tasks. Pressing a button to select a desired simulation, conversion, preprocessing, or ambiguity function/MFI task brings up another GUI containing a number of fields or conditions that you can specify for the run. The simulation / conversion / preprocessing GUIs have been set up so that they can be run using default values, and each run will result in a data file being produced. The ambiguity function and search/optimization GUIs require user specification of parameters before a run can be launched. The contents of certain of these data files can be visualized or viewed using the appropriate button in the initial GUI, as described later in this report.

## Setting up for Parallel Processing

To allow parallel processing of multi-frequency matched field computations on a network of Windows or Linux computers, the following additional installation steps are needed.

- First, after installing the IDL VM on the master computer, copy the files “idl_tools_nodelay.dll” and “idl_tools_nodelay.dlm” from the \bin directory created above to the C:\rsi\idlXX\bin\bin.x86 directory on the client computer, where “XX” is the corresponding IDL version number (e.g., idl62).

- Then, for each (remote) Windows computer in the network:

    - Create a directory C:\Program Files\Orca.
    - Copy the two executables “orcaserver.exe” and “orca.exe” and the batch file “orca.bat” from the \bin directory created above into the newly created directory on the remote computer.

- Then, for each (remote) Linux computer in the network:
    - Copy the executables “orcaserver” and “orca90” from the \linux directory created above to /usr/local/bin on the remote computer. You will need root permission to do this.
    - Copy the tar file “fortranlib.tar.gz” from the \linux directory created above to /usr/local/lib on the remote computer. You will need root permission to do this.

- cd to /usr/local/lib on the remote computer, and run the two commands “gunzip fortranlib.tar.gz” and “tar xf fortranlib.tar”.
- Make sure that the directory /usr/local/lib is in included in the value of the LD_LIBRARY_PATH environment variable for the session running orcaserver on the remote machine. To set LD_LIBRARY_PATH in the C shell (csh), type “setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/usr/local/lib”. To set this environment variable in the bash shell, type
  “ export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/usr/local/lib”.
  To check the value of LD_LIBRARY_PATH, type
  “echo $LD_LIBRARY_PATH”. You may wish to include the setenv or export command in your account's .cshrc (for C shell) or .bashrc (for bash) file so that it does not need to be entered each time orcaserver is run.

- To actually run the server on either Windows or Linux remote computers, cd to the directory containing the orcaserver executable on the remote computer and enter “orcaserver *N*”, where *N* is the remote port number to be used (e.g., “orcaserver 1501”). These orcaserver processes must be running on the remote servers for parallel processing to be enabled. Note that the port numbers can be arbitrarily chosen but must correspond to those selected in the matched field interfaces described in later sections. Also note that the same port numbers can be used with the different computers, providing the numbers in the orcaserver commands on the remote computers agree with the entries in the interfaces.

## Overview of BCOMFI Functionality

### Processing Components

The BCOMFI system consists of 16 separate components for simulation, data conversion, preprocessing, ambiguity function generation, MFI, and display. These components, which correspond to the initial GUI, are listed and briefly described in this section, while the following section gives some representative examples of processing tasks that may be accomplished using these components in a sequence. In the following descriptions and in later sections, “TD” denotes time domain, while “FD” denotes frequency domain.

***Simulate TD shot data***. This component uses ORCA to generate the impulse responses that would be observed at the sensors of an array for the specified geoacoustic environment and source-array geometry. The resulting traces are written to a Shot file, which can then be viewed or processed into a Ship file or an Hpdt file.

***Simulate ship FD hpdt data***. This component uses ORCA or RAM to generate frequency domain data at selected frequencies that would be observed at the sensors of an array for the specified geoacoustic environment and source-array geometry (with one or two moving sources). Noise of various types can be added to this data, and cross-spectral matrices can be generated from sequential segments. The resulting complex data are written to an Hpdt file, which can be used for input to ambiguity function generation and MFI.

***Convert TD shot data to TD ship data***. This component takes the impulse responses at the sensors in a Shot file and applies a separate phase randomization at each frequency (but not between sensors) in order to generate simulated time domain ship data with the same spectra as the traces in the Shot file. The resulting traces are written to a Ship file, which can then be viewed or processed into an Hpdt file.

***Convert p16 data to shot or ship data***. This component reads in real data files in p16 format (a binary integer format in which the data from the actual hydrophone arrays are collected and stored) and converts the data in each file to Shot or Ship format, as specified by the user. These can then be processed into Hpdt files.

***Preprocess TD shot data to FD hpdt data***. This component reads in a Shot file, FFTs the impulse responses for the sensors, forms "measured" cross-spectral matrices within a specified band for each of a number of selected frequencies, and writes the result to an Hpdt file.

***Preprocess TD ship data to FD hpdt data***. This component reads in a Ship file, FFTs the trace data according to the specified segmentation, accumulates "measured" cross-spectral matrices within a specified band for a number of selected frequencies, and writes the results to an Hpdt file.

***Generate 1D ambiguity function***. This component inputs an Hpdt file and generates an ambiguity function (matching function) as a single parameter of the 23-parameter model is varied between specified limits, with fixed values for the other parameters. The result is written to an Am1d file.

***Generate 2D ambiguity function***. This component inputs an Hpdt file and generates an ambiguity function (matching function) as two parameters of the 23-parameter model are varied between specified limits, with fixed values for the other parameters. The result is written to an Am2d file. This component also provides a means of generating precomputed depth-range replica vectors (in the form of a Repl file) to be used in the monitoring.

***Generate 3D ambiguity function***. This component inputs an Hpdt file and generates an ambiguity function (matching function) as three parameters of the 23-parameter model are varied between specified limits, with fixed values for the other parameters. The result is written to an Am3d file.

***Perform MFI search/optimization***. This component inputs an Hpdt file and performs matched field inversion of selected parameters of the 23-parameter model using a global search / local optimization technique with multiple starting points optimized until convergence. The estimates are grouped to reduce nonuniqueness, and 1D ambiguity functions of the specified parameters passing through the best estimate are computed to characterize the parameter space. The results are written to an Mfop file. This component also provides a means of generating a Sopt (search/optimization) file for

input by the monitoring component; this file contains the selected conditions to be used in the search/optimization stage of the monitoring algorithm.

***Perform MFI monitoring***. This component inputs a sequence of Hpdt files and analyzes each one to estimate selected geoacoustic parameters from each set by a two-stage process. In the first stage, a depth-range ambiguity function is generated using precomputed replicas. In the second stage, the maximum in this function is used as input to the search / optimization algorithm to estimate a multi-parameter geoacoustic model. This process is repeated for each data set. . The results are written to a Moni file.

***Display shot data***. This component inputs a Shot file and displays the traces for the sensors in one waterfall plot and their corresponding spectra in another.

***Display ship data***. This component inputs a Ship file and displays the traces for the sensors in one waterfall plot and their corresponding spectra in another.

***Display 1D ambiguity function***. This component reads in an Am1d file and displays the 1D ambiguity function, using IDL's iplot component of its itools suite of interactive visualization utilities.

***Display 2D ambiguity function***. This component reads in an Am2d file and displays the 2D ambiguity function, using IDL's isurface or iimage components of its itools suite of interactive visualization utilities.

***Display 3D ambiguity function***. This component reads in an Am3d file and displays the 3D ambiguity function, using IDL's ivolume component of its itools suite of interactive visualization utilities or its slicer3 volume visualization tool.

***Display MFI search/optimization results***. This component reads in a Mfop file and allows visualization of:

- the matching functions computed during the search stage;
- the evolution of the objective function and parameters during any of the multiple convergences;
- the 1D ambiguity functions at the best optimum found.

It also displays text information describing the groups of optima found and the characteristics of the parameter space at the best optimum.

***Display MFI monitoring results***. This component reads in a Moni file and allows visualization of the time evolution of the mismatch function and parameter estimates for each of the data sets in the monitoring sequence.

## Examples of Processing using BCOMFI Components

Some examples of how the above processing components of BCOMFI can be used to perform some representative tasks are given in this section. Note that, for brevity, it is

assumed that the files required for the runs (e.g., for ORCA and RAM) have been set up. (See later sections for details on which files are involved in the runs and how they should be set up.)

***Determine the impulse response for specified conditions***

- Run the *Simulate TD shot data* component to generate a Shot file.
- View this file using the *Display shot data* component.

***Check the correspondence of ORCA and RAM***

- Run the *Simulate ship FD hpdt data* component, selecting the ORCA option, to generate Hpdt file A.
- Run the *Simulate ship FD hpdt data* component, selecting the RAM option, to generate Hpdt file B.
- Run the *Generate 1D ambiguity function* component for selected parameters, using Hpdt file A, and producing Am1d file C. Also note the numerical values in the text log window of this component.
- Run the *Generate 1D ambiguity function* component for the same selected parameters, using Hpdt file B, and producing Am1d file D. Also note the numerical values in the text log window of this component.
- Compare the text values from the two runs. Also use the *Display 1D ambiguity function* component (twice) to view plots of the data in files C and D. Verify the closeness of the results.

***Check that simulated shot and ship data produce the same matching as direct generation of hpdt data***

- Run the *Simulate TD shot data* component to generate a Shot file.
- Run the *Convert TD shot data to TD ship data* component to input the Shot file and output a corresponding Ship file.
- Run the *Preprocess TD shot data to FD hpdt data* component on the Shot file to produce Hpdt file A.
- Run the *Preprocess TD ship data to FD hpdt data* component on the Ship file to produce Hpdt file B.
- Run the *Simulate ship FD hpdt data* component under equivalent conditions to produce Hpdt file C.
- Run the *Generate 1D ambiguity function* component for selected parameters, using Hpdt files A, B and C, and producing Am1d files D, E, and F. Compare the numerical values in the log window as above, and use the *Display 1D ambiguity function* component to view plots of the data in files D, E and F. Verify the closeness of the results.

***Perform MFI on p16 data obtained for several shots***

- Run the *Convert p16 data to shot or ship data* component, selecting the shot option to generate Shot files from the selected input files.
- For each Shot file, run the *Preprocess TD shot data to FD hpdt data* component to generate an Hpdt file.

- For each Hpdt file, run the *Perform MFI search/optimization* component to generate an Mfop file (this is a time-consuming process).
- View the results of the inversions using the *Display MFI search/optimization results* component.

## Data Files

### BCOMFI Data Files

The files in the "\data" directory have names of the form "xxxxnnnnn.ttt" and contain input and output data that correspond to internal data structures used by the programs. Files of type ".dat" are user-editable while those of type ".sav" are non-editable IDL save files. A brief description of the data files and their use is given here.

***Am1dnnnnn.sav***. This file is produced by the "Generate 1D ambiguity function" process. It contains the conditions used for generating the 1D ambiguity function (i.e., the values for fixed parameters, and the values for the parameter varied) as well as the results of the run (i.e., the matches for each value of the varied parameter, computed using the Bartlett processor). The ambiguity function can then be visualized by pressing the "Display 1D ambiguity function" button in the Main Menu interface.

***Am2dnnnnn.sav***. This file is produced by the "Generate 2D ambiguity function" process. It contains the conditions used for generating the 2D ambiguity function (i.e., the values for fixed parameters, and the values for the two parameters varied) as well as the results of the run (i.e., the matches for each pair of varied parameters, computed using the Bartlett processor). The 2D ambiguity function can then be visualized by pressing the "Display 2D ambiguity function" button in the Main Menu interface.

***Am3dnnnnn.sav***. This file is produced by the "Generate 3D ambiguity function" process. It contains the conditions used for generating the 3D ambiguity function (i.e., the values for fixed parameters, and the values for the three parameters varied) as well as the results of the run (i.e., the matches for each triplet of varied parameters, computed using the Bartlett processor). The 3D ambiguity function can then be visualized by pressing the "Display 3D ambiguity function" button in the Main Menu interface.

***Araynnnnn.dat***. This input file contains the specification for the array to be used in the simulations. Note that the arrays are used by the simulation, ambiguity function and MFI options, but not by the conversion, preprocessing, and display options. These text files are user-editable – see the example in \data\aray00000.dat.

***Grpsnnnnn.dat***. This file is produced by the "Display MFI search/optimization result" process, and is derived from data in an Mfop file. It contains the run conditions and a summary of the results of the grouping and peak analysis stages of a run of the MFI search/optimization. It contains the main results of the MFI run, in the form of a list of the various grouped convergences (optima) that were found, as well as the statistical properties for each group, when the group contained two or more convergences to the

same optimum.

***Hpdtnnnnn.sav***. This file is produced by the "Simulate ship FD hpdt data", "Preprocess TD shot data to FD hpdt data", and "Preprocess TD ship data to FD hpdt data" processes. It contains the conditions used for simulation and the complex field vectors or cross-spectral matrices for the array at user-selected frequencies. It can also contain a sequence of such data sets, at each of a number of times as specified by the user. Hpdt files contain the acoustic field data that will be used for MFI.

***Mfopnnnnn.sav***. This file is produced by the "Perform MFI search/optimization" process. It contains values used in the setup of the search/optimization, information about the fixed and varied parameters, the results for the global search stage, the results of each separate local optimization, grouping of the optima, and 1D ambiguity functions through the best optimum found. This information can be viewed or visualized by pressing the "Display MFI search/optimization results" button in the Main Menu interface.

***Moninnnnn.sav***. This file is produced by the "Perform MFI monitoring" process. It contains a Moni structure, which includes the input conditions and a series of Mfop structures, each of which contains the output of a search/optimization run on a separate data set in the sequence of input data. This information can be viewed or visualized by pressing the "Display MFI monitoring results" button in the Main Menu interface.

***Replnnnnn.sav***. This file can be produced by the "Generate 2D ambiguity function" process. It contains complex replica data in the form of a 4D array, where the dimensions are sensor, frequency, depth, and range, respectively. The file is then used as precomputed replica vectors by the monitoring process.

***Shipnnnnn.sav***. This file is produced by the "Convert TD shot data to FD ship data" option and can also be used for input by the "Preprocess TD ship data to FD hpdt data" option. It contains the conditions used for simulation and the simulated time domain ship data traces (generated from shot traces). These traces can be visualized by pressing the "Display ship data" button in the Main Menu interface.

***Shotnnnnn.sav***. This file is produced by the "Simulate TD shot data" option and can also be used for input by the "Convert TD shot data to FD ship data" and "Preprocess TD shot data to FD hpdt data" processes. It contains the conditions used for simulation and the simulated time domain shot data traces. These traces can be visualized by pressing the "Display shot data" button in the Main Menu interface.

***Soptnnnnn.sav***. This file can be produced by the "Perform MFI search/optimization" process. It provides a convenient means of specifying the contents of the Params and Mfop structures that together define the conditions to be used in the search/optimization stage of monitoring. The file is input by the monitoring process.

***Wavennnnn.dat***. This input file contains the specification for the wavelet to be used in

the generation of shot traces by the “Simulate TD shot data” option. These text files are user-editable – see the example in \data\wave00000.dat.

***Wsspnnnnn.dat***. This input file contains the water-column sound speed profile to be used by the ambiguity function generation and search/optimization MFI procedures. The data format consists of the number of points in the profile, and the (depth, sound speed) pairs defining the profile, one pair per line. These text files are user-editable – see the example in \data\wssp00000.dat.

## ORCA Data Files

### \Source Directory

***Orca_in***. Produced automatically by all runs involving ORCA. Contains the “_opt” and “_svp” files to use for the run.
***Orca_status***. Produced automatically by all runs involving ORCA. Contains the exit status of the ORCA run.

### \Inputs Directory

***Sim_shot_svp***. Must be provided by the user prior to an ORCA run simulating TD shot data, and contain the sound velocity profile (water and bottom layers/halfspace) for the environment. An example file is provided in \inputs\sim_shot_svp.
***Sim_ship_svp***. Must be provided by the user prior to an ORCA run simulating ship FD hpdt data, and contain the sound velocity profile (water and bottom layers/halfspace) for the environment. An example file is provided in \inputs\sim_ship_svp.
***Mfi_svp***. Produced automatically by all ORCA runs involving ambiguity function generation or search/optimization.

***Sim_shot_opt***. Produced automatically by runs involving simulating TD shot data. Contains options for the ORCA run.
***Sim_ship_opt***. Produced automatically by runs involving simulating ship FD hpdt data using ORCA. Contains options for the ORCA run.
***Mfi_opt***. Produced automatically by all runs involving ambiguity function generation or search/optimization. Contains options for the ORCA run.

***Sim_shot_array_geom***. Produced automatically by runs involving simulating TD shot data. Contains the array geometry for the ORCA run.
***Sim_ship_array_geom***. Produced automatically by runs involving simulating ship FD hpdt data. Contains the array geometry for the ORCA run.
***Mfi_array_geom***. Produced automatically by runs involving ambiguity function generation or search/optimization. Contains the array geometry for the ORCA run.

**\Outputs Directory**

***Sim_field.dat***. Produced automatically by ORCA runs involving simulating ship FD hpdt data. Contains the complex-valued fields at the sensors of the array, which are then read in by the IDL code.
***Mfi_field.dat***. Produced automatically by ORCA runs involving ambiguity function generation or search/optimization. Contains the complex-valued fields at the sensors of the array, which are then read in by the IDL code.

***Sim_modes***. Produced automatically by ORCA runs involving simulating ship FD hpdt data. Contains the properties of the modes (not used).
***Mfi_modes***. Produced automatically by ORCA runs involving ambiguity function generation or search/optimization. Contains the properties of the modes (not used).

***Sim_out***. Produced automatically by ORCA runs involving simulating ship FD hpdt data. Contains the input data for the run and a summary of the number of modes and timing data (not used).
***Mfi_out***. Produced automatically by ORCA runs involving ambiguity function generation or search/optimization. Contains the input data for the run and a summary of the number of modes and timing data (not used).

***Sim_shot_fft***. Produced automatically by ORCA runs involving simulating TD shot data. Contains a Fortran binary version of the FFT file output by ORCA for the run (not used).
***Sim_shot_fft.dat***. Produced automatically by ORCA runs involving simulating TD shot data. Contains ASCII output constituting the FFT file output by ORCA for the run, which is then input by the BCOMFI code that generates the impulse response.

## RAM Data Files

**\Source Directory**

***Ramgeo.dat***. Must be provided by the user prior to a RAM run simulating ship FD hpdt data, and contain the sound velocity profile (water and bottom layers/halfspace) for the environment in the same format as that in the input file ramgeo.in. Ramgeo.dat acts as a source and template for the setting of parameters other than frequency, source range and source depth (which are set during BCOMFI simulation), and allows the automated generation of ramgeo.in files during the RAM runs. An example file is provided in \source\ramgeo.dat.

***Ramgeo.in***. Produced automatically just prior to each RAM run simulating ship FD hpdt data and used as input for the RAM run.

**\Ramout Directory**

***Field.dat***. Produced automatically by RAM runs. Contains a Fortran binary version of the complex fields computed during the RAM run (not used).
***Field.out***. Produced automatically by RAM runs. Contains an ASCII version of the complex fields computed during the RAM run, which is then used by BCOMFI to compute the fields at the sensors of the array.
***Tl.grid***. Produced automatically by RAM runs. Contains a Fortran binary version of the transmission loss grid computed during the RAM run (not used).
***Tl.line***. Produced automatically by RAM runs. Contains an ASCII version of the transmission loss along a constant-depth line computed during the RAM run (not used).

## P16 Data Files

These files, which will contain real data from the array, must have names of the form pnnnnn.p16, where nnnnn is a five-character identification (ID) number. However, they can be placed in any user-accessible directory, as they are specified interactively during a run using a special purpose "pickfile" dialog box. Example files are provided in the directories \p16_data\ship and \p16_data\shot.

### *Servers Data File*

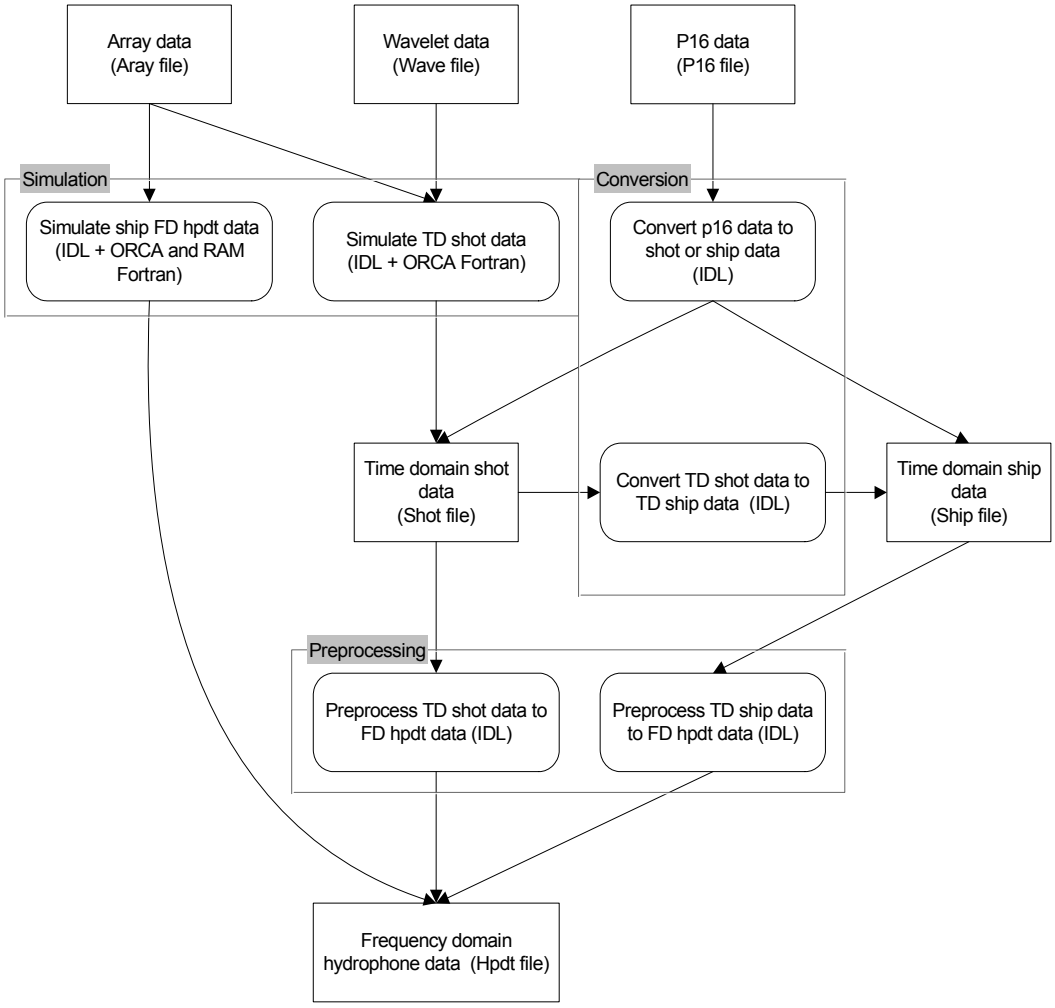
The file "servers.dat" must be present in the \source directory in order to run the matched field computations. It contains the default IP addresses and ports for each server process to be run on the remote computers, and can be edited to correspond to the distributed environment to be used. If the parallel processing option is not selected, the information in this file is simply ignored. An example of the file structure is provided in the \source directory of the software delivery.

# BCOMFI Software: Main Menu

The Main Menu provides an interface for invoking any of the functional components of the BCOMFI system listed in the Processing Components section above, i.e., the simulation, conversion, preprocessing, ambiguity function generation, MFI, monitoring and display components. Note that, in general, the result of using the simulation, conversion and preprocessing components is ultimately an Hpdt file. This Hpdt file is then used as input to the ambiguity function generation and MFI components.

## Overall Design: Simulation, Conversion and Preprocessing Components

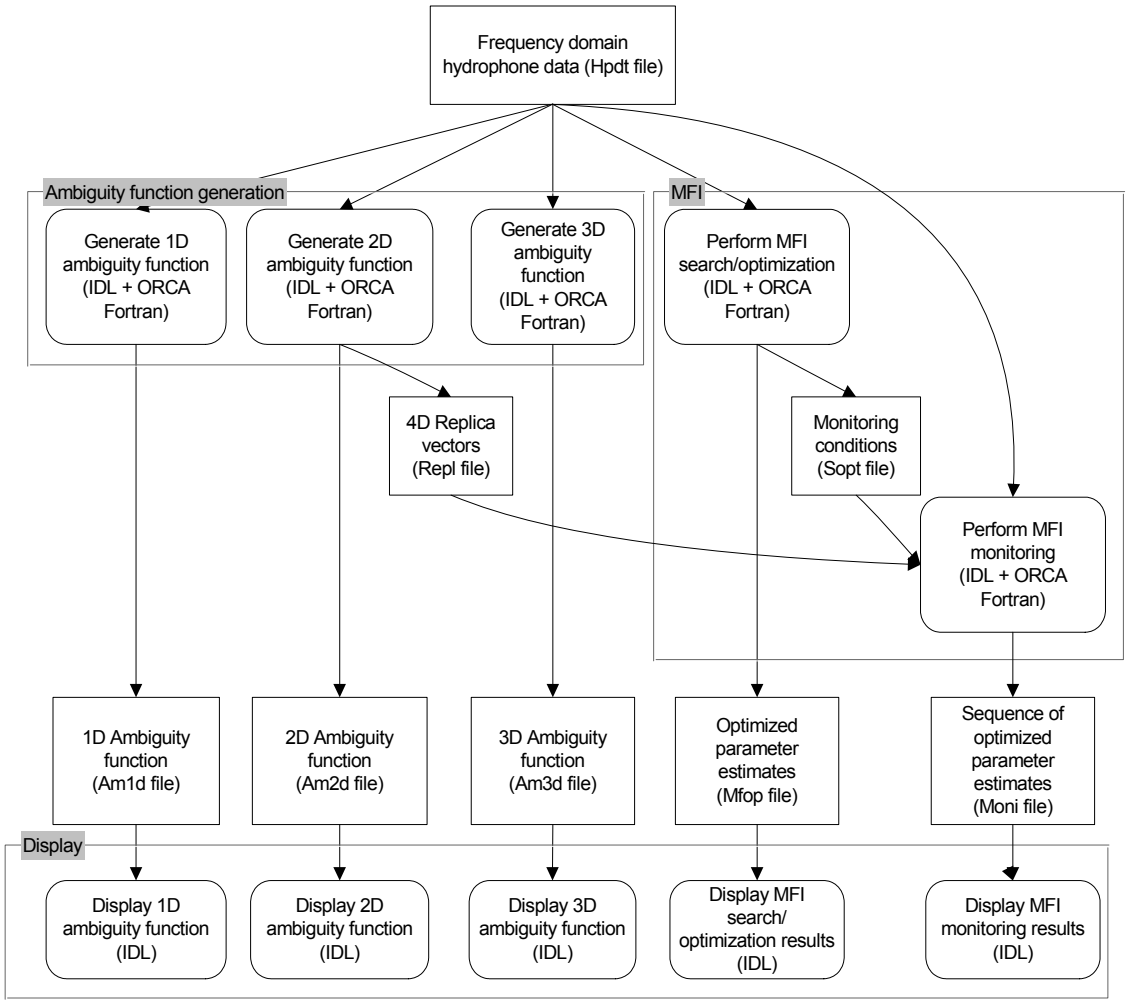
In this design, data and/or files are indicated by rectangles and processes are represented by rounded rectangles.




**Overview of BCOMFI Simulation, Conversion and Preprocessing Components**

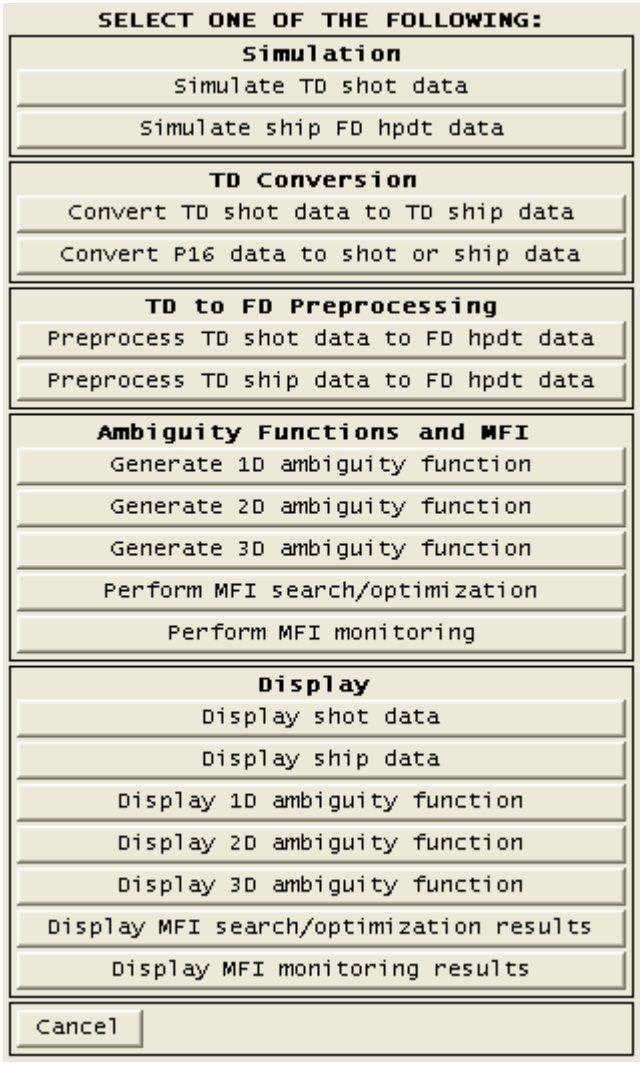
# Overall Design: Ambiguity Function Generation, MFI and Display Components

In this design, data and/or files are indicated by rectangles and processes are represented by rounded rectangles.




**Overview of BCOMFI Ambiguity Function Generation, MFI and Display Components**

Graphical User Interface




Pressing one of the buttons (other than Cancel) brings up another interface (described below) that allows the specification of the conditions for a run of that selected component of BCOMFI.

## Simulation of Time Domain Shot Data

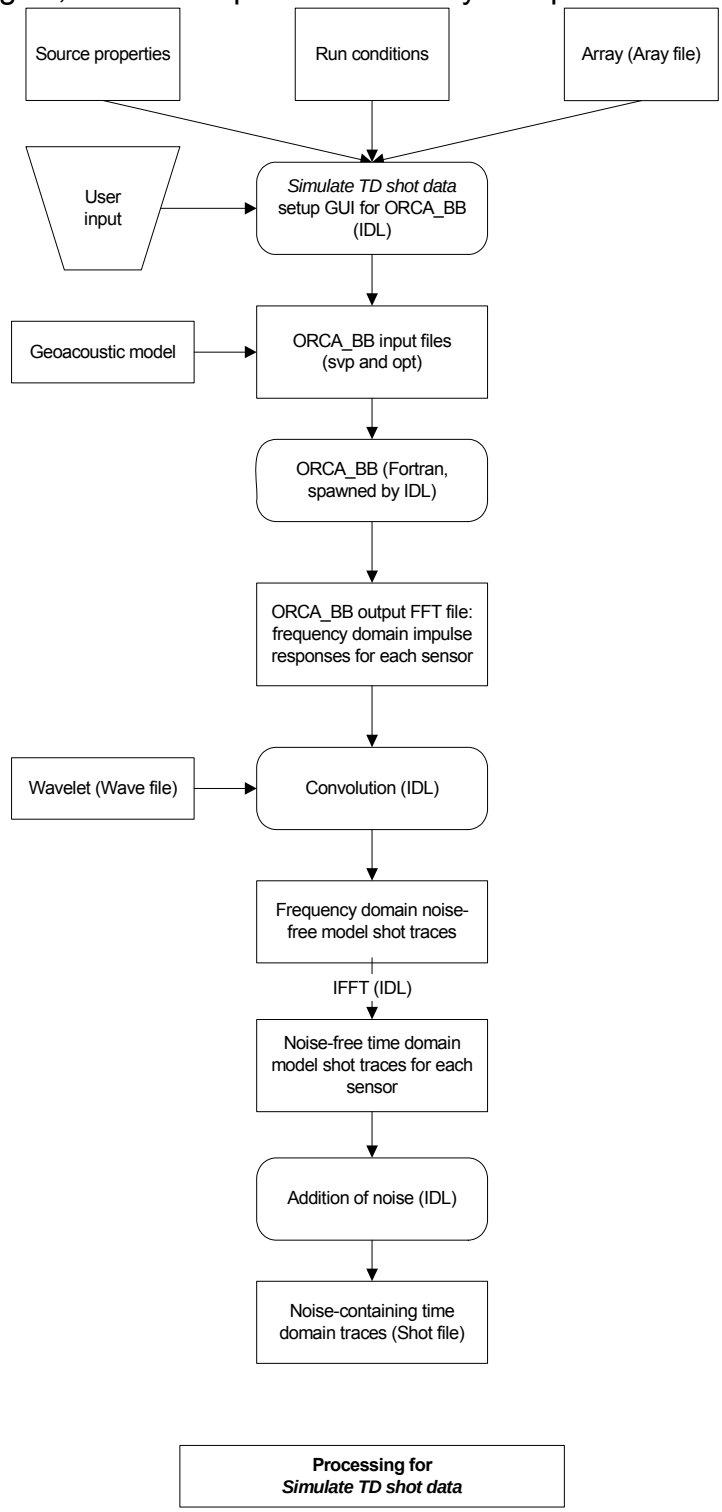
In order to develop and verify methods for MFI of the shot data to be obtained, it was essential to first be able to generate synthetic shot traces for specified source-array geometries and geoacoustic environments and use them to test the ambiguity function generation and MFI methods. These synthetic data consist of impulse responses (generated by propagation modeling) convolved with a representative source wavelet, with additive noise. The resulting model traces can then be FFT'd and selected frequencies used to perform and evaluate the subsequent MFI processing.

In the *Simulate TD shot data* component of the BCOMFI system, the broadband option of ORCA is used to generate the impulse responses. This option allows the user to specify a frequency band (e.g., 1 – 250 Hz), a sampling frequency, and a time window (or equivalently, a number of FFT points). ORCA then generates the frequency domain impulse response for each discrete frequency in the specified band, and outputs this result to an FFT file. This file is then read in and convolved with a specified (shot) waveform. An option to add Gaussian noise is provided, and the resulting frequency domain traces are then inverse FFT'd to yield the corresponding time domain traces.

Implementation of this software for simulating shot data allowed the methods for processing the real shot data to be developed and tested in advance of the real data becoming available.

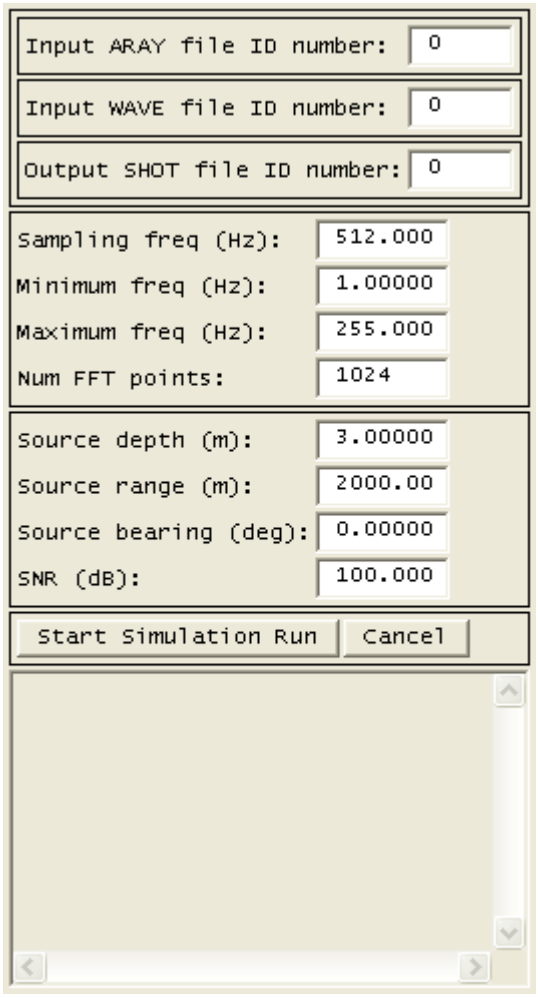
# Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface

The GUI for *Simulate TD shot data* is illustrated in the following diagram.




- The first three items allow the user to specify the input files (Aray and Wave) to be used for the run, and the Shot file to contain the results.
- The next four items allow the user to specify the conditions for the broadband ORCA run.
- The following four items (in the “Source” box) allow the user to specify the source-receiver geometry to be used in the run and the (power) signal-to-noise ratio (dB) for each trace.
- The “Start Simulation Run” and “Cancel” buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for simulating the shot traces is outlined in the following pseudocode:

*Set up an svp (sound velocity profile) file for ORCA to use as the geoacoustic environmental model.*
*Set up an Aray file to contain the array to be used in the simulation and a Wave file to contain the wavelet.*
*Using the GUI, specify the files to be used as the input array (Aray) and wavelet (Wave) files and the output Shot file, and specify the run conditions for the source and ORCA broadband.*
*Check that the run conditions are consistent.*
*Apply a tilt to the array.*
*Generate an array geometry file and an ORCA options file.*
*Spawn a process to perform an ORCA broadband run:*
*ORCA process:*
*For each frequency:*
*Compute the complex fields that would be observed at each sensor for a source at the specified position.*
*Output an ORCA FFT file containing the complex fields for the impulse responses for each of the sensors.*
*Read in the FFT file generated by ORCA.*
*Read in and FFT the wavelet from the specified Wave file.*
*Generate time domain traces: For each sensor:*
*Reflect/conjugate the FFT about the Nyquist frequency to obtain the Fourier transform of the impulse response.*
*Convolve the wavelet with the impulse response.*
*Inverse FFT the result to give a time domain trace.*
*Add Gaussian noise at specified signal-to-noise ratio.*
*Output the traces to a Shot file.*

## Using the Component

Press the “Simulate TD shot data” button to bring up the GUI for generating time domain traces that would be observed at the sensors of an array for a particular source-array geometry, geoacoustic model and source wavelet. The process uses the broadband option of ORCA to generate, for each of the sensors, the complex fields at each frequency within the specified band that would be observed on the basis of the geometry and geoacoustic model. These complex vectors are written to an FFT file by ORCA, and this file is then read in and the sensor data are multiplied with the Fourier transform of the wavelet. The result is inverse Fourier transformed and written to a Shot save file.

To run this model, first ensure that the file “sim_shot_svp” exists in the \data directory noted above, and that it contains the desired sound velocity profile in ORCA format. Then specify the following information in the GUI:

- the ID number of the Aray file containing the array to be used (sensor (*x, y, z*) points, tether depth for the *z* points, and tilt data);
- the ID number of the Wave file containing the wavelet to be used;
- the ID number of the output Shot file that will be produced;
- the sampling frequency, which should be at least twice the maximum frequency specified below;
- the minimum frequency in the band for which the complex fields are to be generated by ORCA;
- the maximum frequency in the band for which the complex fields are to be generated by ORCA (this should be at most half the sampling frequency);
- the number of FFT points; this should be long enough so that the entire trace is contained within the time window (i.e., number of FFT points ÷ sampling frequency); otherwise the traces may wrap around;
- the source depth and its range and bearing with respect to the array;
- the SNR (signal-to-noise ratio) for addition of white noise to the traces; note that for SNR ≥ 100, no noise is added.

When the above have been specified, press the “Start Simulation Run” button to perform the simulation. This will likely take several minutes. The progress of the ORCA broadband run can be monitored by viewing the DOS / Command Prompt window generated by the spawning process by which IDL invokes ORCA.

## Simulation of Frequency Domain Ship Data

As with the shot data, it was essential to have a software component that could generate simulated data in order to test the correctness and performance of the MFI algorithms during development. These synthetic data consist of complex signal vectors or cross-spectral matrices at several user-selected frequencies, each of which can optionally contain noise corresponding to various models. Since there may be more than one actual source, and these sources will generally be moving, the simulation also includes the ability to model and generate data for two moving sources.

As noted above, the main application for the simulations is to generate data for a range-independent environment. For this application, ORCA (in standard rather than broadband mode) is used to generate the fields at the sensors that would be observed for a simulated moving source. However, in view of the fact that the environment has a significant range-dependence, it was also of considerable interest to be able to simulate data for a simple range-dependent environment, and analyze this data with MFI using a range-independent model (ORCA). The range-dependent parabolic equation (PE) code RAM provides the ability to generate data for range-dependent environments, and is freely available over the web. This RAM code was obtained and incorporated into the

software system for generating synthetic data. The availability of two entirely separate propagation modeling codes provided a consistency/validity check by allowing comparison of the fields that are produced by the two models for a range-independent environment.

Hence, both ORCA and RAM options were implemented to generate acoustic fields at multiple frequencies for moving sources. Options were also provided for adding noise and performing cross-spectral matrix estimation. This implementation allowed the methods for processing the real ship data to be developed and tested in advance of the real data being available.

# Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.

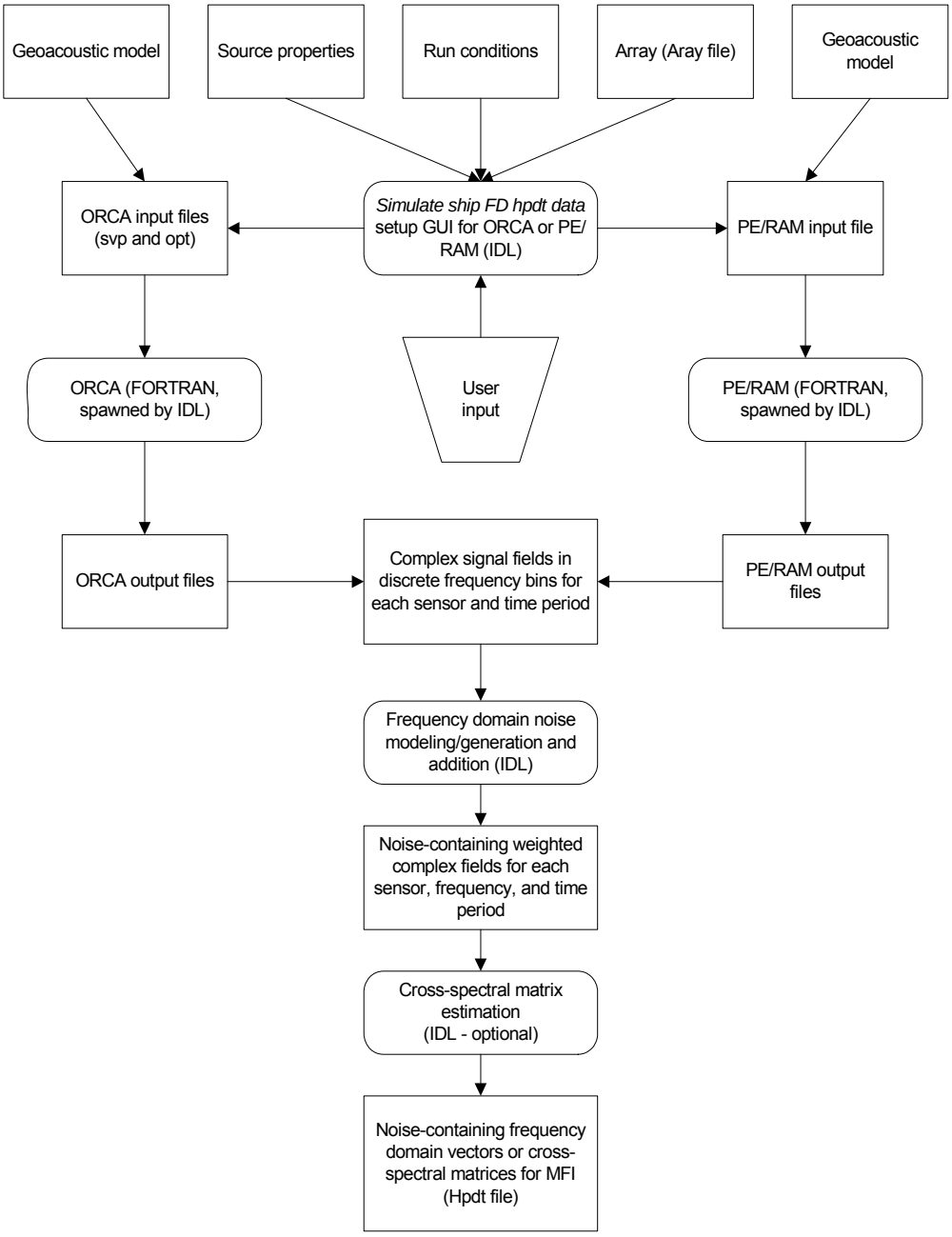
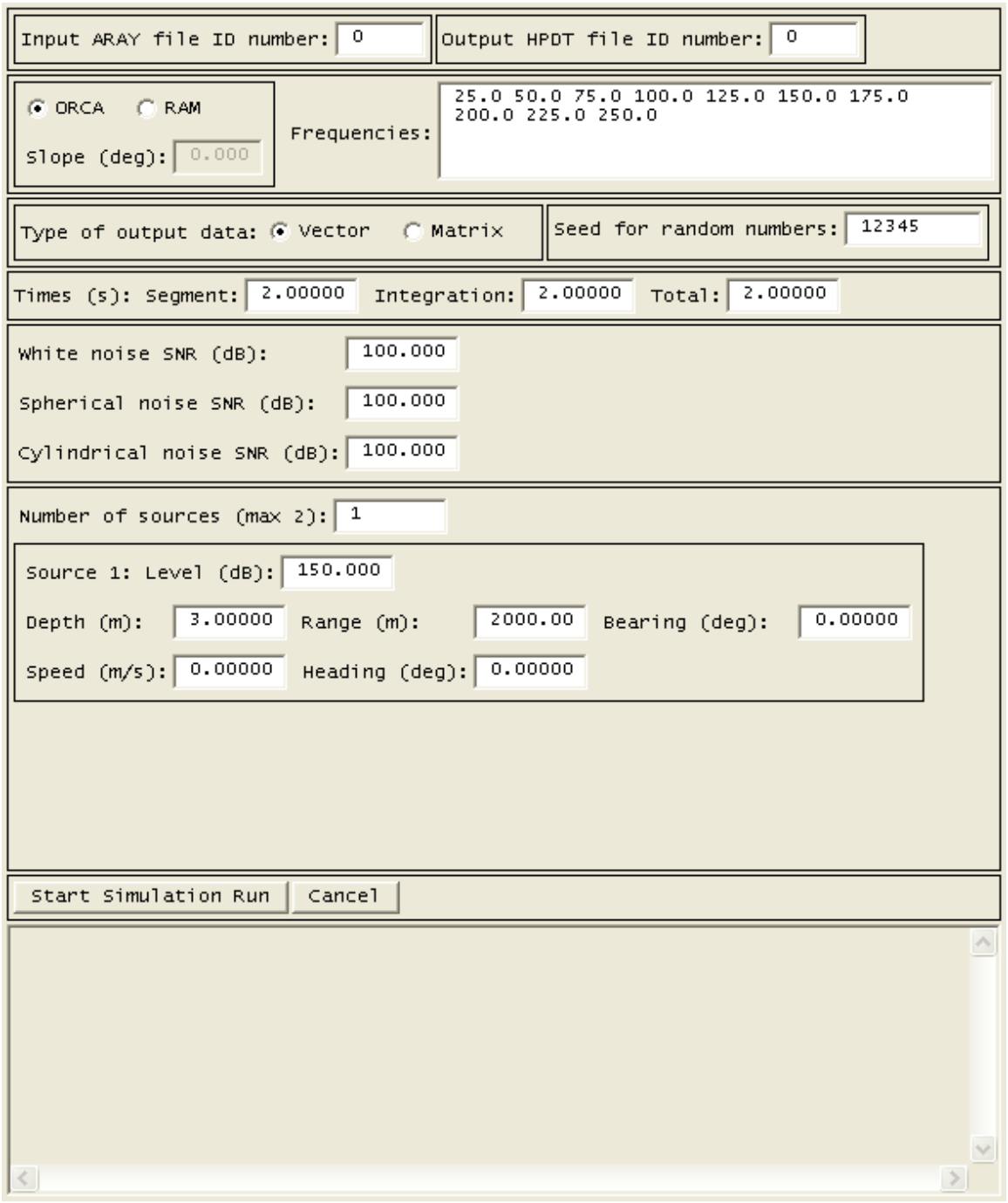



**Figure 6**

## Graphical User Interface




- The first items in the top row allow the user to specify the input Aray file to be used for the run, and the Hpdt file to contain the results.
- The second row allows the user to indicate whether ORCA or RAM is to be used, and to specify the frequencies for which the data are to be generated.

- The third row allows the user to specify whether the data are to be output as vectors or cross-spectral matrices, and to enter a seed for the random number sequence to be used in generating noise realizations. The latter allows repeated runs involving noise to be reproducible.
- The fourth row provides for specification of the segment time, the integration time (the time over which cross-spectral matrix estimation takes place), and the total time (which controls how many vectors/matrices are computed).
- The fifth row provides options for adding noise of specified signal-to-noise ratio from three different distributions: white, spherical, and cylindrical.
- The sixth row allows one or two sources to be specified.
- The seventh row (with the "Source 1" label) allows the user to specify the source level and the source-receiver geometry for Source 1 to be used in the run.
- The eighth row (if present) allows the user to specify the source level and the source-receiver geometry for Source 2 to be used in the run.
- The "Start Simulation Run" and "Cancel" buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for simulating the ship data is outlined in the following pseudocode:

*Set up an svp (sound velocity profile) file for ORCA and/or RAM, to use as the geoacoustic environmental model.*
*Set up an Aray file to contain the array to be used in the simulation.*
*Using the GUI, specify the input Aray file and the output Hpdt file, and select the run conditions.*
*Check that the run conditions are consistent.*
*Apply a tilt to the array.*
*For ORCA, generate an array geometry file and an options file.*
*For each frequency specified:*
    *Generate and Cholesky decompose the noise matrices for white, spherical, cylindrical, and modal noise at that frequency.*
*For each integration time:*
    *For each segment time:*
        *For each source:*
            *Compute new source position.*
            *For each frequency:*
                *Run ORCA or RAM to generate a signal vector.*
                *Scale the signal vector, generate a complex constant and use it to randomize the phase across the array, for that frequency.*
        *For each frequency:*
            *Generate a random noise vector for the array sensors using the noise cross-spectral matrix for that frequency and array.*
            *Add noise vector to signal vector.*

*For each frequency:*
*Accumulate cross-spectral matrices if this option is chosen.*
*Output the vector/matrix data to an Hpdt file.*

## Using the Component

Press the "Simulate ship FD hpdt data" button to bring up the GUI for generating data vectors or cross-spectral matrices, at selected frequencies, that would be observed at the sensors of an array for a particular source-array geometry (where the source can be moving) and geoacoustic model. For a moving source, a quasi-stationary assumption is used, so that the cross-spectral matrices are formed by averaging a number of stationary matrices computed for the source at points along a linear track. The vector or matrix data generated by the run are written to an Hpdt file for input to MFI.

The process uses ORCA or RAM to generate frequency domain pressure field values for each of the sensors, at the selected frequencies. If ORCA is chosen, the geoacoustic model is range-independent. If the RAM option is chosen, a constant slope may be specified for the bathymetry between the source and the array; also, if RAM is chosen, an image of the 2D output transmission loss is displayed at the end of the run for each frequency. Note that runs with RAM are much more time-intensive than ORCA runs.

To run this model, first ensure that, if ORCA is to be used, the file "sim_ship_svp" exists in the \data directory noted above, and that it contains the desired sound velocity profile in ORCA format. If RAM is to be used, ensure that the file "ramgeo.dat" exists in the \source directory and contains the information for the run conditions and the sound velocity profile in RAM format. Note that the parameters that can be influenced by information provided in the GUI are indicated by uppercase labels (e.g, FREQ, ZS) in ramgeo.dat. Also note that the water depth at the array is defined by the first ZB term in ramgeo.dat.

Then specify the following information in the GUI:

- the ID number of the Aray file containing the array to be used (sensor (*x, y, z*) points, tether depth for the *z* points, and tilt data);
- the ID number of the output Hpdt file that will be produced;
- whether ORCA or RAM is to be used, and, if RAM, the slope between the source and the array; positive slopes indicate that the depth increases from the array toward the source;
- the frequencies at which the data are to be generated (the default is ten frequencies, evenly spaced between 25 and 250 Hz);
- whether the output data are to be generated in the form of vectors or cross-spectral matrices;
- a seed for simulated noise generation;

- values for segment time (data length for a single FFT), integration time (time within which outer-product matrices are averaged to accumulate cross-spectral matrices), and total time;
- SNRs for three ocean noise types: white, spherical and cylindrical; note that for SNR ≥ 100, no noise of that type is added;
- the number of sources: currently restricted to 1 or 2; if the latter is chosen, a second region for source specification will appear below;
- the characteristics of the source(s): level (dB re 1 μPa at 1 m), depth, range and bearing with respect to the array, speed and heading.

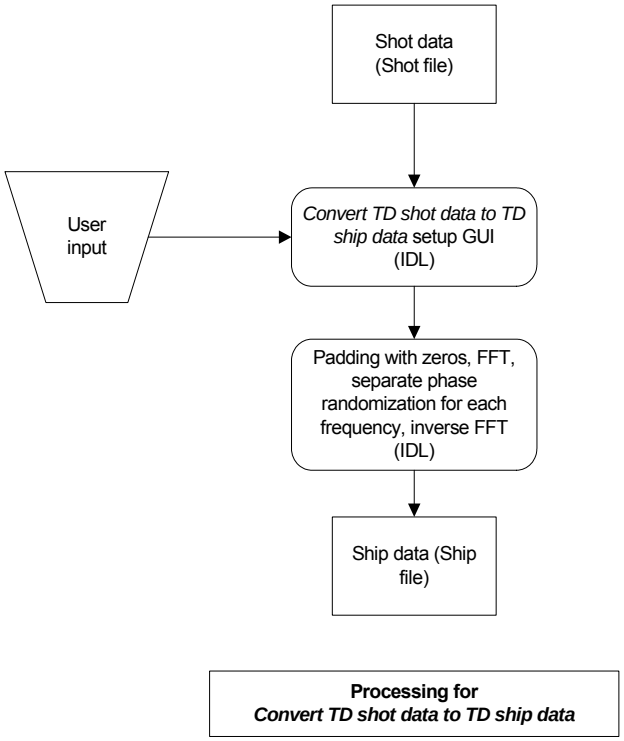
When the above have been specified, press the “Start Simulation Run” button to perform
the simulation. The time taken will depend on the frequencies chosen and on the propagation model: for ORCA, the computations will likely take several seconds, while for RAM, they may take several minutes.

## Conversion of Time Domain Shot Data to Time Domain Ship Data

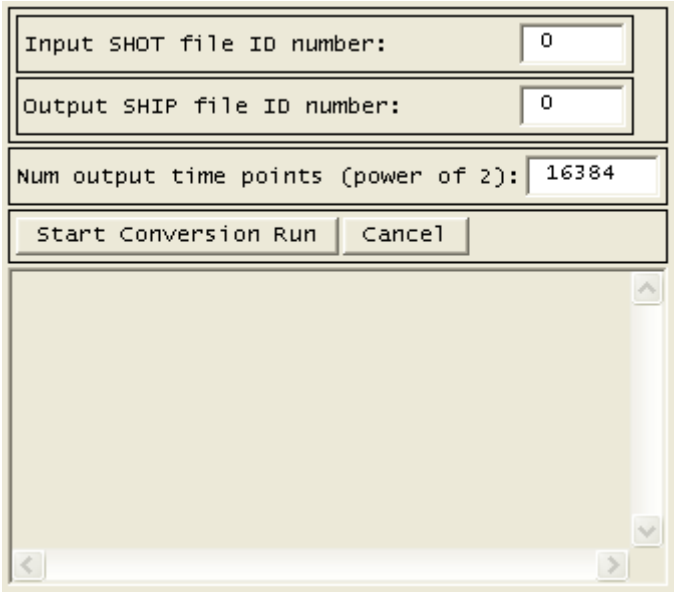
This functionality was developed in order to allow the generation of time domain ship data from previously-generated synthetic shot data for a specified environment. These ship data could then be used as input to a process that would use these data to generate Hpdt files for input to MFI. This software component provided a way of validating the processing of ship data for MFI in advance of the real ship data from the monitoring station being available.

## Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.




## Graphical User Interface



- The first two rows allow the user to specify the input Shot file to be used for the run, and the Ship file to contain the results.
- The next row allows the specification of the number of points in the output time series for each sensor. This should be at least the number of input points in the Shot file and should also be a power of 2 (otherwise, it will be set to a power of 2).
- The "Start Conversion Run" and "Cancel" buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for simulating the ship data is outlined in the following pseudocode:

*Set up a Shot file to contain the array to be used in the simulation.*
*Using the GUI, specify the input Aray file and the output Hpdt file, and select the run conditions.*
*Check that the run conditions are consistent.*
*For each sensor:*
    *Pad the trace with zeroes and FFT.*
*For each frequency:*
    *Generate a random complex constant and multiply it with the frequency bin value for each sensor to randomize the phase.*
*For each sensor:*
    *Inverse FFT the trace.*
*Output the result to a Ship file.*

## Using the Component

Press the "Convert TD shot data to FD ship data" button to bring up the GUI for converting shot traces to the type of data that would correspond more closely to that being collected for ships of opportunity. These data have the same spectra, but are different in that shot traces are coherent in frequency while the ship data are not; both data types are, of course, spatially coherent. In addition, the shot data are confined to a small region of time, while the ship data can go on for an arbitrary length of time. The conversion process involves reading in the shot traces, padding with zeros to increase their length, and Fourier transforming. Then phase randomization is applied to each frequency bin by generating a random rotation and applying the same rotation to each sensor of the array. The result is then inverse transformed to yield the simulated ship data, which is written to a Ship file.

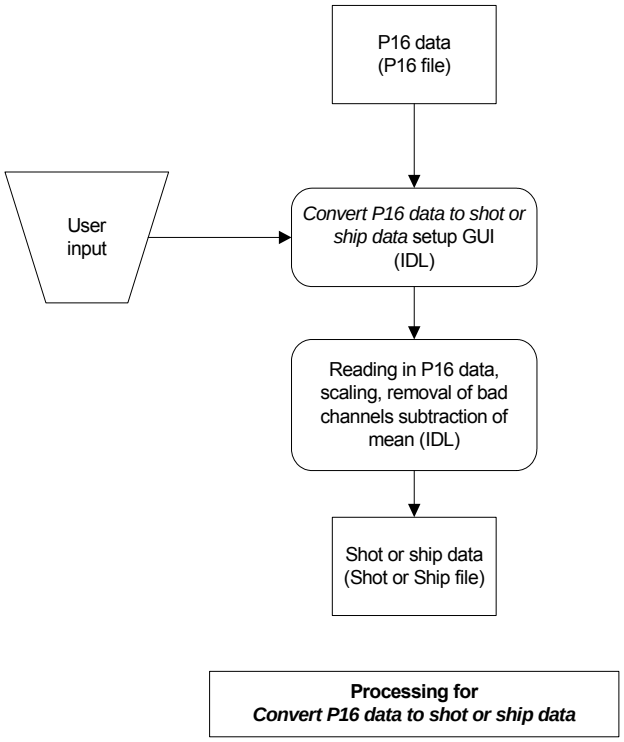
To run this conversion, specify:
- the ID number of the Shot file containing the array to be converted;
- the ID number of the output Ship file that will be produced;
- the number of time points in the output traces (power of 2).

When the above have been specified, press the “Start Conversion Run” button to perform the conversion. The process should be almost instantaneous.

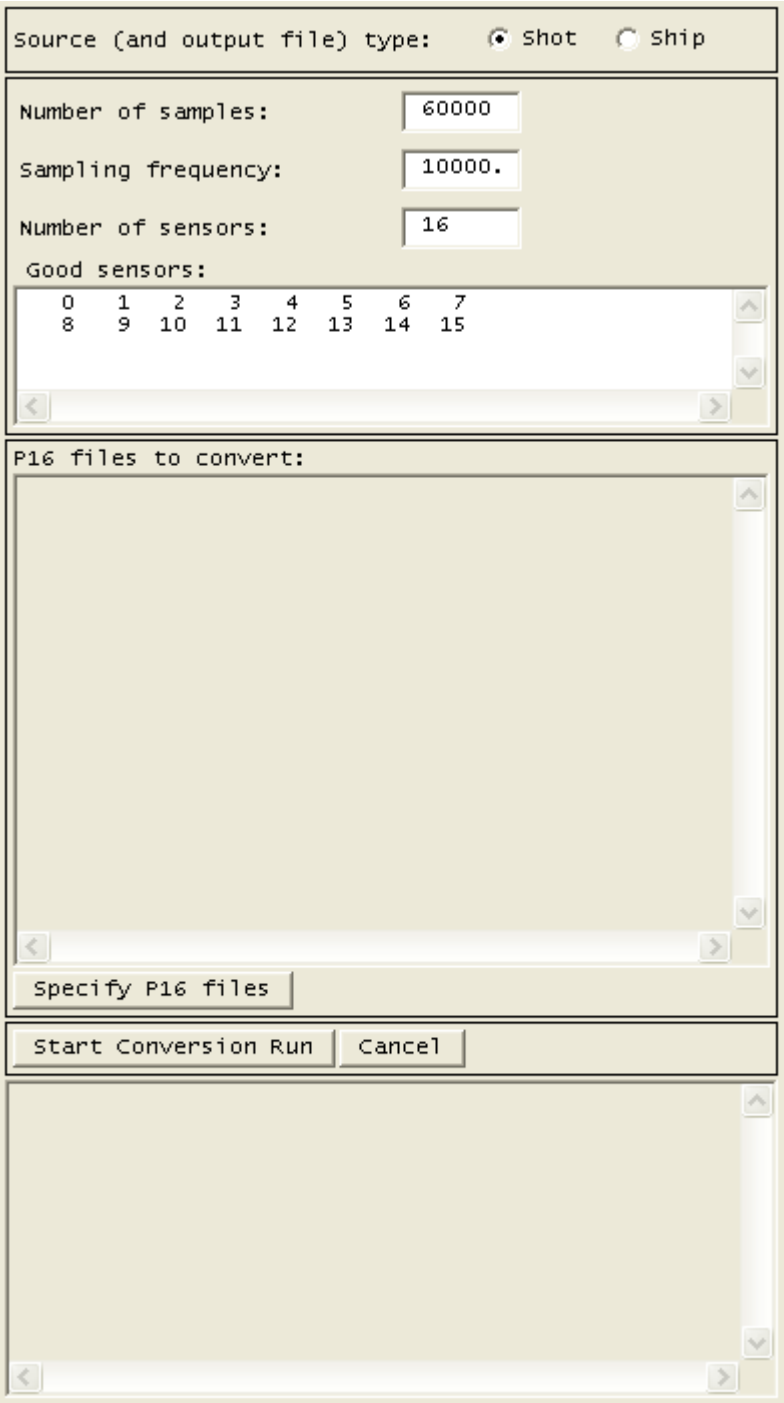
## Conversion of Time Domain P16 Real Data to Time Domain Shot or Ship Data

### Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface




- The first row allows selection of the source and output file type: either Shot or Ship. This should be matched to the type of input data in the P16 file(s).
- The second row allows the user to specify the number of time points that have been recorded for each sensor in the input P16 file(s).
- The third row allows specification of the sampling rate (Hz).

- The fourth row specifies the number of sensors in the input data.
- The fifth row allows the specification of which sensors are "good" and which should be retained in the output file(s).
- The next large initially empty area provides a field for displaying the P16 files selected for conversion.
- The "Specify P16 files" button brings up a dialog box that allows the user to specify which file(s) are to be converted from P16 to Shot or Ship files.
- The "Start Conversion Run" and "Cancel" buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for simulating the ship data is outlined in the following pseudocode:

*Using the GUI, specify the input P16 file(s) and select the run conditions.*
*Check that the run conditions are consistent.*
*For each P16 file:*
    *Read the binary data.*
    *Scale the data to lie between –1 and 1.*
    *Remove bad channels from the data.*
    *For each trace:*
        *Subtract the mean.*
    *Output the traces to a Ship or Shot file with the same ID as the P16 file.*

## Using the Component

Press the "Convert P16 data to shot or ship data" button to bring up the GUI for converting real array data in P16 format to shot or Ship files with the same ID number as the P16 files.

To run this conversion, specify:
- the type of data in the P16 files to be converted – either ship or shot;
- the number of time points in the output traces (power of 2), the sampling frequency, and number of sensors in the array;
- the "good" sensors, for which the data are to be retained;
- the P16 files to be converted, by pressing the "Specify P16 files" button, navigating to the directory containing those files, and selecting the desired ones – note that they should correspond to the specified source type (shot or ship); the selected files will be displayed in the text window above the button;

When the above have been specified, press the "Start Conversion Run" button to perform the conversion. The process should take only a few seconds.

Note that only the "good" sensors data will appear in the output data files. Hence, you

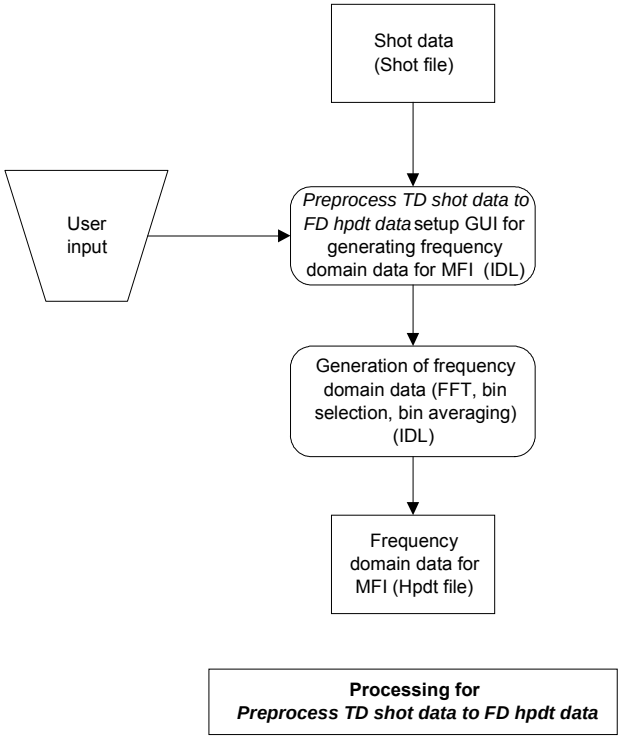
will have to create a corresponding Aray file to match these sensors before performing ambiguity function generation or MFI.

## Preprocessing of Time Domain Shot Data to Frequency Domain Hpdt Data

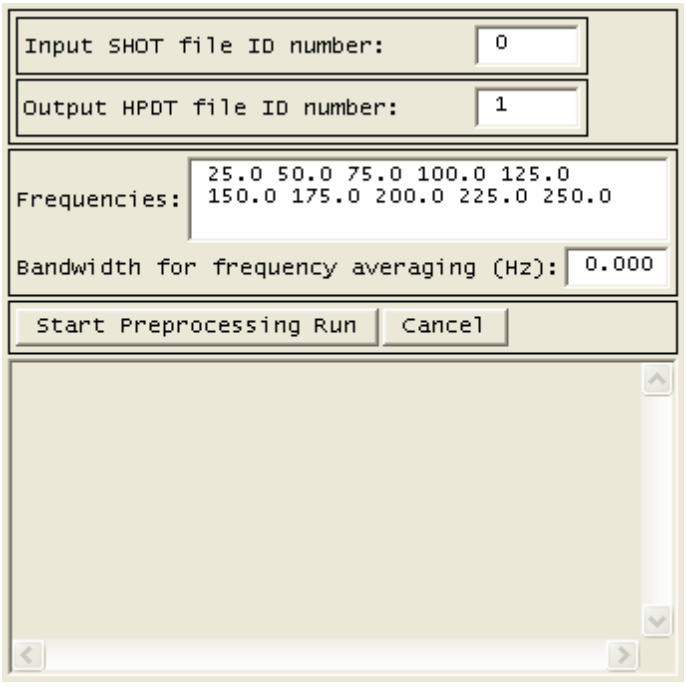
This component allows the data for each shot to be processed into a form amenable to MFI analysis. The strategy for processing the shot data is to FFT the traces for the sensors and form cross-spectral matrices at user-specified frequencies, with the option for averaging within a small frequency band at each of these frequencies. These matrices are output to an Hpdt file, which can be used as input for the ambiguity function and MFI components of BCOMFI.

### Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface




- The first items in the top row allow the user to specify the input Shot file to be used for the run, and the output Hpdt file to contain the results.
- The next two lines allow the user to specify the frequencies at which the FFT'd shot data are to be retained and the bandwidth within which frequency averaging centered at these frequencies is to be performed.
- The "Start Preprocessing Run" and "Cancel" buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for processing the shot data is outlined in the following pseudocode:

*Specify the input Shot file and output Hpdt file.*
*Select the frequencies at which signal vectors are to be computed.*
*For each shot:*
    *FFT the traces for each sensor.*
*For each selected frequency:*
    *Form a frequency-averaged cross spectral matrix by accumulating the outer products of vectors within a specified band of frequencies centered at the selected frequency (optional).*
*Write matrices to an Hpdt file.*

## Using the Component

Press the “Preprocess TD shot data to FD hpdt data” button to bring up the GUI for preprocessing shot traces to hpdt form, where they can be used as input to MFI. The preprocessing involves Fourier transforming the traces, and then, for each selected frequency, forming an averaged cross-spectral matrix corresponding to the sensors of the array. The resulting data are written to an Hpdt file.

To run this conversion, specify:

- the ID number of the Shot file containing the array to be converted;
- the ID number of the output Hpdt file that will be produced;
- the frequencies at which the complex fields are to be generated from the shot traces;
- the bandwidth to use for averaging the cross-spectral matrices at each frequency.

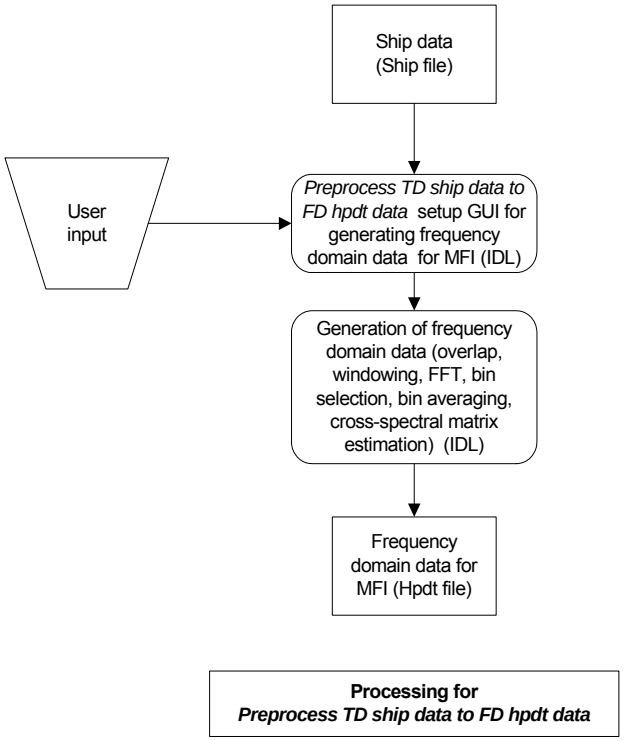
When the above have been specified, press the “Start Preprocessing Run” button to perform the preprocessing. The process should be almost instantaneous.

# Preprocessing of Time Domain Ship Data to Frequency Domain Hpdt data

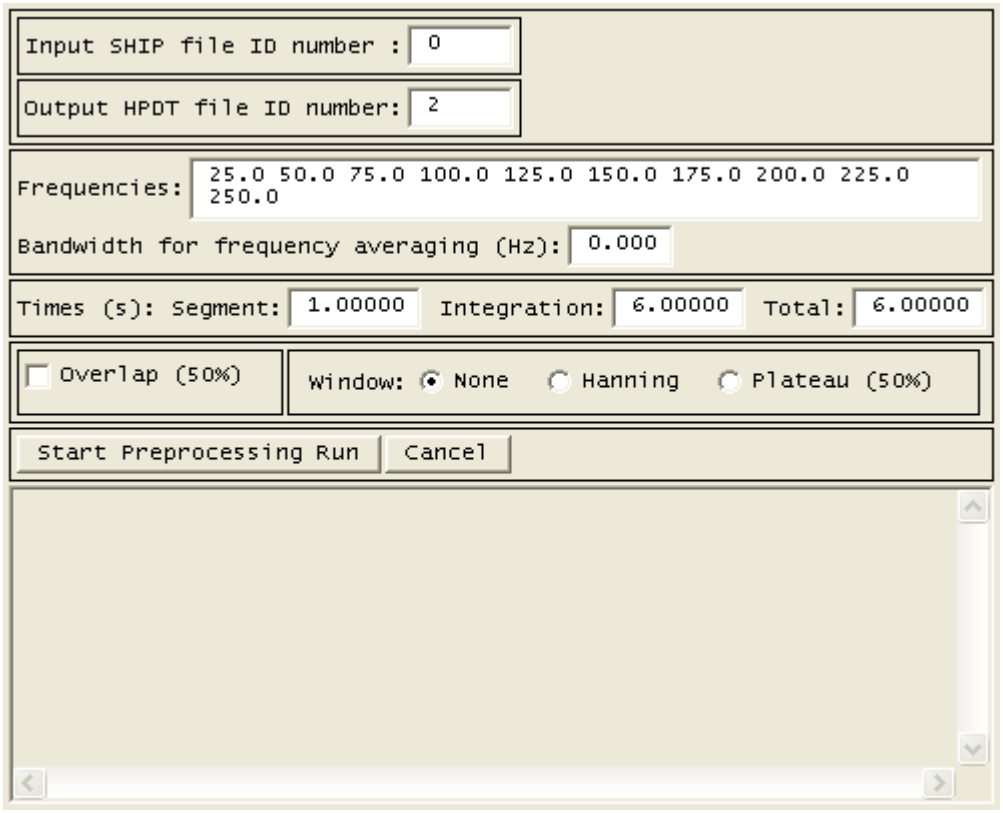
The time domain ship data are processed as a series of (possibly overlapping and windowed) segments to yield cross-spectral matrices for MFI analysis. The strategy for processing the ship data is to segment the time series, optionally overlap the time segments by 50% and apply a window (e.g., Hanning), and then FFT the sensor data for each segment. The data at user-specified frequencies will then be used to form cross-spectral matrices, which will be output to an Hpdt file for MFI.

# Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface




- The items in the top row allow the user to specify the input Ship file to be used for the run, and the Hpdt file to contain the results.
- The next two lines allow the user to specify the frequencies at which the FFT'd ship data are to be retained and the bandwidth within which frequency averaging centered at these frequencies is to be performed.
- The third row provides for specification of the segment time, the integration time (the time over which cross-spectral matrix estimation takes place), and the total time (which controls how many vectors/matrices are computed). Note that at present only the data for the first integration time can be analyzed by MFI.
- The fourth row allows the user to specify the overlap to use for the time series to be FFT'd and the window to use prior to performing the FFT.
- The "Start Preprocessing Run" and "Cancel" buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for preprocessing the ship data is outlined in the following pseudocode:

*Specify the input Ship file and the output Hpdt file.*
*Select the frequencies at which signal vectors are to be computed.*
*Repeat for each integration time:*
    *For each sensor:*
        *Obtain the data for the segment (optional overlap).*
        *Window the data.*
        *FFT the data.*

    *For each frequency:*

        *Form a frequency-averaged cross spectral matrix by accumulating the outer products of vectors within a specified band of frequencies centered at the selected frequency (optional).*
    *Output the cross-spectral matrices to an Hpdt file.*

## Using the Component

Press the "Preprocess TD ship data to FD hpdt data" button to bring up the GUI for preprocessing ship traces to hpdt form, where they can be used as input to MFI. The preprocessing involves Fourier transforming sequential segments of the traces, and accumulating cross-spectral matrices at selected frequencies. Options are provided for overlapping the segments by 50%, and for windowing the segments prior to Fourier transformation for each selected frequency. The transformed data are used to form a sequence of complex cross-spectral matrices representing the fields observed at the sensors of the array. An option is also provided in the interface to average the outer products of sensor field vectors for frequency bins within a window centered on each of the chosen frequencies. The matrices generated by the preprocessing are written to an Hpdt file.

To run this conversion, specify:

- the ID number of the Shot file containing the array to be converted;
- the ID number of the output Hpdt file that will be produced;
- the frequencies at which the complex fields are to be generated from the shot traces;
- the bandwidth within which the outer products of the complex field data are to be averaged for formation of the cross-spectral matrices;
- values for segment time (data length for a single FFT), integration time (time within which outer-product matrices are averaged to accumulate cross-spectral matrices), and total time;
- whether the segments are to be overlapped by 50%;
- the window, if any, to be applied to the traces before FFT.

When the above have been specified, press the "Start Preprocessing Run" button to

perform the preprocessing. The process should be very rapid.

## Matched Field Techniques: Brief Overview

Matched field processing and matched field inversion are array signal processing methods which search over a parameter space for unknown model parameters by matching a replica field vector computed using a parameter-based model with "measured" data. In the present context, the replica data are computed using acoustic propagation modeling to generate the fields at an array that would be observed for a particular realization of source-array geometry and geoacoustic model. The "measured" data are obtained from an array of hydrophones or from propagation modeling using a reference source-array geometry and geoacoustic model. The replica data are matched with the "measured" data using a power processor, with high processor output indicating good matches. A maximum processor value can indicate a reasonable correspondence between the model parameters and the actual structure of the environment.

The normalized Bartlett power processor is used here to compute the matching (ambiguity) function between the measured data and the replica vectors generated using ORCA. For a measured data vector **m** and a replica vector **r**(**p**) computed for parameter set **p** at a single frequency**,** the Bartlett processor is defined as

$$B(\mathbf{r}(\mathbf{p}),\mathbf{m}) = \frac{|\,\mathbf{r}(\mathbf{p})^{*}\mathbf{m}\,|^{2}}{|\,\mathbf{r}(\mathbf{p})\,|^{2}|\,\mathbf{m}\,|^{2}}\,.$$

For a measured data cross-spectral matrix **M,** it is defined as

$$B(\mathbf{r}(\mathbf{p}),\mathbf{M}) = \frac{|\,\mathbf{r}(\mathbf{p})^{*}\mathbf{M}\mathbf{r}(\mathbf{p})\,|}{|\,\mathbf{r}(\mathbf{p})\,|^{2}||\,\mathbf{M}\,||}\,,$$

where ||**M**|| is the spectral norm of the cross-spectral matrix **M**.

For multi-frequency measured data, a processor that combines the Bartlett outputs is defined as

$$C(\mathbf{p}) = \sum_{j=1}^{N} w_j B(\mathbf{r}_j(\mathbf{p}),\mathbf{m}_j) \quad \text{or} \quad C(\mathbf{p}) = \sum_{j=1}^{N} w_j B(\mathbf{r}_j(\mathbf{p}),\mathbf{M}_j)$$

for vector and cross-spectral matrix data, respectively, where $\mathbf{r}_j(\mathbf{p})$ is the replica vector for parameter set **p** at the *j*th frequency, $\mathbf{m}_j$ is the measured vector at the *j*th frequency, *N* is the number of frequencies, and $w_j$ is a weight ($\sum w_j = 1$).

Matched field techniques always involve ambiguity, in that many local maxima may be present in the parameter space of interest, and these matches can often be comparable

to those obtained using the true parameter set. Also, matches can be very insensitive to some parameters, or two parameters may be highly correlated, giving rise to other forms of ambiguity. The presence of noise introduces additional ambiguity. Hence, the ability to visualize the ambiguity in matched field techniques provides valuable insight into the characteristics of the parameter space and the validity of estimation of the geometric or geoacoustic model parameters. Based on these considerations, functionality has been provided in BCOMFI to allow the computation and visualization of ambiguity functions of up to three dimensions. The use of these ambiguity functions is described in the following section.

Matched field inversion is routinely accomplished by repeated forward modeling to generate replicas and matching the fields with the “measured” data. The use of optimization methods allows the parameters to be adjusted so as to achieve the best fit between the replica and “measured” data. Since there can be many local optima in the parameter space, it is essential to have an approach that has both global and local aspects. The approach to MFI taken in BCOMFI is to have an initial global search stage followed by local optimization of each of a number of the best matches found during the search stage. This approach is described in the Matched-Field Techniques: Inversion section below.

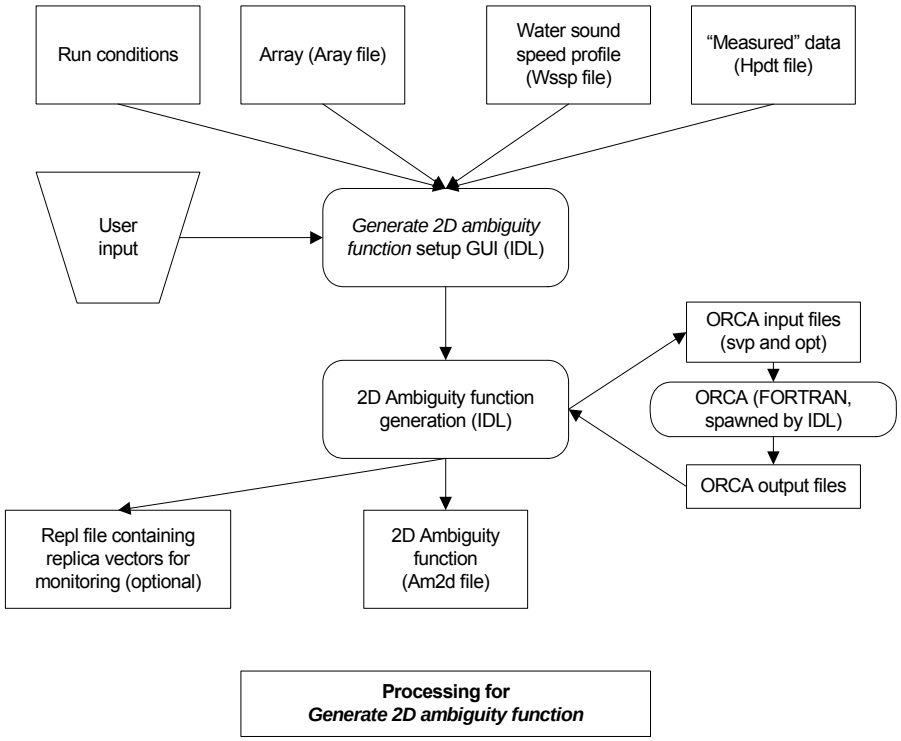
## Matched Field Techniques: Ambiguity Function Generation

In investigations involving MFI, it is useful to be able to examine how the ambiguity function (the matching function used as an objective function in MFI) depends on the individual parameters, and, sometimes, groups of parameters. To allow the visualization of this behavior, a component of BCOMFI has been provided that allows the generation of 1D, 2D, and 3D ambiguity functions (higher-dimension grids are too time-consuming to generate and more difficult to visualize). The user can then use IDL display software to examine the characteristics of these functions, including dynamic range, peak widths, presence of multiple optima, parameter sensitivity, and, in the case of 2D or 3D ambiguity functions, parameter interdependency. This last item is of particular significance, since it can lead to ill-posed MFI problems and inconsistent results in parameter estimation. Visualization of the ambiguity functions for the parameter space can assist in the interpretation of such results.

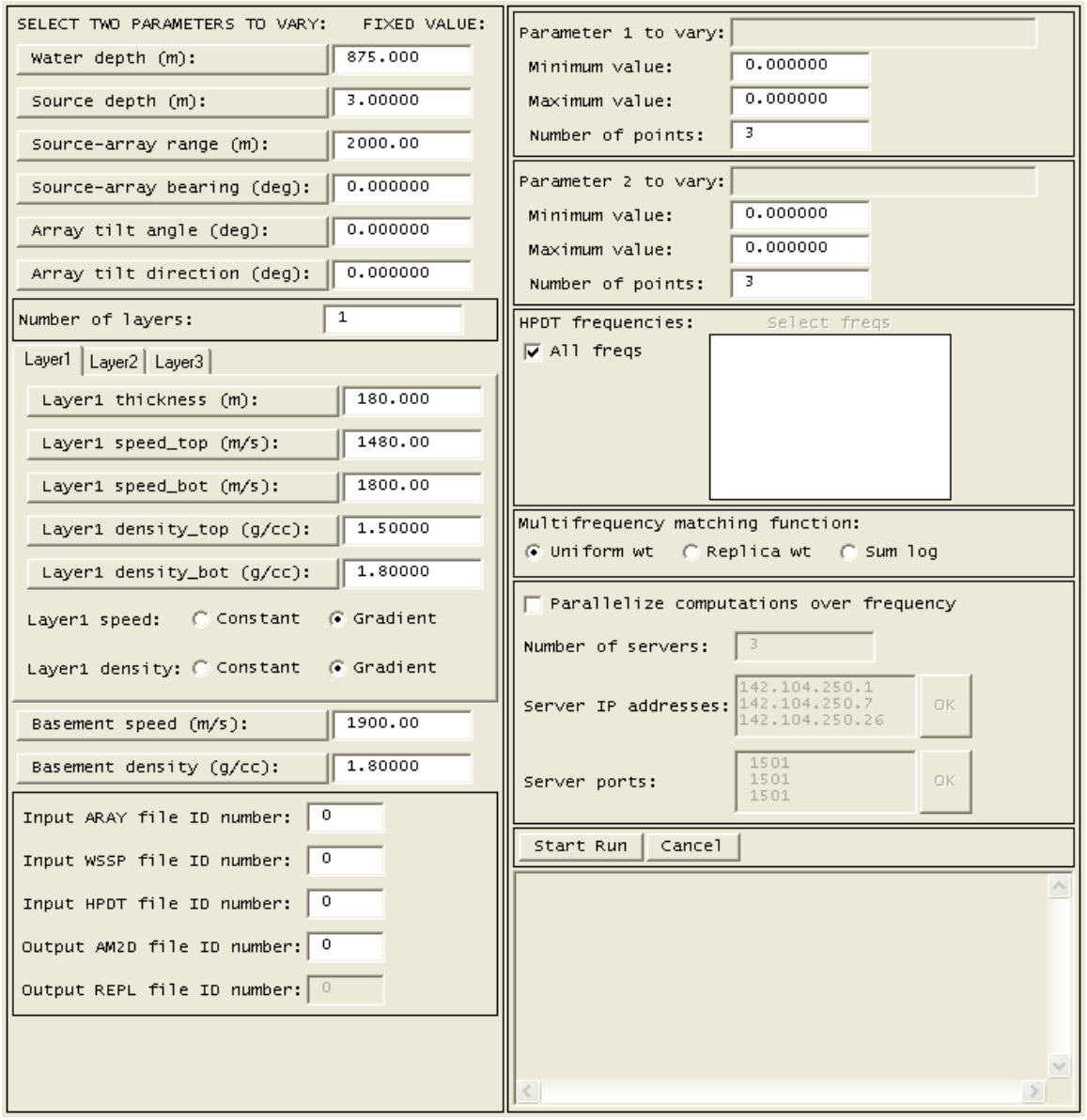
This section describes the ambiguity function generation component, using the 2D case as an example; the 1D and 3D versions are analogous to the 2D version.

# Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



# Graphical User Interface




- The block (including tabs) on the top left lists the 23 parameters that can be varied to generate the ambiguity function, and provides default fixed values for those parameters. Pressing on the button containing a parameter name chooses that parameter as one of the two to be varied and populates one of the “Parameter 1 to vary” or “Parameter 2 to vary” fields in the blocks at the top right.
- The block under the tabs provides an option for the user to force the indicated parameter pairs for the top (e.g., density1) and bottom (e.g., density2) of the corresponding layer to be the same.
- The lower block on the left allows the user to specify the input files (Aray, Wssp, and Hpdt) to be used for the run, and the Am2d and Repl (optional) files to contain the results.

- The top two blocks on the right allow specification of the domain over which the parameters are to be varied, and the number of points in each dimension.
- The third block on the right allows the user to view the frequencies of the data in the Hpdt file and select a subset of these to use in computing the multi-frequency ambiguity function. An option to select all the frequencies in the data is also provided.
- The fourth block on the right provides options for how the matches at the multiple frequencies are to be combined in the overall match. For the uniform and replica options, the Bartlett outputs at the frequencies are weighted uniformly or according to the squared modulus of the replica vector, respectively. For the Sum log option, the expression $\sum_{j=1}^{N} \log(1 - B(\mathbf{r}_j(\mathbf{p}), \mathbf{m}_j))$ (or $\sum_{j=1}^{N} \log(1 - B(\mathbf{r}_j(\mathbf{p}), \mathbf{M}_j))$ for matrix data) is used.
- The fifth block on the right allows for parallel processing over frequency, with the server IP addresses and ports of the remote computers specified in the text boxes.
- The "Start Run" and "Cancel" buttons are self-explanatory.
- The large empty area at the bottom right provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for generating the 2D ambiguity function is outlined in the following pseudocode:

*Specify the input Aray file and the output Hpdt file, and specify the run conditions.*
*Check that the run conditions are consistent.*
*Apply a tilt to the array.*
*For each value of parameter 1:*
    *For value of parameter 2:*
        *For each frequency selected (optionally parallelized):*
            *Apply tilt values to the array positions.*
            *Generate an array geometry file for ORCA.*
            *Generate an opt file for ORCA.*
            *Generate an svp file for ORCA.*
            *Spawn an ORCA process to generate a replica vector.*
            *If depth-range function, save the replica vector.*
            *Compute the Bartlett power of the match for that frequency.*
        *Sum the weighted Bartlett powers for the frequencies to give the ambiguity function for (parameter 1, parameter 2).*
*Output the ambiguity function to an Am2d file.*
*If depth-range function, output the saved replica vectors to a Repl file.*

## Using the Component

Press the "Generate 2D ambiguity function" button to bring up the GUI for computing a

2D ambiguity function. This will compute a 2D array of Bartlett matches for pairs of selected parameters being varied, with fixed values for the other parameters. Note that if a depth-range ambiguity function is chosen, replica vectors for each frequency, depth and range will be computed and saved to a Repl file for later use in monitoring. To generate the 2D ambiguity function, do the following:

- Select the parameters to vary, by clicking on the corresponding text buttons containing the names in the left column; these parameter names will then appear in the two text fields on the top right.
- Specify the values to use for the fixed parameters in the editable fields on the left portion of the interface. To change the number of layers (which can be from 0 up to 3) specify this number in the "Number of layers" field. To access the parameters in a layer other than the present layer, click on one of the tabs: Layer1, Layer2, Layer3.
- Ensure that radio buttons are selected to enable or disable gradients for sound speed and density in the corresponding layer.
- Specify the upper and lower limits for the two parameters to vary, and the number of points to evaluate in each parameter dimension.
- Specify ID numbers for the input Aray, Wssp, and Hpdt files, and the output Am2d and (optional) Repl file to contain the computed ambiguity function. Note that only if depth and range are chosen as parameters to vary will the Repl file ID number field become active.
- Specify the frequencies in the Hpdt file to use in the matching. Checking the "All freqs" checkbox uses all the frequencies in the file. Unchecking the box enables the option to select one or more of the individual frequencies in that file. Note that the Hpdt file with the specified ID number must exist in order to be able to use this option.
- Specify the option to use for combining the frequencies in the final matching function.
- If the computations are to be run in parallel over frequency, check the "Parallelize computations over frequency" checkbox and ensure that the numbers of servers, the IP addresses and the ports specified in the text boxes are correct. If you edit these fields, note that you must press the respective "OK" button for the changes to take effect. (If you wish to change the default values, edit the file "servers.dat" in the \source directory.) Also ensure that orcaserver processes are running on each of the computers selected (see the Installing the BCOMFI Software section above for information on installing and running these processes on Windows and Linux computers).
- Press the "Start run" button.

The result of the run will be an Am2d file, which can be viewed using the "Display 2D ambiguity function" option on the Main Menu interface. If a depth-range function was chosen, a Repl file is also produced by the run, which can be used in monitoring.

# Matched-Field Techniques: Inversion

## Background

The purpose of the MFI component is two-fold:

- to provide a test bed for investigation of MFI techniques, and
- to allow the analysis of real data by means of MFI.

The aim of the MFI software component of BCOMFI is to provide an environment for modeling and MFI using synthetic data and various parameterizations. Using this software, studies can then be performed to determine which approaches will be effective for detecting changes within the hydrate stability zone. Using synthetic data generated by the above components, the software can allow investigation of the following questions, for example:

- What are the relative sensitivities, peak widths, oscillations, and dynamic ranges of the matching function with respect to each of the 23 model parameters?
- How do the above parameter characteristics vary with range and frequency?
- Can we estimate the numbers of optima in the entire search region?
- Under what conditions can we ignore density optimization in the matching?
- What is the effect of errors in one or more of the fixed parameters?
- What is the effect of allowing a gradient in the sediment when the parameter is constant, or forcing it to be constant when there is in fact a gradient?
- What is the effect of multiple sediment layers in the data above the hydrate stability zone, when the matching model contains fewer layers?
- What is the effect of additional layers below the hydrate stability zone, when the matching model contains fewer layers?
- What is the effect of noise of various types, both uncorrelated and correlated?
- What is the effect of multiple sources and source motion on the matching?
- What is the effect of a gas layer at an interface?
- When is regularization required to obtain consistent inversion results?
- Under what conditions and for what parameters can we expect to detect significant changes that would indicate alterations in the hydrate-containing layer? That is, how substantial would changes have to be before they could be reliably detected?
- How do the answers to the above questions change when we move to a range-dependent environment?

The MFI component also provides an environment for the future analysis of real data (both shot and ship) with the intended result of defining a standard model, or perhaps a sector-dependent set of models, that provide a reasonable representation of the environment in the region of the array. These models could then be used as a basis for real-time monitoring.

In designing and implementing the MFI component, we considered that MFI is a nonlinear process that is generally approached using optimization techniques that

repeatedly solve the forward problem for varying sets of parameters until a suitable good match to the data is obtained. MFI optimization approaches must be able to deal with the following challenges:

- There are typically 5 – 25 parameters, and it is required to be able to optimize any or all combinations of these.
- There are generally multiple local optima present in the parameter space.
- It is desirable to restrict the domain of the parameters, usually by bounds constraints.
- The sensitivities of the parameters can be very different (by a factor of 100 or more).
- The parameters need to be scaled for optimization.
- Derivatives are unavailable except by numerical approximation.
- Certain parameters can be correlated, leading to ill-posed problems.
- There is a possibility of discontinuities in the matching function, particularly if the propagation algorithms do not always successfully converge/complete (as has been observed to be the case with ORCA under some conditions).

MFI has been implemented using a global search / local optimization approach. The advantages of this approach are the ability to obtain multiple estimates of the various optima, and the moderate number of function evaluations required (e.g., 2000 for search space sampling and 3000 for 10 optimizations). While a potential drawback of this method is that the individual optimized estimates are local, a sufficiently comprehensive search stage will increase the likelihood that one of these optima is, in fact, global.

The intended outcome of investigations conducted using the MFI component will be to compute effective range-independent geoacoustic models of the environment in the region of the monitoring station. Based on the data acquired during the calibration stage, it is likely that these models will be somewhat region-dependent; i.e., a different model may be derived by MFI for each of a number of regions in the general area. One possible approach to monitoring would be to use one or more of these models as standards and then look for mismatches between the models and the actual data obtained from passing ships.

## Implementation

The overall method implemented for MFI involves the following stages:

***A global search stage***. In this stage, sets of values for the parameters to be varied are generated that are randomly distributed between the lower and upper bounds for those parameters. ORCA is then used to generate a replica vector for this parameter set, and the matching function is then computed. This process is repeated a specified number of times to provide a sampling of the overall search space.

***A local optimization stage***. In this stage, each of a specified number of the best matches found in the global search stage is used as a starting point for optimization of

the parameters. Here the objective function $F(\mathbf{p})$ of the parameter set $\mathbf{p}$ to be minimized is:

$$F(\mathbf{p}) = 1 - C(\mathbf{p}) + P(\mathbf{p}) + R(\mathbf{p}),$$

where $C(\mathbf{p})$ is the processor output, $P(\mathbf{p})$ is a penalty function for values of the parameters outside their bounds, and $R(\mathbf{p})$ is a regularization function.

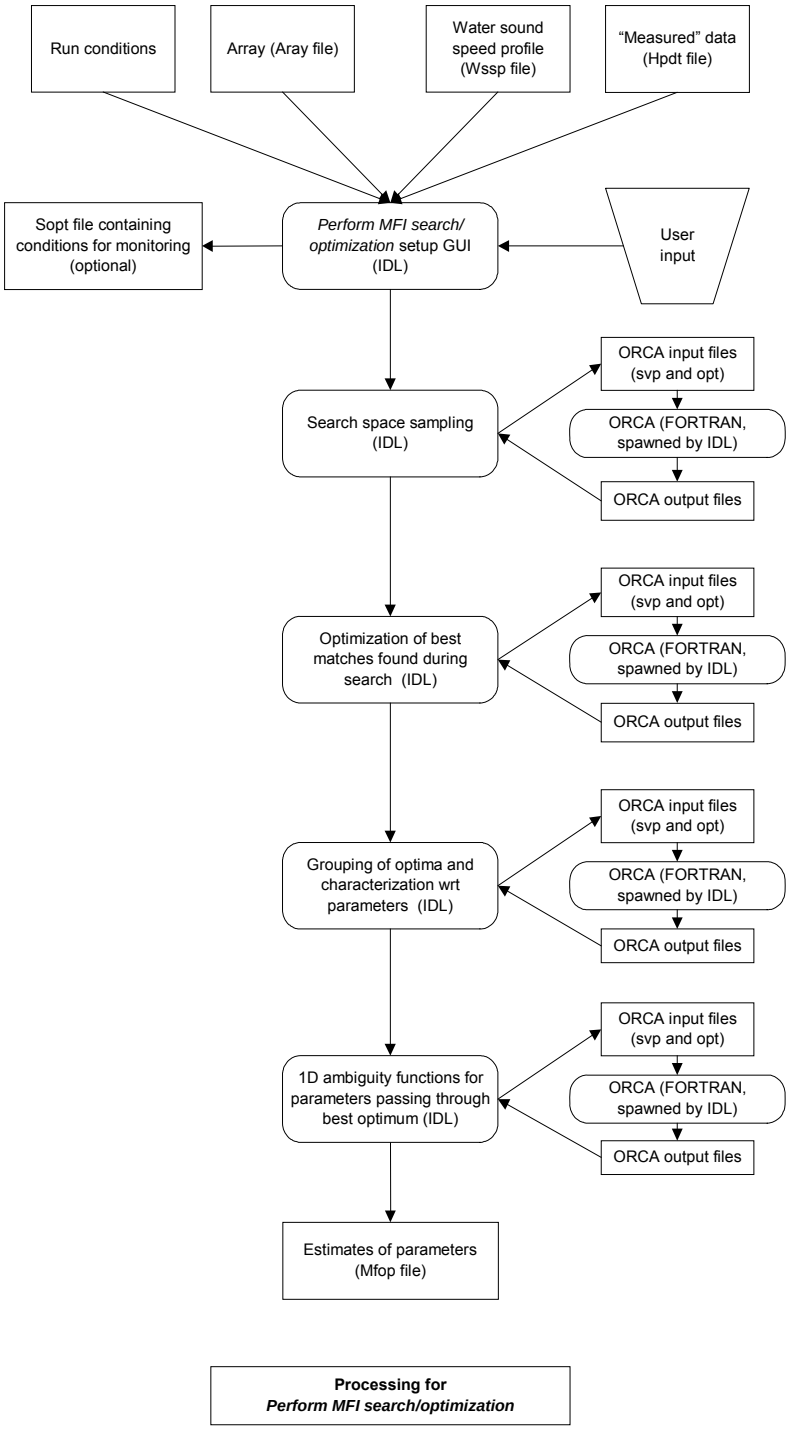
The IDL routine DFPMIN (a quasi-Newton method for which the required derivatives can be approximated numerically by central differences) is used to perform the optimizations. The result is a set of independent estimates of optimized parameter values.

***A grouping stage***. In this stage, the optima are analyzed and multiple estimates of the same optimum are grouped together. This is done by starting with the optimum with the lowest objective function value and determining other optima with sufficiently similar values for the function value. The objective function is evaluated along a line in the parameter space connecting the optima and they are grouped if the objective function along this line does not exceed a specified threshold.

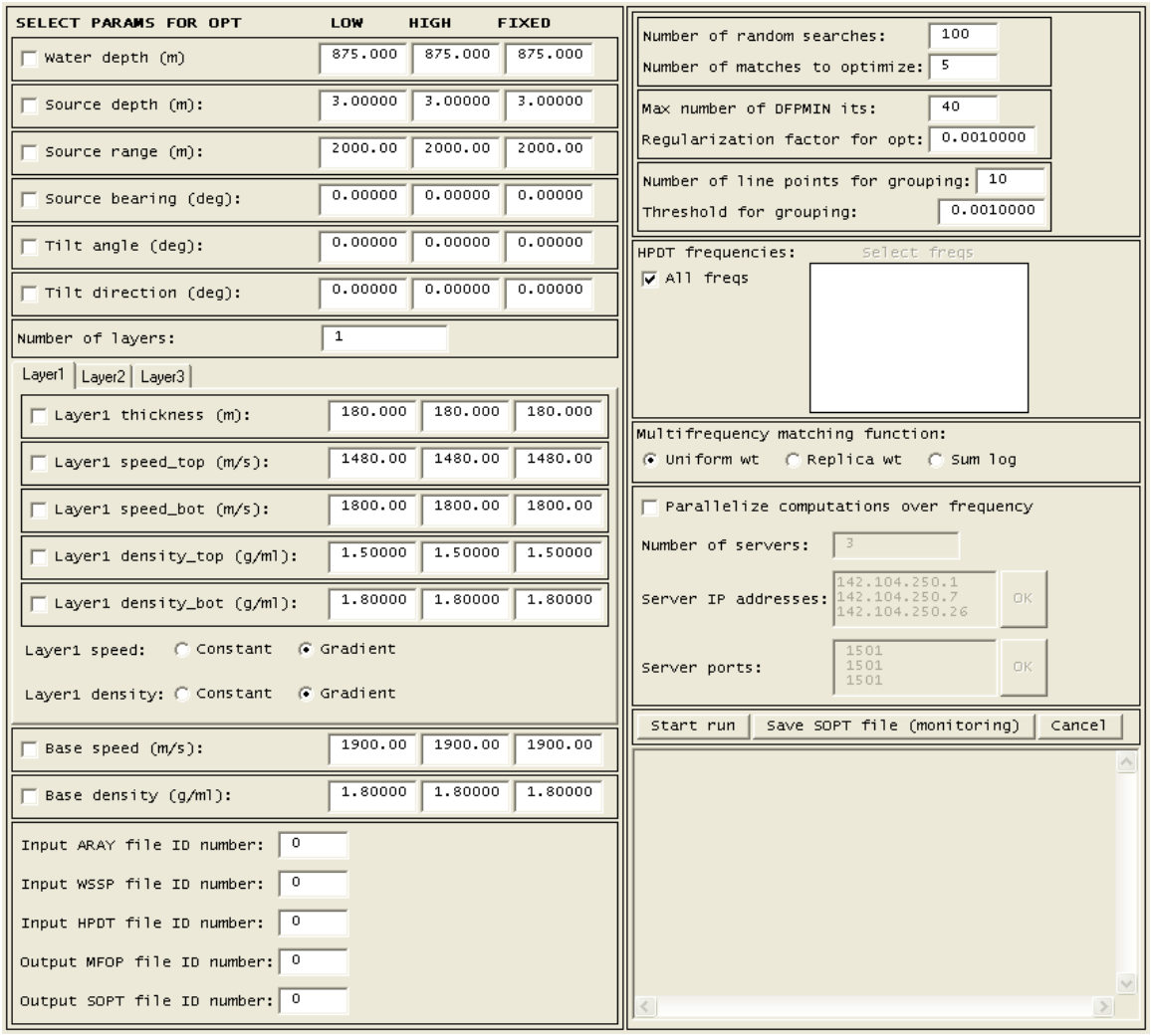
***A parameter characterization stage***. In this stage, the best optimum is identified and the objective function is evaluated independently for each varied parameter along a line spanning the region between its lower and upper bounds and intersecting the peak. This provides an estimate of the sensitivities of the individual parameters, and an indication of the peak width, dynamic range, and oscillations of the function with respect to each parameter.

# Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.




# Graphical User Interface




- The block on the top left (including tabs) lists the 23 parameters that can be chosen for optimization and provides bounds and default fixed values for those parameters. The checkboxes are used to select those parameters which are to be optimized. The lower and upper bounds for selected parameters can then be set using the "LOW" and "HIGH" fields for the parameter (in which case the "FIXED" value is ignored). Note that the "Layer*n* speed" and "Layer*n* density" buttons under the tabs provide an option for the user to force the indicated parameter pairs for the top and bottom of the corresponding layer *n* to be the same.
- The lower block on the left allows the user to specify the input files (Aray, Wssp, and Hpdt) to be used for the run, and the Mfop file (and optionally the Sopt file) to contain the results. The Sopt file is output when the "Save SOPT file (monitoring)" option is chosen, and will contain the current values of the Params and Mfop data structures; these are used to specify the search/optimization conditions for monitoring (see below).

- The top block on the right allows the user to specify the conditions for the search stage, i.e., the number of random samples of the parameter space and the number of best matches to optimize.
- The second block on the right allows specification of convergence and regularization factors for the optimization.
- The third block on the right allows the user to specify conditions for the grouping analysis of the multiple peaks.
- The fourth block on the right allows the user to view the frequencies of the data in the Hpdt file and select a subset of these to use in computing the multi-frequency ambiguity function. An option to select all the frequencies in the data is provided.
- The fifth block on the right provides options for how the matches at the multiple frequencies are to be combined in the overall match. For the uniform and replica options, the Bartlett outputs at the frequencies are weighted uniformly or according to the squared modulus of the replica vector, respectively. For the Sum log option, the expression $\sum_{j=1}^{N} \log(1 - B(\mathbf{r}_j(\mathbf{p}), \mathbf{m}_j))$ (or $\sum_{j=1}^{N} \log(1 - B(\mathbf{r}_j(\mathbf{p}), \mathbf{M}_j))$ for matrix data) is used.
- The sixth block on the right allows for parallel processing over frequency, with the server IP addresses and ports of the remote computers specified in the text boxes.
- The "Start Run" and "Cancel" buttons are self-explanatory. The "Save SOPT file (monitoring)" button saves the current conditions for use in monitoring.
- The large empty area at the bottom right provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used to perform matched-field inversion using the search/gradient optimization method is outlined in the following pseudocode:

*Specify the input Aray and Wssp files and the output Hpdt file, and specify the run conditions, including those parameters that are to be optimized and their bounds.*
*Check that the run conditions are consistent.*
*Generate specified number of random samples of the search space and rank the results in order of the best matches.*
*For each of a specified number of the best matches:*
    *Using the current best match as an initial estimate, call the optimization function to optimize the parameters.*
*Identify those converged optima which are estimates of the same peak and group them together to form unique estimates.*
*Generate 1D ambiguity functions passing through the best optimum for each parameter, and estimate the characteristics of the parameters (number of peaks, peak width, etc.).*
*Write results to an Mfop file.*

The algorithm for computing the objective function, given a set of parameters provided by the optimization algorithm, is as follows:

*For each frequency selected (optionally parallelized):*
*Apply tilt values to the array positions.*
*Generate an array geometry file for ORCA based on the input parameters.*
*Generate an opt file for ORCA based on the input parameters.*
*Generate an svp file for ORCA based on the input parameters .*
*Spawn an ORCA process to generate a replica vector.*
*Compute the Bartlett power of the match for that frequency.*
*Combine the weighted Bartlett powers for the frequencies to give the ambiguity function for the input parameters.*
*Apply penalty function for parameters outside the bounds, and add a (small) regularization function term which increases quadratically with distance from the midpoint of the parameter range.*

## Using the Component

To perform MFI using the search/optimization procedure, press the "Perform MFI search/optimization" button on the Main Menu to bring up the GUI. Then do the following:

- Select the parameters to optimize, by clicking on the checkboxes beside the names in the left column.
- Specify the lower and upper bounds for each parameter to be varied in the corresponding text fields (under columns LOW and HIGH) for that parameter. To change the number of layers (which can be from 0 up to 3), specify this number in the "Number of layers" field. To access the parameters in a layer other than the present layer, click on one of the tabs: Layer1, Layer2, Layer3.
- Specify the values to use for the fixed parameters in the corresponding fields for those parameters (under column FIXED).
- Ensure that radio buttons are selected to enable or disable gradients for sound speed and density in the corresponding layer.
- Specify ID numbers for the input Aray, Wssp, and Hpdt files, and the output Mfop file to contain the results of the run.
- Specify the number of random samples to be generated during the global search stage, and the number of best matches found during this stage to optimize in the next stage.
- Specify the maximum number of iterations for the IDL optimization algorithm DFPMIN, and the regularization scaling factor for the quadratic function used in the regularization.
- Specify the number of points along a line connecting two possibly different optima when performing the grouping, and the threshold for the grouping.
- Specify the frequencies in the Hpdt file to use in the matching. Checking the "All freqs" checkbox uses all the frequencies in the file. Unchecking the box enables the option to select one or more of the individual frequencies in that file. Note that the

Hpdt file with the specified ID number must exist in order to be able to use this option.

- Specify the option to use for combining the frequencies in the final matching function.
- If the computations are to be run in parallel over frequency, check the "Parallelize computations over frequency" checkbox and ensure that the numbers of servers, the IP addresses and the ports specified in the text boxes are correct. If you edit these fields, note that you must press the respective "OK" button for the changes to take effect. (If you wish to change the default values, edit the file "servers.dat" in the \source directory.) Also ensure that orcaserver processes are running on each of the computers selected (see the Installing the BCOMFI Software section above for information on installing and running these processes on Windows and Linux computers).
- If setting up for monitoring, press the "Save SOPT file (monitoring)" button. The conditions for search/optimization that you have specified will be saved to a Sopt file which can then be used during a monitoring run. Note that you do not have to specify depth and range as parameters to vary as this is automatically done by the monitoring program Mfi monitor (see below).
- Press the "Start run" button. During each stage, graphical information (objective function and parameter values) will be plotted to the screen every 20$^{th}$ function call. This provides a convenient way of monitoring the progress of the search, optimization, and other stages.

The result of the run will be a Mfop file (and optionally a Sopt file); the former can be viewed using the "Display MFI search/optimization results" option on the Main Menu interface.

## Matched-Field Techniques: Monitoring

The aim of monitoring using MFI is to be able to detect large-scale changes in sub-bottom hydrates structures by analyzing acoustic array data obtained using the sound of passing ships. During monitoring, data sets or data streams collected for this array are processed to estimate a sequence of geoacoustic models for the region, and systematic changes in the model parameters estimated by MFI could then be taken as evidence of changes in the hydrates.

As noted in the above section, MFI provides a way to estimate geoacoustic model parameters for an ocean environment, including sub-bottom and water column sound speed profiles. MFI can be used with either impulsive or continuous wave sources. Its performance can be improved by using multiple frequencies and this does not require that these frequencies be coherent. While it can an advantage for MFI if the source location is known, this is not necessarily required, as the location can generally be estimated during the inversion process. Hence, in many respects MFI is very well-suited to the intended application of long-term hydrates monitoring using the sound generated by passing ships.

However, MFI does have some potential limitations in the monitoring application, as noted in the following list:

- MFI is not an imaging method, but rather estimates the values of geometric and geoacoustic parameters through an optimization process. These values are themselves averaged or effective estimates of the parameters over the source-receiver path rather than single estimates at a particular location.
- MFI N-D ambiguity surfaces generally contain multiple optima even when multiple frequencies are used, and so the results of MFI are often ambiguous or non-unique.
- The vertical array to be used initially in the monitoring application spans only one quarter of the water column, which results in suboptimal sampling of the modes (and increased ambiguity in MFI).
- Low impedance contrast profiles are difficult to estimate by MFI – this is likely to be the case in the region of the monitoring station and hydrates-bearing region of interest.
- MFI requires sufficient source-receiver separation to provide significant bottom interaction in order to obtain reasonable estimates. However, as the source-receiver distance increases, the SNR decreases and the effects of range dependence increase, both effects degrading the performance of MFI.
- MFI processing is very time-intensive, as large numbers of function evaluations are required during optimization. This can be mitigated somewhat by precomputation, but at the expense of restricting the number of parameters (since each corresponds to a separate dimension of an N-D grid of replica vectors).
- Range-independent models are less computationally expensive but are not applicable to environments with significant range dependence.
- Range-dependent models are applicable more generally but are more time-consuming to compute and also result in more parameters to estimate.
- A particular choice of which model parameters to optimize must be made, and this choice will affect the results obtained.
- MFI can be sensitive to errors in the sound speed profile of the water column as frequency increases; estimation of this profile can be done in principle but would further increase the number of parameters in the model, and the computation times.

In the light of the above considerations, the approach we have taken to implementing monitoring based on MFI is intended to provide relevant information about the sub-bottom while balancing model complexity with computational tractability. At this stage, a single geoacoustic model is assumed to apply to the region as a whole, and the parameters of this model are estimated in a two-stage process:

1. Replica vectors, precomputed for a standard geoacoustic model for the region, are used to compute a 2D ambiguity surface for the geometric parameters water-depth and range, and the best match in this surface is identified.
2. A window is defined around the optimum water-depth and range, and MFI is performed on these two parameters (using this window as the lower and upper bounds) along with other selected geoacoustic parameters such as layer thickness

and sound speed at the top and bottom of the layer. For efficiency, only a single optimization is performed.

This process is repeated for each data set computed from the input data stream, to generate a sequence of estimates for the sub-bottom parameters. The geoacoustic model is specified by the user, and can be as simple or as complex as desired (though simplicity would no doubt be preferred at this stage).

Precomputation allows the large water-depth/range subspace to be rapidly searched, since no real-time model generation is required. In particular, the range subspace can be decreased from several thousands of meters to 100 m or so, thereby greatly reducing the search region for the optimization stage by a factor of 20 – 50. Precomputation could, in principle, be extended to include additional parameters, but the resulting increase in the numbers of dimensions would rapidly render it prohibitive both in terms of computational resources and storage space. Since experimentation indicated that estimation of the geometric parameters source range and water depth could be done with approximate estimates of the geoacoustic parameters, we selected these parameters as being suitable for precomputation of the replica vectors.

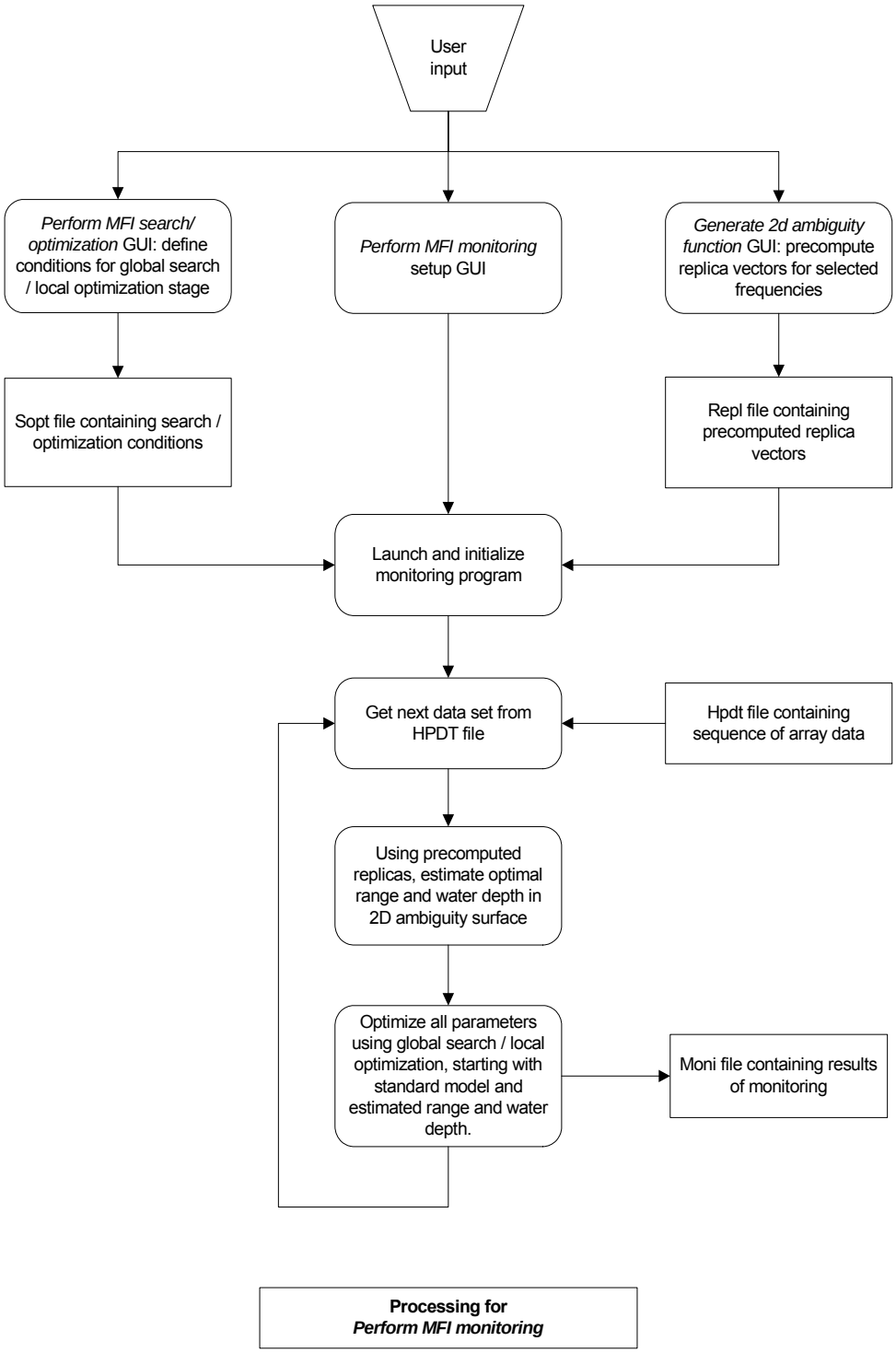
The approach to applying MFI to monitoring uses several other components of BCOMFI during the setup and the actual processing, including those which:

- generate the synthetic data (“Simulate ship FD HPDT data”);
- generate replica vectors (“Generate 2D ambiguity function”);
- define the conditions for and actually perform the search/optimization (“Perform MFI search/optimization)”.

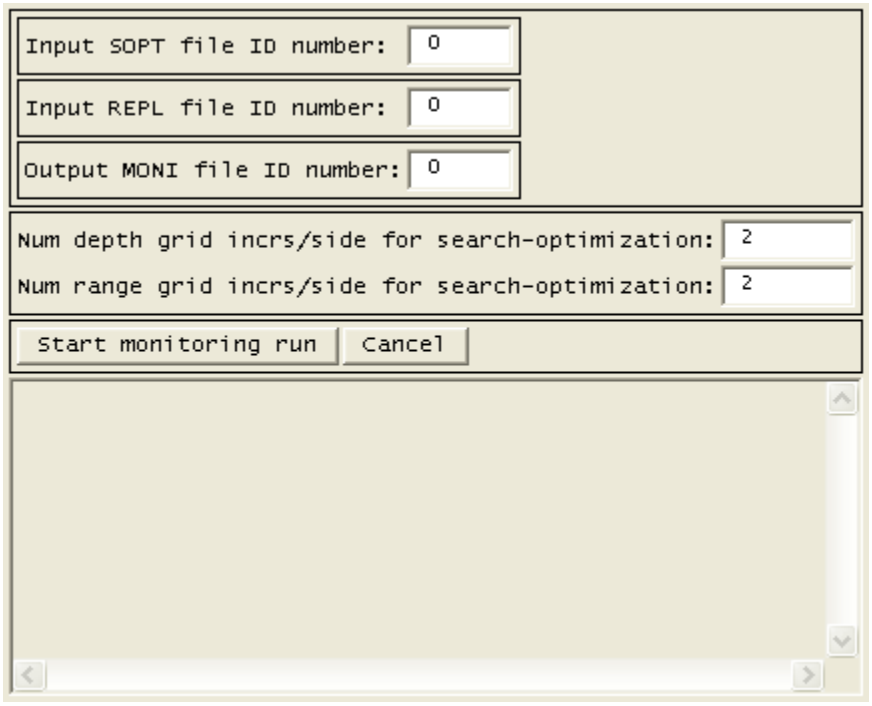
The processes for setting up and performing the MFI monitoring are described in more detail in the remainder of this section.

# Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface




- The top block allows the user to specify the input Sopt and Repl files to be used for the run, and the Moni file to contain the results.
- The second block allows the user to specify the numbers of grid increments in the 2D depth-range ambiguity function to be used to define the bounds for depth and range in the search/optimization stage of monitoring.
- The “Start monitoring run” and “Cancel” buttons are self-explanatory.
- The large empty area at the bottom right provides a field for status messages to be displayed.

## Algorithm

The algorithm used to perform matched-field monitoring is outlined in the following pseudocode:

*Specify the input Sopt and Repl files and the output Moni file, and the numbers of depth and range increments to be used bracketing the search region for these parameters.*
*Check that the run conditions are consistent.*
*Input the precomputed replica vectors from the Repl file and the search/optimization conditions from the Sopt file.*
*For each data set in the monitoring input stream:*
*Generate a 2D depth-range ambiguity function using the replica data in the Repl file, and using the frequencies specified in the Sopt file.*

*Determine the position of the maximum in the ambiguity surface and use this to define the bounds for the parameters water depth and range, based on the grid increments specified above.*

*Using the conditions specified in the Sopt file, perform a search/optimization run, which will yield a single estimate of the chosen parameters for that data set.*

*Write results to a Moni file.*

## Using the Component

To perform MFI monitoring, first generate the simulated data for the monitoring run, by pressing the "Simulate ship FD hpdt data" button and performing a run for the desired conditions.

Next you must generate a set of replica vectors to be used in the depth-range estimation stage of the computations. This is done by pressing the "Generate 2D ambiguity function" button on the Main Menu, choosing water depth as parameter 1 and source range as parameter 2, specifying the lower and upper bounds and the frequencies to use, making any other desired adjustments, and then performing the run. This will produce a Repl file containing the replica vectors.

Then you should generate a Sopt file containing the conditions (i.e., the Params and Mfop structures) to be used in the search/optimization stage of the computations. This is done by pressing the "Perform MFI search/optimization" button in the Main Menu and selecting the conditions to be used in this stage. Note that you do not have to specify that depth and range are to be optimized, as this is done automatically by the monitoring software. In addition, note that this software also always sets the number of matches to optimize to one, no matter what its setting in the search/optimization GUI. You should specify any other parameters to be optimized, and set their bounds, as well as the values for the fixed parameters. When you have specified the desired conditions, press the "Save SOPT file (monitoring)" to save the conditions to a Sopt file.

Once you have prepared the above files, press the "Generate 2D ambiguity function" button to bring up the GUI for monitoring. Then do the following:

- Specify ID numbers for the input Repl and Sopt files, and the output Moni file to contain the results of the run.
- Specify the numbers of grid increments in the 2D depth-range ambiguity function to be used to define the bounds for depth and range in the search/optimization stage of monitoring.
- Press the "Start Monitoring Run" button. After the 2D ambiguity function generation stage, a plot of this function will be displayed. Then, during the search/optimization stage, graphical information (objective function and parameter values) will be plotted to the screen every $20^{th}$ function call. This provides a convenient way of monitoring the progress of the various stages.

The result of the run will be a Moni file, which can be viewed using the "Display MFI

monitoring results" option on the Main Menu interface.

## Matched Field Techniques: Parallelization

Matched-field methods are computationally intensive, and to reduce the times involved it is necessary to develop implementations of these techniques where the computations are parallelized in some way. Parallelization is advantageous both for the computation of ambiguity functions (particularly in the case of higher dimensions) and especially for MFI runs, which can involve many thousands of function evaluations.

Since the basic unit of computation here is a run of ORCA at a single frequency, a parallelization option for BCOMFI was developed in which the computations are distributed among processors according to frequency. This approach also allows convenient separation of client and server processes, with IDL code running on the client and C/Fortran code running on the server. The parallelization was implemented as follows:

- An IDL client sockets routine was written that used the IP addresses of a number of servers (on single or multi-processor computers) and an array containing the names of the input and output files for ORCA at each of a number of frequencies.
- An ORCA executable module was produced and installed on each of a number of server processors.
- A C server routine was written that accepted a set of files for a single frequency from a client, performed an ORCA run using these files, and transferred the output file containing complex field values back to the client. Note that the server processes should not be run on the machine that is running the client.
- The IDL code for the matched field computations was modified to provide an option for parallelized multi-frequency processing. In this implementation, the input files for ORCA at all these frequencies are generated, and the IDL client sockets routine is called. This routine controls the transfer of the files to the servers and handles the receiving of the resulting output files. It continually assigns tasks to servers as they become available, until the fields at all selected frequencies have been computed. The IDL code then inputs the complex field data and performs the matching.

The implementation was designed to be general, in that it allowed distribution of N tasks (i.e., runs at a single frequency) among M processors. The servers were implemented in both Windows and Linux, with ORCA executables compiled for both environments.

## BCOMFI Display Components

The BCOMFI display functions allow the viewing of the simulated shot and ship traces, the ambiguity functions, and the results of MFI search/optimization. Certain of these applications (e.g., plots of shot and ship traces) have been implemented using IDL direct graphics, while others make use of the IDL "itools" interfaces (which include iplot,

isurface, iimage and ivolume). These interfaces provide useful functionality that is not available in direct graphics (e.g., copy and paste), at the cost of speed and memory requirements (for which reason it was not used in plotting traces from Shot and Ship files). Components of the itools interfaces are used in displaying the 1D, 2D and 3D ambiguity functions, with an additional special-purpose IDL slicer tool being provided for 3D volume visualization, while direct graphics are used in display of the search/optimization results. (Note that for the latter, the screen resolution should be set to at least 1280 by 1024 to view the full extent of some of the plots.) The itools interfaces are designed for a high level of user interaction, and some effort should be made to become familiar with their functionality (a tutorial is available at the RSI website http://www.rsinc.com/idl/idl_itools.asp).

## Displaying Time Domain Shot Data Traces

Press the "Display shot data" button to plot the traces in a particular Shot file. A file browser dialog box will appear, which will display all files of the type "shotnnnnn.sav", where nnnnn is an ID number. After you select the desired file, the traces in that file, and their corresponding amplitude spectra, will be displayed in separate windows as waterfall plots.

## Displaying Time Domain Ship Data Traces

Press the "Display ship data" button to plot the data streams (actually, long traces) in a particular Ship file. A file browser dialog box will appear, which will display all files of the type "shipnnnnn.sav", where nnnnn is an ID number. After you select the desired file, the data streams in that file, and their corresponding amplitude spectra, will be displayed in separate windows as waterfall plots.

## Displaying 1D Ambiguity Functions

Press the "Display 1D ambiguity function" button to plot a 1D ambiguity function. A file browser dialog box will appear, which will display all files of the type "am1dnnnnn.sav", where nnnnn is an ID number. After you select the desired file, there may be a momentary delay while itools loads, and an itools splash screen may appear and vanish. The plot will then be displayed in an iplot window.

## Displaying 2D Ambiguity Functions

Press the "Display 2D ambiguity function" button to plot a 2D ambiguity function. A file browser dialog box will appear, which will display all files of the type "am2dnnnnn.sav", where nnnnn is an ID number. After you select the desired file, a setup GUI will then appear, in which you can specify the conditions for the display. First, choose whether you want the display as a surface (which can be rotated in 3D) or as an image. You then have the option of resizing the 2D function before display (which you will usually want to do if the Image option is chosen). If you have chosen the Surface option, you can then specify the shading algorithm to be used (either flat or Gouraud). If you have

chosen the Resize option, you can also specify the interpolation scheme (bilinear or cubic).

Once you have specified the conditions, press the Go button. There may be a momentary delay while itools loads, and an itools splash screen may appear and vanish. If the Surface option was chosen, the ambiguity function will presently be displayed in an isurface window. You can use the Rotate button on this window to rotate and examine the surface in 3D. If the Image option was chosen, the function will be displayed as an iimage window. Note that this plot does not display and label the axes, but these may be added using options in the itools interface.

## Displaying 3D Ambiguity Functions

Press the "Display 3D ambiguity function" button to plot a 3D ambiguity function. A file browser dialog box will appear, which will display all files of the type "am3dnnnnn.sav", where nnnnn is an ID number. After you select the desired file, there may be a momentary delay while itools loads, and an itools splash screen may appear and vanish. Then two separate interfaces will be displayed.

One of these is the ivolume tool, which displays a bounding box with labeled axes, and allows rendering using the button on the right side, and image plane and isosurface generation using the Operations→Volume menu item. Note that you may sometimes have to invoke the Edit→Select All menu item to enable this functionality. You can use the Rotate button on this window to rotate and examine the isosurface in 3D.

The second interface is the 3D Data Visualizer (Slicer3) interface, which provides a different interface for slice, isosurface, and projection operations through its Mode droplist. Note that you may need to use the Tools→Erase menu item to clear the display area of existing images before displaying a new one. Also note that while the ivolume interface has the capability of copying and pasting the display, the Slicer3 interface does not, and so to save a Slicer3 image, a screen dump utility would have to be used.

## Displaying Results of MFI Search Optimization

Press the "Display MFI search/optimization results" button to show several different types of results from an MFI search/optimization run. A file browser dialog box will appear, which will display all files of the type "mfopnnnnn.sav", where nnnnn is an ID number. After you select the desired file, a Setup GUI will then appear, from which you may select the following options for display:

***Search***. This option plots the successive values of mismatch during the random global search stage. Note that, since mismatch is plotted, the best matches are those with the *lowest* values.

***Optimization convergence***. This option plots the evolution of the objective function value and the corresponding parameters during the course of an optimization. It is used in conjunction with the droplist at the bottom of the GUI, which specifies which one of the *M* best matches found during the search stage is to be viewed. (For example, if 0 is chosen, the optimization course for the best match is shown; if 1 is chosen, the course for the second best match is chosen, and so on.) The plots are displayed as objective function, or parameter value, versus the number of function calls.

***1D ambiguity functions at optimum***. This option plots the Bartlett matches obtained as each separate parameter is varied from its lower to its upper bound along a line passing though the "best" optimum (i.e., that with the lowest objective function value). It provides an indication of the sensitivity and ambiguity associated with each parameter.

***Show groups found***. This displays a text window containing information about the conditions for the run and the results of grouping the various optima found. Each group is displayed separately, along with the number of optima (convergences) in that group, the mean function value and parameter values for the group, and, where possible, the standard deviations of the multiple estimates for that group. The results of the peak analysis for the 1D ambiguity functions computed for the best group are then listed separately for each parameter. These include the number of peaks and extrema in the function, the dynamic range (on a scale from 0 to 1), the peak width in both relative units (where 1.0 represents the range between lower and upper bounds) and absolute (physical) units, and the sensitivity (i.e., the RMS variation of the function value between adjacent points along the line).

***Print groups found***. This sends the information displayed in the above text window to the default printer.

When you have selected one of the above options, press the Go button. The program will then perform the specified action.

## Displaying Results of MFI Monitoring

Press the "Display MFI monitoring results" button to show the results from an MFI monitoring run. A file browser dialog box will appear, which will display all files of the type "moninnnnn.sav", where nnnnn is an ID number. After you select the desired file, several plots will then be displayed in an IDL graphics window. The top left plot contains the values of the optimized objective function (mismatch) for each of the time periods during the monitoring run, while the other plots contain the optimized estimates for the selected parameters for each of the time periods. Note that the range plot can provide an indication of the consistency of the source position estimation, and may help in the interpretation of the other parameter estimation results.

## Computing Environment

The BCOMFI development was done on a Windows XP computer running the IDL Development Environment (IDL DE), and on which executables of the Fortran programs ORCA and RAM were prepared and installed.

The BCOMFI software delivered in the zipfile "bcomfi.sav" requires a Windows computer on which the IDL VM has been installed. The BCOMFI executable runs under the IDL VM and calls the ORCA and RAM executables provided with the delivery.

The parallelized option for matched field computations is designed for the client process to run on a Windows computer, but to make use of ORCA servers running on either Windows or Linux computers to perform the propagation modeling for the matched field computations. (Note that RAM is currently not involved in the matched field components.) Hence, this parallelized option can be run with a client Windows computer containing the IDL VM and a network of server computers – either Windows or Linux – on which ORCA server processes are running. Note that the server processes should not be run on the machine that is running the client.

Hence, the computing environment recommended for using BCOMFI is:

- a Windows computer on which the IDL VM is installed, which allows the simulation, conversion, preprocessing, matched field, and display components of BCOMFI to be run;
- optionally, a network of Windows and Linux computers with ORCA servers installed and running, which allows the matched field components (ambiguity function generation and MFI) to be efficiently run.

## Appendix A: Data Structures

We note here that the BCOMFI data structures described in this section also correspond to files, such that a file with a particular five-character prefix (e.g., aray00023.dat) would hold a data structure of the same name (an Aray structure).

### Am2d

The Am2d structure is used as an example of structures used in ambiguity function generation (the Am1d and Am3d structures are analogous). The structure contains the following tags:

- ***param1***. The first parameter to vary.
- ***init1***. The lower limit of the domain to vary param1.
- ***final1***. The upper limit of the domain to vary param1.

- ***incr1***. The increment for varying param1.
- ***num1***. The number of points for varying param1.
- ***param2***. The second parameter to vary (for 2D only).
- ***init2***. The lower limit of the domain to vary param2 (for 2D only).
- ***final2***. The upper limit of the domain to vary param2 (for 2D only).
- ***incr2***. The increment for varying param2 (for 2D only).
- ***num2***. The number of points for varying param2 (for 2D only).
- ***layer1_speed_constant***. A flag indicating whether the compressional speed in layer 1 is to be held constant.
- ***layer1_density_constant***. A flag indicating whether the density in layer 1 is to be held constant.
- ***layer2_speed_constant***. A flag indicating whether the compressional speed in layer 2 is to be held constant.
- ***layer2_density_constant***. A flag indicating whether the density in layer 2 is to be held constant.
- ***layer3_speed_constant***. A flag indicating whether the compressional speed in layer 3 is to be held constant.
- ***layer3_density_constant***. A flag indicating whether the density in layer 3 is to be held constant.
- ***aray_id***. The ID number for the input Aray file containing the array geometry.
- ***wssp_id***. The ID number for the input Wssp file containing the water column sound speed profile.
- ***hpdt_id***. The ID number for the input Hpdt file containing the data for MFI.
- ***am2d_id***. The ID number for the output Am2d file to contain the ambiguity function.
- ***all_freq***. A flag indicating whether all the frequencies in the Hpdt file are to be used for matching.
- ***num_freq***. The number of frequencies in the Hpdt file that are to be used for matching.
- ***freq***. The frequencies in the Hpdt file that are to be used for matching.
- ***use_freq***. A flag indicating which of the frequencies in the Hpdt file (in order) are to be used for matching.
- ***match_fun***. A string indicating the scheme for combining the results at multiple frequencies.
- ***matches***. A 3D array containing the individual matches at each frequency and pair of parameter values.
- ***amb_fun_incoh***. A 2D array containing the ambiguity function computed using the standard Bartlett processor (i.e., incoherent with frequency).

## Aray

The Aray structure holds the data for the array, and contains the following tags:

- ***description***. A text description of the array type of characteristics.
- ***num_sens***. The number of sensors in the array.
- ***tilt_angle***. The tilt angle from vertical (degrees).

- ***tilt_direction***. The direction in which the array is tilted (degrees true).
- ***sens_x***. The *x*-coordinates of the sensors (before tilt is applied).
- ***sens_y***. The *y*-coordinates of the sensors (before tilt is applied).
- ***sens_z***. The *z*-coordinates of the sensors (before tilt is applied).
- ***tether_z***. The tether depths of the sensors (allowing the application of tilt).

## Hpdt

The Hpdt structure holds the simulated and real frequency domain data to be used for input to MFI, and contains the following tags:

- ***hpdt_id***. The ID number for the Hpdt file containing the data for MFI.
- ***aray_id***. The ID number for the Aray file containing the array geometry.
- ***num_sens***. The number of sensors in the array from which the data were obtained.
- ***prop_model***. A string containing the propagation model to be used ('ram' or 'orca').
- ***slope***. The bottom slope in degrees, if RAM is to be used (with positive slopes the depth increases from the array to the source).
- ***num_freq***. The number of frequencies at which data are present.
- ***freq***. A vector containing the actual frequencies at which data are present.
- ***data_type***. A string with a value of either "vector" or "matrix".
- ***seed***. A seed for the random number generator (used for noise and phase randomization).
- ***num_seg_int***. The number of segments in one integration time.
- ***num_int_tot***. The total number of integration times.
- ***seg_time***. The time for a single data segment.
- ***int_time***. The integration time for the data.
- ***tot_time***. The total time for the full set of data.
- ***wn_level_db***. The white noise level at a sensor.
- ***sn_level_db***. The spherical noise level at a sensor.
- ***cn_level_db***. The cylindrical noise level at a sensor.
- ***num_source***. The number of sources (max 2).
- ***source_level_db***. The source intensity levels (dB re 1 μPa at 1 m).
- ***source_depth***. The depth(s) of the source(s).
- ***source_range***. The range(s) of the source(s).
- ***source_bearing***. The bearing(s) of the source(s).
- ***source_speed***. The speed(s) of the source(s).
- ***source_heading***. The heading(s) of the source(s)
- ***fdata***. A multi-dimensional array containing the complex fields for the sensors, frequencies, and times.

## Mfop

The Mfop structure holds the conditions for, and the results of, an MFI run; it is made up of six other structures, as follows:

## Ctrl

The Ctrl structure holds the basic conditions for an MFI run, and contains the following tags:

- ***layer1_speed_constant***. A flag indicating whether the compressional speed in layer 1 is to be held constant.
- ***layer1_density_constant***. A flag indicating whether the density in layer 1 is to be held constant.
- ***layer2_speed_constant***. A flag indicating whether the compressional speed in layer 2 is to be held constant.
- ***layer2_density_constant***. A flag indicating whether the density in layer 2 is to be held constant.
- ***layer3_speed_constant***. A flag indicating whether the compressional speed in layer 3 is to be held constant.
- ***layer3_density_constant***. A flag indicating whether the density in layer 3 is to be held constant.
- ***aray_id***. The ID number for the input Aray file containing the array geometry.
- ***wssp_id***. The ID number for the input Wssp file containing the water column sound speed profile.
- ***hpdt_id***. The ID number for the input Hpdt file containing the data for MFI.
- ***mfop_id***. The ID number for the output Mfop file to contain the results of the MFI.
- ***all_freq***. A flag indicating whether all the frequencies in the Hpdt file are to be used for matching.
- ***num_freq***. The number of frequencies in the Hpdt file that are to be used for matching.
- ***freq***. The frequencies in the Hpdt file that are to be used for matching.
- ***use_freq***. A flag indicating which of the frequencies in the Hpdt file (in order) are to be used for matching.
- ***match_fun***. A string indicating the scheme for combining the results at multiple frequencies.
- ***num_search***. The number of random samples of parameters to use in the search stage of the algorithm.
- ***num_best***. The number of best matches from the search stage that are to be optimized during the optimization stage of the MFI.
- ***num_its***. The maximum number of iterations for the DFPMIN (Davidon-Fletcher-Powell) optimization algorithm to perform.
- ***reg_factor***. The regularization factor to apply in computing the objective function.
- ***num_inter***. The number of intermediate points on a hyperspace line between two optima along which to evaluate the objective function in order to detect whether the optima are estimates of the same peak and should be grouped.
- ***thresh_group***. The threshold used to determine whether two optima (minima) are estimates of the same peak. If the objective functions at the two optima differ by this amount or more, or if any point on the above line connecting the optima is

more than this amount greater than the optimum value at the endpoints, the optima are taken to be different and are not grouped.

- ***num_params***. The total number of parameters in the overall model here, 23).

## Opt

Opt is a vector of structures characterizing how each parameter is to be involved in the optimization and giving information about the widgets for the GUI. Each structure of Opt contains the following tags:

- ***wid_yesno***. The ID of the IDL widget for the checkbox in the GUI.
- ***wid_lower***. The ID for the IDL widget for the lower bound field in the GUI.
- ***wid_upper***. The ID for the IDL widget for the upper bound field in the GUI.
- ***wid_fixed***. The ID for the IDL widget for the fixed field in the GUI.
- ***optimize***. An flags indicating whether this parameter is to be optimized.
- ***lower***. The lower bound for this parameter, if it is to be optimized.
- ***upper***. The upper bound for this parameter, if it is to be optimized.
- ***fixed***. The fixed value for this parameter, if it is not to be optimized

## Search

The Search structure contains the results of the global search stage of MFI, and has the following tags:

- ***f***. A vector containing the function values for the searches.
- ***x***. A 2D array containing the parameter values used for the searches.

## Optim

The Optim structure contains the results of all the optimizations done during the optimization stage of MFI, and has the following tags:

- ***num_best***. The number of best matches from the search stage that were optimized.
- ***p_f***. A vector of pointers; the *i*th pointer points to a vector containing the function values that were computed during the course of the *i*th optimization.
- ***p_x***. A vector of pointers; the *i*th pointer points to a 2D array containing the parameter values used during the course of the *i*th optimization.
- ***f_min***. A vector containing the optimized function values for the optimizations.
- ***x_min***. A 2D array containing the parameters corresponding to the optimized function values for the optimizations.

## Groups

The Groups structure contains the results of the grouping analysis and has the following tags:

- ***num_group***. The number of different groups found.
- ***num_in_group***. A vector containing the number of convergences (equivalent optima) in each group.
- ***p_f***. A vector of pointers; the *i*th pointer points to a vector containing the function values for the individual optima in *i*th group.

- ***p_x***. A vector of pointers; the *i*th pointer points to a 2D array containing the parameter values for the individual optima in the *i*th group.
- ***x_mean***. A 2D array containing the means for the parameters in the groups.
- ***x_sd***. A 2D array containing the standard deviations for the parameters in the groups.
- ***x_corr***. A 3D array containing the correlation matrices for the parameters in the groups.
- ***x_prob***. A 3D array containing the significance levels of the correlations in the correlation matrices for the parameters in the groups.

### Peak

The Peak structure contains the results of the 1D ambiguity function for each parameter passing through the best optimum found, and has the following tags:

- ***num_peak***. A vector containing the number of peaks (local maxima) in the function for each parameter.
- ***num_extrema***. A vector containing the number of extrema (local minima and maxima) in the function for each parameter.
- ***dymanic_range***. A vector containing the dynamic range (maximum – minimum) in the function for each parameter.
- ***width***. A vector containing the estimated width of the peak at half height for each parameter.
- ***sens_rms***. A vector containing the estimated sensitivity (RMS difference of the adjacent function points) for each parameter.

## Moni

The Moni structure contains the setup conditions and results for the monitoring run, and contains the following tags:

- ***sopt_id***. The ID number for the input Sopt file containing the conditions for the search/optimization stage of the MFI monitoring.
- ***repl_id***. The ID number for the input Repl file containing the replica vectors for the 2D ambiguity function stage of the MFI monitoring.
- ***moni_id***. The ID number for the output Moni file containing the conditions and results of the MFI monitoring.
- ***num_depth_grid_incr***. The number of water-depth increments on either side of the maximum in the water-depth/range 2D ambiguity function grid to use for setting the water-depth bounds during the search/optimization stage.
- ***num_range_grid_incr***. The number of range increments on either side of the maximum in the water-depth/range 2D ambiguity function grid to use for setting the range bounds during the search/optimization stage.
- ***data_type***. . A string with a value of either “vector” or “matrix” (obtained from Hpdt file).
- ***num_seg_int***. The number of segments in one integration time (obtained from Hpdt file).

- ***num_int_tot***. The total number of integration times (obtained from Hpdt file).
- ***seg_time***. The time for a single data segment (obtained from Hpdt file).
- ***int_time***. The integration time for the data (obtained from Hpdt file).
- ***tot_time***. The total time for the full set of data (obtained from Hpdt file).
- ***p_mfop***. A vector of pointers to Mfop structures which contain the conditions for, and the results of, the monitoring run.

## Repl

- ***param1***. The first parameter to vary ('water_depth').
- ***init1***. The lower limit of the domain to vary param1.
- ***final1***. The upper limit of the domain to vary param1.
- ***incr1***. The increment for varying param1.
- ***num1***. The number of points for varying param1.
- ***param2***. The second parameter to vary ('source_range').
- ***init2***. The lower limit of the domain to vary param2.
- ***final2***. The upper limit of the domain to vary param2.
- ***incr2***. The increment for varying param2.
- ***num2***. The number of points for varying param2.
- ***freq***. The frequencies in the Hpdt file that are to be used for matching.
- ***vecs***. A 4D (sensor, frequency, water_depth, source_range) complex-valued array containing the replica vectors for the depth/range ambiguity function.

## Ship

The Ship structure holds the traces for a single shot, and contains the following tags:

- ***aray_id***. The ID number for the Aray file containing the array geometry.
- ***shot_id***. The ID number for the Shot file containing the traces for the shots.
- ***ship_id***. The ID number for this Ship file.
- ***num_sens***. The number of sensors in the array.
- ***num_time_pt***. The number of samples in each time series in the structure.
- ***samp_freq***. The sampling frequency for the traces.
- ***source_depth***. The depth of the source.
- ***source_range***. The range of the source.
- ***source_bearing***. The bearing of the source.
- ***tdata***. A 2D array containing the time series for the sensors.

## Shot

The Shot structure holds the traces for a single shot, and contains the following tags:

- ***aray_id***. The ID number for the Aray file containing the array geometry.
- ***wave_id***. The ID number for the Wave file containing the wavelet.
- ***shot_id***. The ID number for the Shot file containing the traces for the shots.

- ***num_sens***. The number of sensors in the array.
- ***samp_freq***. The sampling frequency for the traces.
- ***min_freq***. The minimum frequency in the band for which ORCA broadband should compute a normal mode model (must be positive).
- ***max_freq***. The maximum frequency in the band for which ORCA broadband should compute a normal mode model (must be less than samp_freq).
- ***num_fft***. The number of points in the FFT to be performed on the frequency domain data, and which corresponds to the number of points in the traces (power of 2). Care should be taken to make this large enough to prevent wrap-around artifacts in the traces.
- ***source_depth***. The depth of the source.
- ***source_range***. The range of the source.
- ***source_bearing***. The bearing of the source.
- ***tdata***. A 2D array containing the shot traces for the sensors.

### Wave

The Wave structure holds an acoustic wavelet, and contains the following tags:

- ***num***. The number of elements in the wavelet.
- ***samp_freq***. The sampling frequency for the wavelet points.
- ***wdata***. A vector containing the elements of the wavelet.

### Wssp

The Wssp structure holds a water column sound speed profile, and contains the following tags:

- ***depth***. A vector containing the depths for the profile.
- ***speed***. A vector containing the sound speeds for the profile at the corresponding depths.

## Appendix B: Program Modules

### Simulate TD Shot Data

This component is implemented by the Sim_shot_data.pro module. Its main component functions and procedures are as follows:

- ***Sim_shot_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_sim_shot_data***. Initializes default values for the fields of the GUI.
- ***Check_sim_shot_data***. Checks for existence of files and consistency of the data entered by the user.
- ***Sim_shot_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_sim_shot_data***. Performs a run using the input data specified by the user, i.e., calling ORCA broadband, processing the resulting FFT file to convolve the

impulse response with a wavelet, generating traces for each sensor of the array and adding noise to the traces.

- ***Read_aray***. Reads in the data for the array.
- ***Read_wave***. Reads in the wavelet to convolve with the impulse responses.
- ***Read_orca_fft***. Reads in the ORCA "FFT file" – the frequency domain impulse responses for each sensor of the array.
- ***Gen_array_geom_file***. Uses the data in the Aray file to generate an array geometry file for use by ORCA.
- ***Gen_bb_opt_file***. Uses the data input by the user to generate an options file for ORCA to perform a run using the broadband option.

In addition, an executable ORCA program is required which will input the specified data and run conditions and output the FFT file.

The Sim_shot_data component uses the Aray, Wave, and Shot data structures. A description of these structures and their tags is given in the previous section.

## Simulate Ship FD Hpdt Data

This component is implemented by the Sim_ship_data.pro module. Its component functions and procedures are as follows:

- ***Sim_ship_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_sim_ship_data***. Initializes default values for the fields of the GUI.
- ***Check_sim_ship_data***. Checks for existence of files and consistency of the data entered by the user.
- ***Sim_ship_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_sim_ship_data***. Performs a run using the input data specified by the user, i.e., calling ORCA or RAM, adding noise, performing cross-spectral matrix estimation if specified, and writing the data to an Hpdt file.
- ***Read_aray***. Reads in the data for the array.
- ***Gen_sim_orca_sv***. Sets up files for an ORCA run and then calls ORCA to simulate a signal vector at an array for a particular source, array, and frequency.
- ***Gen_array_geom_file***. Uses the data in the Aray file to generate an array geometry file for use by ORCA.
- ***Gen_cw_opt_file***. Generates an options file for the ORCA run in cw mode.
- ***Gen_ram_sv***. Sets up files for a RAM run and then calls RAM to simulate a signal vector at an array for a particular source, array, and frequency.
- ***Gen_ramgeo_file***. Generates an input data file for RAM.
- ***Read_ram_grid***. Reads in a grid of field values computed by RAM.
- ***Gen_wn_matrix***. Generates a white noise matrix.
- ***Gen_sn_matrix***. Generates a spherical noise matrix.
- ***Gen_cn_matrix***. Generates a cylindrical noise matrix.
- ***Chol_matrix***. Performs a Cholesky decomposition on the input matrix.

- ***Gen_nv***. Generates an estimated noise vector, based on the Cholesky decomposition.

In addition, executable ORCA and RAM programs are required which will input the specified data and run conditions and output the complex fields to be used as signal vectors.

The Sim_ship_data component uses the Aray and Hpdt data structures. A description of these structures and their tags is given in a later section.

## Convert TD Shot Data to TD Ship Data

This component is implemented by the Shot_to_ship_data.pro module. Its component functions and procedures are as follows:

- ***Shot_to_ship_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_shot_to_ship_data***. Initializes default values for the fields of the GUI.
- ***Check_shot_to_ship_data***. Checks for existence of files and consistency of the data entered by the user.
- ***Shot_to_ship_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_shot_to_ship_data***. Performs a run using the input data specified by the user, i.e., reading in the shot data, generating ship data and writing to a Ship file.

## Convert P16 Data to Shot or Ship Data

This component is implemented by the P16_to_ss_data.pro module. Its component functions and procedures are as follows:

- ***P16_to_ss_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_ p16_to_ss_data***. Initializes default values for the fields of the GUI.
- ***Check_ p16_to_ss_data***. Checks for existence of files and consistency of the data entered by the user.
- ***P16_to_ss_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_ p16_to_ss_data***. Performs a run using the input data specified by the user, i.e., reading in the P16 data, converting to ship data and writing to a Shot or Ship file.

## Preprocess TD Shot Data to FD Hpdt Data

This component is implemented by the Prep_shot_data.pro module. Its component functions and procedures are as follows:

- ***Prep_shot_data***. Sets up the GUI for specification of the files and input conditions.

- ***Set_prep_shot_data***. Initializes default values for the fields of the GUI.
- ***Check_prep_shot_data***. Checks for existence of files and consistency of the data entered by the user.
- ***Prep_shot_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_prep_shot_data***. Performs a run using the input data specified by the user, i.e., reading in and FFTing the shot data, and writing the cross-spectral matrices to an Hpdt file.

## Preprocess TD Ship Data to FD Hpdt Data

This component is implemented by the Prep_ship_data.pro module. Its component functions and procedures are as follows:

- ***Prep_ship_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_prep_ship_data***. Initializes default values for the fields of the GUI.
- ***Check_prep_ship_data***. Checks for existence of files and consistency of the data entered by the user.
- ***Prep_ship_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_prep_ship_data***. Performs a run using the input data specified by the user, i.e., reading in, overlapping, windowing and FFTing the ship data, performing cross-spectral matrix estimation, and writing the cross-spectral matrices to an Hpdt file.

## Generate nD Ambiguity Function

These components are implemented by the modules Mfi_1d_amb_fun.pro, Mfi_2d_amb_fun.pro, and Mfi_3d_amb_fun.pro. The component functions and procedures for the 2D case (the others are analogous) are as follows:

- ***Amb_2d***. Sets up the GUI for specification of the files and input conditions.
- ***Set_params***. Initializes default values for the parameters in the GUI.
- ***Set_amb_2d***. Initializes default values for the non-parameter fields of the GUI.
- ***Check_amb_2d_parms***. Checks for existence of files and consistency of the data entered by the user.
- ***Amb_2d_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_amb_2d***. Performs a run using the input data specified by the user, i.e., sets up the values of the parameters and computes the matches at these values.
- ***Read_aray***. Reads in the data for the array.
- ***Read_wssp***. Reads in the sound speed profile to use for the run.
- ***Read_servers***. Reads in the default values for the IP addresses and ports for the remote orcaservers.
- ***Ip_addr_array_to_string***. Converts a string vector of IP addresses to their numeric 2D array equivalents.

- ***Ip_addr_string_to_array***. Converts a numeric 2D array of IP addresses to their string vector equivalents.
- ***Gen_mfi_orca_sv***. Sets up and performs an ORCA run to generate a signal (replica) vector based on the run conditions.
- ***Gen_array_geom_file***. Uses the data in the Aray file to generate an array geometry file for use by ORCA.
- ***Gen_cw_opt_file***. Uses the data input by the user to generate an options file for ORCA to perform a run using the continuous wave option.
- ***Gen_svp_file***. Generates an svp file for ORCA, based on the parameter values.
- ***Bartlett***. Computes the output of the Bartlett power processor, for either vector or matrix data.

## Perform MFI Search/Optimization

This component is implemented by the Mfi_search_opt.pro module. Its component functions and procedures are as follows:

- ***Mfi_search_opt***. Sets up the GUI for specification of the files and input conditions.
- ***Set_params***. Initializes default values for the parameters in the GUI.
- ***Check_mfi_search_opt_parms***. Checks for existence of files and consistency of the data entered by the user.
- ***Mfi_search_opt_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_mfi_search_opt***. Performs an MFI run using the input data specified by the user, i.e., performs the initial search, optimizes the objective function for multiple starting estimates and performs the grouping and peak analysis.
- ***Read_aray***. Reads in the data for the array.
- ***Read_wssp***. Reads in the sound speed profile to use for the run.
- ***Read_servers***. Reads in the default values for the IP addresses and ports for the remote orcaservers.
- ***Ip_addr_array_to_string***. Converts a string vector of IP addresses to their numeric 2D array equivalents.
- ***Ip_addr_string_to_array***. Converts a numeric 2D array of IP addresses to their string vector equivalents.
- ***Gen_mfop***. Returns an Mfop structure based on the input parameters, which are the component structures of the Mfop structure.
- ***Mfi_func***. Evaluates the objective function to be minimized in MFI. This is a composite function consisting of the sum of the Bartlett processor, a penalty function if any parameter exceeds its bounds, and a regularization term.
- ***Mfi_dfunc***. Estimates the gradient of the objective function using central differences.
- ***Plot_opt***. Produces a plot of the evolving function value and the values of the parameters.
- ***Gen_mfi_orca_sv***. Sets up and performs an ORCA run to generate a signal (replica) vector based on the run conditions.

- ***Gen_array_geom_file***. Uses the data in the Aray file to generate an array geometry file for use by ORCA.
- ***Gen_cw_opt_file***. Uses the data input by the user to generate an options file for ORCA to perform a run using the continuous wave option.
- ***Gen_svp_file***. Generates an svp file for ORCA, based on the parameter values.
- ***Gen_group***. Groups multiple optima corresponding to the same peak.
- ***Analyze_group***. Performs statistical analysis in the multiple parameter estimates for each group.
- ***Analyze_peak***. Performs post-processing on the best optimum to estimate peak width, sensitivity, etc., for each parameter.
- ***Bartlett***. Computes the output of the Bartlett power processor, for either vector or matrix data.

## Perform MFI Monitoring

This component is implemented by the Mfi_monitor.pro module. Its component functions and procedures are as follows:

- ***Mfi_monitor***. Sets up the GUI for specification of the files and input conditions.
- ***Set_mfi_monitor***. Initializes default values for the fields of the GUI.
- ***Check_mfi_monitor***. Checks for existence of files and consistency of the data entered by the user.
- ***Mfi_monitor_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_mfi_monitor***. Performs an MFI monitoring run using the input data specified by the user, i.e., for each data set, estimates the water depth and source range using precomputed replicas, uses thee result to define a multi-parameter search region, and performs search/optimization to estimate the geoacoustic parameters for that data set.
- ***Run_mfi_search_opt***. Performs an MFI search/optimization run using the input data specified by the user, i.e., performs the initial search, optimizes the objective function for multiple starting estimates and performs the grouping and peak analysis.
- ***Bartlett***. Computes the output of the Bartlett power processor, for either vector or matrix data.

DMAS and MFI Software Development for Hydrates Monitoring Observatory: October 2005 – March 2006
Version 1.0

**Barrodale Computing Services Ltd. (BCS)**
**Hut R, McKenzie Ave., University of Victoria**
**Victoria BC V8W 3W2 Canada**
**www.barrodale.com**

**March 31, 2006**

# Table of Contents

# Executive Summary

The Gulf of Mexico Hydrates Research Consortium and the Center for Marine Resources and Environmental Technologies (CMRET) at the University of Mississippi are currently developing a multi-sensor Seafloor Observatory to be installed on the continental slope of the northern Gulf of Mexico. The purpose of this station is to investigate and monitor the hydrocarbon system within the hydrate stability zone of the northern Gulf of Mexico, and to remotely observe changes in the physical and chemical parameters of gas hydrates.

In a previous twelve-month contract with the University of Mississippi, Barrodale Computing Services Ltd. (BCS) performed work to support the database and data processing requirements of the Observatory. The results of these efforts included data and sensor characterization, the design of a preliminary database, and the development of a comprehensive simulation and matched field software system termed BCOMFI, for **B**arrodale **CO**mputing **M**atched **F**ield **I**nversion.

In the present six-month contract, BCS continued its database and matched field inversion (MFI) software development, and completed development and tasks in the following areas, as described in more detail in the body of this report:

***Database design and implementation***

- A decision was made as to which database management system (DBMS) to use for the initial implementation of the Observatory database. The object-relational DBMS PostgreSQL was chosen, as it is open source, free, and provides the necessary power and flexibility.
- A detailed design was formulated for the prototype database, including definition of entities and their relationships, and containing an extensive data dictionary (listed in an appendix to this report).
- A prototype database was implemented in PostgreSQL, using an AJAX (Asynchronous Javascript with XML) interface. This system provides options for viewing a catalog of the data, storing and viewing associated metadata, selecting individual data records and displaying these data in graphical format. This system was loaded with test data from the Consortium and from the MBARI web site, and was made available for demonstration purposes on BCS's web site http://www.barrodale.com/olemiss/web/catalog.php.
- SQL was generated to create the database and this is listed in an appendix.
- Contact was made with personnel at the Texas Advanced Computing Center to obtain information on their database environments and arrange for future collaboration.

***MFI software development***

- The simulation and matched field components of BCOMFI were extended to be used with 3D arrays of hydrophones; these modifications were verified by comparing results using two different acoustic propagation models.
- A two-stage approach was implemented for monitoring using the acoustic data that are expected to be produced by the array. This approach involved the analysis of each of a sequence of data sets by first generating a 2D ambiguity surface for geometric parameters and identifying the optimal values from the largest peak, and then using these in concert with user-selected parameters to estimate a geoacoustic model by search-optimization.
- The BCOMFI software system was extended to allow matched field techniques to be applied when the range-dependent bathymetry for the environment is known. (Previous to this, the software could only be applied to range-independent environments.) Although this extension to range-dependent MFI was not included in the statement of work, BCS proceeded with this task since it is clearly essential to successfully apply MFI in the range-dependent region of the Observatory.
- The User Guide to the BCOMFI software was extended to include descriptions of the above enhancements, and is provided as a section of this report.

***MFI and monitoring options***

- The advantages, as well as several potential limitations, of the current layer-based matched field techniques were identified.
- Enhancements or alternatives to these matched field inversion approaches were identified and are described. These include 3D MFI, MFI using measured data, and techniques based on cross correlation of ambient noise.
- Our research indicates that, so far, MFI techniques appear not to have been applied to shear wave data such as those to be generated by the multicomponent sensors planned for some of the Observatory arrays. An outline of general processing techniques that could be used to process these data during monitoring is presented.

***Consortium support***

- Ian Barrodale and Cedric Zala attended the February 16-17 meeting of the Gulf of Mexico Hydrates Research Consortium at Oxford, Mississippi, and gave a presentation entitled *Database design, software for MFI processing, and some alternative approaches to hydrates monitoring*.

The major deliverables of the present contract are a prototype database which can now be extended and used to house the Observatory data when they become available, and an enhanced acoustic simulation and matched field inversion (and monitoring) software system suitable for use with 3D arrays and range-dependent bathymetry, that can be applied to estimate geoacoustic models from the Observatory data.

# Introduction

The Gulf of Mexico Hydrates Research Consortium and the Center for Marine Resources and Environmental Technology (CMRET) at the University of Mississippi are currently developing a multi-sensor Seafloor Observatory to be installed on the continental slope of the northern Gulf of Mexico. The aim of this station is to monitor and investigate the hydrocarbon system within the hydrate stability zone of the northern Gulf of Mexico, and to remotely observe changes in the physical and chemical parameters of gas hydrates.

Barrodale Computing Services Ltd. (BCS) has been contracted by the University of Mississippi to design and develop data management and processing software for this monitoring station. In six reports written during the earlier stages of this project, BCS has characterized the data to be produced by the station[18], proposed a design for a data management and archiving system for these data[19], formulated a design for a software system for simulating the data to be acquired by the vertical acoustic array of the station and analyzing these data using matched field inversion (MFI) techniques[20], implemented this design[21], provided a User Guide to the initial version of the software[22], and extended the implementation to include prototype matched field monitoring capability[23].

Our focus in the present project was two-fold:

1. to design and implement a prototype object-relational database management system and populate it with sample data, as well as to develop an interface that included a catalog, the ability to handle metadata, and a data plotting capability.
2. to enhance the simulation and matched field software system BCOMFI to use 3D arrays, implement a two-stage monitoring procedure, and determine some enhancements of alternatives to matched field inversion of the acoustic data.

This report describes the design and implementation of a PostgreSQL database along with an AJAX interface to provide convenient user access. It also describes the enhancements to the BCOMFI software, including the development of a capability for range-dependent matched field inversion using the bathymetry defined by a digital elevation model for the site of the Observatory. A User Guide to the enhanced BCOMFI software system is also provided.

[18] "Sensor and Data Characterization for the Gulf of Mexico Hydrates Monitoring Station" (Jan. 31, 2005).
[19] "Data Management Architecture Design for the Gulf of Mexico Hydrates Seafloor Observatory" (Feb. 28, 2005).
[20] "Software Design for Simulation, Matched Field Inversion, and Monitoring for the Gulf of Mexico Hydrates Seafloor Observatory" (April 26, 2005).
[21] Software for Simulation and Matched Field Inversion for the Gulf of Mexico Hydrates Seafloor Observatory (September 30, 2005).
[22] "BCOMFI Software User Guide" (August 12, 2005).
[23] Software for Simulation, Matched Field Inversion and Monitoring for the Gulf of Mexico Hydrates Seafloor Observatory (February 14, 2006).

# Design of Prototype Database

## Introduction

The data generated by the Seafloor Observatory will need to be archived in an appropriately structured data management and archive system (DMAS). In two previous reports for the University of Mississippi[24], BCS laid the framework for this DMAS development by characterizing the sensors and data for the monitoring station, and by designing the basic architecture of such a system. In the present report, we extend this effort by presenting a conceptual design for the DMAS, outlining the required tables and defining their relationships.

In formulating the database design, it was first necessary to update the information on which sensors were currently targeted as array components and to characterize these sensors and their data. Based on the information provided in response to inquiries that we initiated, a description and list of sensor components for each of the various arrays was finalized; this list is presented in Appendix 1. Several sensors that were not included in the list in our original report, and also those for which additional information is now available, were also identified and characterized; this supplementary information is presented in Appendix 2. Additional advice about the sensor and array configurations was also obtained in two phone interviews with sensor experts; synopses of these interviews are contained in Appendix 3. Finally, a detailed design of the DMAS (an expansion of the diagrammed design, in tabular form) is given in Appendix 4.

We note that many decisions about how the sensors are to be configured and the format of the data to be produced have not yet been made by the researchers involved. This situation has required a flexible approach to database design, and so the design presented here falls between a simple conceptual model and a detailed physical design. This design reflects the basic architecture decided on to date, and utilizes placeholders to house the details that will be determined later.

A further issue to be addressed is the computer environment where the DMAS is to be housed. To this end, Paul Murray of the Bureau of Economic Geology at the University of Texas at Austin (a member of the Gulf of Mexico Hydrates Research Consortium) has met with staff from the Texas Advanced Computing Center (TACC), who provided the name of a contact (Tomislav Urban). BCS then corresponded with Mr. Urban and confirmed that TACC is running a suitable database (PostgreSQL 7.x) and would be able to arrange a PostgreSQL test environment with 200 GB of available space.

## Choice of ORDBMS

While the design presented in this document could be implemented using a variety of

[24] *Sensor and Data Characterization for the Gulf of Mexico Hydrates Seafloor Observatory – Version 1.2* (January 31, 2005) and *Data Management Architecture Design for the Gulf of Mexico Hydrates Seafloor Observatory – Version 1.1* (February 28, 2005)

conventional relational database management systems (RDBMS), future enhancements will benefit from a database management system with object relational support[25].

In the short term, data from the instruments will be retrieved on a discrete schedule. The data will be stored in files, these files can easily be loaded into conventional database binary large objects (blobs), and metadata for these files can be precomputed and stored in conventional simple integer, float, and text columns. The files (and hence blobs) will be relatively small, so there will be no real penalty in fetching entire blobs and processing them exclusively by the client applications.

In the longer term, however, data will be reaching the database, via fiber cable, as a stream rather than as a prepackaged set of files. While the data will ultimately have to be broken into discrete chunks, object relational extensions can be used to preserve the "illusion" of a continuous stream. Using a single SQL statement it will be easy, for example, to extract all the data that falls between two specific time values, regardless of whether this data is stored in a single chunk or in parts of many chunks. These object relational extensions will require that the data be stored in "smart" chunks, not simple blobs, and this "smartness" cannot be designed until the communication and control mechanisms of the continuously-feeding-data Observatory are finalized.

As a result of these considerations, candidate database management systems were restricted to ones that enable user-defined object-relational extensions. There are two of these: PostgreSQL (http://www.postgresql.org/) and IBM Informix (http://www-306.ibm.com/software/data/informix). Either product would be suitable for the Gulf of Mexico Hydrates Seafloor Observatory DMAS; the following is a brief comparison of the two products, with the ***P*** or ***I*** symbol used to indicate which product has an advantage in a particular area:

6) PostgreSQL is free; Informix has a (high) up-front cost (***P***).
7) PostgreSQL is open-source; Informix is proprietary. This makes maintaining the object-relational extensions easier in PostgreSQL (***P***).
8) PostgreSQL has a more cumbersome garbage collection mechanism than Informix ***(I)***.
9) PostgreSQL is single-threaded, while Informix is multi-threaded. This can make Informix more efficient in situations with many concurrent users ***(I)***.
10) The smart blob implementation in Informix is more efficient than that used in PostgreSQL ***(I)***.

The initial database will use conventional features common to both Informix and PostgreSQL, so a decision can be deferred, without cost, until the time comes to develop database extensions. It is therefore recommended that PostgreSQL be used for the initial database and that this decision be revisited once initial data has been loaded and user access requirements are better understood.

[25] The benefits of using an object relational database approach were outlined in *Data Management Architecture Design for the Gulf of Mexico Hydrates Seafloor Observatory – Version 1.1* (February 28, 2005)

# Design for DMAS

This section presents a design for the Gulf of Mexico Hydrates Seafloor Observatory data management and archive system. Database designs range from simple conceptual models, which lay out the basic entities and relationships of a database, to complex physical designs that define exactly what is stored, how it is stored, and where it is stored. The design presented here falls midway between these two extremes. It is premature at this time to present a detailed physical design, since the details of such a design would depend on factors not yet decided, for example, the configuration options to use on a particular instrument. Instead, this design reflects the basic architecture decided on to date. It will likely be the case that some of these details will not be filled in until the data from the instruments is actually retrieved.

## Design Diagrams

The following five diagrams illustrate various aspects of the DMAS design. The notation used in these diagrams is explained immediately following the last diagram (Figure 5).

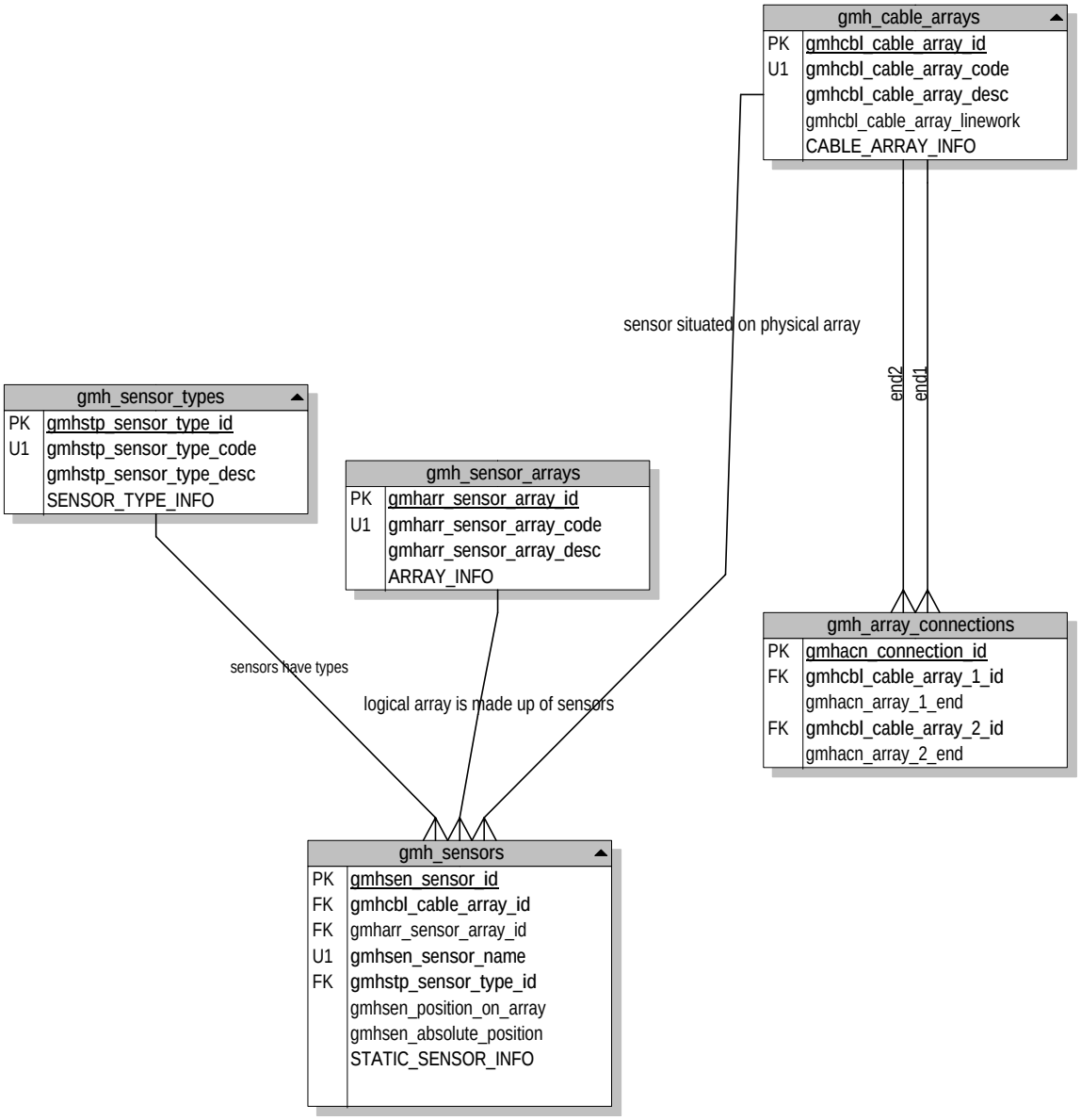



**Figure 7: DMAS Sensor and Array Configuration**

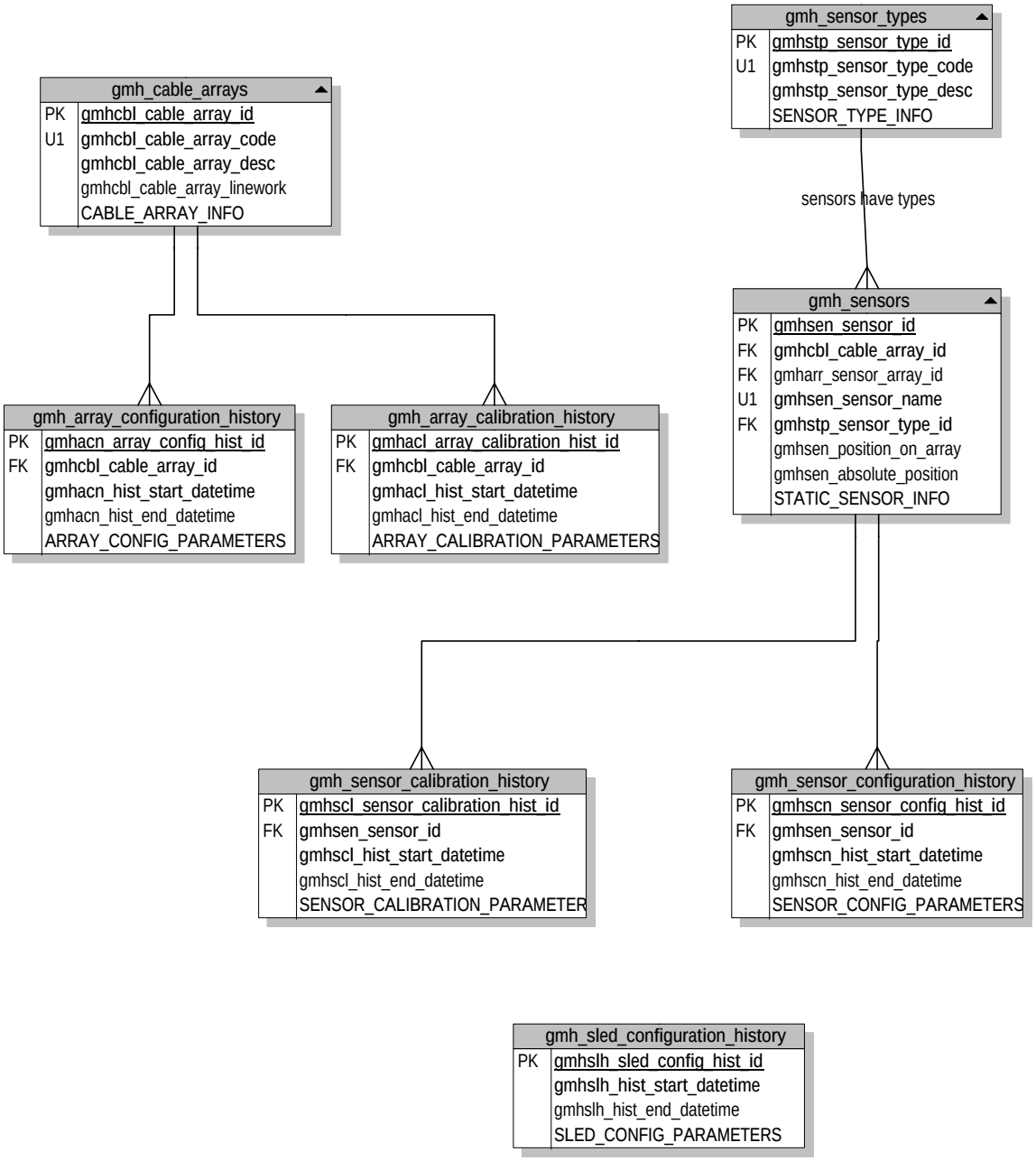



**Figure 8: DMAS Calibration and Configuration History**

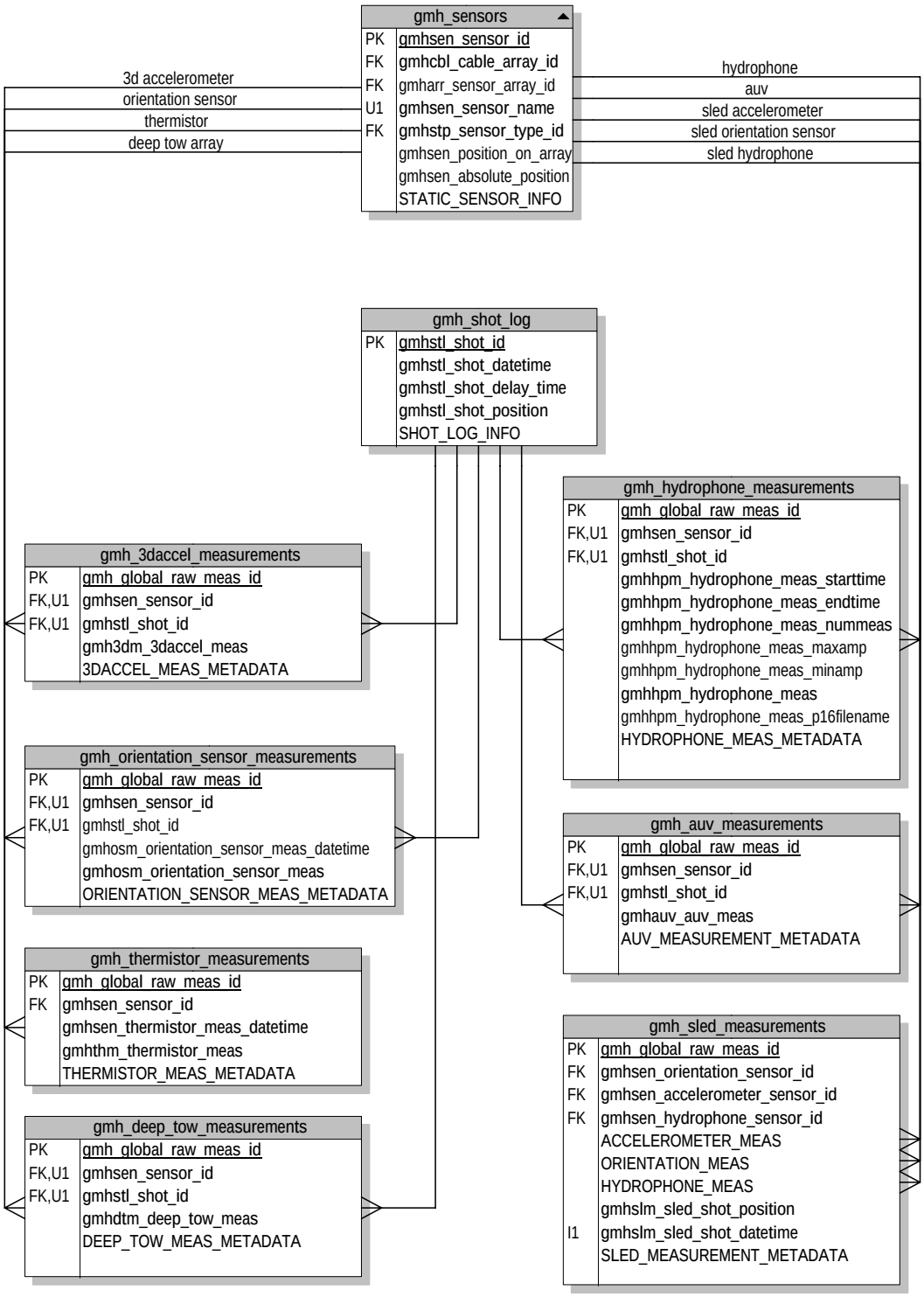



**Figure 9: DMAS Seismic Sensor Measurements**

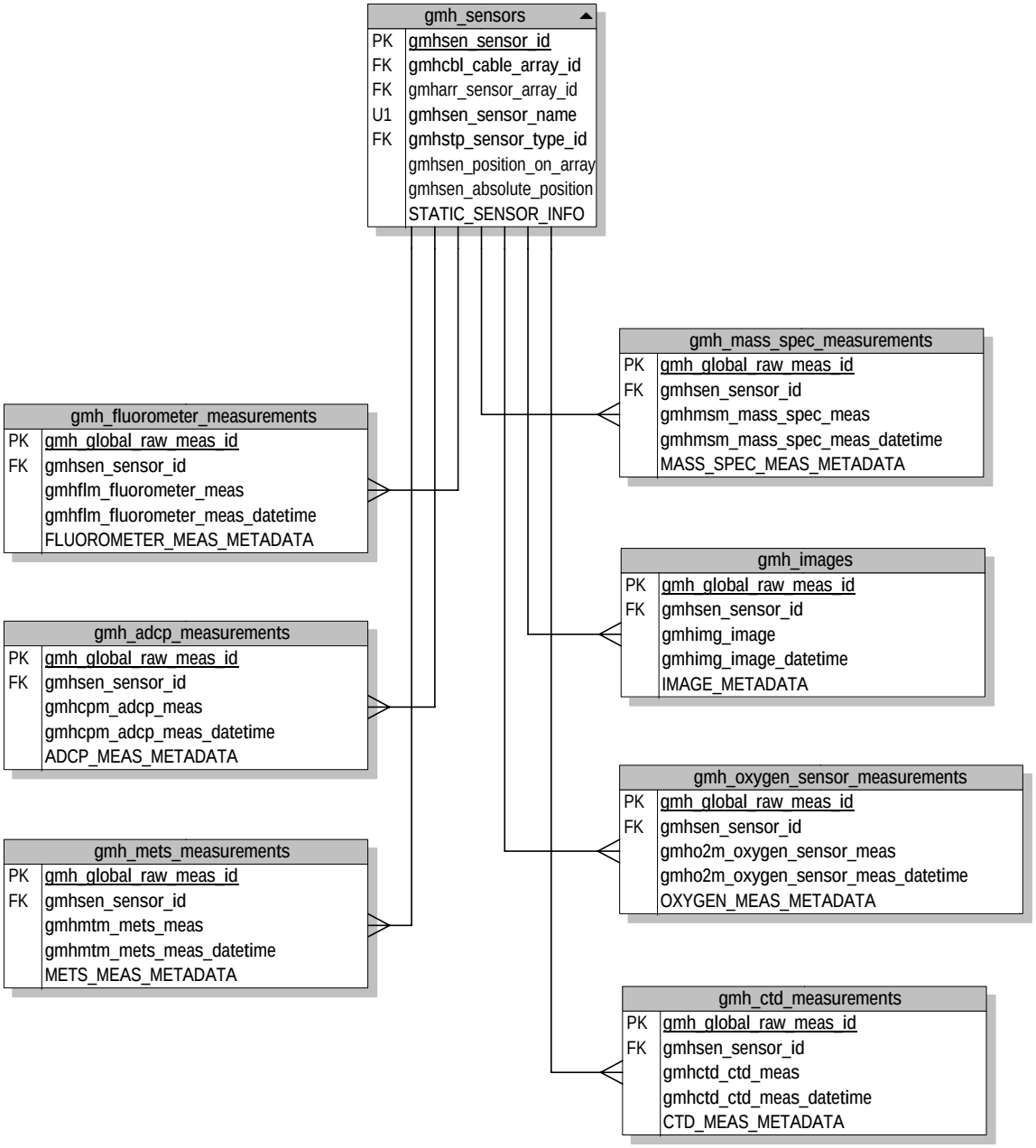



**Figure 10: DMAS Environmental Sensor Measurements**

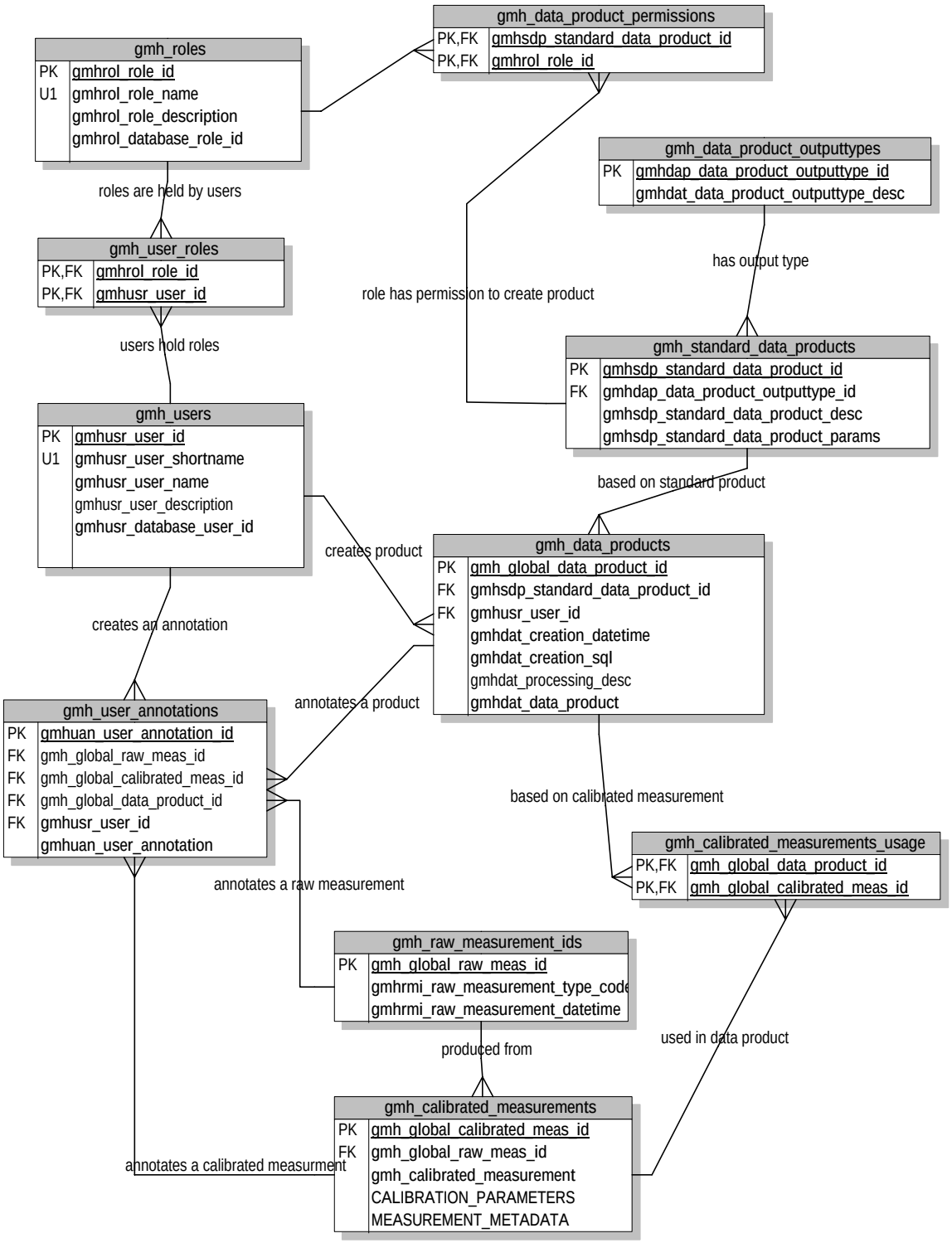



**Figure 11: DMAS Users, Roles, Measurements and Data Products**

## Diagram Notation

12) Each box represents a table; text in the gray area at the top of the box is the table name.
13) Table columns are listed in the white part of the box.
14) The primary key[26] column(s) of each table are underlined and have PK written next to them on the left.
15) Foreign key[27] columns have an FK written next to them on the left.
16) Columns with the same U*n* value written next to them together form a unique key; together their values uniquely identify a row in the table.
17) All PK, FK, and U*n* columns are indexed; other indexed columns are identified by an I*n* designator appearing next to them.
18) Column names written in **bold** lower-case represent columns that must always have a value (cannot be left NULL); column names written in non-bold lower-case are nullible columns.
19) Column names written in UPPER CASE represent placeholders; they will be replaced by real columns in a later design.
20) Lines between boxes represent 1-to-many relationships between tables. The line end with a just a single part is the "1" end of the relationship; the line end with the three parts ("crows feet") is the "many" end of the relationship. For example, in Figure 1, the line between the table gmh_sensor_types and gmh_sensors can be interpreted as follows: For each sensor type there are 0 or more ("many") sensors with that type and for each sensor there is just a single ("1") sensor type. Specifically, the relationship ties the PK column in the "1" table to the FK column having the same name in the "many" table.
21) Tables can be related in more than one way. In Figure 1 there are two lines between gmh_cable_arrays and gmh_array_connections. A cable can be connected to one set of other cables at one end and a different set of cables at the other end.
22) Some lines have "verb phrases" written across them; these help to explain the nature of the relationship.

The tables and relationships shown in Figures 1 through 5 are described extensively in the database dictionary report reproduced in Appendix 4. In addition, some features of the five views of the database shown in Figures 1 through 5 are highlighted in the following sections.

[26] The primary key for a table is a column or a combination of columns whose value(s) are used to uniquely identify a single row in the table.
[27] A foreign key in a table is a column or a combination of columns whose value(s) are used to uniquely identify a single related row in another table.

## Sensor and Array Configuration

Figure 1 illustrates the notions of ***sensor***, ***sensor array***, and ***cable array***. The Observatory houses a number of sensors of various types (e.g., hydrophones, thermistors, orientation sensors). These sensors may provide data independently of other sensors, or they may act as a group with other sensors to produce coordinated measurements. The group of one or more sensors producing coordinated measurements is referred to as a *logical* array. Each individual sensor is placed on a *physical* array. These physical arrays can be connected, at either end, to other physical arrays.

The assignment of these conceptual objects to tables is as follows:

| | |
|---|---|
| Individual sensors: | gmh_sensors |
| Sensor types: | gmh_sensor_types |
| Logical, or sensor, arrays: | gmh_sensor_arrays |
| Physical, or cable, arrays: | ghm_cable_arrays |
| Connections between physical arrays: | gmh_array_connections |

## Calibration and Configuration History

Figure 2 illustrates the relationships between sensors / physical arrays and their configuration and calibration histories.

Throughout the life of the Gulf of Mexico Hydrates Seafloor Observatory, the Observatory's sensors and physical arrays will undergo changes in configuration and calibration. In order to reprocess historical data it may be necessary to access the calibration and configuration data that was in effect at that time. The *gmh…history* tables will store the information needed for this.

## Seismic Sensor Measurements

The HLA, VLA, and BLA physical arrays described in Appendix 1 will house the instruments and sensors needed to conduct matched field inversion (MFI)[28]. Figure 3 illustrates how the tables associated with these sensor measurements are related to one another.

There are seven measurement tables. Each includes the foreign key values necessary to represent the relationship of a measurement to the sensor from which it is taken and the shot number it represents, if any. The tables also include the measurements themselves, in addition to any other metadata useful in identifying or finding

[28] MFI will require knowledge of the locations of the source and hydrophone sensors, and the measurements received from the array of sensors.

measurements needed for a particular application.

Five of these tables (gmh_3daccel_measurements, gmh_auv_measurements, gmh_orientation_sensor_measurements, gmh_hydrophone_measurements, and gmh_deep_tow_measurements) are related to both the shot log (stored in gmh_shot_log) and to snesors (information about which is stored in gmh_sensors). Each of these measurements relates to a single shot (identified by the gmhstl_shot_id value) and to a single sensor (identified by the gmhsen_sensor_id column). No time value is explicitly stored with the measurement, since the appropriate time value can be accessed via the shot log.

Thermistor measurements, on the other hand, are not tied to a specific shot and so the table for thermistor measurements (gmh_thermistor_measurements) includes a specific datetime value.

The sled system includes its own acoustic source, so again there is no need to relate the sled system table (gmh_sled_measurements) to the shot log.

## Environmental Sensor Measurements

The Benthic Boundary Layer Array (BBLA) and the Chimney Array described in Appendix 1 will house a variety of instruments and sensors. The measurements (or images in the case of the cameras) will be stored in the seven tables shown in the lower portion of Figure 4. Each row of each of these tables will include a measurement (or image), information identifying the sensor to which the measurement / image was taken, the datetime of the measurement or image, and any other metadata useful in identifying or finding measurements/images needed for a particular application.

## Users, Roles, Measurements, and Data Products

Figure 5 illustrates many aspects of the processes involved in accessing and managing data in the Gulf of Mexico Hydrates Seafloor Observatory DMAS. Processes reflected by Figure 5 include:

5) calibration of raw data to produce calibrated data,
6) creation of data products from calibrated data,
7) annotation of raw data, calibrated data, and data products, and
8) user access to raw data, calibrated data, and data products.

## Calibration of Raw Data

As mentioned previously, raw measurement data will be stored in the tables shown in Figures 3 and 4. In Figure 5, these individual tables are referred to collectively by the

“GENERIC RAW MEASUREMENT TABLE” box. When viewing Figure 5, it should be kept in mind that this one box actually represents 14 measurement tables, which will all have the same relationships with the gmh_user_annotations and gmh_calibrated_measurements tables shown in Figure 5.

Each raw measurement may be calibrated one or more times in order to produce a set of calibrated measurements. These calibrated measurements are stored in the gmh_calibrated_measurement table.

## Creation of Data Products

Once raw measurements have been calibrated they may be used, perhaps in concert with other calibrated measurements, to produce a data product. Each data product is based on a particular pre-defined “standard” data product, although the design does accommodate the need to sometimes create products in an *ad hoc* fashion. Each of these standard data product types has a particular output type (e.g., image, spreadsheet, or textual report), and these output types are defined by the gmh_data_product_outputtypes table.

## Annotation of Data

Each user of the DMAS can produce and store annotations for raw measurements, calibrated measurements, and/or data products. These annotations are stored in the gmh_user_annotations table.

## User Access

The Gulf of Mexico Hydrates Seafloor Observatory DMAS may have many different types of users, with these types distinguished by the sorts of things they can do with the data. These user types, called “roles”, are stored in the gmh_roles table, while the DMAS users are enumerated in the gmh_users table. Each user might play multiple roles, so the gmh_user_roles table keeps track of the roles that each user can play. As mentioned previously, users can create annotations and data products; those two relationships are illustrated by the relationships appearing on the center left side of Figure 5.

It may be the case that not every user is allowed to create every type of data product. The relationship between user roles and the types of data products that can be created is represented by the gmh_data_product_permissions table.

# Implementation of Prototype Database

## Database Installation and Population

The Gulf of Mexico Hydrates Observatory ("gmh") prototype database was installed on a Pentium 4 2x3.06 GHz machine running Fedora Core 3 Linux and PostgreSQL version 8.0.3. The SQL used to create the database is provided in Appendix 5.

In order to produce the prototype ("catalog retrieval") application described in the next section it was necessary to populate some of the database tables with sample data, as illustrated in the following:

| Database Table | Number of Rows | Data Source / Description |
|---|---|---|
| gmh_sensor_types | 4 | CTD, ADCP, hydrophone, fluorometer |
| gmh_sensors | 19 | 1 CTD, 1 ADCP, 16 hydrophones, 1 fluorometer |
| gmh_cable_arrays | 2 | VLA and OVA arrays |
| gmh_shot_log | 1063 | data supplied by the University of Victoria |
| gmh_hydrophone_measurements | 17008 | data supplied by the University of Victoria |
| gmh_fluorometer_measurements | 93245 | MBARI (OPeNDAP)<br>http://dods.mbari.org:8085/mbariMoos/ncarchive.jsp |
| gmh_adcp_measurements | 20482 | MBARI data<br>http://www.mbari.org/MUSE/Processed/Mooring/NPS-Adcp/muse_adcp_vcomponent.txt<br>http://www.mbari.org/MUSE/Processed/Mooring/NPS-Adcp/muse_adcp_ucomponent.txt |
| gmh_ctd_measurements | 31038 | MBARI (OPeNDAP), data from mooring M0<br>http://dods.mbari.org:8085/mbariMoos/ncarchive.jsp |

## Catalog Retrieval Prototype Application

In order to demonstrate the framework for data extraction from the Gulf of Mexico Hydrates Observatory, a prototype web application for accessing the "gmh" database was developed. This prototype can be viewed at http://www.barrodale.com/olemiss/web/catalog.php.

The prototype makes use of the following technologies:

1) client-side image maps, allowing the user to select the array/sensor to be queried from a picture;
2) asynchronous Javascript with XML (AJAX), providing a smoother, more "Windows-like" interface than that of conventional web applications;
3) PHP scripting for server-side HTML page preparation;
4) CGI/C/Gnuplot processing for plot preparation.

### Browser Support

The prototype uses features of Javascript that are standard parts of recent versions of both Internet Explorer and Netscape-based browsers such as Mozilla and Firefox. No special plug-ins (e.g., the Java runtime environment, Macromedia Flash, etc.) are required. The prototype has been tested using the following browsers:

- Internet Explorer 6.0
- Mozilla 1.7.2
- Firefox 1.0

## AJAX Interface

In conventional web applications, updates are performed “a page at a time.” When the user presses a submit button, a two-way synchronous communication takes place: data is sent from the client (browser) to the web server, an entire web page is (re)created on the web server, and it is then sent back to the client (browser) for (re)display. Such an interface can appear to be slow and cumbersome, since:

- the entire page is redrawn, rather than just the part that has changed;
- there is often a noticeable delay while the page is calculated and redrawn, especially over slower Internet connections.

In contrast, AJAX (Asynchronous Javascript and XML) allows for two-way asynchronous communication, providing a means of updating just parts of a page while allowing other activity on the rest of the page to continue. For example, an event on one part of a form (e.g., the selection of an item from a drop-down) can trigger the refresh of another part of the form.

As an example of AJAX, consider the following pages from the prototype.

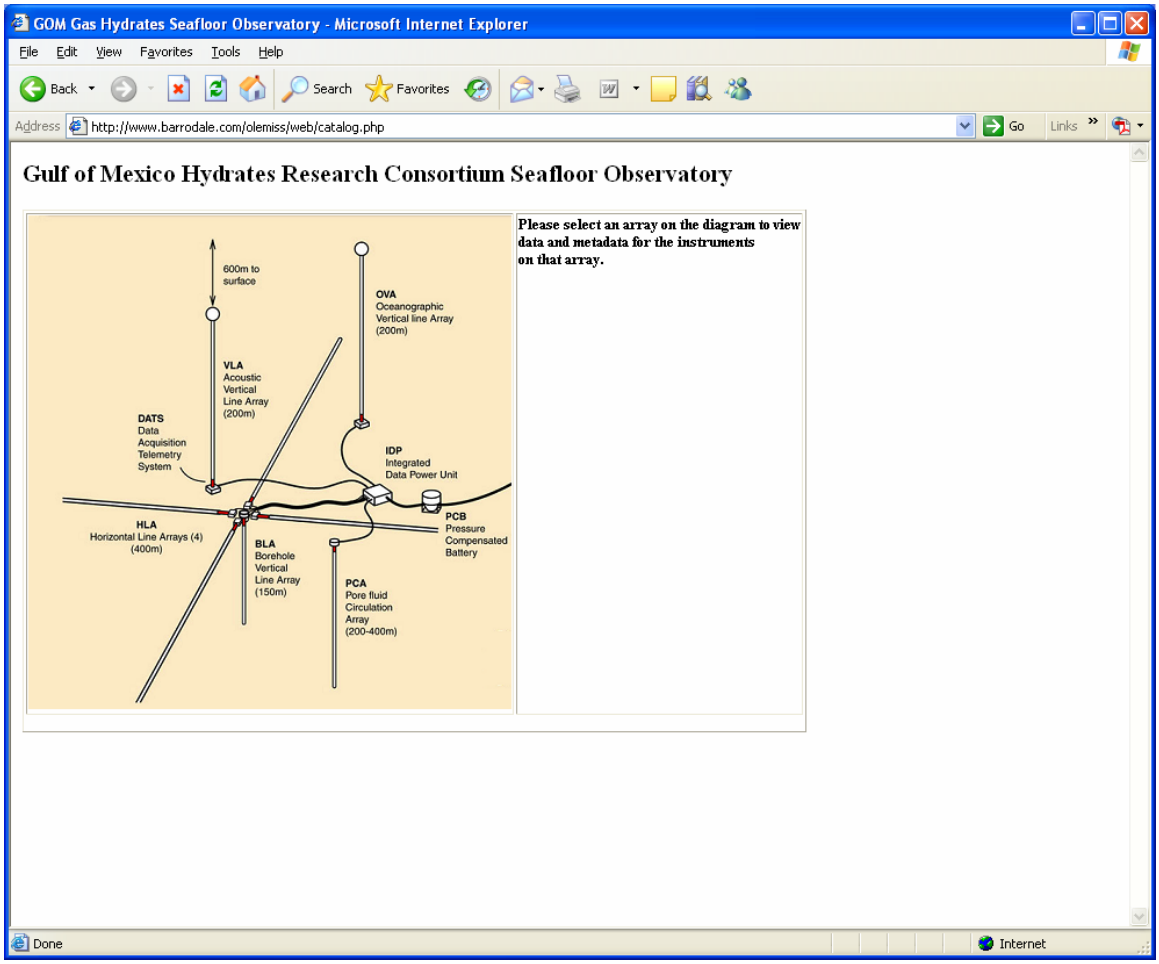



**Figure 12: Catalog Prototype: Initial Form.**

When the user first accesses the web site they see the page shown above.

As soon as they click on the OVA portion of the image, a Javascript trigger calls AJAX to extract (from the gmh database) and display metadata about this array. In particular, a drop-down list of sensors is created and populated (from the database), as shown in the following image.

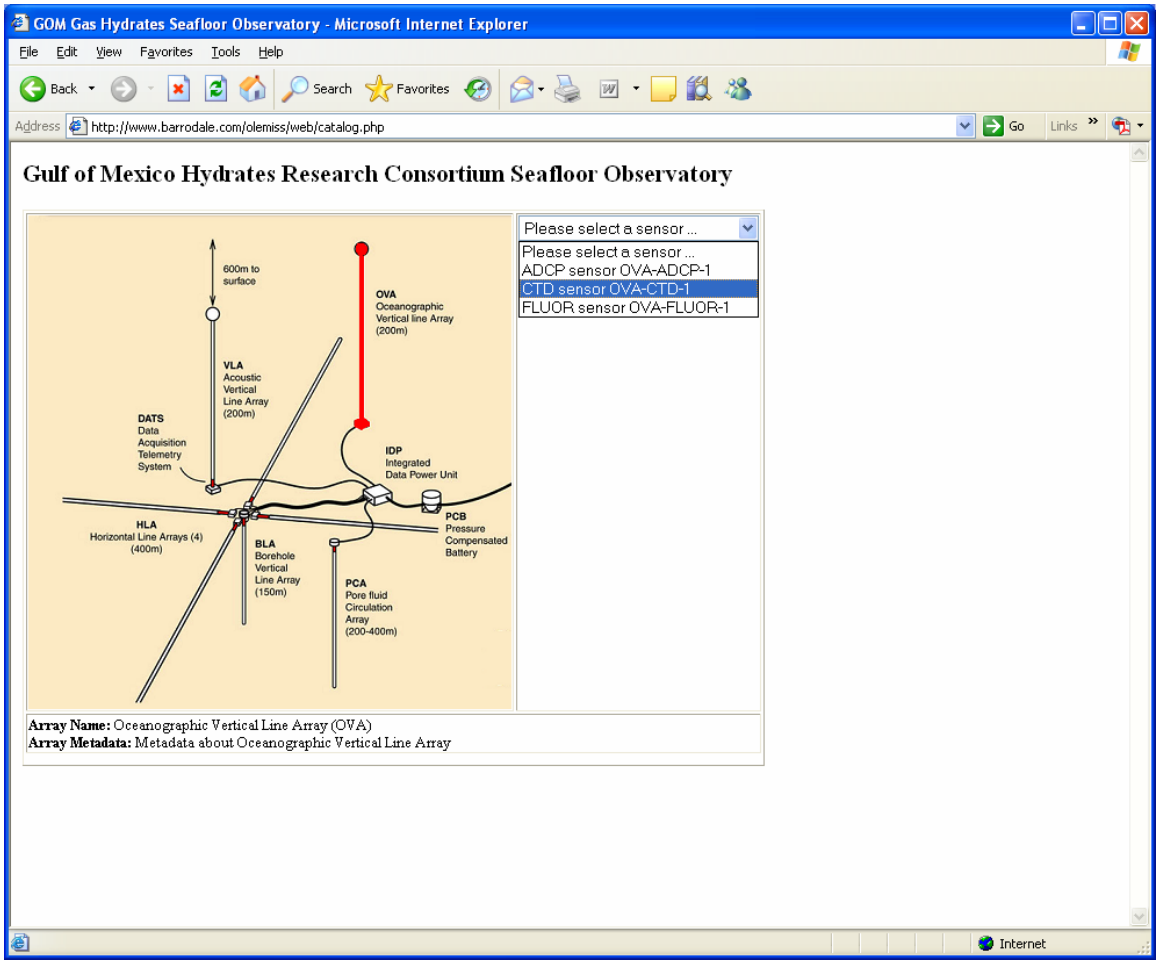



**Figure 13: Sensor List Has Been Populated.**

Note that in this prototype, the metadata shown is just a placeholder. Once the nature of the metadata is better understood, the prototype will be modified to show the actual metadata fields.

As soon as the user selects one of the sensors, another Javascript trigger calls AJAX to fetch and display information about the selected sensor, such as the start and end times found in the database.

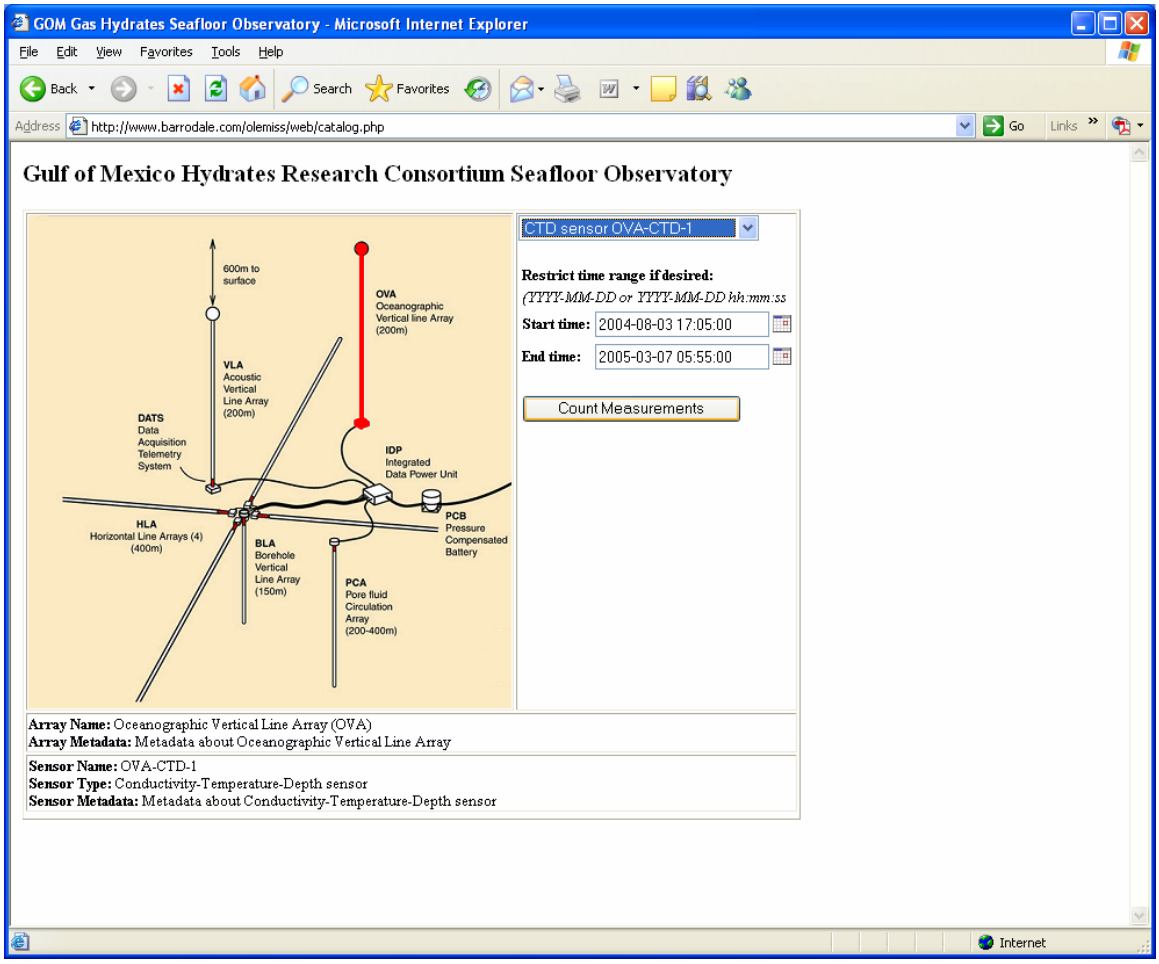



**Figure 14: Actual Data Time Ranges for CTD Have Been Populated.**

This client-server interaction continues, with the user drilling down into the data until a single measurement can be selected for plotting:

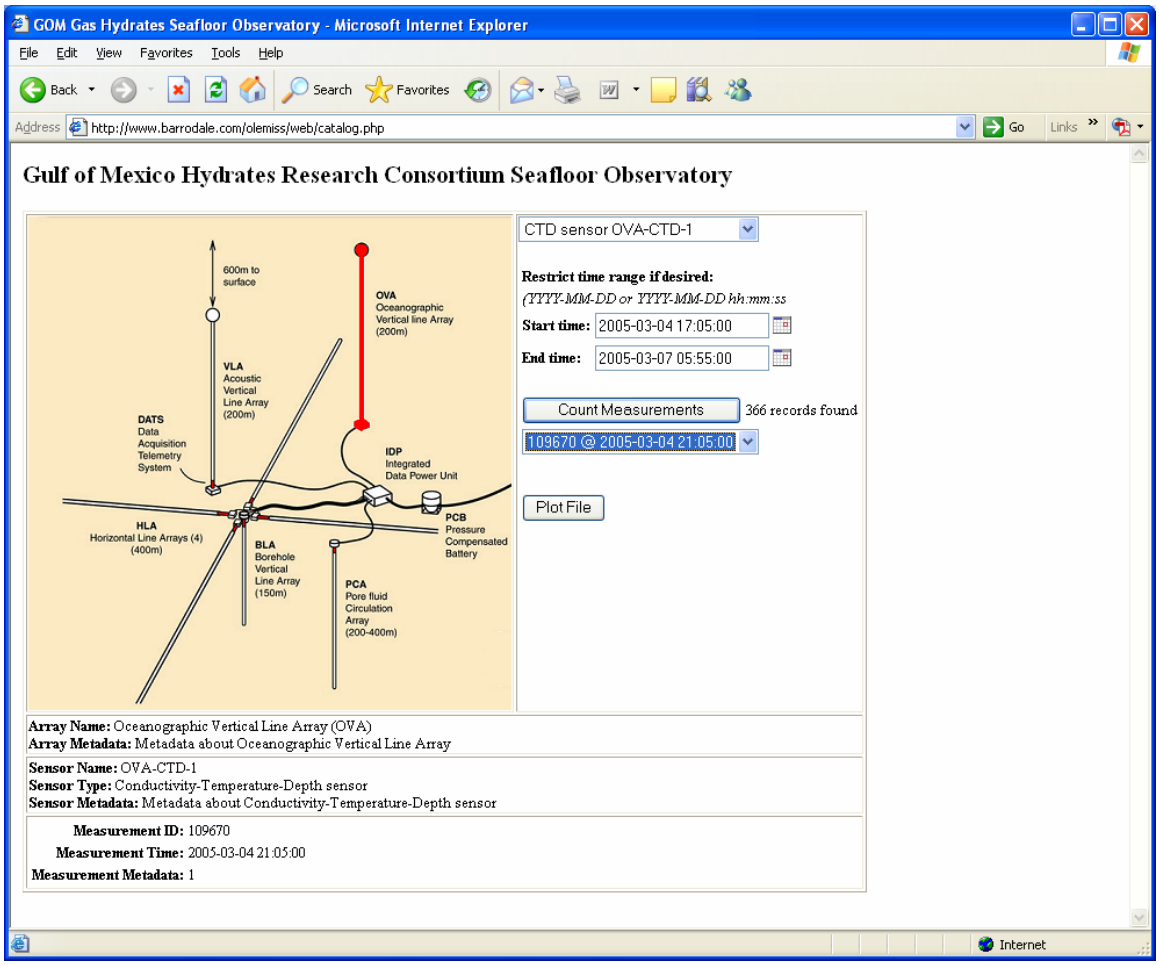



**Figure 15: Placeholder Metadata for CTD Sensor Is Displayed.**

Note that as the user drills down, the extent of the metadata display (displayed at the bottom of the web page) advances from *Array* to *Sensor* to *Actual Sensor Measurement*.

## Plotting

Once the user presses the "Plot File" button shown in Figure 10, CGI[29] is used to invoke a plotting program on the server. In this prototype, the plotting library used is *gnuplot_i* (http://ndevilla.free.fr/gnuplot/), a freely-available interface to the package *gnuplot* (http://www.gnuplot.info/).

The following plots have been implemented in the prototype:

[29] CGI stands for *Common Gateway Interface*, a mechanism for communicating between a web form and a processing application, in this case a graphics program, running on the web server.

1) plot of hydrophone output vs. time for a particular 6 second, 10,000 Hz recording;
2) plot of hydrophone output spectra for a particular 6 second, 10,000 Hz recording;
3) north and east velocity components vs. depth, for a particular ADCP sensor record;
4) conductivity, salinity and temperature vs. pressure (depth), for a particular CTD sensor record;
5) fluorometer voltage output vs. time for a user-specified time period.

A sample fluorometer plot is shown in Figure 11.

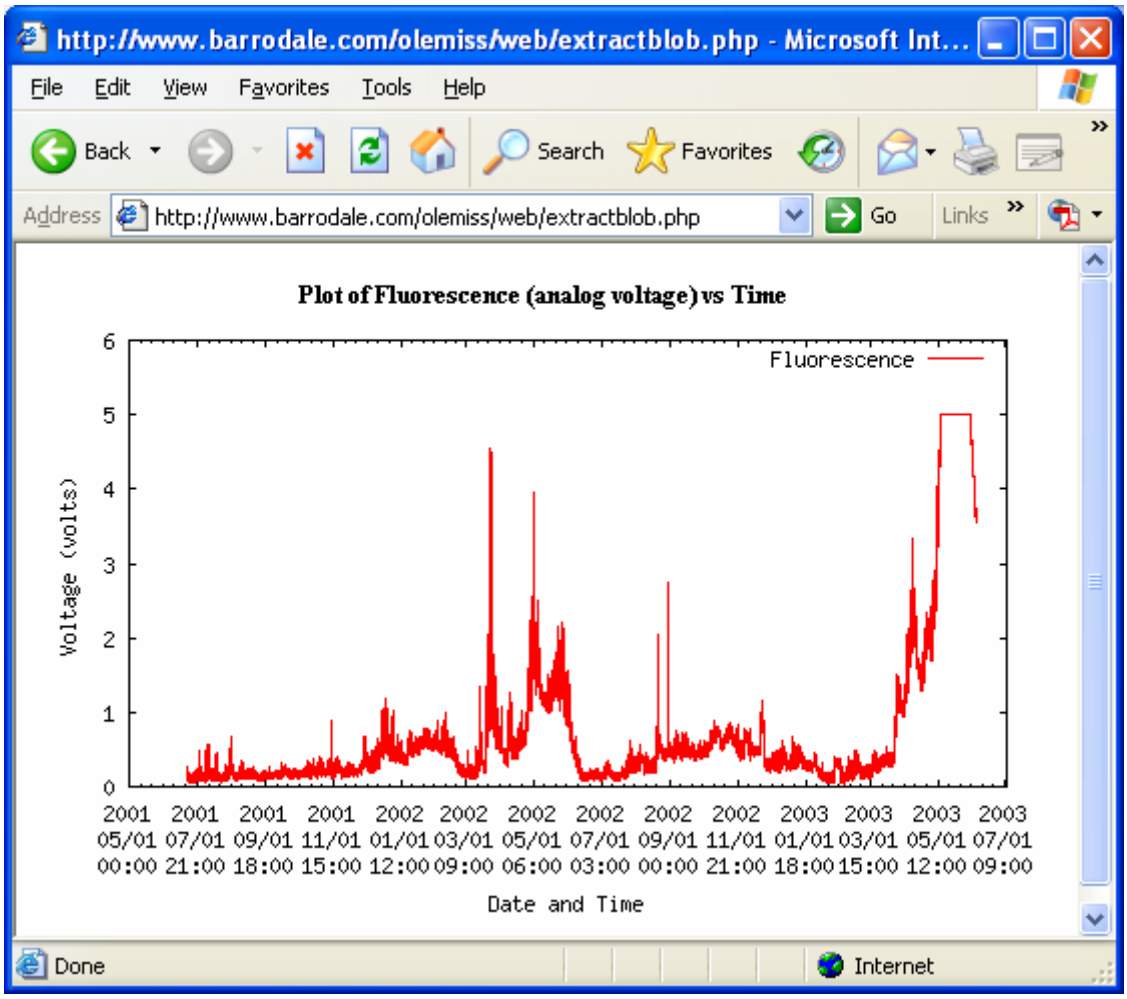



**Figure 16: Fluorescence vs. Time plot.**

## MFI Software Enhancements

### 3D Arrays

The ability to use 3D arrays was incorporated into the simulation and matched field components of the BCOMFI system and for both ORCA and RAM propagation models. Since ORCA already has an option to make use of an array of arbitrary geometry, the changes required here were minor and simply involved setting up the options file in a more general way than previously and modifying the code that reads in the computed field file.

Unlike ORCA, RAM does not have an explicit provision for using 3D arrays. The output of RAM is simply a 2D grid of complex field values along a vertical plane intersecting the source and the receiver positions. If the array being used is vertical or if the array has all its sensors in this plane, then using RAM is most efficient, since it is simply required to interpolate the grid at the appropriate ranges to obtain the field values at the array locations. However, if the array is truly 3D, many of the sensors will no longer lie in the plane, and the corresponding fields will vary with bearing as well as range.

We implemented an approximate but efficient approach to using 3D arrays with RAM, which assumes that the horizontal extent of the array is small enough so that the source-receiver paths are effectively independent of bearing. In this case, a single RAM run is performed, starting from the source to a suitable reference point of the array (e.g., the geometric center). For each sensor, the source-sensor range is then computed and used to interpolate the grid in a similar way to the approach taken for arrays in the plane of propagation.

Another possible option considered for full implementation of 3D arrays using RAM was to perform a separate run for each sensor with a different source-receiver path in the horizontal plane. If the environment is range-independent, the two approaches will produce identical results. If it is not, then the results will differ by amounts which increase with the difference between the respective range-dependent paths. This option would be extremely time-consuming, however, since numerous paths would be required for a 3D array, and the RAM runs themselves are already very time-intensive compared to ORCA. It is anticipated that for relatively small discrepancies in the paths (say, 100 m in a range of 2 km), a mildly range-dependent environment, and suitable choice of frequencies, the agreement between the full and approximate approaches should be close. Hence the approximate approach was implemented, thereby allowing the generation of fields by RAM for a 3D array in a range-dependent environment in a reasonable time frame.

Following these enhancements, field values were computed for 3D arrays using ORCA and RAM for a range-independent environment and these were compared using the matched field techniques in BCOMFI at a range of 2 km. It was verified that the two approaches gave high multi-frequency Bartlett matches for large (up to 96 elements tested) 3D arrays.

## Monitoring Software

The BCOMFI software system was extended to include the ability to apply monitoring to hydrophone array data. In the absence of real data from fiber optic cable, previously developed software was used to simulate the data from the array that would be obtained when the data streams are processed and transformed to the frequency domain. A two-stage process was then developed for monitoring by estimating the parameters of a layered geoacoustic model, for each time period in the data sequence, as follows:

- In the first stage, a 2D depth-range (ORCA option) or range-bearing (RAM option) ambiguity surface with contributions from multiple frequencies was generated using precomputed replica vectors, and the parameter values corresponding to the best match in this surface were identified.
- In the second stage, bounds for the two parameters were established based on the best match in the 2D ambiguity surface. Then the parameter space was expanded to include user-selected parameters and their bounds, and these chosen parameters were optimized using the search-optimization technique developed for MFI. The result of the optimization provided an estimate of the geoacoustic parameters.

This approach to monitoring was implemented in IDL as an additional option in BCOMFI. As part of this implementation, the 2D ambiguity function component of BCOMFI was modified to precompute the replica vectors to be used for monitoring. In addition, the search-optimization code was modified to allow the specification of the conditions for using this technique in monitoring, and to actually perform the search/optimization computations required by the monitoring process. A display routine was also written to show the parameter estimates as a function of time.

In the absence of fiber-optic data, the simulation software was used to generate test data for the monitoring. Using a simple test model with a moving source, it was verified that the geometric and selected geoacoustic parameters could be recovered using this technique.

## Range-Dependent MFI

It is known that the ocean bathymetry in the region of the Observatory is not range-independent, but generally slopes towards the southeast at an angle of about 1.5 – 2 degrees. Simulations performed earlier in this project indicated that range-independent methods could only be expected to provide reasonable estimates of the layered geoacoustic parameters up to an angle of 0.5 degree. Hence the use of only a range-independent model in MFI would restrict its usefulness to sectors that fell within these slope limits. This would clearly be an undesirable constraint on the use of MFI in parameter estimation and monitoring.

To overcome this limitation, and although it was beyond the scope of the current contract, BCS extended the BCOMFI software system to implement the ability to

generate ambiguity functions and apply MFI in a range-dependent environment with known bathymetry. The approach was designed to use a digital elevation model (DEM) for the bathymetry, from which the depth profiles along the source-receiver paths can be computed at user-specified intervals. These profiles are used in building up an input file for RAM, which then generates the fields that would be observed at the sensors. A DEM for Mississippi Canyon 118 was obtained from the University of Mississippi and subsampled at 50 m intervals for use in BCOMFI.

The user can then choose whether the environment is to be range-independent (in which case ORCA is used) or range-dependent with a DEM (in which case RAM is used). In the latter case, the user can place the array at a specified location within the horizontal extents of the DEM grid and specify the step size at which the depths along the path are provided to RAM. During the computation of these depths, cubic interpolation was used to ensure that the depth derivatives at the edges of the DEM grid cells are continuous, and so to prevent possible problems that can arise with optimization when bilinear interpolation is used.

An additional advantage of range-dependent MFI is that it can potentially allow the resolution of source bearing, since the environment is no longer symmetric. Numerical simulations confirmed this for the DEM of MC 118, with Bartlett processor values being observed to vary between 0.5 and 1.0 as a source at 2 km range was moved through 360 degrees of bearing.

## Documentation

The User Guide to the BCOMFI software system was extended to include descriptions of the above enhancements to functionality, and also to describe the new and modified interfaces that were developed as part of these enhancements. This extended User Guide forms the next section of this report.

# BCOMFI Software User Guide Version 1.1

## Introduction

Barrodale Computing Services Ltd. (BCS) has been contracted by the University of Mississippi to design and develop data management and processing software for this monitoring station. In six reports written during the earlier stages of this project, BCS has characterized the data to be produced by the station[30], proposed a design for a data management and archiving system for these data[31], formulated a design for a software system for simulating the data to be acquired by the vertical acoustic array of the station and analyzing these data using matched field inversion (MFI) techniques[32], implemented this design[33], provided a User Guide to the initial version of the software[34], and extended the implementation to include prototype matched field monitoring capability[35]. The present User Guide contains a full description of the simulation and matched field software, and includes descriptions of the recent implementation of simulation and MFI components for range-dependent bathymetry using a digital elevation model for the site of the Observatory.

The software system has been termed BCOMFI, for **B**arrodale **CO**mputing **M**atched **F**ield **I**nversion. The simulation and conversion software component of BCOMFI provides several types of functionality. First, since the real data will consist of both time-limited shots and continuous data from nearby moving ships, the software has been implemented to simulate both shot traces and ambient noise data streams that would be observed at the sensors of an array. It also provides the ability to preprocess these data into a form where the resulting complex pressure fields can be used for MFI. In addition, to provide for greater flexibility in modeling correlated noise, and to promote efficiency, the software allows direct generation of frequency domain data (i.e., complex pressure fields) at selected frequencies, to serve as input to the MFI algorithms. These complex "measured" data can be generated in the form of either a signal vector **m** or a cross-spectral matrix $\mathbf{M} = \sum_j \mathbf{m}_j \mathbf{m}_j^*$, where (*) denotes the conjugate transpose operation. By using appropriate acoustic propagation models, the complex fields can be computed for range-independent environments, constant-slope bathymetry environments or arbitrary bathymetry environments. A further component of BCOMFI allows real data "P16" files, which will be produced by the hydrophone array, to be read in and converted to a format where the data can then be analyzed by MFI.

The ambiguity function generation, MFI and monitoring software components of BCOMFI allow "measured" frequency domain data to be matched with replica data

[30] "Sensor and Data Characterization for the Gulf of Mexico Hydrates Monitoring Station" (Jan. 31, 2005).
[31] "Data Management Architecture Design for the Gulf of Mexico Hydrates Seafloor Observatory" (Feb. 28, 2005).
[32] "Software Design for Simulation, Matched Field Inversion, and Monitoring for the Gulf of Mexico Hydrates Seafloor Observatory" (April 26, 2005).
[33] Software for Simulation and Matched Field Inversion for the Gulf of Mexico Hydrates Seafloor Observatory (September 30, 2005).
[34] "BCOMFI Software User Guide" (August 12, 2005).
[35] Software for Simulation, Matched Field Inversion and Monitoring for the Gulf of Mexico Hydrates Seafloor Observatory (February 14, 2006).

vectors generated from a test model. This range-independent model includes 23 geometric and geoacoustic parameters, and can provide for up to three separate layers over a halfspace. The Bartlett power processor is used as a quantitative measure of the match between replica vectors generated using the model and the "measured" data. The software provides options for generating 1D, 2D, and 3D ambiguity functions at the points of a regular grid. This functionality allows the investigation of parameter sensitivity and correlation, and can provide an indication of the numbers of optima and other characteristics of the overall parameter space. MFI has been implemented using a global search / local optimization approach.[36] In the search stage, matches given by random choices of selected parameter values are obtained; then the best matches are chosen as starting points and for each of these, the matching function is optimized. This software allows investigation of issues such as the conditions where the model and geometry can be estimated with reasonable confidence, and could also be applied to use data obtained during calibration to generate "standard" range-independent models. The monitoring component of BCOMFI analyzes a sequence of data sets to estimate selected geoacoustic parameters from each set by a two-stage process. In the first stage, a 2D ambiguity function is generated using precomputed replicas. In the second stage, the maximum in this function is used as input to the search / optimization algorithm to estimate a multi-parameter geoacoustic model. This process is repeated for each data set.

An additional software component of BCOMFI provides the ability to display the results of the simulation and processing. It allows visualization of the simulated time domain data, the ambiguity functions, and the results of the MFI and monitoring processing.

The capabilities and use of the software are described in detail below. Very generally, the BCOMFI software consists of two basic types of code:

- ***Two Fortran executable modules for generating the acoustic fields***. These "computational engines" are: ORCA, a normal-mode code for range-independent environments, and RAM (Range-dependent Acoustic Model), a parabolic equation program that can be used for both range-independent and range-dependent environments. ORCA is generally more than an order of magnitude faster than RAM. Note that the availability of two entirely separate propagation models provides a consistency check by allowing comparison of the fields that are produced by the two models.
- ***IDL code for the front end and the back end***. This software provides the environment for setting up GUIs, controlling the simulation and MFI, calling the computational engines, processing the computed fields, and displaying the results. The software has been developed in such a way that it can be run using the IDL Virtual Machine (IDL VM – available at no cost from RSI) and therefore does not require an IDL Development Environment. The actual code has been delivered in the form of a save file (analogous to an executable file) that can run under the IDL VM.

[36] Zala, C. A. and Ozard, J. M. *Estimation of geoacoustic parameters from narrowband data using a search/optimization technique*, J. Comput. Acoust. **6**, 223–243 (1998).

In this report, the geoacoustic environment in the region of the proposed site is first described and the parameterization of this environment is outlined. Then the approaches for propagation modeling are outlined: the normal mode program ORCA would normally be used to compute the fields for a range-independent environment, for both simulation and MFI, while RAM would be used to simulate data and perform MFI for range-dependent environments. The design and use of the individual simulation and MFI components are then described: each description includes a diagram of the overall process, an image and explanation of the GUI, a statement of the algorithm, and a description of how to use the component. Our implementation of a parallel processing option for matched field computations is also described, and the use of the display components is outlined. A computing environment to support this processing is then outlined. Finally, two Appendices describe the data structures and tags used in the software and detail the modular structure of each of the components.

# Geoacoustic Environment

## Description

In order to formulate effective designs for the functionality of the software to be developed, it was necessary to obtain some information about the general geoacoustic environment in which the monitoring station is to be deployed. At a meeting of the Hydrates Research Consortium in November 2004, a desirable site for the station was identified by the attendees, in Mississippi Canyon Block 118. Early in 2005, a final choice of the location within this block was made, several seismic surveys were performed along four different tracks, and a digital elevation model (DEM) was acquired. These data indicated that:

- The depth at the site was about 875 m.
- There was a bottom slope up to about 2 degrees, with depth increasing to the southeast.
- The sloping region immediately surrounding the site was reasonably planar, but there was a canyon running northwest-southeast about 1.5 km to the northeast, and a flatter plateau, which showed high acoustic reflectivity, about 2 km to the south.
- There was a strong sub-bottom reflection about 250 msec below the bottom reflection, with two or three weaker reflections above this region and occasional strong deeper reflections. It was suggested that the reflection at 250 msec could possibly represent the bottom of the hydrate stability zone.
- There was a strong sub-bottom domed feature 1.5 – 2 km to the southwest of the site.
- There were several bright spots in the seismic sections, also at about the 250 msec horizon. These were thought to correspond to pockets of gas.

These data clearly showed that this environment is significantly range-dependent, both with respect to the water depth (approximately planar with a maximum slope of about 2°) and the characteristics of the 250 msec reflection. Because of this range

dependence, the question arose of the extent to which this environment might be satisfactorily modeled by a fast range-independent propagation model such as ORCA.

The initial approach taken in this project was to assume that there is some region close to the station within which the environment may be taken to be approximately range-independent. This assumption simplified the parameterization and reduced the times required for matched field inversion using optimization. We therefore designed and implemented a software system that allowed generation of synthetic data for range-independent or planar range-dependent environments and performed matched field analysis of synthetic and real data using a range-independent environment. The implementation included the ability to generate 1D, 2D and 3D ambiguity functions for specified parameters, and to perform parameter optimization.

The initial implementation was then extended during the present contract to include a prototype monitoring capability, to use full 3D arrays, and to allow simulation and matched field analysis with range-dependent bathymetry using a DEM for the region of the Observatory.

## Parameterization

It was essential to define a geoacoustic and geometric parameterization that is acceptably realistic for use in range-independent modeling of the environment, yet is sufficiently simple to allow matched field inversion to be performed in a reasonable time frame. Based on the above description of the environment, a set of variables was defined that consisted of source-array geometry parameters, up to three sediment layers that could each have a gradient in their acoustic parameters, and a basement[37] halfspace with constant parameter values. The 23-parameter model for the environment is as follows:

- ***Water depth***. Some variations in water depth due to tides suggested that that water depth should be a parameter in the inversion.
- ***Source depth***. This will generally be close to the surface (0.5 – 5 m) for both shot data and sources of opportunity, but was included for investigative purposes.
- ***Source range***. Matching is known to be strongly dependent on range.
- ***Source bearing***. For a vertical array in a range-independent environment, bearing is not resolvable (the fields are bearing-independent). However, for range-dependent environments or when the array is tilted, bearing does have an effect and is potentially invertible.
- ***Array tilt angle***. Arrays will be tilted in practice, and it is known that this tilt can have a significant effect on the field matches. Optimization of this parameter (and the following parameter, i.e., tilt direction) could significantly enhance the matching. In the present implementation, it is assumed that the tilted array is linear.

[37] The term “basement” is used here in a modeling, rather than geological, context.

- ***Array tilt direction***. As for previous parameter.
- ***Layer parameters***. These consist of five parameters for each of a prespecified number of layers (0 – 3: note that the number of layers is not optimizable):
  - ***Layer thickness***. This may be a useful parameter for detecting changes at the base of the hydrate stability zone.
  - ***Compressional sound speed at layer top***. Changes in this parameter (or the following parameter, or both) could be indicative of changes that occur within the hydrate stability zone.
  - ***Compressional sound speed at layer bottom***. As for previous parameter.
  - ***Density at layer top***. Although this parameter is generally quite insensitive with respect to matching, it has been included for investigative purposes.
  - ***Density at layer bottom***. As for previous parameter.
- ***Compressional sound speed in basement halfspace***. Changes in this parameter could also reflect changes that occur at the base of the hydrate stability zone.
- ***Density in basement halfspace***. Although this parameter is generally quite insensitive with respect to matching, it has been included for investigative purposes.

This parameterization was designed to give considerable flexibility in the modeling while still being tractable in terms of matched field inversion.

An option was provided in the matched field applications to force the top and bottom compressional sound speed and density parameters in the layers to be equal (i.e., no gradient), so as to allow investigation of the effect of matching using a constant value when there is in fact a gradient in the layer. Note that shear sound speed and compressional and shear attenuations are not included as parameters. Also, the sound-speed profile in the water column is not included in the parameter set, but rather, a means is provided of specifying this data in an input file. This file could contain a generic, seasonally appropriate, sound-speed profile to be used in the matched field applications.

It is expected that the most useful optimizations will involve the geometric parameters water depth, source-array range and the thicknesses and sound speeds of the first one or two layers. The densities are not expected to have a substantial effect on the matches, but are provided nonetheless as a research tool.

For modeling using RAM in a range-dependent environment, it is assumed that the bottom is layered as above and that the layers parallel the bathymetry. Hence the above parameterization for matched field techniques can be used unchanged.

## Propagation Modeling

The ORCA normal mode acoustic propagation model is the “engine” used to compute the acoustic fields required for simulation and matched field-inversion in range-independent environments. For a given environment, specified by the sound-speed profile in the water column and a geoacoustic profile of the ocean bottom, ORCA finds the normal modes and computes the acoustic field at the sensors of an array. The model includes the effects of sound-speed gradients in the water and the bottom layers, shear waves in the bottom layers, steep-angle propagation represented by leaky modes, and attenuation in the bottom layers. It may be used to predict narrowband or

broadband propagation. ORCA is unique among underwater acoustic propagation codes because it is largely automatic: the user does not need to guess at any convergence parameters such as depth- or range-sampling resolutions. It is also computationally efficient, typically requiring a few tenths of a second (on a 3 GHz Windows computer) to compute a propagation model at frequencies of interest in the present application.

ORCA is written in Fortran, requires several input files, and was modified to produce an output file with the field values at the sensors. Since the simulation and MFI code are written in IDL, a simple communication protocol and data transfer scheme between IDL and ORCA was defined. In this scheme, the IDL process writes out files with the information required by ORCA, and then spawns an ORCA process that reads in these files and produces an output file containing the complex field values. The IDL process then reads in this file and uses the data in its internal computations.

Despite its advantages, ORCA is a range-independent propagation model and is not directly applicable to range-dependent environments. The environment in the region of the Observatory, however, is known to have significant range dependence, and it is desirable to be able to investigate the effects of this by using range-dependent data. While it might be possible to implement an adiabatic mode approximation using ORCA to achieve this, it was deemed preferable to implement a separate range-dependent code for performing such simulations. In addition, such a code provided the opportunity to perform a validation check on the correctness of the ORCA results.

Accordingly, the RAM parabolic equation model was implemented and modified as required to integrate it into the BCOMFI system, to allow simulation and matched field analysis of data in a range-dependent environment. This model is also written in Fortran and so the same general communication strategy as used for ORCA (i.e., IDL spawning a RAM process and reading in the output file) was used for RAM.

## Installing the BCOMFI Software

The BCOMFI software that runs under the IDL VM is provided in the form of the zipfile "bcomfi.zip". It sets up an appropriate directory structure and contains various data files as well as the software. To install this software, simply unzip the file into a directory of your choosing. The following directory structure and files will be produced:

- ***\Bin***. This contains the Windows version of the executable files for ORCA, RAM, and the orca server enabling parallel processing of matched field computations, i.e.,
  - orca.exe
  - rampe.exe
  - orcaserver.exe
- ***\Data***. This contains a set of files of the form xxxxnnnnn.ttt, where xxxx is a four-character file prefix, nnnnn is a five-character identification (ID) number, and ttt is either "dat" for a user-editable data file or "sav" for an IDL save file. These files are used to provide input and to store the results. (Certain of them may be overwritten during the processing, if the ID numbers are not changed during GUI setup.) See below for more detailed descriptions of these files, which correspond to IDL data structures used internally by the BCOMFI system.
- ***\Doc***. This contains User Guides for ORCA (PDF file) and RAM (PostScript file), and the present BCOMFI report.
- ***\Inputs***. This contains input files used by ORCA. In particular, the files sim_shot_svp and sim_ship_svp must be present in this directory; these files can be edited to modify the sound speed profile to be used by ORCA and/or RAM during the simulation. The other file types are generated during runs of the BCOMFI simulation system.
- ***\Linux***. This contains the Linux version of the executable files for ORCA and the orca server enabling parallel processing of matched field computations.
- ***\Outputs***. This contains files used by ORCA for output.
- ***\P16_data***. This contains sample ship and shot real data in p16 format in its subdirectories \ship and \shot.
- ***\Ramout***. This is initially empty but will contain files output by RAM during a simulation run where RAM is used.
- ***\Source***. This contains the files "bcomfi.sav", "ramgeo.in", "dem.dat" and "servers.dat", and will contain some input files for ORCA (orca_in) and/or RAM (ramgeo.dat) that are required for, or generated during, runs.

## Installing the IDL VM and Launching BCOMFI

The IDL Virtual Machine can be obtained from the RSI web site. To download the IDL VM, go to http://www.rsinc.com/idlvm/, press the "Download IDL VM" option, enter the required registration information, check the "Windows" option, and follow the instructions. When the download is complete, run the install script. This will install the IDL VM on your computer and create an icon on the desktop.

The IDL VM allows you to locate an IDL save (.sav) file – in this case, bcomfi.sav – and

run it. Pressing the IDL VM icon will first display a splash screen; you must click on this screen to proceed. When you have done so, a new file browser screen will appear. Navigate to the directory containing the bcomfi.sav file and select this file by double-clicking or selecting it into the “File name:” field. The file browser will then disappear and the BCOMFI program will be launched.

For convenience, it is a good idea to edit the “Properties” field of the IDL VM screen icon, select the “Shortcut” tab, and set the “Start in:” field to the directory containing the bcomfi.sav file. This will allow fast launching of the BCOMFI system.

The BCOMFI system consists of an initial GUI (the “Main Menu”) that allows selection of a number of processing tasks. Pressing a button to select a desired simulation, conversion, preprocessing, or ambiguity function/MFI task brings up another GUI containing a number of fields or conditions that you can specify for the run. The simulation / conversion / preprocessing GUIs have been set up so that they can be run using default values, and each run will result in a data file being produced. The ambiguity function and search/optimization GUIs require user specification of parameters before a run can be launched. The contents of certain of these data files can be visualized or viewed using the appropriate button in the initial GUI, as described later in this report.

## Setting up for Parallel Processing

To allow parallel processing of multi-frequency matched field computations on a network of Windows or Linux computers, the following additional installation steps are needed.

- First, after installing the IDL VM on the master computer, copy the files “idl_tools_nodelay.dll” and “idl_tools_nodelay.dlm” from the \bin directory created above to the C:\rsi\idlXX\bin\bin.x86 directory on the client computer, where “XX” is the corresponding IDL version number (e.g., idl62).

- Then, for each (remote) Windows computer in the network:

    - Create a directory C:\Program Files\Orca.
    - Copy the two executables “orcaserver.exe” and “orca.exe” and the batch file “orca.bat” from the \bin directory created above into the newly created directory on the remote computer.

- Then, for each (remote) Linux computer in the network:
    - Copy the executables “orcaserver” and “orca90” from the \linux directory created above to /usr/local/bin on the remote computer. You will need root permission to do this.
    - Copy the tar file “fortranlib.tar.gz” from the \linux directory created above to /usr/local/lib on the remote computer. You will need root permission to do this.

- cd to /usr/local/lib on the remote computer, and run the two commands “gunzip fortranlib.tar.gz” and “tar xf fortranlib.tar”.
- Make sure that the directory /usr/local/lib is in included in the value of the LD_LIBRARY_PATH environment variable for the session running orcaserver on the remote machine. To set LD_LIBRARY_PATH in the C shell (csh), type “setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/usr/local/lib”. To set this environment variable in the bash shell, type
  “ export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/usr/local/lib”.
  To check the value of LD_LIBRARY_PATH, type
  “echo $LD_LIBRARY_PATH”. You may wish to include the setenv or export command in your account's .cshrc (for C shell) or .bashrc (for bash) file so that it does not need to be entered each time orcaserver is run.

- To actually run the server on either Windows or Linux remote computers, cd to the directory containing the orcaserver executable on the remote computer and enter “orcaserver *N*”, where *N* is the remote port number to be used (e.g., “orcaserver 1501”). These orcaserver processes must be running on the remote servers for parallel processing to be enabled. Note that the port numbers can be arbitrarily chosen but must correspond to those selected in the matched field interfaces described in later sections. Also note that the same port numbers can be used with the different computers, providing the numbers in the orcaserver commands on the remote computers agree with the entries in the interfaces.

# Overview of BCOMFI Functionality

## Processing Components

The BCOMFI system consists of 16 separate components for simulation, data conversion, preprocessing, ambiguity function generation, MFI, and display. These components, which correspond to the initial GUI, are listed and briefly described in this section, while the following section gives some representative examples of processing tasks that may be accomplished using these components in a sequence. In the following descriptions and in later sections, “TD” denotes time domain, while “FD” denotes frequency domain.

***Simulate TD shot data***. This component uses ORCA to generate the impulse responses that would be observed at the sensors of an array for the specified geoacoustic environment and source-array geometry. The resulting traces are written to a Shot file, which can then be viewed or processed into a Ship file or an Hpdt file.

***Simulate ship FD hpdt data***. This component uses ORCA or RAM to generate frequency domain data at selected frequencies that would be observed at the sensors of an array for the specified geoacoustic environment and source-array geometry (with one or two moving sources). Noise of various types can be added to this data, and cross-spectral matrices can be generated from sequential segments. The resulting complex data are written to an Hpdt file, which can be used for input to ambiguity

function generation and MFI.

***Convert TD shot data to TD ship data***. This component takes the impulse responses at the sensors in a Shot file and applies a separate phase randomization at each frequency (but not between sensors) in order to generate simulated time domain ship data with the same spectra as the traces in the Shot file. The resulting traces are written to a Ship file, which can then be viewed or processed into an Hpdt file.

***Convert p16 data to shot or ship data***. This component reads in real data files in p16 format (a binary integer format in which the data from the actual hydrophone arrays are collected and stored) and converts the data in each file to Shot or Ship format, as specified by the user. These can then be processed into Hpdt files.

***Preprocess TD shot data to FD hpdt data***. This component reads in a Shot file, FFTs the impulse responses for the sensors, forms "measured" cross-spectral matrices within a specified band for each of a number of selected frequencies, and writes the result to an Hpdt file.

***Preprocess TD ship data to FD hpdt data***. This component reads in a Ship file, FFTs the trace data according to the specified segmentation, accumulates "measured" cross-spectral matrices within a specified band for a number of selected frequencies, and writes the results to an Hpdt file.

***Generate 1D ambiguity function***. This component inputs an Hpdt file and generates an ambiguity function (matching function) as a single parameter of the 23-parameter model is varied between specified limits, with fixed values for the other parameters. The result is written to an Am1d file.

***Generate 2D ambiguity function***. This component inputs an Hpdt file and generates an ambiguity function (matching function) as two parameters of the 23-parameter model are varied between specified limits, with fixed values for the other parameters. The result is written to an Am2d file. This component also provides a means of generating precomputed depth-range or range-bearing replica vectors (in the form of a Repl file) to be used in the monitoring.

***Generate 3D ambiguity function***. This component inputs an Hpdt file and generates an ambiguity function (matching function) as three parameters of the 23-parameter model are varied between specified limits, with fixed values for the other parameters. The result is written to an Am3d file.

***Perform MFI search/optimization***. This component inputs an Hpdt file and performs matched field inversion of selected parameters of the 23-parameter model using a global search / local optimization technique with multiple starting points optimized until convergence. The estimates are grouped to reduce nonuniqueness, and 1D ambiguity functions of the specified parameters passing through the best estimate are computed to characterize the parameter space. The results are written to an Mfop file. This

component also provides a means of generating a Sopt (search/optimization) file for input by the monitoring component; this file contains the selected conditions to be used in the search/optimization stage of the monitoring algorithm.

***Perform MFI monitoring***. This component inputs a sequence of Hpdt files and analyzes each one to estimate selected geoacoustic parameters from each set by a two-stage process. In the first stage, a 2D depth-range (range-independent – ORCA option) or range-bearing ambiguity function (range-dependent – RAM option) is generated using precomputed replicas. In the second stage, the maximum in this function is used as input to the search / optimization algorithm to estimate a multi-parameter geoacoustic model. This process is repeated for each data set. . The results are written to a Moni file.

***Display shot data***. This component inputs a Shot file and displays the traces for the sensors in one waterfall plot and their corresponding spectra in another.

***Display ship data***. This component inputs a Ship file and displays the traces for the sensors in one waterfall plot and their corresponding spectra in another.

***Display 1D ambiguity function***. This component reads in an Am1d file and displays the 1D ambiguity function, using IDL's iplot component of its itools suite of interactive visualization utilities.

***Display 2D ambiguity function***. This component reads in an Am2d file and displays the 2D ambiguity function, using IDL's isurface or iimage components of its itools suite of interactive visualization utilities.

***Display 3D ambiguity function***. This component reads in an Am3d file and displays the 3D ambiguity function, using IDL's ivolume component of its itools suite of interactive visualization utilities or its slicer3 volume visualization tool.

***Display MFI search/optimization results***. This component reads in a Mfop file and allows visualization of:

- the matching functions computed during the search stage;
- the evolution of the objective function and parameters during any of the multiple convergences;
- the 1D ambiguity functions at the best optimum found.

It also displays text information describing the groups of optima found and the characteristics of the parameter space at the best optimum.

***Display MFI monitoring results***. This component reads in a Moni file and allows visualization of the time evolution of the mismatch function and parameter estimates for each of the data sets in the monitoring sequence.

## Examples of Processing using BCOMFI Components

Some examples of how the above processing components of BCOMFI can be used to perform some representative tasks are given in this section. Note that, for brevity, it is assumed that the files required for the runs (e.g., for ORCA and RAM) have been set up. (See later sections for details on which files are involved in the runs and how they should be set up.)

***Determine the impulse response for specified conditions***

- Run the *Simulate TD shot data* component to generate a Shot file.
- View this file using the *Display shot data* component.

***Check the correspondence of ORCA and RAM***

- Run the *Simulate ship FD hpdt data* component, selecting the ORCA option, to generate Hpdt file A.
- Run the *Simulate ship FD hpdt data* component, selecting the RAM option, to generate Hpdt file B.
- Run the *Generate 1D ambiguity function* component for selected parameters, using Hpdt file A, and producing Am1d file C. Also note the numerical values in the text log window of this component.
- Run the *Generate 1D ambiguity function* component for the same selected parameters, using Hpdt file B, and producing Am1d file D. Also note the numerical values in the text log window of this component.
- Compare the text values from the two runs. Also use the *Display 1D ambiguity function* component (twice) to view plots of the data in files C and D. Verify the closeness of the results.

***Check that simulated shot and ship data produce the same matching as direct generation of hpdt data***

- Run the *Simulate TD shot data* component to generate a Shot file.
- Run the *Convert TD shot data to TD ship data* component to input the Shot file and output a corresponding Ship file.
- Run the *Preprocess TD shot data to FD hpdt data* component on the Shot file to produce Hpdt file A.
- Run the *Preprocess TD ship data to FD hpdt data* component on the Ship file to produce Hpdt file B.
- Run the *Simulate ship FD hpdt data* component under equivalent conditions to produce Hpdt file C.
- Run the *Generate 1D ambiguity function* component for selected parameters, using Hpdt files A, B and C, and producing Am1d files D, E, and F. Compare the numerical values in the log window as above, and use the *Display 1D ambiguity function* component to view plots of the data in files D, E and F. Verify the closeness of the results.

***Perform MFI on p16 data obtained for several shots***

- Run the *Convert p16 data to shot or ship data* component, selecting the shot option to generate Shot files from the selected input files.
- For each Shot file, run the *Preprocess TD shot data to FD hpdt data* component to generate an Hpdt file.
- For each Hpdt file, run the *Perform MFI search/optimization* component to generate an Mfop file (this is a time-consuming process).
- View the results of the inversions using the *Display MFI search/optimization results* component.

# Data Files

## BCOMFI Data Files

The files in the "\data" directory have names of the form "xxxxnnnnn.ttt" and contain input and output data that correspond to internal data structures used by the programs. Files of type ".dat" are user-editable while those of type ".sav" are non-editable IDL save files. A brief description of the data files and their use is given here.

***Am1dnnnnn.sav***. This file is produced by the "Generate 1D ambiguity function" process. It contains the conditions used for generating the 1D ambiguity function (i.e., the values for fixed parameters, and the values for the parameter varied) as well as the results of the run (i.e., the matches for each value of the varied parameter, computed using the Bartlett processor). The ambiguity function can then be visualized by pressing the "Display 1D ambiguity function" button in the Main Menu interface.

***Am2dnnnnn.sav***. This file is produced by the "Generate 2D ambiguity function" process. It contains the conditions used for generating the 2D ambiguity function (i.e., the values for fixed parameters, and the values for the two parameters varied) as well as the results of the run (i.e., the matches for each pair of varied parameters, computed using the Bartlett processor). The 2D ambiguity function can then be visualized by pressing the "Display 2D ambiguity function" button in the Main Menu interface.

***Am3dnnnnn.sav***. This file is produced by the "Generate 3D ambiguity function" process. It contains the conditions used for generating the 3D ambiguity function (i.e., the values for fixed parameters, and the values for the three parameters varied) as well as the results of the run (i.e., the matches for each triplet of varied parameters, computed using the Bartlett processor). The 3D ambiguity function can then be visualized by pressing the "Display 3D ambiguity function" button in the Main Menu interface.

***Araynnnnn.dat***. This input file contains the specification for the array to be used in the simulations. Note that the arrays are used by the simulation, ambiguity function and MFI options, but not by the conversion, preprocessing, and display options. These text files are user-editable – see the example in \data\aray00000.dat.

***Grpsnnnnn.dat***. This file is produced by the "Display MFI search/optimization result"

process, and is derived from data in an Mfop file. It contains the run conditions and a summary of the results of the grouping and peak analysis stages of a run of the MFI search/optimization. It contains the main results of the MFI run, in the form of a list of the various grouped convergences (optima) that were found, as well as the statistical properties for each group, when the group contained two or more convergences to the same optimum.

***Hpdtnnnnn.sav***. This file is produced by the "Simulate ship FD hpdt data", "Preprocess TD shot data to FD hpdt data", and "Preprocess TD ship data to FD hpdt data" processes. It contains the conditions used for simulation and the complex field vectors or cross-spectral matrices for the array at user-selected frequencies. It can also contain a sequence of such data sets, at each of a number of times as specified by the user. Hpdt files contain the acoustic field data that will be used for MFI.

***Mfopnnnnn.sav***. This file is produced by the "Perform MFI search/optimization" process. It contains values used in the setup of the search/optimization, information about the fixed and varied parameters, the results for the global search stage, the results of each separate local optimization, grouping of the optima, and 1D ambiguity functions through the best optimum found. This information can be viewed or visualized by pressing the "Display MFI search/optimization results" button in the Main Menu interface.

***Moninnnnn.sav***. This file is produced by the "Perform MFI monitoring" process. It contains a Moni structure, which includes the input conditions and a series of Mfop structures, each of which contains the output of a search/optimization run on a separate data set in the sequence of input data. This information can be viewed or visualized by pressing the "Display MFI monitoring results" button in the Main Menu interface.

***Replnnnnn.sav***. This file can be produced by the "Generate 2D ambiguity function" process. It contains complex replica data in the form of a 4D array, where the dimensions are sensor, frequency, depth, and range (range-independent – ORCA) or sensor, frequency, range, and bearing (range-dependent – RAM). The file is then used as precomputed replica vectors by the monitoring process.

***Shipnnnnn.sav***. This file is produced by the "Convert TD shot data to FD ship data" option and can also be used for input by the "Preprocess TD ship data to FD hpdt data" option. It contains the conditions used for simulation and the simulated time domain ship data traces (generated from shot traces). These traces can be visualized by pressing the "Display ship data" button in the Main Menu interface.

***Shotnnnnn.sav***. This file is produced by the "Simulate TD shot data" option and can also be used for input by the "Convert TD shot data to FD ship data" and "Preprocess TD shot data to FD hpdt data" processes. It contains the conditions used for simulation and the simulated time domain shot data traces. These traces can be visualized by pressing the "Display shot data" button in the Main Menu interface.

***Soptnnnnn.sav***. This file can be produced by the "Perform MFI search/optimization" process. It provides a convenient means of specifying the contents of the Params and Mfop structures that together define the conditions to be used in the search/optimization stage of monitoring. The file is input by the monitoring process.

***Wavennnnn.dat***. This input file contains the specification for the wavelet to be used in the generation of shot traces by the "Simulate TD shot data" option. These text files are user-editable – see the example in \data\wave00000.dat.

***Wsspnnnnn.dat***. This input file contains the water-column sound speed profile to be used by the ambiguity function generation and search/optimization MFI procedures. The data format consists of the number of points in the profile, and the (depth, sound speed) pairs defining the profile, one pair per line. These text files are user-editable – see the example in \data\wssp00000.dat.

## ORCA Data Files

### \Source Directory

***Orca_in***. Produced automatically by all runs involving ORCA. Contains the "_opt" and "_svp" files to use for the run.
***Orca_status***. Produced automatically by all runs involving ORCA. Contains the exit status of the ORCA run.

### \Inputs Directory

***Sim_shot_svp***. Must be provided by the user prior to an ORCA run simulating TD shot data, and contain the sound velocity profile (water and bottom layers/halfspace) for the environment. An example file is provided in \inputs\sim_shot_svp.
***Sim_ship_svp***. Must be provided by the user prior to an ORCA run simulating ship FD hpdt data, and contain the sound velocity profile (water and bottom layers/halfspace) for the environment. An example file is provided in \inputs\sim_ship_svp.
***Mfi_svp***. Produced automatically by all ORCA runs involving ambiguity function generation or search/optimization.

***Sim_shot_opt***. Produced automatically by runs involving simulating TD shot data. Contains options for the ORCA run.
***Sim_ship_opt***. Produced automatically by runs involving simulating ship FD hpdt data using ORCA. Contains options for the ORCA run.
***Mfi_opt***. Produced automatically by all runs involving ambiguity function generation or search/optimization. Contains options for the ORCA run.

***Sim_shot_array_geom***. Produced automatically by runs involving simulating TD shot data. Contains the array geometry for the ORCA run.

***Sim_ship_array_geom***. Produced automatically by runs involving simulating ship FD hpdt data. Contains the array geometry for the ORCA run.
***Mfi_array_geom***. Produced automatically by runs involving ambiguity function generation or search/optimization. Contains the array geometry for the ORCA run.

### \Outputs Directory

***Sim_field.dat***. Produced automatically by ORCA runs involving simulating ship FD hpdt data. Contains the complex-valued fields at the sensors of the array, which are then read in by the IDL code.
***Mfi_field.dat***. Produced automatically by ORCA runs involving ambiguity function generation or search/optimization. Contains the complex-valued fields at the sensors of the array, which are then read in by the IDL code.

***Sim_modes***. Produced automatically by ORCA runs involving simulating ship FD hpdt data. Contains the properties of the modes (not used).
***Mfi_modes***. Produced automatically by ORCA runs involving ambiguity function generation or search/optimization. Contains the properties of the modes (not used).

***Sim_out***. Produced automatically by ORCA runs involving simulating ship FD hpdt data. Contains the input data for the run and a summary of the number of modes and timing data (not used).
***Mfi_out***. Produced automatically by ORCA runs involving ambiguity function generation or search/optimization. Contains the input data for the run and a summary of the number of modes and timing data (not used).

***Sim_shot_fft***. Produced automatically by ORCA runs involving simulating TD shot data. Contains a Fortran binary version of the FFT file output by ORCA for the run (not used).
***Sim_shot_fft.dat***. Produced automatically by ORCA runs involving simulating TD shot data. Contains ASCII output constituting the FFT file output by ORCA for the run, which is then input by the BCOMFI code that generates the impulse response.

## RAM Data Files

### \Source Directory

***Ramgeo.dat***. Must be provided by the user prior to a RAM run simulating ship FD hpdt data, and contain the sound velocity profile (water and bottom layers/halfspace) for the environment in the same format as that in the input file ramgeo.in. Ramgeo.dat acts as a source and template for the setting of parameters other than frequency, source range and source depth (which are set during BCOMFI simulation), and allows the automated generation of ramgeo.in files during the RAM runs. An example file is provided in \source\ramgeo.dat.

***Ramgeo.in***. Produced automatically just prior to each RAM run simulating ship FD hpdt data and used as input for the RAM run.

### \Ramout Directory

***Field.dat***. Produced automatically by RAM runs. Contains a Fortran binary version of the complex fields computed during the RAM run (not used).
***Field.out***. Produced automatically by RAM runs. Contains an ASCII version of the complex fields computed during the RAM run, which is then used by BCOMFI to compute the fields at the sensors of the array.
***Tl.grid***. Produced automatically by RAM runs. Contains a Fortran binary version of the transmission loss grid computed during the RAM run (not used).
***Tl.line***. Produced automatically by RAM runs. Contains an ASCII version of the transmission loss along a constant-depth line computed during the RAM run (not used).

## P16 Data Files

These files, which will contain real data from the array, must have names of the form pnnnnn.p16, where nnnnn is a five-character identification (ID) number. However, they can be placed in any user-accessible directory, as they are specified interactively during a run using a special purpose "pickfile" dialog box. Example files are provided in the directories \p16_data\ship and \p16_data\shot.

## Servers Data File

The file "servers.dat" must be present in the \source directory in order to run the matched field computations. It contains the default IP addresses and ports for each server process to be run on the remote computers, and can be edited to correspond to the distributed environment to be used. If the parallel processing option is not selected, the information in this file is simply ignored. An example of the file structure is provided in the \source directory of the software delivery.

## DEM Data File

The file "dem.dat" must be present in the \source directory in order to perform modeling using digital elevation model for the bathymetry. It contains the start and end extents of the gridded DEM, the number of points and increments in the x and y directions, and the DEM values.

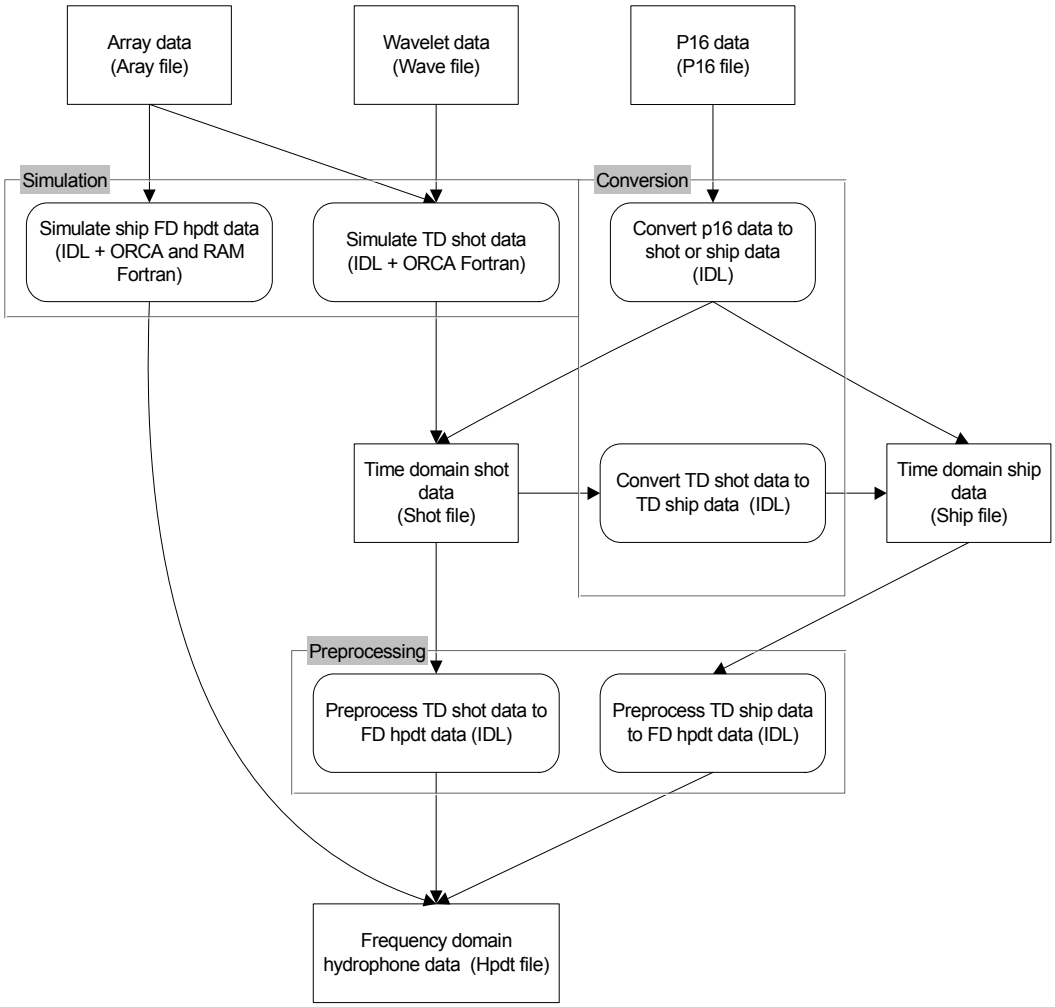
# BCOMFI Software: Main Menu

The Main Menu provides an interface for invoking any of the functional components of

the BCOMFI system listed in the Processing Components section above, i.e., the simulation, conversion, preprocessing, ambiguity function generation, MFI, monitoring and display components. Note that, in general, the result of using the simulation, conversion and preprocessing components is ultimately an Hpdt file. This Hpdt file is then used as input to the ambiguity function generation and MFI components.

## Overall Design: Simulation, Conversion and Preprocessing Components

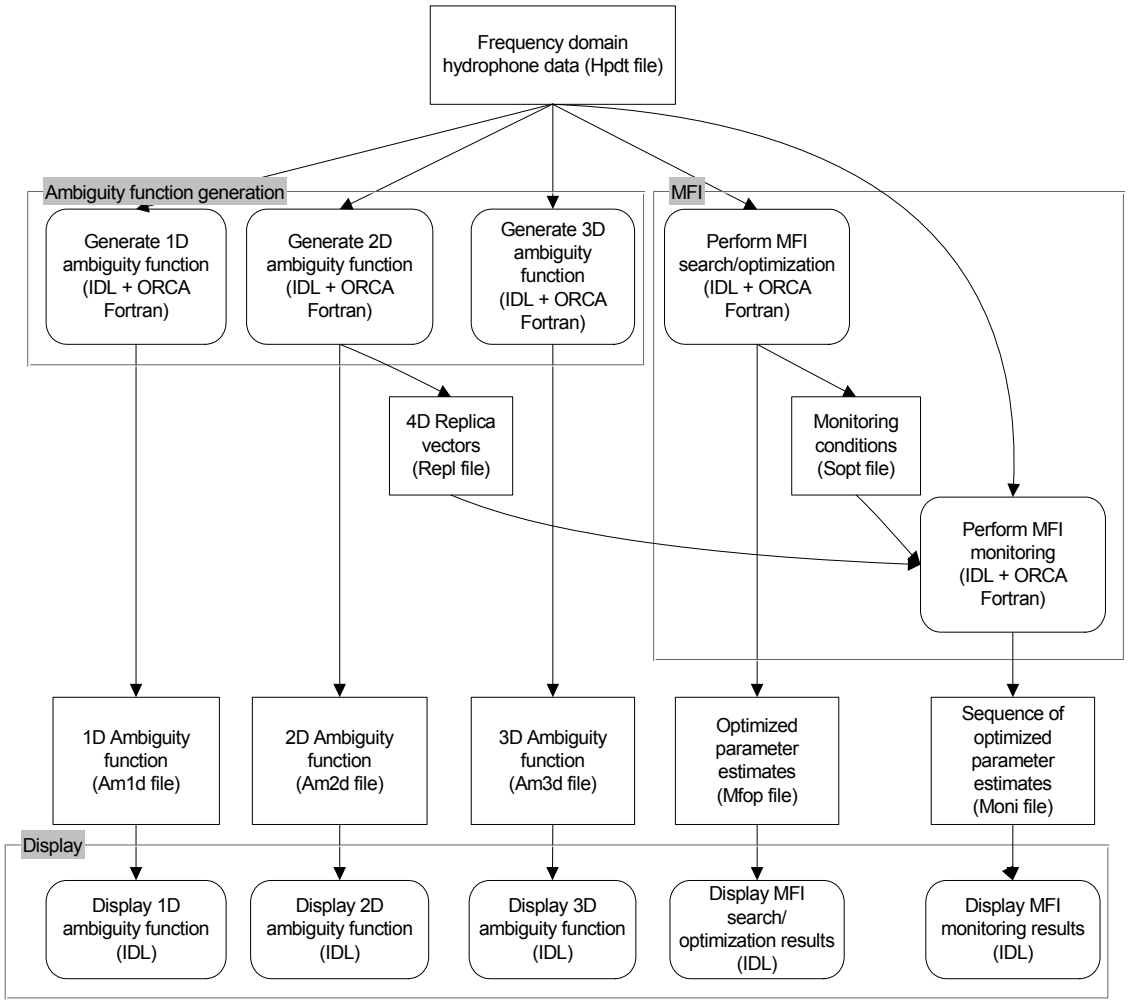
In this design, data and/or files are indicated by rectangles and processes are represented by rounded rectangles.




**Overview of BCOMFI Simulation, Conversion and Preprocessing Components**

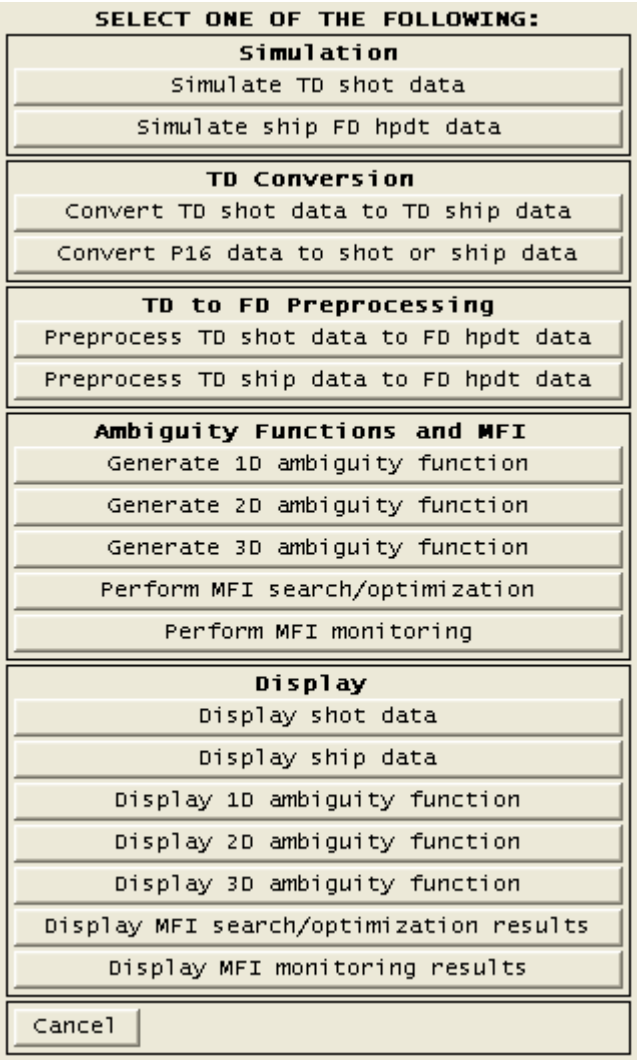
## Overall Design: Ambiguity Function Generation, MFI and Display Components

In this design, data and/or files are indicated by rectangles and processes are represented by rounded rectangles.




**Overview of BCOMFI Ambiguity Function Generation, MFI and Display Components**

## Graphical User Interface




Pressing one of the buttons (other than Cancel) brings up another interface (described below) that allows the specification of the conditions for a run of that selected component of BCOMFI.

### Simulation of Time Domain Shot Data

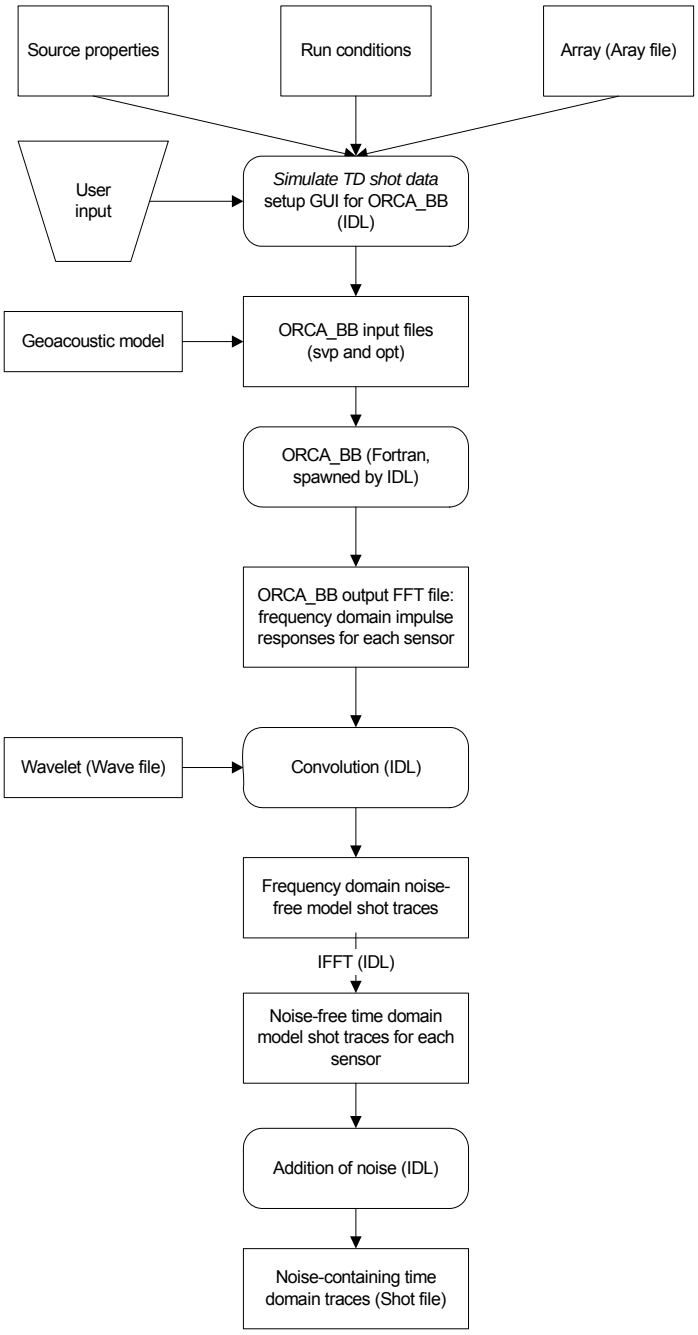
In order to develop and verify methods for MFI of the shot data to be obtained, it was essential to first be able to generate synthetic shot traces for specified source-array geometries and geoacoustic environments and use them to test the ambiguity function generation and MFI methods. These synthetic data consist of impulse responses (generated by propagation modeling) convolved with a representative source wavelet, with additive noise. The resulting model traces can then be FFT'd and selected frequencies used to perform and evaluate the subsequent MFI processing.

In the *Simulate TD shot data* component of the BCOMFI system, the broadband option of ORCA is used to generate the impulse responses. This option allows the user to specify a frequency band (e.g., 1 – 250 Hz), a sampling frequency, and a time window (or equivalently, a number of FFT points). ORCA then generates the frequency domain impulse response for each discrete frequency in the specified band, and outputs this result to an FFT file. This file is then read in and convolved with a specified (shot) waveform. An option to add Gaussian noise is provided, and the resulting frequency domain traces are then inverse FFT'd to yield the corresponding time domain traces.

Implementation of this software for simulating shot data allowed the methods for processing the real shot data to be developed and tested in advance of the real data becoming available.

## Overall Design

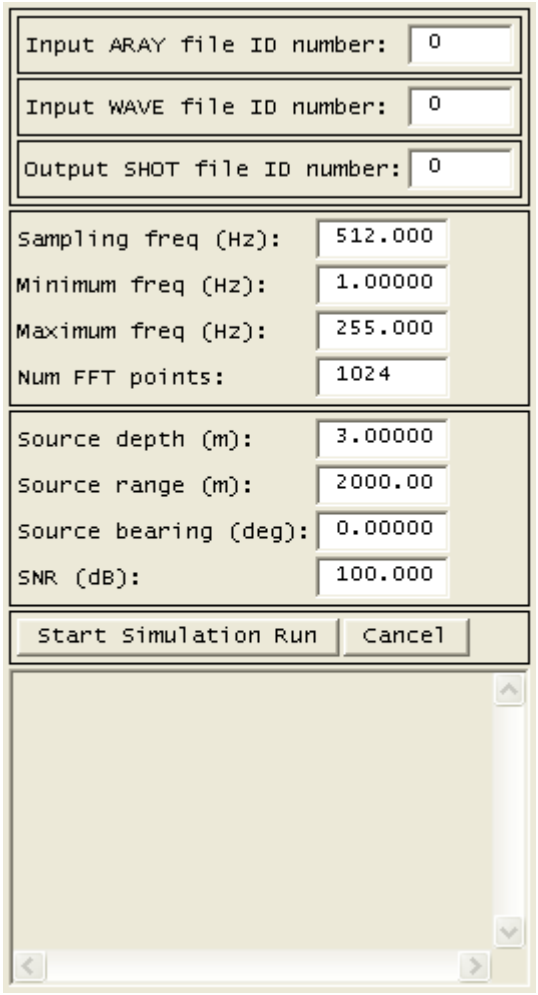
In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.




**Processing for**
***Simulate TD shot data***

## Graphical User Interface

The GUI for *Simulate TD shot data* is illustrated in the following diagram.




- The first three items allow the user to specify the input files (Aray and Wave) to be used for the run, and the Shot file to contain the results.
- The next four items allow the user to specify the conditions for the broadband ORCA run.
- The following four items (in the “Source” box) allow the user to specify the source-receiver geometry to be used in the run and the (power) signal-to-noise ratio (dB) for each trace.
- The “Start Simulation Run” and “Cancel” buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for simulating the shot traces is outlined in the following pseudocode:

*Set up an svp (sound velocity profile) file for ORCA to use as the geoacoustic environmental model.*
*Set up an Aray file to contain the array to be used in the simulation and a Wave file to contain the wavelet.*
*Using the GUI, specify the files to be used as the input array (Aray) and wavelet (Wave) files and the output Shot file, and specify the run conditions for the source and ORCA broadband.*
*Check that the run conditions are consistent.*
*Apply a tilt to the array.*
*Generate an array geometry file and an ORCA options file.*
*Spawn a process to perform an ORCA broadband run:*
*ORCA process:*
*For each frequency:*
*Compute the complex fields that would be observed at each sensor for a source at the specified position.*
*Output an ORCA FFT file containing the complex fields for the impulse responses for each of the sensors.*
*Read in the FFT file generated by ORCA.*
*Read in and FFT the wavelet from the specified Wave file.*
*Generate time domain traces: For each sensor:*
*Reflect/conjugate the FFT about the Nyquist frequency to obtain the Fourier transform of the impulse response.*
*Convolve the wavelet with the impulse response.*
*Inverse FFT the result to give a time domain trace.*
*Add Gaussian noise at specified signal-to-noise ratio.*
*Output the traces to a Shot file.*

## Using the Component

Press the “Simulate TD shot data” button to bring up the GUI for generating time domain traces that would be observed at the sensors of an array for a particular source-array geometry, geoacoustic model and source wavelet. The process uses the broadband option of ORCA to generate, for each of the sensors, the complex fields at each frequency within the specified band that would be observed on the basis of the geometry and geoacoustic model. These complex vectors are written to an FFT file by ORCA, and this file is then read in and the sensor data are multiplied with the Fourier

transform of the wavelet. The result is inverse Fourier transformed and written to a Shot save file.

To run this model, first ensure that the file "sim_shot_svp" exists in the \data directory noted above, and that it contains the desired sound velocity profile in ORCA format. Then specify the following information in the GUI:

- the ID number of the Aray file containing the array to be used (sensor (*x, y, z*) points, tether depth for the *z* points, and tilt data);
- the ID number of the Wave file containing the wavelet to be used;
- the ID number of the output Shot file that will be produced;
- the sampling frequency, which should be at least twice the maximum frequency specified below;
- the minimum frequency in the band for which the complex fields are to be generated by ORCA;
- the maximum frequency in the band for which the complex fields are to be generated by ORCA (this should be at most half the sampling frequency);
- the number of FFT points; this should be long enough so that the entire trace is contained within the time window (i.e., number of FFT points ÷ sampling frequency); otherwise the traces may wrap around;
- the source depth and its range and bearing with respect to the array;
- the SNR (signal-to-noise ratio) for addition of white noise to the traces; note that for SNR ≥ 100, no noise is added.

When the above have been specified, press the "Start Simulation Run" button to perform the simulation. This will likely take several minutes. The progress of the ORCA broadband run can be monitored by viewing the DOS / Command Prompt window generated by the spawning process by which IDL invokes ORCA.

## Simulation of Frequency Domain Ship Data

As with the shot data, it was essential to have a software component that could generate simulated data in order to test the correctness and performance of the MFI algorithms during development. These synthetic data consist of complex signal vectors or cross-spectral matrices at several user-selected frequencies, each of which can optionally contain noise corresponding to various models. Since there may be more than one actual source, and these sources will generally be moving, the simulation also includes the ability to model and generate data for two moving sources.

For a range-independent environment, ORCA (in standard rather than broadband mode) was implemented to generate the fields that would be observed at the sensors for a simulated moving source. However, in view of the fact that the environment has a significant range-dependence, it was also desirable to be able to simulate data for a sloping range-dependent environment, or for an environment with bathymetry defined by a DEM, and to analyze this data with MFI. As noted above, the range-dependent parabolic equation (PE) code RAM provides the ability to generate data for range-

dependent environments, and is available at no cost over the web. This RAM code was incorporated into the software system for generating synthetic data. The availability of two entirely separate propagation modeling codes provided a consistency/validity check by allowing comparison of the fields that are produced by the two models for a range-independent environment.

Hence, both ORCA and RAM options were implemented to generate acoustic fields at multiple frequencies for moving sources. Options were also provided for adding noise and performing cross-spectral matrix estimation. This implementation allowed the methods for processing the real ship data to be developed and tested in advance of the real data being available.

## Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.

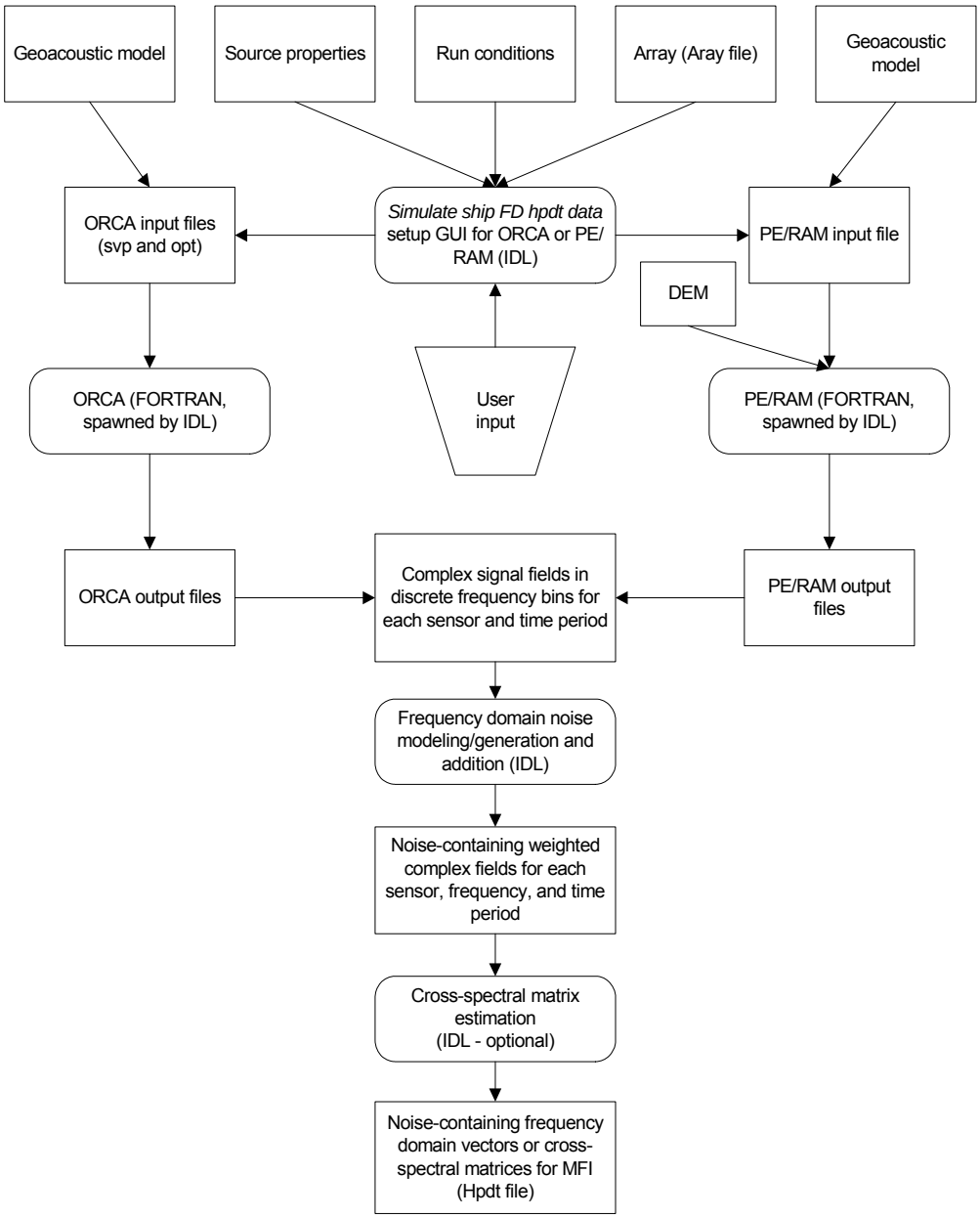
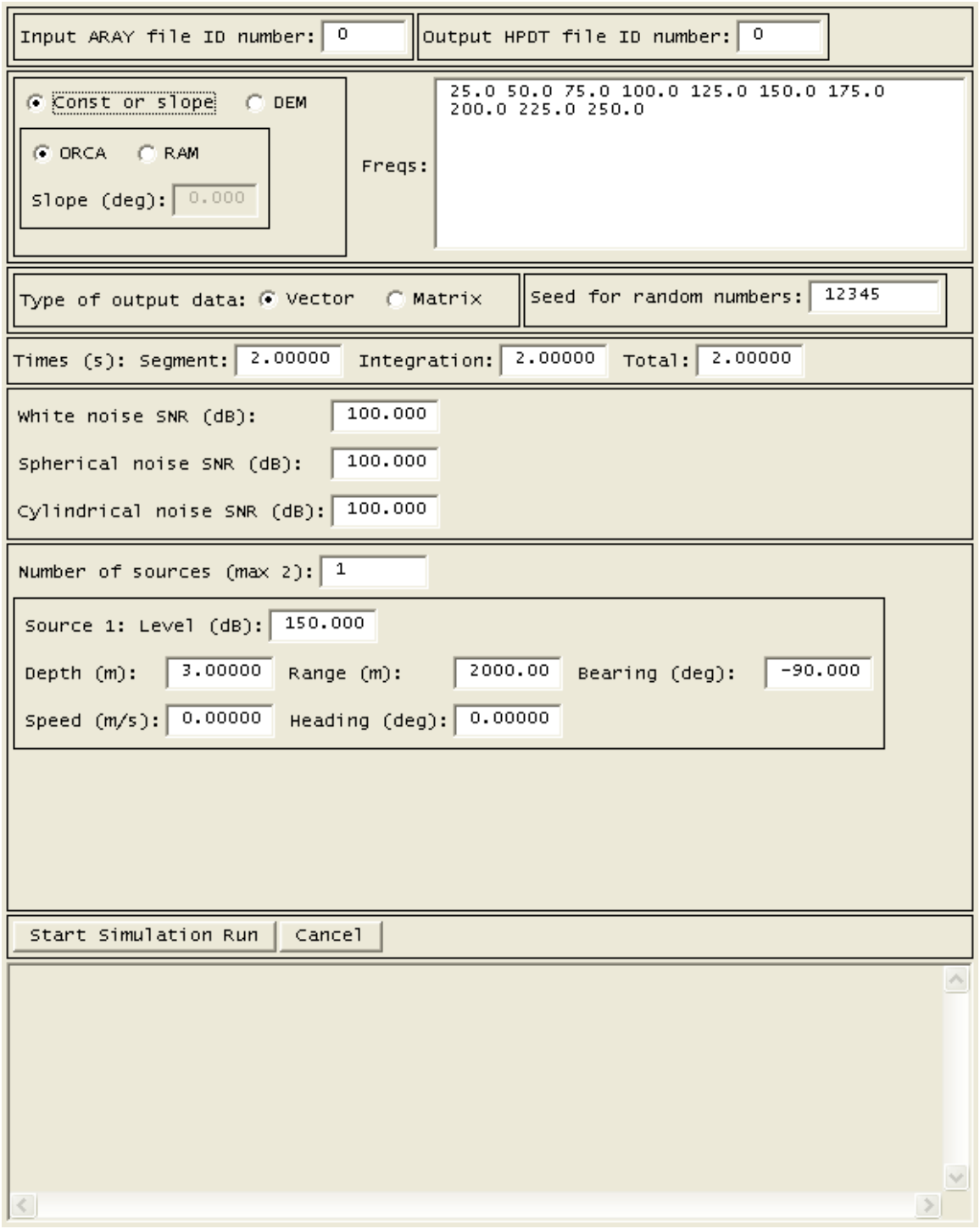



**Figure 17**

## Graphical User Interface




- The first items in the top row allow the user to specify the input Aray file to be used for the run, and the Hpdt file to contain the results.

- The second row allows the user to specify the environment / propagation model and to select the frequencies at which the data are to be generated. The area on the left indicates whether on the one hand, a constant-depth or constant-slope environment, or on the other, a DEM for the bathymetry, is to be used. If the former case is chosen, ORCA or RAM can be used if the environment is range-independent, while RAM can be used for a sloping environment. If the latter option (DEM) is chosen, RAM will be used, and the contents of the screen area change to allow input of additional conditions for the run.
- The third row allows the user to specify whether the data are to be output as vectors or cross-spectral matrices, and to enter a seed for the random number sequence to be used in generating noise realizations. The latter allows repeated runs involving noise to be reproducible.
- The fourth row provides for specification of the segment time, the integration time (the time over which cross-spectral matrix estimation takes place), and the total time (which controls how many vectors/matrices are computed).
- The fifth row provides options for adding noise of specified signal-to-noise ratio from three different distributions: white, spherical, and cylindrical.
- The sixth row allows one or two sources to be specified.
- The seventh row (with the “Source 1” label) allows the user to specify the source level and the source-receiver geometry for Source 1 to be used in the run.
- The eighth row (if present) allows the user to specify the source level and the source-receiver geometry for Source 2 to be used in the run.
- The “Start Simulation Run” and “Cancel” buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for simulating the ship data is outlined in the following pseudocode:

*Set up an svp (sound velocity profile) file for ORCA and/or RAM, to use as the geoacoustic environmental model.*
*Set up an Aray file to contain the array to be used in the simulation.*
*Set up a dem file if this option is to be used.*
*Using the GUI, specify the input Aray file and the output Hpdt file, and select the run conditions.*
*Check that the run conditions are consistent.*
*Apply a tilt to the array.*
*For ORCA, generate an array geometry file and an options file.*
*For each frequency specified:*
    *Generate and Cholesky decompose the noise matrices for white, spherical, cylindrical, and modal noise at that frequency.*
*For each integration time:*
    *For each segment time:*

*For each source:*
*Compute new source position.*
*For each frequency:*
*Run ORCA or RAM to generate a signal vector.*
*Scale the signal vector, generate a complex constant and use it to randomize the phase across the array, for that frequency.*
*For each frequency:*
*Generate a random noise vector for the array sensors using the noise cross-spectral matrix for that frequency and array.*
*Add noise vector to signal vector.*
*For each frequency:*
*Accumulate cross-spectral matrices if this option is chosen.*
*Output the vector/matrix data to an Hpdt file.*

## Using the Component

Press the "Simulate ship FD hpdt data" button to bring up the GUI for generating data vectors or cross-spectral matrices, at selected frequencies, that would be observed at the sensors of an array for a particular source-array geometry (where the source can be moving) and geoacoustic model. For a moving source, a quasi-stationary assumption is used, so that the cross-spectral matrices are formed by averaging a number of stationary matrices computed for the source at points along a linear track. The vector or matrix data generated by the run are written to an Hpdt file for input to MFI.

The process uses ORCA or RAM to generate frequency domain pressure field values for each of the sensors, at the selected frequencies. If the constant-or-slope option is chosen, then either ORCA or RAM options may be selected. If ORCA is selected, then the geoacoustic model is range-independent. If RAM is selected, a constant slope may be specified for the bathymetry between the source and the array. If the DEM option is chosen, then RAM along with a DEM for the region is used; the location of the array in the DEM, along with the range step through the DEM, may be specified in the fields that then appear. Whenever RAM is chosen, an image of the 2D output transmission loss is displayed at the end of the run for each frequency. Note that runs with RAM are much more time-intensive than ORCA runs.

To run this model, first ensure that, if ORCA is to be used, the file "sim_ship_svp" exists in the \data directory noted above, and that it contains the desired sound velocity profile in ORCA format. If RAM is to be used, ensure that the file "ramgeo.dat" exists in the \source directory and contains the information for the run conditions and the sound velocity profile in RAM format. Note that the parameters that can be influenced by information provided in the GUI are indicated by uppercase labels (e.g, FREQ, ZS) in ramgeo.dat. Also note that the water depth at the array is defined by the first ZB term in ramgeo.dat. If the DEM option is used, also ensure that the file "dem.dat" exists in the \source directory.

Then specify the following information in the GUI:

- the ID number of the Aray file containing the array to be used (sensor (*x, y, z*) points, tether depth for the *z* points, and tilt data);
- the ID number of the output Hpdt file that will be produced;
- whether a constant-or-slope environment or a DEM is to be used; if the former, specify whether ORCA or RAM is to be used, and, if RAM, specify the slope between the source and the array (positive slopes indicate that the depth increases from the array toward the source); if the latter, specify the location for the array (default is center of the DEM grid) and the range step though the bathymetry grid;
- the frequencies at which the data are to be generated (the default is ten frequencies, evenly spaced between 25 and 250 Hz);
- whether the output data are to be generated in the form of vectors or cross-spectral matrices;
- a seed for simulated noise generation;
- values for segment time (data length for a single FFT), integration time (time within which outer-product matrices are averaged to accumulate cross-spectral matrices), and total time;
- SNRs for three ocean noise types: white, spherical and cylindrical; note that for SNR ≥ 100, no noise of that type is added;
- the number of sources: currently restricted to 1 or 2; if the latter is chosen, a second region for source specification will appear below;
- the characteristics of the source(s): level (dB re 1 μPa at 1 m), depth, range and bearing with respect to the array, speed and heading.

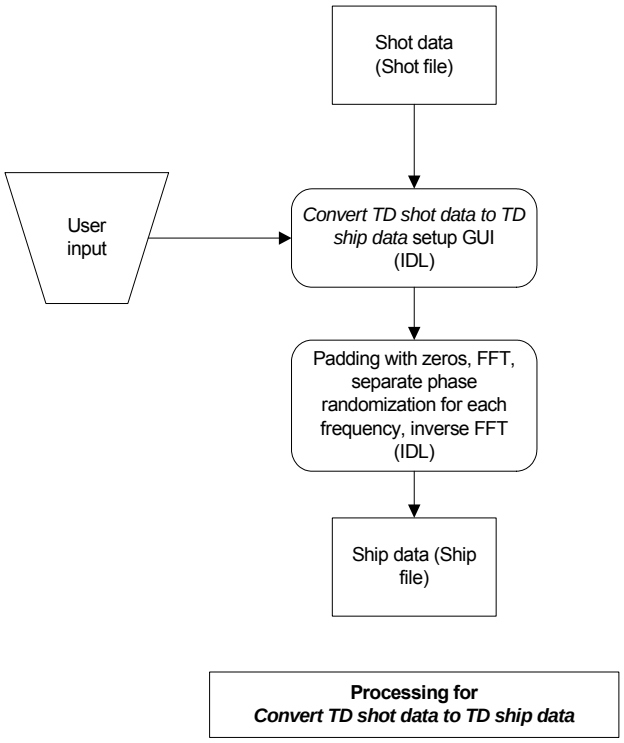
When the above have been specified, press the “Start Simulation Run” button to perform
the simulation. The time taken will depend on the frequencies chosen and on the propagation model: for ORCA, the computations will likely take several seconds, while for RAM, they may take several minutes.

## Conversion of Time Domain Shot Data to Time Domain Ship Data

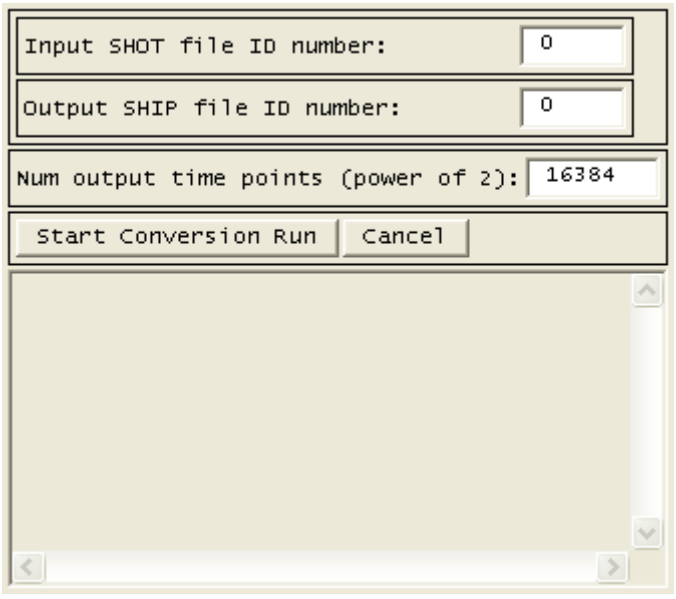
This functionality was developed in order to allow the generation of time domain ship data from previously-generated synthetic shot data for a specified environment. These ship data could then be used as input to a process that would use these data to generate Hpdt files for input to MFI. This software component provided a way of validating the processing of ship data for MFI in advance of the real ship data from the monitoring station being available.

## Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.




## Graphical User Interface



- The first two rows allow the user to specify the input Shot file to be used for the run, and the Ship file to contain the results.
- The next row allows the specification of the number of points in the output time series for each sensor. This should be at least the number of input points in the Shot file and should also be a power of 2 (otherwise, it will be set to a power of 2).
- The "Start Conversion Run" and "Cancel" buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for simulating the ship data is outlined in the following pseudocode:

*Set up a Shot file to contain the array to be used in the simulation.*
*Using the GUI, specify the input Aray file and the output Hpdt file, and select the run conditions.*
*Check that the run conditions are consistent.*
*For each sensor:*
    *Pad the trace with zeroes and FFT.*
*For each frequency:*
    *Generate a random complex constant and multiply it with the frequency bin value for each sensor to randomize the phase.*
*For each sensor:*
    *Inverse FFT the trace.*
*Output the result to a Ship file.*

## Using the Component

Press the "Convert TD shot data to FD ship data" button to bring up the GUI for converting shot traces to the type of data that would correspond more closely to that being collected for ships of opportunity. These data have the same spectra, but are different in that shot traces are coherent in frequency while the ship data are not; both data types are, of course, spatially coherent. In addition, the shot data are confined to a small region of time, while the ship data can go on for an arbitrary length of time. The conversion process involves reading in the shot traces, padding with zeros to increase their length, and Fourier transforming. Then phase randomization is applied to each frequency bin by generating a random rotation and applying the same rotation to each sensor of the array. The result is then inverse transformed to yield the simulated ship data, which is written to a Ship file.

To run this conversion, specify:
- the ID number of the Shot file containing the array to be converted;

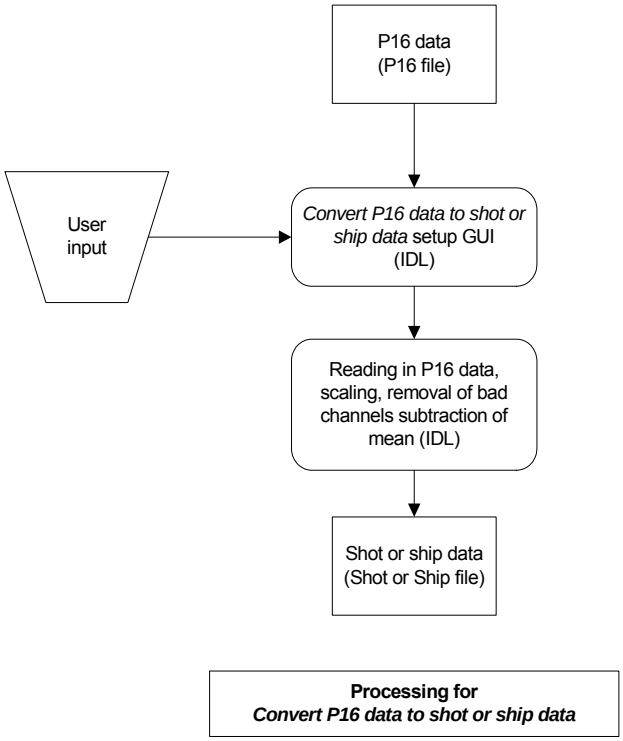
- the ID number of the output Ship file that will be produced;
- the number of time points in the output traces (power of 2).

When the above have been specified, press the “Start Conversion Run” button to perform the conversion. The process should be almost instantaneous.

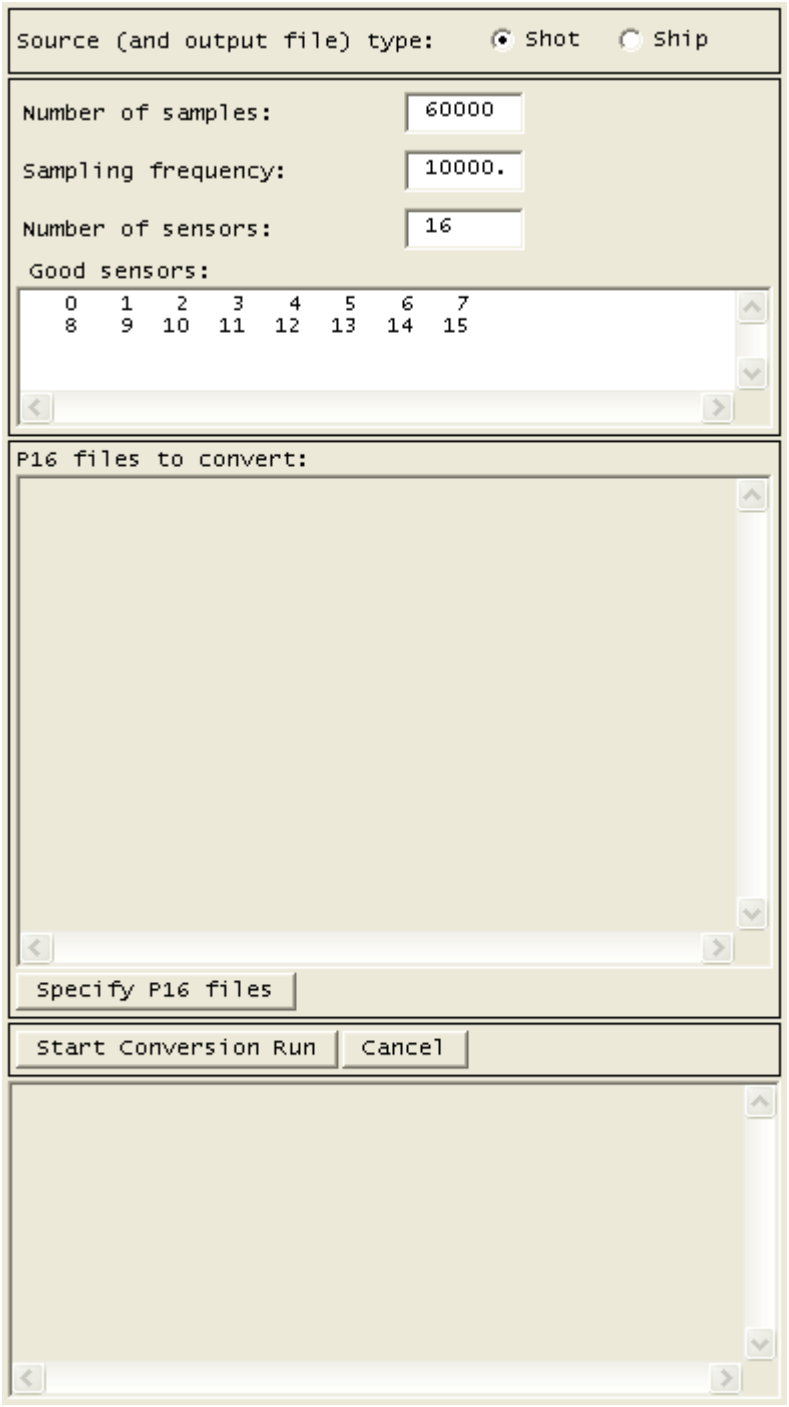
## Conversion of Time Domain P16 Real Data to Time Domain Shot or Ship Data

### Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface




- The first row allows selection of the source and output file type: either Shot or Ship. This should be matched to the type of input data in the P16 file(s).
- The second row allows the user to specify the number of time points that have been recorded for each sensor in the input P16 file(s).

- The third row allows specification of the sampling rate (Hz).
- The fourth row specifies the number of sensors in the input data.
- The fifth row allows the specification of which sensors are “good” and which should be retained in the output file(s).
- The next large initially empty area provides a field for displaying the P16 files selected for conversion.
- The “Specify P16 files” button brings up a dialog box that allows the user to specify which file(s) are to be converted from P16 to Shot or Ship files.
- The “Start Conversion Run” and “Cancel” buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for simulating the ship data is outlined in the following pseudocode:

*Using the GUI, specify the input P16 file(s) and select the run conditions.*
*Check that the run conditions are consistent.*
*For each P16 file:*
    *Read the binary data.*
    *Scale the data to lie between –1 and 1.*
    *Remove bad channels from the data.*
    *For each trace:*
        *Subtract the mean.*
    *Output the traces to a Ship or Shot file with the same ID as the P16 file.*

## Using the Component

Press the “Convert P16 data to shot or ship data” button to bring up the GUI for converting real array data in P16 format to shot or Ship files with the same ID number as the P16 files.

To run this conversion, specify:
- the type of data in the P16 files to be converted – either ship or shot;
- the number of time points in the output traces (power of 2), the sampling frequency, and number of sensors in the array;
- the “good” sensors, for which the data are to be retained;
- the P16 files to be converted, by pressing the “Specify P16 files” button, navigating to the directory containing those files, and selecting the desired ones – note that they should correspond to the specified source type (shot or ship); the selected files will be displayed in the text window above the button;

When the above have been specified, press the “Start Conversion Run” button to perform the conversion. The process should take only a few seconds.

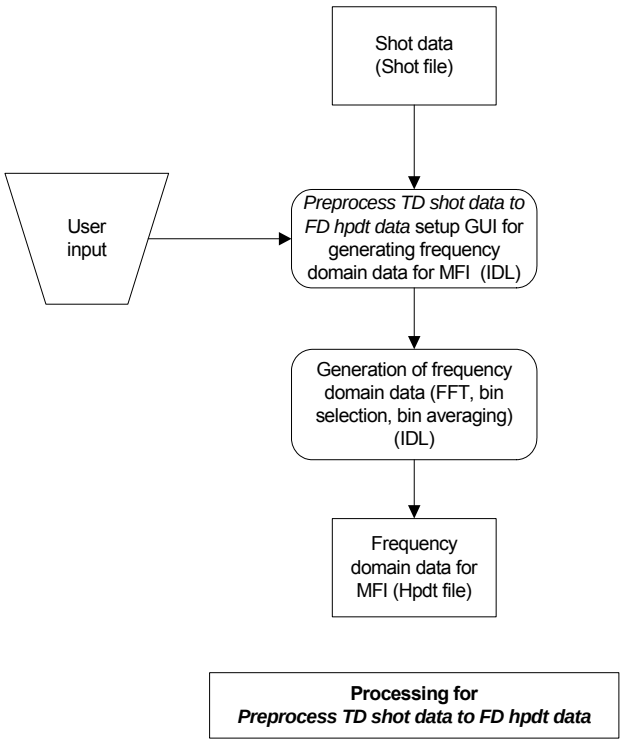
Note that only the “good” sensors data will appear in the output data files. Hence, you will have to create a corresponding Aray file to match these sensors before performing ambiguity function generation or MFI.

## Preprocessing of Time Domain Shot Data to Frequency Domain Hpdt Data

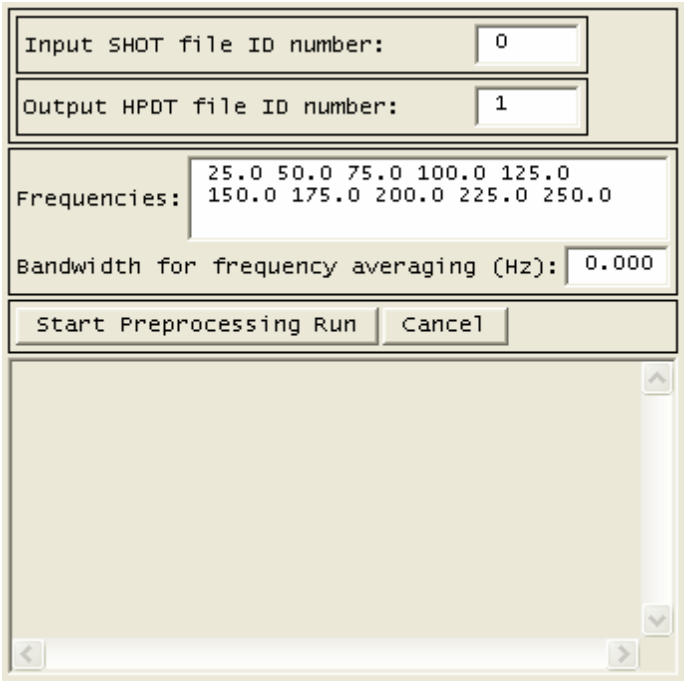
This component allows the data for each shot to be processed into a form amenable to MFI analysis. The strategy for processing the shot data is to FFT the traces for the sensors and form cross-spectral matrices at user-specified frequencies, with the option for averaging within a small frequency band at each of these frequencies. These matrices are output to an Hpdt file, which can be used as input for the ambiguity function and MFI components of BCOMFI.

### Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface




- The first items in the top row allow the user to specify the input Shot file to be used for the run, and the output Hpdt file to contain the results.
- The next two lines allow the user to specify the frequencies at which the FFT'd shot data are to be retained and the bandwidth within which frequency averaging centered at these frequencies is to be performed.
- The "Start Preprocessing Run" and "Cancel" buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for processing the shot data is outlined in the following pseudocode:

*Specify the input Shot file and output Hpdt file.*
*Select the frequencies at which signal vectors are to be computed.*
*For each shot:*
    *FFT the traces for each sensor.*
*For each selected frequency:*
    *Form a frequency-averaged cross spectral matrix by accumulating the outer products of vectors within a specified band of frequencies centered at the selected frequency (optional).*
*Write matrices to an Hpdt file.*

## Using the Component

Press the “Preprocess TD shot data to FD hpdt data” button to bring up the GUI for preprocessing shot traces to hpdt form, where they can be used as input to MFI. The preprocessing involves Fourier transforming the traces, and then, for each selected frequency, forming an averaged cross-spectral matrix corresponding to the sensors of the array. The resulting data are written to an Hpdt file.

To run this conversion, specify:

- the ID number of the Shot file containing the array to be converted;
- the ID number of the output Hpdt file that will be produced;
- the frequencies at which the complex fields are to be generated from the shot traces;
- the bandwidth to use for averaging the cross-spectral matrices at each frequency.

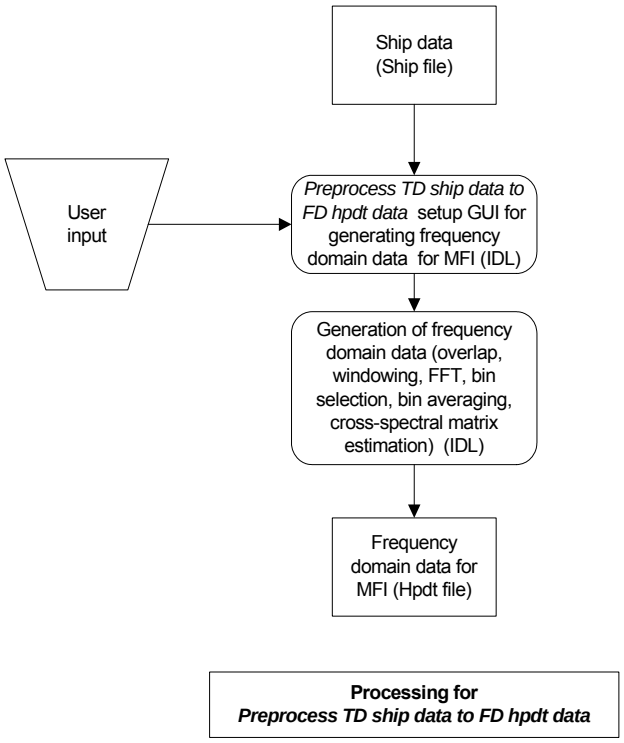
When the above have been specified, press the “Start Preprocessing Run” button to perform the preprocessing. The process should be almost instantaneous.

## Preprocessing of Time Domain Ship Data to Frequency Domain Hpdt data

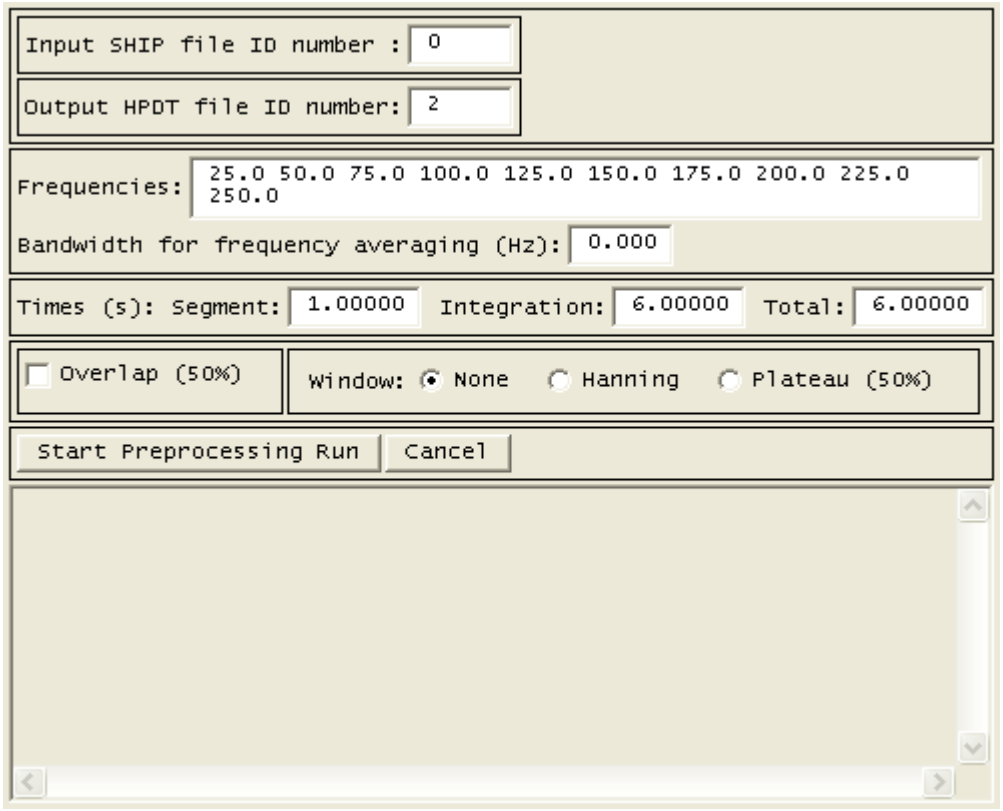
The time domain ship data are processed as a series of (possibly overlapping and windowed) segments to yield cross-spectral matrices for MFI analysis. The strategy for processing the ship data is to segment the time series, optionally overlap the time segments by 50% and apply a window (e.g., Hanning), and then FFT the sensor data for each segment. The data at user-specified frequencies will then be used to form cross-spectral matrices, which will be output to an Hpdt file for MFI.

## Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface




- The items in the top row allow the user to specify the input Ship file to be used for the run, and the Hpdt file to contain the results.
- The next two lines allow the user to specify the frequencies at which the FFT'd ship data are to be retained and the bandwidth within which frequency averaging centered at these frequencies is to be performed.
- The third row provides for specification of the segment time, the integration time (the time over which cross-spectral matrix estimation takes place), and the total time (which controls how many vectors/matrices are computed). Note that at present only the data for the first integration time can be analyzed by MFI.
- The fourth row allows the user to specify the overlap to use for the time series to be FFT'd and the window to use prior to performing the FFT.
- The "Start Preprocessing Run" and "Cancel" buttons are self-explanatory.
- The large empty area at the bottom provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for preprocessing the ship data is outlined in the following pseudocode:

*Specify the input Ship file and the output Hpdt file.*
*Select the frequencies at which signal vectors are to be computed.*
*Repeat for each integration time:*
*For each sensor:*
*Obtain the data for the segment (optional overlap).*
*Window the data.*
*FFT the data.*

*For each frequency:*

*Form a frequency-averaged cross spectral matrix by accumulating the outer products of vectors within a specified band of frequencies centered at the selected frequency (optional).*

*Output the cross-spectral matrices to an Hpdt file.*

## Using the Component

Press the "Preprocess TD ship data to FD hpdt data" button to bring up the GUI for preprocessing ship traces to hpdt form, where they can be used as input to MFI. The preprocessing involves Fourier transforming sequential segments of the traces, and accumulating cross-spectral matrices at selected frequencies. Options are provided for overlapping the segments by 50%, and for windowing the segments prior to Fourier transformation for each selected frequency. The transformed data are used to form a sequence of complex cross-spectral matrices representing the fields observed at the sensors of the array. An option is also provided in the interface to average the outer products of sensor field vectors for frequency bins within a window centered on each of the chosen frequencies. The matrices generated by the preprocessing are written to an Hpdt file.

To run this conversion, specify:

- the ID number of the Shot file containing the array to be converted;
- the ID number of the output Hpdt file that will be produced;
- the frequencies at which the complex fields are to be generated from the shot traces;
- the bandwidth within which the outer products of the complex field data are to be averaged for formation of the cross-spectral matrices;
- values for segment time (data length for a single FFT), integration time (time within which outer-product matrices are averaged to accumulate cross-spectral matrices), and total time;
- whether the segments are to be overlapped by 50%;
- the window, if any, to be applied to the traces before FFT.

When the above have been specified, press the “Start Preprocessing Run” button to perform the preprocessing. The process should be very rapid.

## Matched Field Techniques: Brief Overview

Matched field processing and matched field inversion are array signal processing methods which search over a parameter space for unknown model parameters by matching a replica field vector computed using a parameter-based model with “measured” data. In the present context, the replica data are computed using acoustic propagation modeling to generate the fields at an array that would be observed for a particular realization of source-array geometry and geoacoustic model. The “measured” data are obtained from an array of hydrophones or from propagation modeling using a reference source-array geometry and geoacoustic model. The replica data are matched with the “measured” data using a power processor, with high processor output indicating good matches. A maximum processor value can indicate a reasonable correspondence between the model parameters and the actual structure of the environment.

The normalized Bartlett power processor is used here to compute the matching (ambiguity) function between the measured data and the replica vectors generated using ORCA. For a measured data vector **m** and a replica vector **r**(**p**) computed for parameter set **p** at a single frequency**,** the Bartlett processor is defined as

$$B(\mathbf{r}(\mathbf{p}),\mathbf{m}) = \frac{|\,\mathbf{r}(\mathbf{p})^{*}\mathbf{m}\,|^{2}}{|\,\mathbf{r}(\mathbf{p})\,|^{2}|\,\mathbf{m}\,|^{2}}\,.$$

For a measured data cross-spectral matrix **M,** it is defined as

$$B(\mathbf{r}(\mathbf{p}),\mathbf{M}) = \frac{|\,\mathbf{r}(\mathbf{p})^{*}\mathbf{M}\mathbf{r}(\mathbf{p})\,|}{|\,\mathbf{r}(\mathbf{p})\,|^{2}\|\,\mathbf{M}\,\|},$$

where ||**M**|| is the spectral norm of the cross-spectral matrix **M**.

For multi-frequency measured data, a processor that combines the Bartlett outputs is defined as

$$C(\mathbf{p}) = \sum_{j=1}^{N} w_j B(\mathbf{r}_j(\mathbf{p}),\mathbf{m}_j) \quad \text{or} \quad C(\mathbf{p}) = \sum_{j=1}^{N} w_j B(\mathbf{r}_j(\mathbf{p}),\mathbf{M}_j)$$

for vector and cross-spectral matrix data, respectively, where $\mathbf{r}_j(\mathbf{p})$ is the replica vector for parameter set **p** at the *j*th frequency, $\mathbf{m}_j$ is the measured vector at the *j*th frequency, *N* is the number of frequencies, and $w_j$ is a weight ($\sum w_j = 1$).

Matched field techniques always involve ambiguity, in that many local maxima may be present in the parameter space of interest, and these matches can often be comparable to those obtained using the true parameter set. Also, matches can be very insensitive to some parameters, or two parameters may be highly correlated, giving rise to other forms of ambiguity. The presence of noise introduces additional ambiguity. Hence, the ability to visualize the ambiguity in matched field techniques provides valuable insight into the characteristics of the parameter space and the validity of estimation of the geometric or geoacoustic model parameters. Based on these considerations, functionality has been provided in BCOMFI to allow the computation and visualization of ambiguity functions of up to three dimensions. The use of these ambiguity functions is described in the following section.

Matched field inversion is routinely accomplished by repeated forward modeling to generate replicas and matching the fields with the “measured” data. The use of optimization methods allows the parameters to be adjusted so as to achieve the best fit between the replica and “measured” data. Since there can be many local optima in the parameter space, it is essential to have an approach that has both global and local aspects. The approach to MFI taken in BCOMFI is to have an initial global search stage followed by local optimization of each of a number of the best matches found during the search stage. This approach is described in the Matched-Field Techniques: Inversion section below.

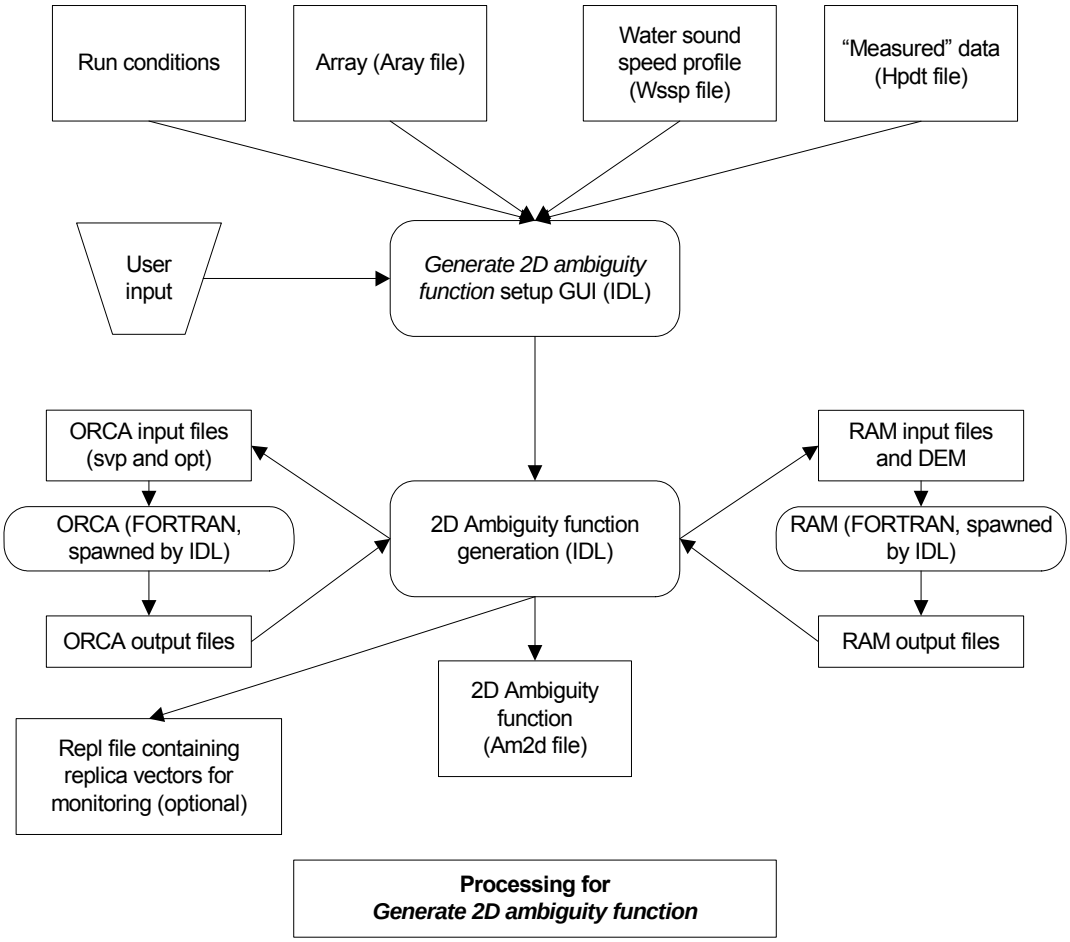
## Matched Field Techniques: Ambiguity Function Generation

In investigations involving MFI, it is useful to be able to examine how the ambiguity function (the matching function used as an objective function in MFI) depends on the individual parameters, and, sometimes, groups of parameters. To allow the visualization of this behavior, a component of BCOMFI has been provided that allows the generation of 1D, 2D, and 3D ambiguity functions (higher-dimension grids are too time-consuming to generate and more difficult to visualize). The user can then use IDL display software to examine the characteristics of these functions, including dynamic range, peak widths, presence of multiple optima, parameter sensitivity, and, in the case of 2D or 3D ambiguity functions, parameter interdependency. This last item is of particular significance, since it can lead to ill-posed MFI problems and inconsistent results in parameter estimation. Visualization of the ambiguity functions for the parameter space can assist in the interpretation of such results.

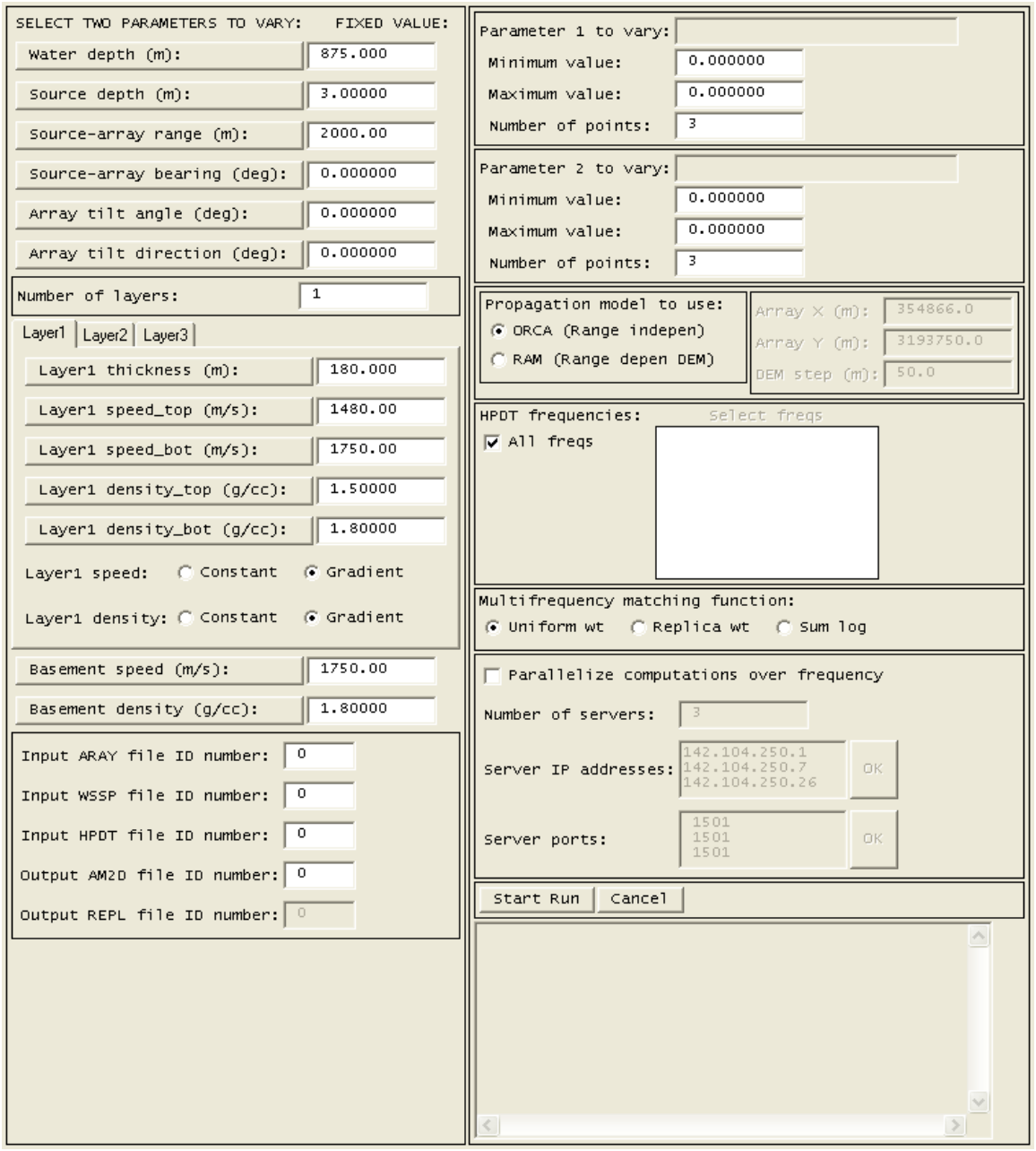
This section describes the ambiguity function generation component, using the 2D case as an example; the 1D and 3D versions are analogous to the 2D version.

## Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface




- The block (including tabs) on the top left lists the 23 parameters that can be varied to generate the ambiguity function, and provides default fixed values for those parameters. Pressing on the button containing a parameter name chooses that parameter as one of the two to be varied and populates one of the “Parameter 1 to vary” or “Parameter 2 to vary” fields in the blocks at the top right.

- The block under the tabs provides an option for the user to force the indicated parameter pairs for the top (e.g., density1) and bottom (e.g., density2) of the corresponding layer to be the same.
- The lower block on the left allows the user to specify the input files (Aray, Wssp, and Hpdt) to be used for the run, and the Am2d and Repl (optional) files to contain the results.
- The top two blocks on the right allow specification of the domain over which the parameters are to be varied, and the number of points in each dimension.
- The third block on the right allows the user to select the propagation model to use: ORCA (for a range-independent environment) or RAM (for a DEM containing the range-dependent bathymetry). If RAM is chosen, the location of the array and the range step for the DEM bathymetry mat then be specified.
- The fourth block on the right allows the user to view the frequencies of the data in the Hpdt file and select a subset of these to use in computing the multi-frequency ambiguity function. An option to select all the frequencies in the data is also provided.
- The fifth block on the right provides options for how the matches at the multiple frequencies are to be combined in the overall match. For the uniform and replica options, the Bartlett outputs at the frequencies are weighted uniformly or according to the squared modulus of the replica vector, respectively. For the Sum log option, the expression $\sum_{j=1}^{N} \log(1 - B(\mathbf{r}_j(\mathbf{p}), \mathbf{m}_j))$ (or $\sum_{j=1}^{N} \log(1 - B(\mathbf{r}_j(\mathbf{p}), \mathbf{M}_j))$ for matrix data) is used.
- The sixth block on the right allows for parallel processing over frequency, with the server IP addresses and ports of the remote computers specified in the text boxes.
- The “Start Run” and “Cancel” buttons are self-explanatory.
- The large empty area at the bottom right provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used for generating the 2D ambiguity function is outlined in the following pseudocode:

```
Specify the input Aray file and the output Hpdt file, and specify the run conditions.
Check that the run conditions are consistent.
Apply a tilt to the array.
For each value of parameter 1:
    For value of parameter 2:
        For each frequency selected (optionally parallelized if ORCA is chosen):
            Apply tilt values to the array positions.
            If the ORCA option is chosen:
                Generate an array geometry file for ORCA.
                Generate an opt file for ORCA.
```

*Generate an svp file for ORCA.*
*Spawn an ORCA process to generate a replica vector.*
*Else if the RAM option is chosen:*
*Generate a ramgeo.in file from the ramgeo.dat file, the dem file, and the current set of run conditions.*
*Spawn a RAM process to generate a replica vector.*
*If depth-range or range-bearing ambiguity function, save the replica vector.*
*Compute the Bartlett power of the match for that frequency.*
*Sum the weighted Bartlett powers for the frequencies to give the ambiguity function for (parameter 1, parameter 2).*
*Output the ambiguity function to an Am2d file.*
*If depth-range or range-bearing ambiguity function, output the saved replica vectors to a Repl file.*

## Using the Component

Press the "Generate 2D ambiguity function" button to bring up the GUI for computing a 2D ambiguity function. This will compute a 2D array of Bartlett matches for pairs of selected parameters being varied, with fixed values for the other parameters. Note that if a depth-range or range-bearing ambiguity function is chosen, replica vectors for each frequency will be computed and saved to a Repl file for later use in monitoring. To generate the 2D ambiguity function, do the following:

- Select the parameters to vary, by clicking on the corresponding text buttons containing the names in the left column; these parameter names will then appear in the two text fields on the top right.
- Specify the values to use for the fixed parameters in the editable fields on the left portion of the interface. To change the number of layers (which can be from 0 up to 3) specify this number in the "Number of layers" field. To access the parameters in a layer other than the present layer, click on one of the tabs: Layer1, Layer2, Layer3.
- Ensure that radio buttons are selected to enable or disable gradients for sound speed and density in the corresponding layer.
- Specify the upper and lower limits for the two parameters to vary, and the number of points to evaluate in each parameter dimension.
- Specify ID numbers for the input Aray, Wssp, and Hpdt files, and the output Am2d and (optional) Repl file to contain the computed ambiguity function. Note that only if depth and range, or range and bearing, are chosen as parameters to vary will the Repl file ID number field become active.
- Select the propagation model to use. If ORCA is chosen, the replicas will be computed using a range-independent environment. If RAM is selected, then the replicas will be computed using the DEM in the file "dem.dat". Also, if RAM is selected, the box on the right is then enabled and you may use it to specify three additional parameters. These are the *x* and *y* placement of the array in the region of the DEM (the default is in the center of the grid) and the range step for the bathymetry along the source-array path.

- Specify the frequencies in the Hpdt file to use in the matching. Checking the "All freqs" checkbox uses all the frequencies in the file. Unchecking the box enables the option to select one or more of the individual frequencies in that file. Note that the Hpdt file with the specified ID number must exist in order to be able to use this option.
- Specify the option to use for combining the frequencies in the final matching function.
- For ORCA, and if the computations are to be run in parallel over frequency, check the "Parallelize computations over frequency" checkbox and ensure that the numbers of servers, the IP addresses and the ports specified in the text boxes are correct. If you edit these fields, note that you must press the respective "OK" button for the changes to take effect. (If you wish to change the default values, edit the file "servers.dat" in the \source directory.) Also ensure that orcaserver processes are running on each of the computers selected (see the Installing the BCOMFI Software section above for information on installing and running these processes on Windows and Linux computers). Note that this parallelization option is not available for the RAM propagation model.
- Press the "Start run" button.

The result of the run will be an Am2d file, which can be viewed using the "Display 2D ambiguity function" option on the Main Menu interface. If a depth-range or range-bearing function was chosen, a Repl file is also produced by the run, which can be used in monitoring.

## Matched Field Techniques: Inversion

### Background

The purpose of the MFI software component of BCOMFI is to provide an environment for modeling and MFI using appropriate parameterizations. Using this software, studies can then be performed to determine which approaches will be effective for detecting changes within the hydrate stability zone. The MFI component also provides an environment for the analysis of real data (both shot and ship) with the intended result of defining a model, or set of models, that provide a reasonable representation of the environment in the region of the array. These models could then be used as a basis for real-time monitoring.

In designing and implementing the MFI component, we considered that MFI is a nonlinear process that is generally approached using optimization techniques to repeatedly solve the forward problem for varying sets of parameters until a suitable good match to the data is obtained. MFI optimization approaches must be able to deal with the following challenges:

- There are typically 5 – 25 parameters, and it is required to be able to optimize any or all combinations of these.
- There are generally multiple local optima present in the parameter space.

- It is desirable to restrict the domain of the parameters, usually by bounds constraints.
- The sensitivities of the parameters can be very different (by a factor of 100 or more).
- The parameters need to be scaled for optimization.
- Derivatives are unavailable except by numerical approximation.
- Certain parameters can be correlated, leading to ill-posed problems.
- There is a possibility of discontinuities in the matching function, particularly if the propagation algorithms do not always successfully converge/complete (as has been observed to be the case with ORCA under some conditions).

MFI has been implemented using a global search / local optimization approach. The advantages of this approach are the ability to obtain multiple estimates of the various optima, and the moderate number of function evaluations required (e.g., 2000 for search space sampling and 3000 for 10 optimizations). While a potential drawback of this method is that the individual optimized estimates are local, a sufficiently comprehensive search stage will increase the likelihood that one of these optima is, in fact, global.

The intended outcome of investigations conducted using the MFI component will be to compute effective geoacoustic models of the environment in the region of the monitoring station. Under some circumstances (i.e., for source-receiver paths corresponding to approximately constant bottom depth) it would be appropriate to use ORCA to compute the replicas for MFI. However, it is likely that in most cases the variable bathymetry of the region would preclude the use of ORCA for MFI, and so RAM should be used instead. In due course, calibration studies using real data will address this question, and allow the better definition of those conditions where it would be appropriate to use ORCA and those where RAM should be used. Note that the greater modeling flexibility of RAM comes at the cost of the greater computational times involved, which are generally at least an order of magnitude greater than those for ORCA.

## Implementation

The overall method implemented for MFI involves the following stages:

***A global search stage***. In this stage, sets of values for the parameters to be varied are generated that are randomly distributed between the lower and upper bounds for those parameters. ORCA is then used to generate a replica vector for this parameter set, and the matching function is then computed. This process is repeated a specified number of times to provide a sampling of the overall search space.

***A local optimization stage***. In this stage, each of a specified number of the best matches found in the global search stage is used as a starting point for optimization of the parameters. Here the objective function $F(\mathbf{p})$ of the parameter set $\mathbf{p}$ to be minimized is:

$$F(\mathbf{p}) = 1 - C(\mathbf{p}) + P(\mathbf{p}) + R(\mathbf{p}),$$

where $C(\mathbf{p})$ is the processor output, $P(\mathbf{p})$ is a penalty function for values of the parameters outside their bounds, and $R(\mathbf{p})$ is a regularization function.

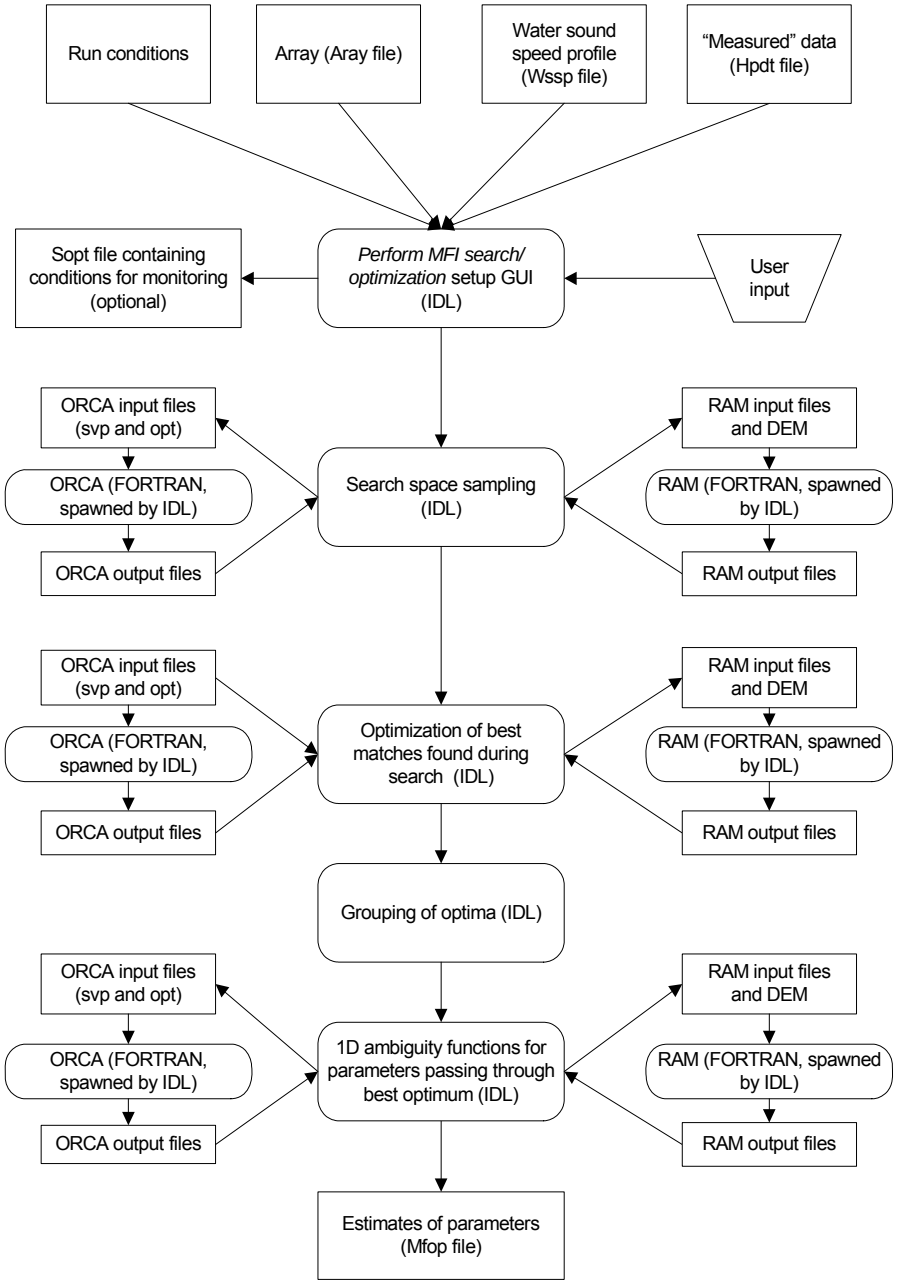
The IDL routine DFPMIN (a quasi-Newton method for which the required derivatives can be approximated numerically by central differences) is used to perform the optimizations. The result is a set of independent estimates of optimized parameter values.

***A grouping stage***. In this stage, the optima are analyzed and multiple estimates of the same optimum are grouped together. This is done by starting with the optimum with the lowest objective function value and determining other optima with sufficiently similar values for the function value. The objective function is evaluated along a line in the parameter space connecting the optima and they are grouped if the objective function along this line does not exceed a specified threshold.

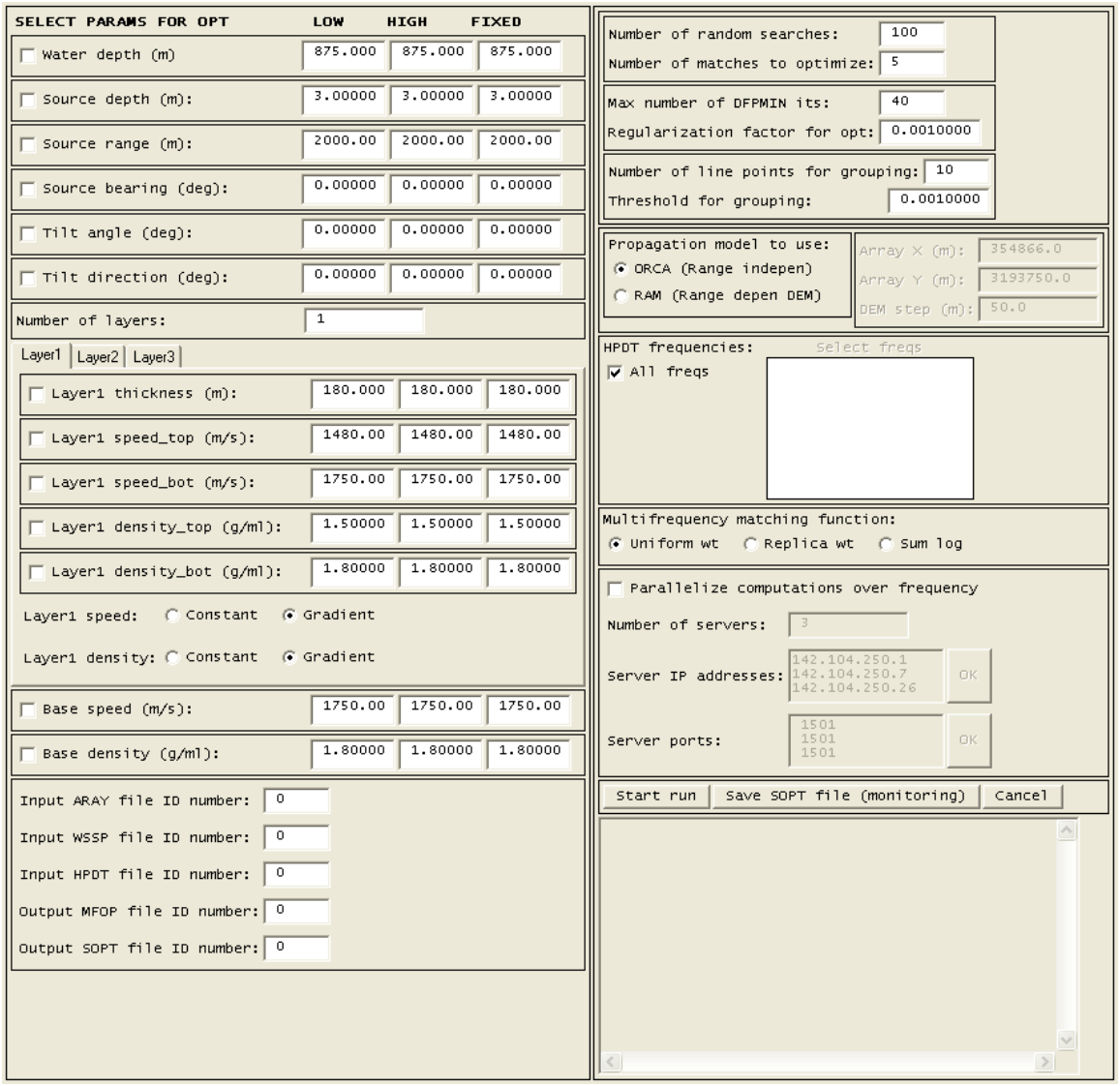
***A parameter characterization stage***. In this stage, the best optimum is identified and the objective function is evaluated independently for each varied parameter along a line spanning the region between its lower and upper bounds and intersecting the peak. This provides an estimate of the sensitivities of the individual parameters, and an indication of the peak width, dynamic range, and oscillations of the function with respect to each parameter.

## Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface




- The block on the top left (including tabs) lists the 23 parameters that can be chosen for optimization and provides bounds and default fixed values for those parameters. The checkboxes are used to select those parameters which are to be optimized. The lower and upper bounds for selected parameters can then be set using the "LOW" and "HIGH" fields for the parameter (in which case the "FIXED" value is ignored). Note that the "Layer*n* speed" and "Layer*n* density" buttons under the tabs provide an option for the user to force the indicated parameter pairs for the top and bottom of the corresponding layer *n* to be the same.
- The lower block on the left allows the user to specify the input files (Aray, Wssp, and Hpdt) to be used for the run, and the Mfop file (and optionally the Sopt file) to contain the results. The Sopt file is output when the "Save SOPT file (monitoring)" option is chosen, and will contain the current values of the Params and Mfop data structures;

these are used to specify the search/optimization conditions for monitoring (see below).

- The top block on the right allows the user to specify the conditions for the search stage, i.e., the number of random samples of the parameter space and the number of best matches to optimize.
- The second block on the right allows specification of convergence and regularization factors for the optimization.
- The third block on the right allows the user to specify conditions for the grouping analysis of the multiple peaks.
- The fourth block on the right allows the user to select the propagation model to use: ORCA (for a range-independent environment) or RAM (for a DEM containing the range-dependent bathymetry). If RAM is chosen, the location of the array and the range step for the DEM bathymetry mat then be specified.
- The fifth block on the right allows the user to view the frequencies of the data in the Hpdt file and select a subset of these to use in computing the multi-frequency ambiguity function. An option to select all the frequencies in the data is provided.
- The sixth block on the right provides options for how the matches at the multiple frequencies are to be combined in the overall match. For the uniform and replica options, the Bartlett outputs at the frequencies are weighted uniformly or according to the squared modulus of the replica vector, respectively. For the Sum log option, the expression $\sum_{j=1}^{N} \log(1 - B(\mathbf{r}_j(\mathbf{p}), \mathbf{m}_j))$ (or $\sum_{j=1}^{N} \log(1 - B(\mathbf{r}_j(\mathbf{p}), \mathbf{M}_j))$ for matrix data) is used.
- The seventh block on the right allows for parallel processing over frequency, with the server IP addresses and ports of the remote computers specified in the text boxes.
- The “Start Run” and “Cancel” buttons are self-explanatory. The “Save SOPT file (monitoring)” button saves the current conditions for use in monitoring.
- The large empty area at the bottom right provides a field for status messages to be displayed.

## Algorithm

The algorithm to be used to perform matched field inversion using the search/gradient optimization method is outlined in the following pseudocode:

*Specify the input Aray and Wssp files and the output Hpdt file, and specify the run conditions, including those parameters that are to be optimized and their bounds.*
*Check that the run conditions are consistent.*
*Generate specified number of random samples of the search space and rank the results in order of the best matches.*
*For each of a specified number of the best matches:*
    *Using the current best match as an initial estimate, call the optimization function to optimize the parameters.*

*Identify those converged optima which are estimates of the same peak and group them together to form unique estimates.*
*Generate 1D ambiguity functions passing through the best optimum for each parameter, and estimate the characteristics of the parameters (number of peaks, peak width, etc.).*
*Write results to an Mfop file.*

The algorithm for computing the objective function, given a set of parameters provided by the optimization algorithm, is as follows:

*For each frequency selected (optionally parallelized of ORCA is chosen):*
*Apply tilt values to the array positions.*
*If the ORCA option is chosen:*
*Generate an array geometry file for ORCA.*
*Generate an opt file for ORCA.*
*Generate an svp file for ORCA.*
*Spawn an ORCA process to generate a replica vector.*
*Else if the RAM option is chosen:*
*Generate a ramgeo.in file from the ramgeo.dat file, the dem file, and the current set of run conditions.*
*Spawn a RAM process to generate a replica vector.*
*Compute the Bartlett power of the match for that frequency.*
*Combine the weighted Bartlett powers for the frequencies to give the ambiguity function for the input parameters.*
*Apply penalty function for parameters outside the bounds, and add a (small) regularization function term which increases quadratically with distance from the midpoint of the parameter range.*

## Using the Component

To perform MFI using the search/optimization procedure, press the “Perform MFI search/optimization” button on the Main Menu to bring up the GUI. Then do the following:

- Select the parameters to optimize, by clicking on the checkboxes beside the names in the left column.
- Specify the lower and upper bounds for each parameter to be varied in the corresponding text fields (under columns LOW and HIGH) for that parameter. To change the number of layers (which can be from 0 up to 3), specify this number in the “Number of layers” field. To access the parameters in a layer other than the present layer, click on one of the tabs: Layer1, Layer2, Layer3.
- Specify the values to use for the fixed parameters in the corresponding fields for those parameters (under column FIXED).
- Ensure that radio buttons are selected to enable or disable gradients for sound speed and density in the corresponding layer.

- Specify ID numbers for the input Aray, Wssp, and Hpdt files, and the output Mfop file to contain the results of the run.
- Specify the number of random samples to be generated during the global search stage, and the number of best matches found during this stage to optimize in the next stage.
- Specify the maximum number of iterations for the IDL optimization algorithm DFPMIN, and the regularization scaling factor for the quadratic function used in the regularization.
- Specify the number of points along a line connecting two possibly different optima when performing the grouping, and the threshold for the grouping.
- Select the propagation model to use. If ORCA is chosen, the replicas will be computed using a range-independent environment. If RAM is selected, then the replicas will be computed using the DEM in the file "dem.dat". Also, if RAM is selected, the box on the right is then enabled and you may use it to specify three additional parameters. These are the *x* and *y* placement of the array in the region of the DEM (the default is in the center of the grid) and the range step for the bathymetry along the source-array path.
- Specify the frequencies in the Hpdt file to use in the matching. Checking the "All freqs" checkbox uses all the frequencies in the file. Unchecking the box enables the option to select one or more of the individual frequencies in that file. Note that the Hpdt file with the specified ID number must exist in order to be able to use this option.
- Specify the option to use for combining the frequencies in the final matching function.
- For ORCA, and if the computations are to be run in parallel over frequency, check the "Parallelize computations over frequency" checkbox and ensure that the numbers of servers, the IP addresses and the ports specified in the text boxes are correct. If you edit these fields, note that you must press the respective "OK" button for the changes to take effect. (If you wish to change the default values, edit the file "servers.dat" in the \source directory.) Also ensure that orcaserver processes are running on each of the computers selected (see the Installing the BCOMFI Software section above for information on installing and running these processes on Windows and Linux computers). Note that this parallelization option is not available for the RAM propagation model.
- If setting up for monitoring, press the "Save SOPT file (monitoring)" button. The conditions for search/optimization that you have specified will be saved to a Sopt file which can then be used during a monitoring run. Note that you do not have to specify depth and range (if the ORCA option is chosen) or range and bearing (if the RAM option is chosen) as parameters to vary, as this is done automatically by the monitoring program Mfi_monitor (see below).
- Press the "Start run" button. During each stage, graphical information (objective function and parameter values) will be plotted to the screen every 20$^{th}$ function call. This provides a convenient way of monitoring the progress of the search, optimization, and other stages.

The result of the run will be a Mfop file (and optionally a Sopt file); the former can be

viewed using the “Display MFI search/optimization results” option on the Main Menu interface.

## Matched Field Techniques: Monitoring

The aim of monitoring using MFI is to be able to detect large-scale changes in sub-bottom hydrates structures by analyzing acoustic array data obtained using the sound of passing ships. During monitoring, data sets or data streams collected for this array are processed to estimate geoacoustic models for the region. Systematic changes in the model parameters estimated by MFI could then be taken as evidence of changes in the hydrates.

As noted above, MFI provides a way to estimate geoacoustic model parameters for an ocean environment, including sub-bottom and water column sound speed profiles. MFI can be used with either impulsive or continuous wave sources. Its performance can be improved by using multiple frequencies and this does not require that these frequencies be coherent. While it can an advantage for MFI if the source location is known, this is not necessarily required, as the location can generally be estimated during the inversion process. Hence, in many respects MFI is very well-suited to the intended application of long-term hydrates monitoring using the sound generated by passing ships.

However, MFI does have some potential limitations in the monitoring application, as noted in the following list:

- MFI is not an imaging method, but rather estimates the values of geometric and geoacoustic parameters through an optimization process. These values are themselves averaged or effective estimates of the parameters over the source-receiver path rather than single estimates at a particular location.
- MFI N-D ambiguity surfaces generally contain multiple optima even when multiple frequencies are used, and so the results of MFI are often ambiguous or non-unique.
- The vertical array to be used initially in the monitoring application spans only one quarter of the water column, which results in suboptimal sampling of the modes (and increased ambiguity in MFI).
- Low impedance contrast bottom profiles can be difficult to estimate by MFI – this is likely to be the case in the region of the monitoring station and hydrates-bearing region of interest.
- MFI requires sufficient source-receiver separation to provide significant bottom interaction in order to obtain reasonable parameter estimates. However, as the source-receiver distance increases, the SNR will decrease, degrading the performance of MFI.
- MFI processing is very time-intensive, as large numbers of function evaluations are required during optimization. This can be mitigated somewhat by precomputation, but this can only be done when the number of parameters N is small (since each parameter corresponds to a separate dimension of an N-D grid of replica vectors).
- Range-independent models are less computationally expensive but are not applicable to environments with substantial range dependence.

- Range-dependent models are applicable more generally but are more time-consuming to compute and also can involve more parameters to estimate.
- A particular choice of which model parameters to optimize must be made, and this choice will affect the results obtained.
- MFI can be sensitive to errors in the sound speed profile of the water column as frequency increases; estimation of this profile can be done in principle but would further increase the number of parameters in the model, and the computation times.

In the light of the above considerations, the approach we have taken to implementing monitoring based on MFI is intended to provide relevant information about the sub-bottom while balancing model complexity with computational tractability. At this stage, a single geoacoustic model (which parallels the bottom in the case of a range-dependent environment) is assumed to apply to the region as a whole, and the parameters of this model are estimated in a two-stage process:

3. Replica vectors, precomputed for a standard geoacoustic model for the region, are used to compute a 2D ambiguity surface for the geometric parameters depth and range (if ORCA is selected) or range and bearing (if RAM is selected) and the best match in this surface is identified.
4. A window is defined around the optimum depth and range (or range and bearing) in the 2D ambiguity surface, and MFI is performed on these two parameters, using this window to define the lower and upper bounds, along with other selected geoacoustic parameters such as layer thickness and sound speed at the top and bottom of the layer. For efficiency, only a single optimization is performed.

This process is repeated for each data set computed from the input data stream, to generate a sequence of estimates for the sub-bottom parameters according to the geoacoustic model specified by the user.

Precomputation allows the large depth-range or range-bearing subspace to be rapidly searched, since no real-time model generation is required. In particular, the range subspace can be decreased from several thousands of meters to 100 m or so, thereby greatly reducing the search region for the optimization stage by a factor of 20 – 50. Precomputation could, in principle, be extended to include additional parameters, but the resulting increase in the numbers of dimensions would rapidly render it prohibitive both in terms of computational resources and storage space. Since experimentation indicated that estimation of the geometric parameters depth, range and bearing could be done with approximate estimates of the geoacoustic parameters, we selected these parameters as being suitable for precomputation of the replica vectors.

The approach to applying MFI to monitoring uses several other components of BCOMFI during the setup and the actual processing, including those which:

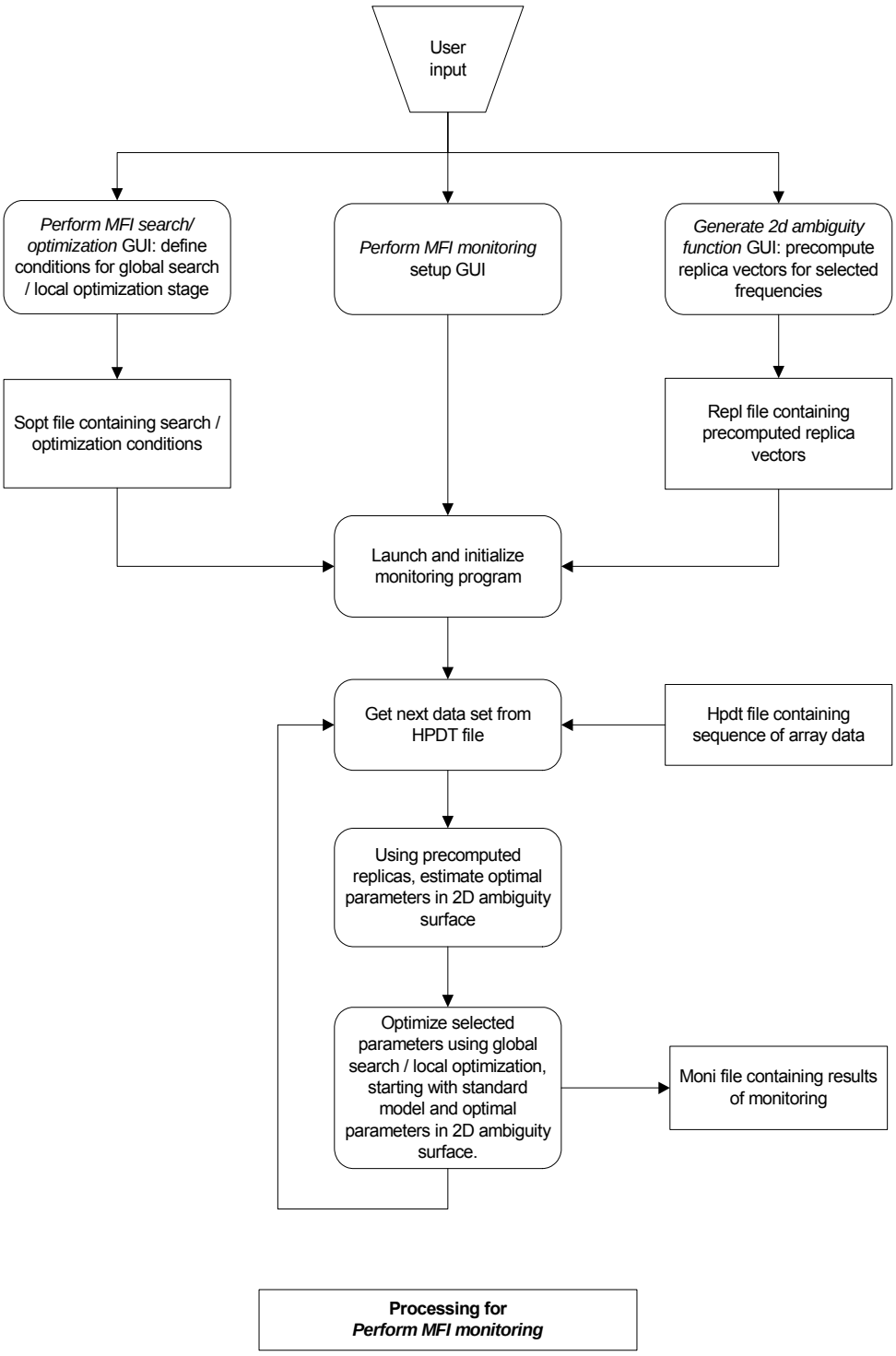
- generate the synthetic data (“Simulate ship FD HPDT data”);
- generate replica vectors (“Generate 2D ambiguity function”);

- define the conditions for and actually perform the search/optimization (“Perform MFI search/optimization)”.

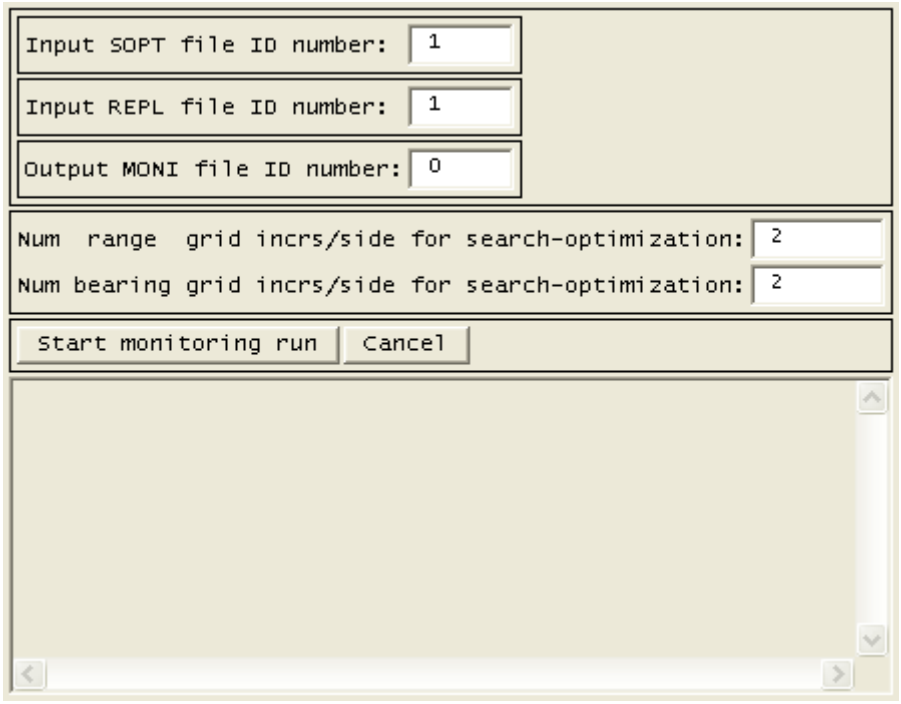
The processes for setting up and performing the MFI monitoring are described in more detail in the remainder of this section.

## Overall Design

In this design, data and/or files are indicated by rectangles, processes are represented by rounded rectangles, and user input is denoted by a trapezoid.



## Graphical User Interface




- The top block allows the user to specify the input Sopt and Repl files to be used for the run, and the Moni file to contain the results.
- The second block allows the user to specify the numbers of grid increments in the 2D ambiguity function to be used to define the bounds in the search/optimization stage of monitoring.
- The “Start monitoring run” and “Cancel” buttons are self-explanatory.
- The large empty area at the bottom right provides a field for status messages to be displayed.

## Algorithm

The algorithm used to perform matched field monitoring is outlined in the following pseudocode:

*Specify the input Sopt and Repl files and the output Moni file, and the numbers of 2D grid increments to be used bracketing the search region for these parameters.*
*Check that the run conditions are consistent.*
*Input the precomputed replica vectors from the Repl file and the search/optimization conditions from the Sopt file.*
*For each data set in the monitoring input stream:*

*Generate a 2D depth-range (ORCA option) or range-bearing (RAM option) ambiguity function using the replica data in the Repl file, and using the frequencies specified in the Sopt file.*

*Determine the position of the maximum in the ambiguity surface and use this to define the bounds for the two parameters, based on the grid increments specified above.*

*Using the conditions specified in the Sopt file, perform a search/optimization run, which will yield a single estimate of the chosen parameters for that data set.*

*Write results to a Moni file.*

## Using the Component

To perform MFI monitoring, you should first generate the simulated data for the monitoring run, by pressing the "Simulate ship FD hpdt data" button and performing a run for the desired conditions.

Next you must generate a set of replica vectors to be used in the depth-range or range-bearing estimation stage of the computations. This is done by pressing the "Generate 2D ambiguity function" button on the Main Menu, choosing (ORCA option) water depth as parameter 1 and source range as parameter 2, or (RAM option) source range as parameter 1 and source bearing as parameter 2, then specifying the lower and upper bounds for the parameters and the frequencies to use, making any other desired adjustments, and then performing the run. This will produce a Repl file containing the replica vectors.

Then you should generate a Sopt file containing the conditions (i.e., the Params and Mfop structures) to be used in the search/optimization stage of the computations. This is done by pressing the "Perform MFI search/optimization" button in the Main Menu and selecting the conditions to be used in this stage. Note that you do not have to specify that depth and range (ORCA option) or range and bearing (RAM option) are to be optimized, as this is done automatically by the monitoring software. In addition, note that this software also always sets the number of matches to optimize to one, no matter what its setting in the search/optimization GUI. You should specify any other parameters to be optimized, and set their bounds, as well as the values for the fixed parameters. When you have specified the desired conditions, press the "Save SOPT file (monitoring)" to save the conditions to a Sopt file.

Once you have prepared the above files, press the "Generate 2D ambiguity function" button to bring up the GUI for monitoring. Then do the following:

- Specify ID numbers for the input Repl and Sopt files, and the output Moni file to contain the results of the run.
- Specify the numbers of grid increments in the 2D ambiguity function to be used to define the bounds for depth and range (ORCA option) or range and bearing (RAM option) in the search/optimization stage of monitoring.

- Press the “Start Monitoring Run” button. After the 2D ambiguity function generation stage, a plot of this function will be displayed. Then, during the search/optimization stage, graphical information (objective function and parameter values) will be plotted to the screen every 20$^{th}$ function call. This provides a convenient way of monitoring the progress of the various stages.

The result of the run will be a Moni file, which can be viewed using the “Display MFI monitoring results” option on the Main Menu interface.

## Matched Field Techniques: Parallelization

Matched field methods are computationally intensive, and to reduce the times involved it is necessary to develop implementations of these techniques where the computations are parallelized in some way. Parallelization is advantageous both for the computation of ambiguity functions (particularly in the case of higher dimensions) and especially for MFI runs, which can involve many thousands of function evaluations.

A parallelization option has been provided for MFI using ORCA. Since the basic unit of computation here is a run of ORCA at a single frequency, the parallelization was implemented so that the computations are distributed among processors according to frequency. This approach also allows convenient separation of client and server processes, with IDL code running on the client and C/Fortran code running on the servers. The parallelization was implemented as follows:

- An IDL client sockets routine was written that used the IP addresses of a number of servers (on single or multi-processor computers) and an array containing the names of the input and output files for ORCA at each of a number of frequencies.
- An ORCA executable module was produced and installed on each of a number of server processors.
- A C server routine was written that accepted a set of files for a single frequency from a client, performed an ORCA run using these files, and transferred the output file containing complex field values back to the client. Note that the server processes should not be run on the machine that is running the client.
- The IDL code for the matched field computations was modified to provide an option for parallelized multi-frequency processing. In this implementation, the input files for ORCA at all these frequencies are generated, and the IDL client sockets routine is called. This routine controls the transfer of the files to the servers and handles the receiving of the resulting output files. It continually assigns tasks to servers as they become available, until the fields at all selected frequencies have been computed. The IDL code then inputs the complex field data and performs the matching.

The implementation was designed to be general, in that it allowed distribution of N tasks (i.e., runs at a single frequency) among M processors. The servers were implemented in both Windows and Linux, with ORCA executables compiled for both environments.

# BCOMFI Display Components

The BCOMFI display functions allow the viewing of the simulated shot and ship traces, the ambiguity functions, and the results of MFI search/optimization. Certain of these applications (e.g., plots of shot and ship traces) have been implemented using IDL direct graphics, while others make use of the IDL "itools" interfaces (which include iplot, isurface, iimage and ivolume). These interfaces provide useful functionality that is not available in direct graphics (e.g., copy and paste), at the cost of speed and memory requirements (for which reason it was not used in plotting traces from Shot and Ship files). Components of the itools interfaces are used in displaying the 1D, 2D and 3D ambiguity functions, with an additional special-purpose IDL slicer tool being provided for 3D volume visualization, while direct graphics are used in display of the search/optimization results. (Note that for the latter, the screen resolution should be set to at least 1280 by 1024 to view the full extent of some of the plots.) The itools interfaces are designed for a high level of user interaction, and some effort should be made to become familiar with their functionality (a tutorial is available at the RSI website http://www.rsinc.com/idl/idl_itools.asp).

## Displaying Time Domain Shot Data Traces

Press the "Display shot data" button to plot the traces in a particular Shot file. A file browser dialog box will appear, which will display all files of the type "shotnnnnn.sav", where nnnnn is an ID number. After you select the desired file, the traces in that file, and their corresponding amplitude spectra, will be displayed in separate windows as waterfall plots.

## Displaying Time Domain Ship Data Traces

Press the "Display ship data" button to plot the data streams (actually, long traces) in a particular Ship file. A file browser dialog box will appear, which will display all files of the type "shipnnnnn.sav", where nnnnn is an ID number. After you select the desired file, the data streams in that file, and their corresponding amplitude spectra, will be displayed in separate windows as waterfall plots.

## Displaying 1D Ambiguity Functions

Press the "Display 1D ambiguity function" button to plot a 1D ambiguity function. A file browser dialog box will appear, which will display all files of the type "am1dnnnnn.sav", where nnnnn is an ID number. After you select the desired file, there may be a momentary delay while itools loads, and an itools splash screen may appear and vanish. The plot will then be displayed in an iplot window.

## Displaying 2D Ambiguity Functions

Press the “Display 2D ambiguity function” button to plot a 2D ambiguity function. A file browser dialog box will appear, which will display all files of the type “am2dnnnnn.sav”, where nnnnn is an ID number. After you select the desired file, a setup GUI will then appear, in which you can specify the conditions for the display. First, choose whether you want the display as a surface (which can be rotated in 3D) or as an image. You then have the option of resizing the 2D function before display (which you will usually want to do if the Image option is chosen). If you have chosen the Surface option, you can then specify the shading algorithm to be used (either flat or Gouraud). If you have chosen the Resize option, you can also specify the interpolation scheme (bilinear or cubic).

Once you have specified the conditions, press the Go button. There may be a momentary delay while itools loads, and an itools splash screen may appear and vanish. If the Surface option was chosen, the ambiguity function will presently be displayed in an isurface window. You can use the Rotate button on this window to rotate and examine the surface in 3D. If the Image option was chosen, the function will be displayed as an iimage window. Note that this plot does not display and label the axes, but these may be added using options in the itools interface.

## Displaying 3D Ambiguity Functions

Press the “Display 3D ambiguity function” button to plot a 3D ambiguity function. A file browser dialog box will appear, which will display all files of the type “am3dnnnnn.sav”, where nnnnn is an ID number. After you select the desired file, there may be a momentary delay while itools loads, and an itools splash screen may appear and vanish. Then two separate interfaces will be displayed.

One of these is the ivolume tool, which displays a bounding box with labeled axes, and allows rendering using the button on the right side, and image plane and isosurface generation using the Operations→Volume menu item. Note that you may sometimes have to invoke the Edit→Select All menu item to enable this functionality. You can use the Rotate button on this window to rotate and examine the isosurface in 3D.

The second interface is the 3D Data Visualizer (Slicer3) interface, which provides a different interface for slice, isosurface, and projection operations through its Mode droplist. Note that you may need to use the Tools→Erase menu item to clear the display area of existing images before displaying a new one. Also note that while the ivolume interface has the capability of copying and pasting the display, the Slicer3 interface does not, and so to save a Slicer3 image, a screen dump utility would have to be used.

## Displaying Results of MFI Search Optimization

Press the “Display MFI search/optimization results” button to show several different types of results from an MFI search/optimization run. A file browser dialog box will appear, which will display all files of the type “mfopnnnnn.sav”, where nnnnn is an ID number. After you select the desired file, a Setup GUI will then appear, from which you may select the following options for display:

***Search***. This option plots the successive values of mismatch during the random global search stage. Note that, since mismatch is plotted, the best matches are those with the *lowest* values.

***Optimization convergence***. This option plots the evolution of the objective function value and the corresponding parameters during the course of an optimization. It is used in conjunction with the droplist at the bottom of the GUI, which specifies which one of the *M* best matches found during the search stage is to be viewed. (For example, if 0 is chosen, the optimization course for the best match is shown; if 1 is chosen, the course for the second best match is chosen, and so on.) The plots are displayed as objective function, or parameter value, versus the number of function calls.

***1D ambiguity functions at optimum***. This option plots the Bartlett matches obtained as each separate parameter is varied from its lower to its upper bound along a line passing though the “best” optimum (i.e., that with the lowest objective function value). It provides an indication of the sensitivity and ambiguity associated with each parameter.

***Show groups found***. This displays a text window containing information about the conditions for the run and the results of grouping the various optima found. Each group is displayed separately, along with the number of optima (convergences) in that group, the mean function value and parameter values for the group, and, where possible, the standard deviations of the multiple estimates for that group. The results of the peak analysis for the 1D ambiguity functions computed for the best group are then listed separately for each parameter. These include the number of peaks and extrema in the function, the dynamic range (on a scale from 0 to 1), the peak width in both relative units (where 1.0 represents the range between lower and upper bounds) and absolute (physical) units, and the sensitivity (i.e., the RMS variation of the function value between adjacent points along the line).

***Print groups found***. This sends the information displayed in the above text window to the default printer.

When you have selected one of the above options, press the Go button. The program will then perform the specified action.

### Displaying Results of MFI Monitoring

Press the “Display MFI monitoring results” button to show the results from an MFI monitoring run. A file browser dialog box will appear, which will display all files of the type “moninnnnn.sav”, where nnnnn is an ID number. After you select the desired file, several plots will then be displayed in an IDL graphics window. The top left plot contains the values of the optimized objective function (mismatch) for each of the time periods during the monitoring run, while the other plots contain the optimized estimates for the selected parameters for each of the time periods. Note that the range plot can provide an indication of the consistency of the source position estimation, and may help in the interpretation of the other parameter estimation results.

## Computing Environment

The BCOMFI development was done on a Windows XP computer running the IDL Development Environment (IDL DE), and on which executables of the Fortran programs ORCA and RAM were prepared and installed.

The BCOMFI software delivered in the zipfile “bcomfi.sav” requires a Windows computer on which the IDL VM has been installed. The BCOMFI executable runs under the IDL VM and calls the ORCA and RAM executables provided with the delivery.

The parallelized option for matched field computations is designed for the client process to run on a Windows computer, but to make use of ORCA servers running on either Windows or Linux computers to perform the propagation modeling for the matched field computations. (Note that RAM is currently not involved in the matched field components.) Hence, this parallelized option can be run with a client Windows computer containing the IDL VM and a network of server computers – either Windows or Linux – on which ORCA server processes are running. Note that the server processes should not be run on the machine that is running the client.

Hence, the computing environment for using BCOMFI is:

- a Windows computer on which the IDL VM is installed, which allows the simulation, conversion, preprocessing, matched field, and display components of BCOMFI to be run;
- optionally, a network of Windows and Linux computers with ORCA servers installed and running, which allows the matched field components (ambiguity function generation and MFI) to be efficiently run.

## Data Structures

We note here that the BCOMFI data structures described in this section also

correspond to files, such that a file with a particular five-character prefix (e.g., aray00023.dat) would hold a data structure of the same name (an Aray structure).

## Am2d

The Am2d structure is used as an example of structures used in ambiguity function generation (the Am1d and Am3d structures are analogous). The structure contains the following tags:

- ***param1***. The first parameter to vary.
- ***init1***. The lower limit of the domain to vary param1.
- ***final1***. The upper limit of the domain to vary param1.
- ***incr1***. The increment for varying param1.
- ***num1***. The number of points for varying param1.
- ***param2***. The second parameter to vary (for 2D only).
- ***init2***. The lower limit of the domain to vary param2 (for 2D only).
- ***final2***. The upper limit of the domain to vary param2 (for 2D only).
- ***incr2***. The increment for varying param2 (for 2D only).
- ***num2***. The number of points for varying param2 (for 2D only).
- ***layer1_speed_constant***. A flag indicating whether the compressional speed in layer 1 is to be held constant.
- ***layer1_density_constant***. A flag indicating whether the density in layer 1 is to be held constant.
- ***layer2_speed_constant***. A flag indicating whether the compressional speed in layer 2 is to be held constant.
- ***layer2_density_constant***. A flag indicating whether the density in layer 2 is to be held constant.
- ***layer3_speed_constant***. A flag indicating whether the compressional speed in layer 3 is to be held constant.
- ***layer3_density_constant***. A flag indicating whether the density in layer 3 is to be held constant.
- ***aray_id***. The ID number for the input Aray file containing the array geometry.
- ***wssp_id***. The ID number for the input Wssp file containing the water column sound speed profile.
- ***hpdt_id***. The ID number for the input Hpdt file containing the data for MFI.
- ***am2d_id***. The ID number for the output Am2d file to contain the ambiguity function.
- ***all_freq***. A flag indicating whether all the frequencies in the Hpdt file are to be used for matching.
- ***num_freq***. The number of frequencies in the Hpdt file that are to be used for matching.
- ***freq***. The frequencies in the Hpdt file that are to be used for matching.
- ***use_freq***. A flag indicating which of the frequencies in the Hpdt file (in order) are to be used for matching.

- ***match_fun***. A string indicating the scheme for combining the results at multiple frequencies.
- ***matches***. A 3D array containing the individual matches at each frequency and pair of parameter values.
- ***amb_fun_incoh***. A 2D array containing the ambiguity function computed using the standard Bartlett processor (i.e., incoherent with frequency).

## Aray

The Aray structure holds the data for the array, and contains the following tags:

- ***description***. A text description of the array type of characteristics.
- ***num_sens***. The number of sensors in the array.
- ***tilt_angle***. The tilt angle from vertical (degrees).
- ***tilt_direction***. The direction in which the array is tilted (degrees true).
- ***sens_x***. The *x*-coordinates of the sensors (before tilt is applied).
- ***sens_y***. The *y*-coordinates of the sensors (before tilt is applied).
- ***sens_z***. The *z*-coordinates of the sensors (before tilt is applied).
- ***tether_z***. The tether depths of the sensors (allowing the application of tilt).

## Hpdt

The Hpdt structure holds the simulated and real frequency domain data to be used for input to MFI, and contains the following tags:

- ***hpdt_id***. The ID number for the Hpdt file containing the data for MFI.
- ***aray_id***. The ID number for the Aray file containing the array geometry.
- ***num_sens***. The number of sensors in the array from which the data were obtained.
- ***env***. A string specifying which type of environment to use ('const_or_slope' or 'dem').
- ***prop_model***. A string containing the propagation model to be used ('ram' or 'orca').
- ***slope***. The bottom slope in degrees, if RAM is to be used (with positive slopes the depth increases from the array to the source).
- ***num_freq***. The number of frequencies at which data are present.
- ***freq***. A vector containing the actual frequencies at which data are present.
- ***data_type***. A string with a value of either "vector" or "matrix".
- ***seed***. A seed for the random number generator (used for noise and phase randomization).
- ***num_seg_int***. The number of segments in one integration time.
- ***num_int_tot***. The total number of integration times.
- ***seg_time***. The time for a single data segment.
- ***int_time***. The integration time for the data.
- ***tot_time***. The total time for the full set of data.

- ***wn_level_db***. The white noise level at a sensor.
- ***sn_level_db***. The spherical noise level at a sensor.
- ***cn_level_db***. The cylindrical noise level at a sensor.
- ***num_source***. The number of sources (max 2).
- ***source_level_db***. The source intensity levels (dB re 1 µPa at 1 m).
- ***source_depth***. The depth(s) of the source(s).
- ***source_range***. The range(s) of the source(s).
- ***source_bearing***. The bearing(s) of the source(s).
- ***source_speed***. The speed(s) of the source(s).
- ***source_heading***. The heading(s) of the source(s)
- ***fdata***. A multi-dimensional array containing the complex fields for the sensors, frequencies, and times.

## Mfop

The Mfop structure holds the conditions for, and the results of, an MFI run; it is made up of six other structures, as follows:

### Ctrl

The Ctrl structure holds the basic conditions for an MFI run, and contains the following tags:

- ***layer1_speed_constant***. A flag indicating whether the compressional speed in layer 1 is to be held constant.
- ***layer1_density_constant***. A flag indicating whether the density in layer 1 is to be held constant.
- ***layer2_speed_constant***. A flag indicating whether the compressional speed in layer 2 is to be held constant.
- ***layer2_density_constant***. A flag indicating whether the density in layer 2 is to be held constant.
- ***layer3_speed_constant***. A flag indicating whether the compressional speed in layer 3 is to be held constant.
- ***layer3_density_constant***. A flag indicating whether the density in layer 3 is to be held constant.
- ***aray_id***. The ID number for the input Aray file containing the array geometry.
- ***wssp_id***. The ID number for the input Wssp file containing the water column sound speed profile.
- ***hpdt_id***. The ID number for the input Hpdt file containing the data for MFI.
- ***mfop_id***. The ID number for the output Mfop file to contain the results of the MFI.
- ***sopt_id***. The ID number for the input Sopt file containing the Mfop data to be used for monitoring.
- ***prop_model***. A string indicating the propagation model to be used (‘orca’ or ‘ram’).

- ***all_freq***. A flag indicating whether all the frequencies in the Hpdt file are to be used for matching.
- ***num_freq***. The number of frequencies in the Hpdt file that are to be used for matching.
- ***freq***. The frequencies in the Hpdt file that are to be used for matching.
- ***use_freq***. A flag indicating which of the frequencies in the Hpdt file (in order) are to be used for matching.
- ***match_fun***. A string indicating the scheme for combining the results at multiple frequencies.
- ***parallelize***. A flag indicating whether the MFI should be run in a parallel distributed processing environment.
- ***num_servers***. The number of servers for parallelization.
- ***servers***. The IP addresses of the servers for parallelization.
- ***ports***. The ports on the servers to be used for parallelization.
- ***num_search***. The number of random samples of parameters to use in the search stage of the algorithm.
- ***num_best***. The number of best matches from the search stage that are to be optimized during the optimization stage of the MFI.
- ***num_its***. The maximum number of iterations for the DFPMIN (Davidon-Fletcher-Powell) optimization algorithm to perform.
- ***reg_factor***. The regularization factor to apply in computing the objective function.
- ***num_inter***. The number of intermediate points on a hyperspace line between two optima along which to evaluate the objective function in order to detect whether the optima are estimates of the same peak and should be grouped.
- ***thresh_group***. The threshold used to determine whether two optima (minima) are estimates of the same peak. If the objective functions at the two optima differ by this amount or more, or if any point on the above line connecting the optima is more than this amount greater than the optimum value at the endpoints, the optima are taken to be different and are not grouped.
- ***num_params***. The total number of parameters in the overall model here, 23).

## Opt

Opt is a vector of structures characterizing how each parameter is to be involved in the optimization and giving information about the widgets for the GUI. Each structure of Opt contains the following tags:

- ***wid_yesno***. The ID of the IDL widget for the checkbox in the GUI.
- ***wid_lower***. The ID for the IDL widget for the lower bound field in the GUI.
- ***wid_upper***. The ID for the IDL widget for the upper bound field in the GUI.
- ***wid_fixed***. The ID for the IDL widget for the fixed field in the GUI.
- ***optimize***. An flags indicating whether this parameter is to be optimized.
- ***lower***. The lower bound for this parameter, if it is to be optimized.
- ***upper***. The upper bound for this parameter, if it is to be optimized.
- ***fixed***. The fixed value for this parameter, if it is not to be optimized

## Search

The Search structure contains the results of the global search stage of MFI, and has the following tags:

- ***f***. A vector containing the function values for the searches.
- ***x***. A 2D array containing the parameter values used for the searches.

## Optim

The Optim structure contains the results of all the optimizations done during the optimization stage of MFI, and has the following tags:

- ***num_best***. The number of best matches from the search stage that were optimized.
- ***p_f***. A vector of pointers; the *i*th pointer points to a vector containing the function values that were computed during the course of the *i*th optimization.
- ***p_x***. A vector of pointers; the *i*th pointer points to a 2D array containing the parameter values used during the course of the *i*th optimization.
- ***f_min***. A vector containing the optimized function values for the optimizations.
- ***x_min***. A 2D array containing the parameters corresponding to the optimized function values for the optimizations.

## Groups

The Groups structure contains the results of the grouping analysis and has the following tags:

- ***num_group***. The number of different groups found.
- ***num_in_group***. A vector containing the number of convergences (equivalent optima) in each group.
- ***p_f***. A vector of pointers; the *i*th pointer points to a vector containing the function values for the individual optima in *i*th group.
- ***p_x***. A vector of pointers; the *i*th pointer points to a 2D array containing the parameter values for the individual optima in the *i*th group.
- ***x_mean***. A 2D array containing the means for the parameters in the groups.
- ***x_sd***. A 2D array containing the standard deviations for the parameters in the groups.
- ***x_corr***. A 3D array containing the correlation matrices for the parameters in the groups.
- ***x_prob***. A 3D array containing the significance levels of the correlations in the correlation matrices for the parameters in the groups.

## Peak

The Peak structure contains the results of the 1D ambiguity function for each parameter passing through the best optimum found, and has the following tags:

- ***num_peak***. A vector containing the number of peaks (local maxima) in the function for each parameter.
- ***num_extrema***. A vector containing the number of extrema (local minima and maxima) in the function for each parameter.

- ***dymanic_range***. A vector containing the dynamic range (maximum – minimum) in the function for each parameter.
- ***width***. A vector containing the estimated width of the peak at half height for each parameter.
- ***sens_rms***. A vector containing the estimated sensitivity (RMS difference of the adjacent function points) for each parameter.

## Moni

The Moni structure contains the setup conditions and results for the monitoring run, and contains the following tags:

- ***sopt_id***. The ID number for the input Sopt file containing the conditions for the search/optimization stage of the MFI monitoring.
- ***repl_id***. The ID number for the input Repl file containing the replica vectors for the 2D ambiguity function stage of the MFI monitoring.
- ***moni_id***. The ID number for the output Moni file containing the conditions and results of the MFI monitoring.
- ***num1_grid_incr***. The number of first dimension increments on either side of the maximum in the depth-range (ORCA option) or range-bearing (RAM option) 2D ambiguity function grid to use for setting the parameter bounds during the search/optimization stage.
- ***num1_grid_incr***. The number of second dimension increments on either side of the maximum in the depth-range (ORCA option) or range-bearing (RAM option) 2D ambiguity function grid to use for setting the parameter bounds during the search/optimization stage.
- ***data_type***. . A string with a value of either “vector” or “matrix” (obtained from Hpdt file).
- ***num_seg_int***. The number of segments in one integration time (obtained from Hpdt file).
- ***num_int_tot***. The total number of integration times (obtained from Hpdt file).
- ***seg_time***. The time for a single data segment (obtained from Hpdt file).
- ***int_time***. The integration time for the data (obtained from Hpdt file).
- ***tot_time***. The total time for the full set of data (obtained from Hpdt file).
- ***p_mfop***. A vector of pointers to Mfop structures which contain the conditions for, and the results of, the monitoring run.

## Repl

- ***param1***. The first parameter to vary (‘water_depth’ or ‘source_range’).
- ***init1***. The lower limit of the domain to vary param1.
- ***final1***. The upper limit of the domain to vary param1.
- ***incr1***. The increment for varying param1.

- ***num1***. The number of points for varying param1.
- ***param2***. The second parameter to vary ('source_range' or 'source_bearing').
- ***init2***. The lower limit of the domain to vary param2.
- ***final2***. The upper limit of the domain to vary param2.
- ***incr2***. The increment for varying param2.
- ***num2***. The number of points for varying param2.
- ***freq***. The frequencies in the Hpdt file that are to be used for matching.
- ***vecs***. A 4D (sensor, frequency, num1, num2) complex-valued array containing the replica vectors for the ambiguity function.

## Ship

The Ship structure holds the traces for a single shot, and contains the following tags:

- ***aray_id***. The ID number for the Aray file containing the array geometry.
- ***shot_id***. The ID number for the Shot file containing the traces for the shots.
- ***ship_id***. The ID number for this Ship file.
- ***num_sens***. The number of sensors in the array.
- ***num_time_pt***. The number of samples in each time series in the structure.
- ***samp_freq***. The sampling frequency for the traces.
- ***source_depth***. The depth of the source.
- ***source_range***. The range of the source.
- ***source_bearing***. The bearing of the source.
- ***tdata***. A 2D array containing the time series for the sensors.

## Shot

The Shot structure holds the traces for a single shot, and contains the following tags:

- ***aray_id***. The ID number for the Aray file containing the array geometry.
- ***wave_id***. The ID number for the Wave file containing the wavelet.
- ***shot_id***. The ID number for the Shot file containing the traces for the shots.
- ***num_sens***. The number of sensors in the array.
- ***samp_freq***. The sampling frequency for the traces.
- ***min_freq***. The minimum frequency in the band for which ORCA broadband should compute a normal mode model (must be positive).
- ***max_freq***. The maximum frequency in the band for which ORCA broadband should compute a normal mode model (must be less than samp_freq).
- ***num_fft***. The number of points in the FFT to be performed on the frequency domain data, and which corresponds to the number of points in the traces (power of 2). Care should be taken to make this large enough to prevent wrap-around artifacts in the traces.
- ***source_depth***. The depth of the source.

- ***source_range***. The range of the source.
- ***source_bearing***. The bearing of the source.
- ***tdata***. A 2D array containing the shot traces for the sensors.

### Wave

The Wave structure holds an acoustic wavelet, and contains the following tags:

- ***num***. The number of elements in the wavelet.
- ***samp_freq***. The sampling frequency for the wavelet points.
- ***wdata***. A vector containing the elements of the wavelet.

### Wssp

The Wssp structure holds a water column sound speed profile, and contains the following tags:

- ***depth***. A vector containing the depths for the profile.
- ***speed***. A vector containing the sound speeds for the profile at the corresponding depths.

## Program Modules

### Simulate TD Shot Data

This component is implemented by the Sim_shot_data.pro module. Its main component functions and procedures are as follows:

- ***Sim_shot_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_sim_shot_data***. Initializes default values for the fields of the GUI.
- ***Check_sim_shot_data***. Checks for existence of files and consistency of the data entered by the user.
- ***Sim_shot_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_sim_shot_data***. Performs a run using the input data specified by the user, i.e., calling ORCA broadband, processing the resulting FFT file to convolve the impulse response with a wavelet, generating traces for each sensor of the array and adding noise to the traces.
- ***Read_aray***. Reads in the data for the array.
- ***Read_wave***. Reads in the wavelet to convolve with the impulse responses.
- ***Read_orca_fft***. Reads in the ORCA “FFT file” – the frequency domain impulse responses for each sensor of the array.
- ***Gen_array_geom_file***. Uses the data in the Aray file to generate an array geometry file for use by ORCA.

- ***Gen_bb_opt_file***. Uses the data input by the user to generate an options file for ORCA to perform a run using the broadband option.

In addition, an executable ORCA program is required which will input the specified data and run conditions and output the FFT file.

The Sim_shot_data component uses the Aray, Wave, and Shot data structures. A description of these structures and their tags is given in the previous section.

## Simulate Ship FD Hpdt Data

This component is implemented by the Sim_ship_data.pro module. Its component functions and procedures are as follows:

- ***Sim_ship_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_sim_ship_data***. Initializes default values for the fields of the GUI.
- ***Check_sim_ship_data***. Checks for existence of files and consistency of the data entered by the user.
- ***Sim_ship_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_sim_ship_data***. Performs a run using the input data specified by the user, i.e., calling ORCA or RAM, adding noise, performing cross-spectral matrix estimation if specified, and writing the data to an Hpdt file.
- ***Read_aray***. Reads in the data for the array.
- ***Gen_sim_orca_sv***. Sets up files for an ORCA run and then calls ORCA to simulate a signal vector at an array for a particular source, array, and frequency.
- ***Gen_array_geom_file***. Uses the data in the Aray file to generate an array geometry file for use by ORCA.
- ***Gen_cw_opt_file***. Generates an options file for the ORCA run in cw mode.
- ***Gen_ram_sv***. Sets up files for a RAM run and then calls RAM to simulate a signal vector at an array for a particular source, array, and frequency.
- ***Gen_ramgeo_file***. Generates an input data file for RAM.
- ***Read_ram_grid***. Reads in a grid of field values computed by RAM.
- ***Gen_wn_matrix***. Generates a white noise matrix.
- ***Gen_sn_matrix***. Generates a spherical noise matrix.
- ***Gen_cn_matrix***. Generates a cylindrical noise matrix.
- ***Chol_matrix***. Performs a Cholesky decomposition on the input matrix.
- ***Gen_nv***. Generates an estimated noise vector, based on the Cholesky decomposition.

In addition, executable ORCA and RAM programs are required which will input the specified data and run conditions and output the complex fields to be used as signal vectors.

The Sim_ship_data component uses the Aray and Hpdt data structures. A description

of these structures and their tags is given in a later section.

## Convert TD Shot Data to TD Ship Data

This component is implemented by the Shot_to_ship_data.pro module. Its component functions and procedures are as follows:

- ***Shot_to_ship_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_shot_to_ship_data***. Initializes default values for the fields of the GUI.
- ***Check_shot_to_ship_data***. Checks for existence of files and consistency of the data entered by the user.
- ***Shot_to_ship_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_shot_to_ship_data***. Performs a run using the input data specified by the user, i.e., reading in the shot data, generating ship data and writing to a Ship file.

## Convert P16 Data to Shot or Ship Data

This component is implemented by the P16_to_ss_data.pro module. Its component functions and procedures are as follows:

- ***P16_to_ss_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_ p16_to_ss_data***. Initializes default values for the fields of the GUI.
- ***Check_ p16_to_ss_data***. Checks for existence of files and consistency of the data entered by the user.
- ***P16_to_ss_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_ p16_to_ss_data***. Performs a run using the input data specified by the user, i.e., reading in the P16 data, converting to ship data and writing to a Shot or Ship file.

## Preprocess TD Shot Data to FD Hpdt Data

This component is implemented by the Prep_shot_data.pro module. Its component functions and procedures are as follows:

- ***Prep_shot_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_prep_shot_data***. Initializes default values for the fields of the GUI.
- ***Check_prep_shot_data***. Checks for existence of files and consistency of the data entered by the user.
- ***Prep_shot_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.

- ***Run_prep_shot_data***. Performs a run using the input data specified by the user, i.e., reading in and FFTing the shot data, and writing the cross-spectral matrices to an Hpdt file.

## Preprocess TD Ship Data to FD Hpdt Data

This component is implemented by the Prep_ship_data.pro module. Its component functions and procedures are as follows:

- ***Prep_ship_data***. Sets up the GUI for specification of the files and input conditions.
- ***Set_prep_ship_data***. Initializes default values for the fields of the GUI.
- ***Check_prep_ship_data***. Checks for existence of files and consistency of the data entered by the user.
- ***Prep_ship_data_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_prep_ship_data***. Performs a run using the input data specified by the user, i.e., reading in, overlapping, windowing and FFTing the ship data, performing cross-spectral matrix estimation, and writing the cross-spectral matrices to an Hpdt file.

## Generate nD Ambiguity Function

These components are implemented by the modules Mfi_1d_amb_fun.pro, Mfi_2d_amb_fun.pro, and Mfi_3d_amb_fun.pro. The component functions and procedures for the 2D case (the others are analogous) are as follows:

- ***Amb_2d***. Sets up the GUI for specification of the files and input conditions.
- ***Set_params***. Initializes default values for the parameters in the GUI.
- ***Set_amb_2d***. Initializes default values for the non-parameter fields of the GUI.
- ***Check_amb_2d_parms***. Checks for existence of files and consistency of the data entered by the user.
- ***Amb_2d_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_amb_2d***. Performs a run using the input data specified by the user, i.e., sets up the values of the parameters and computes the matches at these values.
- ***Read_aray***. Reads in the data for the array.
- ***Read_wssp***. Reads in the sound speed profile to use for the run.
- ***Read_servers***. Reads in the default values for the IP addresses and ports for the remote orcaservers.
- ***Ip_addr_array_to_string***. Converts a string vector of IP addresses to their numeric 2D array equivalents.
- ***Ip_addr_string_to_array***. Converts a numeric 2D array of IP addresses to their string vector equivalents.

- ***Gen_mfi_orca_sv***. Sets up and performs an ORCA run to generate a signal (replica) vector based on the run conditions.
- ***Gen_array_geom_file***. Uses the data in the Aray file to generate an array geometry file for use by ORCA.
- ***Gen_cw_opt_file***. Uses the data input by the user to generate an options file for ORCA to perform a run using the continuous wave option.
- ***Gen_svp_file***. Generates an svp file for ORCA, based on the parameter values.
- ***Bartlett***. Computes the output of the Bartlett power processor, for either vector or matrix data.

## Perform MFI Search/Optimization

This component is implemented by the Mfi_search_opt.pro module. Its component functions and procedures are as follows:

- ***Mfi_search_opt***. Sets up the GUI for specification of the files and input conditions.
- ***Set_params***. Initializes default values for the parameters in the GUI.
- ***Check_mfi_search_opt_parms***. Checks for existence of files and consistency of the data entered by the user.
- ***Mfi_search_opt_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_mfi_search_opt***. Performs an MFI run using the input data specified by the user, i.e., performs the initial search, optimizes the objective function for multiple starting estimates and performs the grouping and peak analysis.
- ***Read_aray***. Reads in the data for the array.
- ***Read_wssp***. Reads in the sound speed profile to use for the run.
- ***Read_servers***. Reads in the default values for the IP addresses and ports for the remote orcaservers.
- ***Ip_addr_array_to_string***. Converts a string vector of IP addresses to their numeric 2D array equivalents.
- ***Ip_addr_string_to_array***. Converts a numeric 2D array of IP addresses to their string vector equivalents.
- ***Gen_mfop***. Returns an Mfop structure based on the input parameters, which are the component structures of the Mfop structure.
- ***Mfi_func***. Evaluates the objective function to be minimized in MFI. This is a composite function consisting of the sum of the Bartlett processor, a penalty function if any parameter exceeds its bounds, and a regularization term.
- ***Mfi_dfunc***. Estimates the gradient of the objective function using central differences.
- ***Plot_opt***. Produces a plot of the evolving function value and the values of the parameters.
- ***Gen_mfi_orca_sv***. Sets up and performs an ORCA run to generate a signal (replica) vector based on the run conditions.

- ***Gen_array_geom_file***. Uses the data in the Aray file to generate an array geometry file for use by ORCA.
- ***Gen_cw_opt_file***. Uses the data input by the user to generate an options file for ORCA to perform a run using the continuous wave option.
- ***Gen_svp_file***. Generates an svp file for ORCA, based on the parameter values.
- ***Gen_group***. Groups multiple optima corresponding to the same peak.
- ***Analyze_group***. Performs statistical analysis in the multiple parameter estimates for each group.
- ***Analyze_peak***. Performs post-processing on the best optimum to estimate peak width, sensitivity, etc., for each parameter.
- ***Bartlett***. Computes the output of the Bartlett power processor, for either vector or matrix data.

## Perform MFI Monitoring

This component is implemented by the Mfi_monitor.pro module. Its component functions and procedures are as follows:

- ***Mfi_monitor***. Sets up the GUI for specification of the files and input conditions.
- ***Set_mfi_monitor***. Initializes default values for the fields of the GUI.
- ***Check_mfi_monitor***. Checks for existence of files and consistency of the data entered by the user.
- ***Mfi_monitor_event***. Handles the events generated by the GUI and sets the values in the data structures based on the input by the user.
- ***Run_mfi_monitor***. Performs an MFI monitoring run using the input data specified by the user, i.e., for each data set, estimates the water depth and source range using precomputed replicas, uses thee result to define a multi-parameter search region, and performs search/optimization to estimate the geoacoustic parameters for that data set.
- ***Run_mfi_search_opt***. Performs an MFI search/optimization run using the input data specified by the user, i.e., performs the initial search, optimizes the objective function for multiple starting estimates and performs the grouping and peak analysis.
- ***Bartlett***. Computes the output of the Bartlett power processor, for either vector or matrix data.

# MFI and Monitoring Options

## Introduction

The Gulf of Mexico Hydrates Research Consortium is managing the installation and testing of a Gas Hydrates Seafloor Observatory in the Mississippi Canyon region of the Gulf of Mexico (Block MC 118). The main function and objective of this Observatory is to provide for the long-term monitoring of interactions between the near-seabed hydrocarbon system and the hydrate stability zone section, particularly regarding the formation/dissociation of hydrates, transiting of pore fluids, and possible vertical migration of the base of the hydrate stability zone.

Among the components of the Observatory will be several hydrophone arrays and one or more sets of four-component (4C) sensors. These sensors each consist of a hydrophone for recoding pressure waves and a set of three mutually orthogonal accelerometers for measuring shear waves. It is intended that, during the initial calibration stage involving impulsive sound sources, data from these sensors will be used to provide preliminary information about the sub-bottom structure in the region of the array. Subsequent surveys will allow further estimation of the sub-bottom structure, and provide an opportunity for detection of changes.

During this and a previous project under contract with the University of Mississippi, BCS developed a software system for applying matched field inversion (MFI) techniques to acoustic array data with the aim of estimating sub-bottom geoacoustic parameters. This technique is based on matching the measured fields of acoustic (compressional) waves with model fields computed using an acoustic propagation model. In the most recent version of the MFI software, the ability is provided to use acoustic data and modeling to estimate a layered geoacoustic model for range-dependent environments where the bathymetry is known as well as for range-independent environments.

Even with this capability, however, it may be that further enhancements to the MFI approach are needed in order to detect sub-bottom changes of interest at the Observatory site. We propose here two extensions to MFI, namely 3D MFI, and MFI using replica vectors, that may have the potential to increase effectiveness of this general approach in the present monitoring. We also note the possible application of a third approach to monitoring based on ambient noise.

In addition, the use of shear waves may have the potential to provide key additional information compared to using only compressional waves. For example, the lower sound speed of shear waves results in their having smaller wavelengths, and thereby can allow the resolution of thinner layers, particularly near the surface, where the shear speeds are generally low. Also, the different propagation characteristics of shear waves can permit imaging under some circumstances where compressional waves fail to produce useful images (e.g., when gas clouds are present in the layers).

This section of the report first examines the potential use of shear waves in matched field techniques. It then outlines some possible limitations of the initial implementation

of MFI for the current monitoring application. Extensions and alternatives to the original approach for MFI are then proposed and described. An overview of shear waves is then given, and the properties of the sensors to be used for measuring the compressional and shear waves are provided. Finally, the (tentative) configurations and temporal staging of the 4C sensor arrays are presented and a synopsis of some possible processing methods for 4C data is given.

## Shear Waves and Matched Field Techniques

One of the aims of the present project was to survey methods whereby the shear wave data could also be used for MFI. BCS performed a web literature search and determined that seismic shear wave data have been used in pilot studies to perform matched field processing (MFP) for heavy vehicle tracking applications (Harben et al., Harris et al.) and proposed as a technique for passive seismic monitoring of possible long-term underground storage sites (Cheng et al.). In both these cases, the aim was to localize the sources (tanks and rockfalls, respectively) by assuming a geological model and matching measured fields with computed replica seismic fields that would result at the sensors of a seismic array. The seismic data were modeled using a full 3D seismic wave propagation code (called E3D) developed by Lawrence Livermore National Labs. Although these applications did use matched field techniques, their focus was source localization rather than matched field inversion in the current context. In addition, a study was found where inversion techniques were used with multicomponent data to estimate sub-bottom parameters for simple models (Schalkwijk et al.); however, these techniques were based on other inversion principles than the MFI techniques of interest in the Observatory application.

BCS also contacted Evan Westwood, the author of the ORCA code (which includes shear wave modeling for bottom layers), to ask whether the shear wave potential might be modeled using ORCA, and whether this might be used to support MFI of 4C data. The response was that all the quantities that would be needed (particle velocities and potentials) are calculable within ORCA, but their use has so far been restricted to determining their contribution to the acoustic field. Using ORCA to perform the shear wave modeling would require a deep understanding of the quantities required and the code itself (which is large and complex) and would represent a very large effort. Hence, it is believed that it is possible in principle, although not in practice at this time, to perform MFI using ORCA-generated quantities to match with measured shear wave data.

Overall, however, it does not appear that any studies have been performed to date where MFI techniques have been applied to 4C data to estimate sub-bottom parameters.

## Possible Limitations of Current MFI

When BCS began work on the Observatory data management and processing project in

December 2004, it had been proposed that matched field inversion (MFI) techniques from the vertical array and the hydrophone components of the other arrays might be applied to the analysis of acoustic data to estimate geometric and geoacoustic properties of the environment in the region of the Observatory. The advantages of MFI techniques in this context were identified as follows:

- MFI can use sources of opportunity (passing ships); these noise sources are expected to pass frequently within a few kilometers of the Observatory and to continue to do so over its lifetime.
- It does not require that the source location be separately provided, and can estimate the location as one of the parameters in the inversion.
- It generally yields a layered geoacoustic model with quantitative estimates of the sub-bottom parameters. Hence, it could be adapted for monitoring of a gas-bearing layer by estimating changes in the velocities and densities of the layers.
- It can be used with impulsive or ship noise sources.
- It can use data at various frequencies and does not require that these frequencies be coherent.
- It can be applied in both range-independent and range-dependent environments.

Proceeding on this basis, BCS developed a comprehensive software system for simulation and MFI using the normal mode programs ORCA and the parabolic equation code RAM to match acoustic compressional wave fields for an environment involving a number of layers over a basement halfspace. The overall strategy here was to compute reference geoacoustic models during a calibration stage and use these to look for mismatches with these models during the subsequent monitoring stage.

During this development, a general possible issue concerning the application of MFI has become clearer as the understanding of the nature of the hydrates-bearing region to be monitored by the Observatory has evolved. When BCS first started work on this project in late 2004, the site for the Observatory had just been determined. Although the choice of a feature in Mississippi Canyon 118 was made in 2004, it was early 2005 before seismic and other techniques were applied to characterize the site in some detail. These studies showed a significant degree of range dependence and the presence of some localized features such as a fault and “bright spots”. Further work, the results of which were described during the most recent Consortium meeting on February 16-17, 2006 (which BCS attended), indicated that the hydrates-bearing region of interest is both very localized and very heterogeneous. This region occupies less than one square kilometer and contains mud, deformed sediments, gas pockets, faults, vents, carbonates (arising from both bivalve and microbial activity) and hydrates (which account for between 1 and 10% of the volume) at various levels, all superimposed over a salt dome, with deep sources of thermogenic methane percolating upwards.

Given this new and complex picture of the region of interest, the present MFI approach could possibly have the following potential drawbacks:

- It is currently restricted to layered range-independent environments or environments with known bathymetry and might not provide good matches with data from complex environments with localized sub-bottom features.
- The technique yields averaged or effective parameter estimates over the source-receiver path rather than an image of the structures along that path.
- The output often exhibits multiple optima, ambiguity and/or non-uniqueness in parameter space, so that several different models can sometimes provide comparably good fits to the data. It is expected that the greater the complexity of the model, the greater the potential for ambiguity may be. In addition, some of the parameters are correlated, further adding to the non-uniqueness.
- The low impedance contrast profiles thought to be present in some regions of the Observatory may be difficult to estimate using MFI.
- The results can be sensitive to model errors in the water and bottom.

Hence, it is possible that existing MFI techniques involving layered models, as currently implemented, may not be able to provide the necessary level of detail required for defining and detecting changes in the complex structures of interest for the hydrates monitoring application. It will therefore likely be necessary to develop enhanced MFI techniques and/or to apply other approaches to achieve this aim.

It should be emphasized, however, that these considerations are still tentative, since no suitable acoustic data from the Observatory site have been obtained and analyzed. When such data do become available (anticipated Summer 2006), it will be possible to apply the MFI techniques already developed and to examine the effectiveness of the matching using the existing software system BCOMFI. If sufficiently good matches in fact can be found, layer-based MFI may indeed be a viable option for the monitoring application.

Nevertheless, we do recommend that MFI extensions and alternative methods should also be identified and examined for their applicability to the monitoring application. In addition to the planned seismic surveys at regular intervals, three candidates for monitoring (namely, 3D MFI, measured field MFI, and cross correlation using ambient noise) are suggested. These are described in the following sections.

## MFI Enhancements and Other Monitoring Options

### 3D MFI

MFI has generally been applied using 2D layered models, sometimes with variable bathymetry. In principle, the MFI approach could be extended further to model a moderately complex 3D environment, for example, a layered system with a gas pocket within one of the layers. The main requirements here would be a suitably parameterized model for the 3D environment. One possible approach to this would be to set up a fixed 2D grid with the bathymetry model and to define a number of data structures containing parameterized features of the environment (e.g., layers, gas pockets, and surface carbonate regions), including their 3D extents and geoacoustic

properties. These 3D features could then be used to generate the range-dependent 2D profiles to be provided to RAM. This approach could be designed to keep the parameterization for MFI purposes as sparse as possible, and a chosen subset of these parameters would be optimized during MFI.

It is envisioned that the main benefits of the 3D MFI approach would be:

- it could provide for the modeling of a more realistic and complex overall environment than just using range-independent or variable-bathymetry MFI;
- it would allow the modeling of known surface features with pronouncedly different acoustic properties from other surface regions (e.g., the localized deposits of carbonates);
- it might allow changes in the extents of large sub-bottom features (e.g., the size of a gas pocket) to be detected, as a result of the large impedance changes exhibited by these features.

However, possible drawbacks of this approach might include:

- increased times for propagation modeling, since the parabolic equation code RAM takes a much longer time to run than ORCA;
- increased times for optimization, if more parameters are involved;
- the likelihood of insensitivity to some parameters;
- increased levels of ambiguity.

These disadvantages may be more than offset by the potential of the method to perform 3D modeling of the sub-bottom. For example, based on the current knowledge of the sub-bottom in the region of the Observatory, an initial 3D model for the region could include:

- the known bathymetry;
- the known spatial distribution of surface carbonates, designating a thickness of a few meters and appropriate values for the sound speed and density of these deposits;
- two or three layers (or one or two layers with gradients) paralleling the bathymetry, above a basement halfspace;
- a thin gas-bearing layer within a defined region, situated just above the halfspace (at about 180 m depth).

Seismic section data for the region of the Observatory indicate that the layered structure is quite consistent away from the hydrates-bearing region, and it also generally persists within this region. Hence geoacoustic parameters for the layers could be estimated using MFI techniques applied to calibration data acquired for sources with source-array paths that do not intersect this region. This estimation could be done using a current version of BCOMFI.

The known distribution of surface carbonates could then be incorporated to form an initial 3D layered model. The effect of modeling the carbonates could be tested using

synthetic data or calibration data collected when the source is located on the opposite side of the hydrates-bearing region (which includes the carbonates) from the array. Finally, the effect of including the deep gas-bearing region in the geoacoustic model could also be tested using similar synthetic or calibration data.

During monitoring with a 3D array, the parameters of greatest importance would be the source locations and the spatial extents of the gas-bearing layer. The source locations could be estimated by the approaches described in the next section (note also that using the bathymetry allows the source bearing to be estimated by MFI). The aim would then be to estimate the gas layer parameters using suitable source-generated data by applying MFI procedures based on the derived geoacoustic model for the carbonates-layers-halfspace system. These gas parameters could simply consist of the start and end limits and the thickness of the gas-bearing region along the path between the source and the array.

It should also be noted that the MFI computations could be performed using only the vertical array data provided that range-dependent MFI with bathymetry is used, since bearing can then be estimated during MFI.

## Measured Field MFI

Fialkowski et al. proposed a novel approach to matched field processing that avoids the problem of lack of knowledge of the environment by using measured replicas in the matching rather than replicas computed according to a geoacoustic model. In this approach, a database of replicas would be built up by towing a source over a series of known tracks and using the data received at an array to build up measured field replica vectors over a region of accurately defined source positions. The procedure is computationally much more efficient than geoacoustic model-based MFI since no forward propagation modeling is required. The authors point out that the source must be towed slowly enough so that high-quality cross-spectral matrices can be obtained by averaging over time windows of sufficient duration, and that the acoustic field must be adequately sampled in space. They advocate a technique for suppressing noise by using as a replica vector the eigenvector that corresponds to the largest eigenvalue of the cross-spectral matrix. They also note that the quality of the database can be expected to degrade over time as the environmental parameters vary, and suggest that this problem can be reduced by using relatively low frequencies and accumulating acoustic databases under different environmental conditions.

This measured field MFI approach was developed for source localization using matched field processing, but it could also be adapted to monitoring. This could be accomplished by first generating a database of acoustic replica fields during the calibration stage, with the ship tracks directed along radials centered at the location of the array. Then, in the monitoring stage, as a source of opportunity passed nearby, its position could be estimated and the measured fields matched with those in the database. Assuming that the water column propagation did not change substantially during the time between

calibration and monitoring, a substantial mismatch would then be taken as evidence for a change in the properties of the sub-bottom structures.

Note that this procedure does not require any geoacoustic modeling of the environment and that it can be applied to regions of arbitrary range-dependence or sub-bottom complexity. It does not attempt to image the sub-bottom or to derive parameter estimates, but is simply designed to detect the presence of a significant change from the reference replica fields in the database, and possibly provide an indication of where the change may have occurred, (i.e., along the path between the array and the source at the time of the mismatch).

The goal of the measured field MFI procedure is to detect changes over time in the fields at the array that are produced by acoustic sources at known locations. Hence, the source of opportunity must be localized for the method to work. If it is not localized, then its position within (or outside) the calibration region will be unknown, and a good match (obtained, for example, by correlation with field vectors over the entire grid) might be purely fortuitous. For source localization, it is essential that both the range and the bearing of the source can be accurately estimated from the acoustic data, and that the source location falls within the calibration region. Bearing estimation in this case probably requires that the array must have some horizontal extent, preferably in two dimensions. Assuming such an overall 3D array (with vertical and 2D horizontal components), source localization might be realized by one of the following approaches:

- ***3D Beamforming***. Simple plane-wave beamforming could be applied to the output of the 3D array of hydrophones to estimate a ray in 3-space corresponding to the direct arrival from the source. The intersection of this ray with the sea surface would then give an estimate of the location of the source. Alternatively, for more accurate near-field results, the distances from each surface point to each array element could be computed and used to generate focused steering vectors for beamforming. A third option would be to use ray tracing to compute the path lengths.

- ***Simplified matched field processing***. An MFP technique might be implemented to estimate source range and bearing. This might be done by assuming a simple range-dependent water-halfspace model for bathymetry and using RAM to precompute the replica fields for sources on a 2D grid just below the surface. These fields would then be matched with the measured fields from the source of opportunity to estimate its position. This approach might also have the advantage of allowing the estimation of source position using just a vertical array.

If the estimated source location fell within the calibration region, the measured fields for the source would then be matched against the calibrated fields corresponding to the source location to determine the match for monitoring purposes.

The issue of degradation of matching over short time scales in dynamic environments is a potentially serious one in the context of this proposed MFI application. For example,

Mignerey and Orr found correlation times of a few minutes for frequencies of 300 – 500 Hz in highly dynamic environments (large diurnal internal tides, with concurrent large changes in temperature profiles). Temporal changes in the Gulf of Mexico sound speed profile are believed to be much less dramatic but such changes as do exist will still be expected to influence the acoustic propagation. To address this potential source of error, it is proposed to accumulate several databases taken from measurements made at different times of the year, and for which sound speed profiles (SSPs) were taken during the acoustic calibration measurements. Then a suitable database may be used to generate replica fields for the conditions in effect during the time of the monitoring; these would include data for different SSPs in summer/winter, hurricane season, loop current eddies, etc. Also, the choice of low frequencies would be expected to confer a greater resistance to variations in the water column SSP as well as giving better bottom penetration.

## Long-Term Cross-Correlation Using Ambient Noise

This recently developed technique for Green's function estimation using ambient noise is based on the principle that, for appropriately distributed ambient noise, the derivative of the cross-correlation between two sensors converges to the time domain Green's function for wave propagation between the sensors (Gerstoft et al.; Harrison; Lobkis and Weaver). This provides an estimate of the impulse response modified by the spectrum of the noise sources. In the context of the hydrates monitoring application, the method is potentially appealing, in that it does not require an impulsive source, it is well-suited to the long monitoring times necessary to build up cross-correlations, and it requires modest computational resources.

However, in order for the technique to be successfully applied to the monitoring application, a number of questions/issues should be addressed. These include the following:

- ***Sensor distribution***. To monitor a region of interest, a network of receivers spanning the region is required. The fixed vertical array, of course, does not provide this sampling. The set of four horizontal arrays is proposed to be deployed either as a cross or an "L" on one side of the hydrates mound, but neither of these configurations spans this feature. A cross or pinwheel configuration spanning the mound would be preferred. Likely even better would be a distributed configuration, with the sensors positioned at the vertices of a grid or a network spanning the feature. (A practical consideration here could be that for larger distributed arrays, timing synchronization issues may arise, particularly for higher frequencies.)

- ***Noise characteristics and distribution***. The method works best for ambient sources distributed more or less evenly in azimuth, but can still provide useful information when the noise is azimuthally dependent. However, in applying the method to microseismic data, strong sources have generally been removed from the data, by using thresholding techniques (or 1-bit cross-correlations). This would

presumably also need to be done for the monitoring application, to remove the effects of passing ships. The individual wavefronts in the noise must also be sufficiently coherent, both in time and space, to allow cross-correlation estimation. The noise sources should also have a frequency range to provide sufficient resolution (say, in the tens of meters). In this connection, microseisms (with spectra in the region 0.05 – 2.0 Hz) are probably too low frequency to provide the needed spatial resolution. It is anticipated that ambient noise with power in the region 20-500 Hz would be more appropriate here.

- ***Information provided***. The information obtained is simply the cross-correlation between sensor pairs. This provides a 1D estimate of the impulse response, from which travel times might be estimated. It does not provide a 2D image of either the surface characteristics or the sub-bottom volume of the region. For a network of sensors, the travel time data can be inverted to yield a low-resolution estimate of the surface wave speed on a 2D grid corresponding to the near-ocean floor. It is unknown whether the cross-correlations will be sufficiently detailed to allow estimation of sub-bottom reflections and thereby possibly provide an avenue for imaging the 3D sub-surface volume in the region of the network. This type of information would, of course, be the ultimate goal of the monitoring.

- ***Wave types***. The overall intended sensor configuration includes a 200 m vertical line array (16 hydrophones), a 200 m borehole array (16 hydrophones, 8 x 3-component sensors) and four 400 m horizontal arrays (each 16 hydrophones, 8 x 3-component sensors, the latter subject to funding approval). This configuration allows the estimation of pressure and shear waves, including water-bottom interface waves. The information provided by each wave type must be examined with a view to how the data could best be used to image or otherwise interpret the 3D hydrates structures and ultimately detect changes on various spatial scales. A careful examination of this information and a judicious choice of which wave types to use and how to analyze, process and/or invert the data could be crucial to the success of the ambient noise method in the monitoring application.

## Overview of Shear Waves

As noted above, one of the aims of the current project was to identify methods for applying MFI to shear wave data. Since no such work appears to have been published, efforts were redirected to survey methods that might be applied to the processing of the shear wave data from the accelerometers. This section outlines some properties of shear waves, and is based on information provided by Blaylock and by Barkved et al. Subsequent sections characterize the hydrophone and accelerometer components of the 4C sensors, note the configuration and staging of the 4C arrays, and summarize some possible processing methods.

### Properties

By using data from both compressional and shear waves, seismic analysts can obtain more information about the subsurface than from using just one wave type. Shear waves bring additional knowledge to a seismic study because compressional and shear waves sample different properties of the propagating medium.

Compressional-wave (or P-wave) particle motion parallels the direction of wave travel. P-wave velocity is a function of the medium's density, shear modulus and bulk modulus. From the scalar wave equation, it can be shown that for homogeneous isotropic media, compressional velocity can be represented as

$$Vp = [(k + 4m/3) / r]^{1/2},$$

where $Vp$ is P-wave velocity, $k$ is the bulk modulus (reciprocal of compressibility), $m$ is the rigidity or shear modulus, and $r$ is the bulk density of the medium. Bulk modulus is sensitive to fluid compressibility, making P-waves highly sensitive to a rock's fluid content. The dual dependence on fluid compressibility and shear modulus allows P-waves to propagate in both solids and liquids.

Shear-wave (or S-wave) particle motion is oriented perpendicular to the direction of ray-path travel. Shear-wave velocity is a function of the density and the shear modulus of the medium; a shear wave is almost insensitive to a rock's fluid content. Shear wave velocity, in homogeneous isotropic media, can be represented as

$$Vs = (m / r)^{1/2},$$

where $Vs$ is S-wave velocity, and $m$ and $r$ are as above. Because the elastic constants in these equations are always positive, it is apparent that, for a given medium, $Vp$ is always greater than $Vs$. In a given formation, shear-wave velocity and reflectivity remain unchanged whether the formation contains gas, oil or water. However, S-waves can travel only in media with nonzero shear modulus, so they can originate and propagate only in solids.

Note that for shear waves, measurements of vertical velocity/acceleration correspond to horizontally propagating (radial) waves, while measurements of radial velocity correspond to vertically propagating waves.

Measurement and comparison of P- and S-wave velocity information provide significant insight into sampled media. The most common means of accomplishing this comparison is through the use of $Vp/Vs$ ratios. Because the elastic moduli which dictate P- and S-wave velocities are integrally linked to the geophysical properties of the rock medium sampled by measurement, velocity ratio relationships are useful indicators of a wide range of geophysical properties, including lithology, porosity, pore structure, pore fluid, and other factors.

## S-Wave Generation and Propagation

In most cases, it is impractical to deploy a shear-wave source on the seafloor. Fortunately, although typical marine seismic sources do not directly generate S-waves, the P-waves they do generate can in turn generate S-waves in the subsurface. Compressional waves undergo partial conversion to shear waves at subsurface interfaces and can be detected as S-waves by seabed sensors. Recorded waves that start as P-waves and convert to S-waves are called converted waves, or PS-waves, but they can also be termed C-waves, for converted. Their counterparts, those waves that start and reflect as P-waves, can be called PP-waves.

Converted waves reflect at subsurface interfaces according to Snell's law, which relates the angles of incidence, reflection and transmission to the velocities of propagation of P- and S-waves. For P-waves reflecting at an interface, the angle of reflection equals the angle of incidence. This symmetry simplifies acquisition and processing of P-wave surveys. However, for PS-waves, the S-wave angle of reflection does not equal the P-wave angle of incidence. An S-wave always reflects more vertically than would a P-wave, because the propagation velocity of an S-wave is less than that of a P-wave. This asymmetry can complicate acquisition and processing of converted-wave surveys.

## Measurement by Multicomponent Sensors

P-waves in the ocean can be detected by hydrophones, or pressure sensors, surrounded by water. In the seabed, they can also be recorded by single-component geophones or accelerometers that detect vertical motion in the formation. Recording S-waves requires sensors that detect more than just the vertical component, so standard P-wave recording systems are inadequate. Since the S-wave field is fully three-dimensional, three-component sensors are required to characterize it. The usual form of three-component sensor is an instrument comprising three geophones or accelerometers in mutually orthogonal orientation, which allows detection of S-waves from all possible directions. The S-wave motion recorded on the horizontal components can be converted by mathematical rotation into a radial component of motion in the plane of wave propagation, and a transverse component out of the plane of wave propagation.

The addition of a hydrophone in close proximity to the three-component sensor creates a four-component (4C) sensor and it is these 4C sensors that will be part of the borehole array (and possibly other arrays) of the Seafloor Observatory. This use of more than one receiver has given rise to the term "multicomponent" sensors.

# Description of 4C Sensors

The specifications and sampling characteristics of the hydrophone and accelerometer sensors to be used in the Seafloor Observatory are outlined in tabular form in this section.

## Hydrophone

| Item | Information |
|---|---|
| Consortium contact(s) | Paul Higley, Specialty Devices, Inc. |
| Product name/number | Analog Acoustic Hydrophone |
| Manufacturer | Specialty Devices, Inc. |
| Physical quantity measured | Pressure |
| Dimensionality of measurements | 1D stream of voltages proportional to pressure (units in decibels) |
| Range | 2 Hz – 30 kHz (123 dB peak to RMS) |
| Resolution | 16 bits/sample |
| Sampling Modes: | The hydrophone is designed to be synchronized with shot points of 1-12 seconds in duration (typically 6 seconds)<br>Initially (2005): sporadic, output to data logger during separate specified time periods<br>Operationally (2006+): continuous |
| Typical sampling rate | 10,000 samples/sec |
| Peak sampling rate | 10,000 samples/sec |
| Typical data transfer rate (per sensor) | 160 kbit/sec |
| Peak data transfer rate (per sensor) | 160 kbit/sec |
| Data format(s) (proprietary or public standard) | Pseudo SEG Y (SDI to provide more details) |
| Required metadata and how they are created | A synchronized time signal is required for the acoustic signals. Sensor configuration, number of hydrophones, calibrations, and array geometry are required metadata fields. |
| Metadata standards | SEG Y format headers. (See http://www.seg.org/publications/tech-stand/seg_y_rev1.pdf) |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled (data logged in DATS)<br>Streaming (when operated continuously) |
| Post-processing requirements | Conversion to pressure, signal processing, matched field processing and inversion |
| Brochure/description available | No (third-party sensor information may be available) |
| Other relevant information | A synchronized time signal needs to be embedded in the data stream for many applications. Breaking wave processes in particular require these signals for computing time delays between hydrophones in array configurations. |

## Accelerometer

| Item | Information |
|---|---|
| Consortium contact(s) | Paul Higley, Specialty Devices, Inc. |
| Product name/number | VectorSeis SVSM |
| Manufacturer | Input/Output Inc. |
| Physical quantity measured | Acceleration (g) |
| Dimensionality of measurements | Scalar values in 3 orthogonal planes; units in g |
| Range | +/– 1.225 g static, +/– 0.225 g dynamic |
| Resolution | 24 bit (40 ng) |
| Type of sampling (continuous or sporadic, variability) | Episodic/periodic mode: typically 3 second records every 6-12 seconds at 2 kHz, or 6 second records every 12-18 seconds at 1 kHz.<br>Continuous: 1 kHz – 2 kHz |
| Typical sampling rate | 1 kHz |
| Peak sampling rate | 2 kHz |
| Typical data transfer rate (per sensor) | 72 kbit/sec |
| Peak data transfer rate (per sensor) | 144 kbit/sec |
| Data format(s) (proprietary or public standard) | I/O Inc. has proprietary data formats; SDI has access to these formats and will be able to poll the sensors, manipulate the data, and store them in a format known to SDI and suitable for extraction. |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled (data logged in DATS)<br>Streaming (when operated continuously) |
| Brochure/description available | Yes |

# Staging of 4C Sensors

The number and configuration of 4C sensors that will be used in the Observatory will depend on the stage of the installation. Based on recent documents and discussions, the progression of sensor implementation is likely to be as follows.

***First stage***: This installation will consist of either:

- A 10 m gravity-driven seafloor probe fitted with at least one 4C sensor (from Hot Topics CMRET document, Dec. 31, 2004), or
- A 50 m array with 4C sensors, to be used for sled testing (from CMRET workshop schedule document, handed out at meeting Feb. 16-17, 2006).

***Second stage***: This installation will consist of a 150 m vertical borehole array

containing 13 hydrophones with 12.5 m spacing, and seven 3C sensors with 25 m spacing (from CMRET preprint of article to appear in May 2006 issue of The Leading Edge).

***Future stage (dependent on funding)***: Four horizontal arrays, each 400 m long and containing 16 hydrophones with 25 m spacing and 8 x 3C accelerometers with 50 m spacing. The configuration on the bottom is yet to be decided, but they could be deployed in a cross or a pinwheel pattern. At present, funding is only available for the hydrophone components of these arrays, but the 3C sensors could be added if resources become available.

## 4C Data Processing Methods

This section outlines some processing techniques that can be applied to the output of a single isolated 4C sensor, rather than attempting to describe array signal processing applied to the sensor data. Note that the planned availability of borehole 4C data provides an avenue for the application of vertical seismic profiling (VSP) techniques. A discussion of VSP is beyond the scope of the current report; see Hardage, and Balch and Lee for more details on VSP.

### Calibration / Vector Fidelity

***Purpose***: To determine position, orientation angles, and component sensitivities of each of the 3D accelerometers for each set of array sensors.

***Procedure***: The need for vector fidelity calibration depends on the array and the type of sensor configuration. For example, for the gimbaled sensors on an ocean bottom cable, it is not necessary to apply vertical alignment corrections. For other situations, it will usually be necessary to apply a rotation of the sensor data to give vertical and standard azimuthal components of the shear wave field; this can be done using rotation matrices once the 3D orientation of the sensor is known. Note that the VectorSeis accelerometer data include the direction cosines from vertical for each sensor, and these may be used to perform the vertical alignment. The azimuthal orientation of the horizontal components would be obtained from analysis of shot data or of shear waves generated by the bottom sled device during the calibration stage.

***Reference***: Byerley et al., Gratacos, Input/Output Systems.

### Integration of Accelerometer Data

***Purpose***: If it is desired to obtain velocity rather than acceleration data, the acceleration measurements must be integrated over time. The resulting quantity can then be used in concert with the pressure measurements generated by the hydrophones to achieve wavefield separation.

***Procedure***: This process involves applying a numerical integration algorithm to the data. While simple procedures such as the trapezoid or Simpson's rule can be applied, recently methods have been developed that use spectral information to provide more stability and accuracy. To counteract the effects of drift, a procedure for dynamic estimation of the mean and/or trend estimation and removal may also have to be applied. Since the characteristics of the accelerometers in this regard are not well understood at this stage, some experimentation will likely be required to develop and test appropriate methods for integration.

***References***: Wu and Mugler

## Wavefield Separation

***Purpose***: This technique estimates the upgoing and downgoing components of the PP wavefield using measured pressure and particle velocity data. The separated up-down wavefields may then be used to estimate the PP reflectivity through deconvolution. In addition, a separation technique may be used to isolate the horizontal PS wave (radial) component from the projection of the PP wave field onto this component. The result is an improved estimate of the PS shear wave, from which the contribution of the PP wave has largely been removed.

***Procedure***: The upgoing and downgoing components of the PP wave are separated by taking linear combinations of the pressure sensor and the vertical geophone (or integrated accelerometer) outputs. The PP reflectivity can then be estimated by deconvolution of these components. The effect of the PP wave on the PS wave can be reduced by applying a cross-equalization filter to the hydrophone output and subtracting the result from the radial geophone (or integrated accelerometer) output.

***References***: Backus et al.

## Wavefield Deconvolution

***Purpose***: This method yields an estimate of the reflectivity series for PP or PS waves.

***Procedure***: The downgoing and upgoing PP waves can be separated and the reflectivity can be estimated by deconvolving the downgoing component from the upgoing component, using Wiener deconvolution. A similar approach can be taken to estimate the reflectivity sequence for PS waves. An added benefit is that the results are aligned to the direct arrival time at the seafloor location, so that the seafloor becomes the relative datum.

***References***: Gaiser et al, Backus et al.

## Time Registration and Velocity Modeling

***Purpose***: The process of time registration allows the identification of the times in the PS traces and the PP traces that correspond to the same actual reflection events. This facilitates the interpretation of the data sets, particularly in the presence of gas clouds. Accurate time registration of PP and PS time-migrated images also results in velocity models and images of *Vs*/*Vp* velocity ratios that provide information about the geology/lithology of the sub-bottom.

***Procedure***: Times corresponding to the same event can sometimes be identified by inspection of 2D sections of hydrophone and geophone data, although this process can be both laborious and ambiguous. Simple 1D warping functions have been used to modify the bin spacing based on the time registrations. If 3D data are available, methods based on determining the corresponding times for features using horizontal plan views of the raw or processed data volumes (e.g., coherency volumes) have also been devised. However, since 3D data will not be produced by the monitoring station, such registration will likely be restricted to 1D traces and 2D sections.

***References***: DeAngelo et al., Fomel et al., Murray et al.

## References


1. Backus, M.M., Murray, P.E., Hardage, B.A., and Graebner, R.J., Enhanced PS-wave images of deep water, near-seafloor geology from 2D 4C OBC data in the Gulf of Mexico.
2. Balch, A.H., and Lee, M.W. (Eds.), 1984, *Vertical Seismic Profiling: Techniques, Applications, and Case Histories*, Boston (International Human Resources Development Corporation).
3. Barkved, O., Bartman, B., Compani, B., Gaiser, J., VanDok, R., Johns, T., Kristiansen, P., Probert, T., and Thompson, M., 2004, The many facets of multicomponent seismic data, Oilfield Review, Summer 2004, 42-56.
4. Blaylock, J.J., 1999, Interpretation of a baseline 4D multicomponent seismic survey at vacuum field, Lea County, New Mexico, M. Sc. Thesis, Colorado School of Mines, http://www.mines.edu/academic/geophysics/rcp/theses/old/Blaylock_1999/introjay.htm.
5. Byerley, G., Clausen, C., and Tessman, J., Ekofisk VectorSeis® test: Improvements in vector fidelity of 4C seismic data, http://www.rxt.com/docs/Bryerley_et_al.pdf.
6. Cary, P., 2001, Multicomponent seismic exploration in Canada – One person's perspective, CSEG Recorder, September 2001, 62-67.
7. Cheng, J., Twilley, K., Nurvosh, H., El-Khater, R., Ti, Y., Luke, B., Yfantis, A. and Harris, D. Passive seismic monitoring for rockfall at Yucca Mountain: Concept tests, UCRL-JC-152343, http://www.llnl.gov/tid/lof/documents/pdf/242654.pdf.
8. DeAngelo, M.V., Backus, M., Hardage, B.A., and Murray, P., 2003, Depth registration of P-wave and C-wave seismic data for shallow marine sediment characterization, Gulf of Mexico, The Leading Edge, February 2003, 96-105.
9. Dong, H., Maao, F.A., and Mjaaland, S. Study on vector fidelity of ocean bottom seismic data, http://www.10iwsa.dkrz.de/abstracts/Dongetal.pdf.

10. Flueh, E. R., Klaeschen, D., and Bialas, J., 2002, Options for multicomponent seismic data acquisition in deep water, First Break **20**, 12 December 2002, 764-769.
11. Fomel, S., Backus, M.M., DeAngelo, M.V., Murray, P.E., and Hardage, B.A., 2003, Multicomponent seismic data registration for subsurface characterization in the shallow Gulf of Mexico, Offshore Technology Conference (OTC 15117).
12. Gaiser, J. E., DiSiena, J. P., and Fix, J. E., 1984, Vertical seismic profiles: fundamentals of the downgoing wavefield and applications that improve CDP data interpretation, *in* Toksöz, M. N., and Stewart, R. R., Eds, Vertical Seismic Profiling Part B: Advanced Concepts: I. Seismic Exploration, **14B**, Geophysical Press, 87-112.
13. Gerstoft, P., Sabra., K., Roux, P., Kuperman, W.A. and Fehler, M.C., Green's function extraction and surface wave tomography from microseisms in Southern California, Geophysics (in press) 2006, http://www.mpl.ucsd.edu/people/gerstoft/papers/geophysics2005.pdf.
14. Gerstoft, P., Sabra., K., Roux, P., Kuperman, W.A. and Hodgkiss, W.S., 2005, Passive acoustic and seismic tomography with ocean ambient noise in ORION, Proc. Internat. Conf. "Underwater Acoustic Measurements: Technologies & Results", Heraklion, Crete, Greece, 28 June – 1 July 2005.
15. Glangeaud, F., Mari, J.L. and Meunier, J. 2001, 4C-sensor orientation using multicomponent-SVD method applied on shallow water data, EAGE 63rd Conference and Technical Exhibition, Amsterdam, The Netherlands, 11-15 June.
16. Gratacos, B, 2003, Reorientation and calibration of non-gimbaled multicomponent sensors, 73rd SEG International Exposition and Annual Meeting, Dallas, Texas, 26-31 October 2003, 71-74.
17. Harbin, P.E., Harris, D.B., Myers, S.C., Larsen, S.C., Wagoner, J.L, Trebes, J.E. and Nelson, K.E., Developing smart seismic arrays, Centre for Complex Distributed Systems, LDRD/ETR/FY03, http://engineering.llnl.gov/eng_llnl/01_html/pdfs/FY03_LDRD_p34.pdf.
18. Hardage, B. A., 1985, *Vertical seismic profiling-principles (2nd ed.):* Amsterdam, Geophysical Press, 509 pp.
19. Hardage, B. A., 2001, Developments, trends and future directions in vertical seismic profiling and crosswell seismic profiling, CSEG Recorder, September 2001, 72-78.
20. Harris, D.B., Larsen, S.C., McCallen, D.B., Rock, D.W., Lewis, J.P., McMahon, D.H., and Levatin, J.L., Developing smart seismic arrays, http://engineering.llnl.gov/eng_llnl/01_html/pdfs/ETR_01_PDFs/Vol1_LDRD.pdf.
21. Harrison, C.H., 2004, Sub-bottom profiling using ocean ambient noise, J. Acoust. Soc. Am. **115**, 1505-1515.
22. Hendrick, N. and Hearn, S., 2003, Introduction to vector-processing techniques for multicomponent seismic exploration, ASEG 16th Geophysical Conference and Exhibition, February 2003, Adelaide.
23. Input/Output Systems, VectorSeis vector fidelity and vertical orientation, http://www.i-o.com/content/includes/Vector_fidelity.pdf.
24. Jarvis, K.D. and Knight, R., 2002, Aquifer heterogeneity from SH-wave seismic impedance inversion, Geophysics 67, 1538-1557.
25. Knapp, S., Payne, N., and Johns, T., 2002, Imaging through gas clouds: A case history from the Gulf of Mexico, CSEG Recorder, April 2002, 5-12.

26. Le Bihan, N., Mars, J.I., and Pedersen, H., 2000, Wavefield separation on multicomponent sensors, 62nd meeting of European Association of Geoscientists and Engineers, Glasgow, http://www.lis.inpg.fr/pages_perso/Mars/eage2000NL.pdf.
27. Li, X.-Y., Accuracy and sensitivity analysis for estimating anisotropic parameters from 4D seismic data.
28. Li, X.-Y., 2002, Anisotropic processing of 4C seismic data, 4C/OBS Data Processing and Interpretation – Old Myths, New Insights", 22nd April, 2002 at NPD in Stavanger, http://www.force.org/PDW-Seminars/SeismicMethods/4C/abstract-presentations/Xiang_Yang_Edin.htm.
29. Lobkis, O.I. and Weaver, R.L, 2001, On the emergence of the Green's function in the correlations of a diffuse field, J. Acoust. Soc. Am. **110**, 3011-3017.
30. Mancini, F., Li, X.-Y., Ziolkowski, A., and Pointer, T., Interpreting velocity ratios from 4C seismic data and well logs in the presence of gas and anisotropy.
31. Maxwell, P., Tessman, J. and Reichert, B., 2001, Design through to production of a MEMS digital accelerometer for seismic acquisition, First Break **19**, 141-143.
32. Murray, P.E., DeAngelo, M.V., Hardage, B.A., Backus, M.M., Graebner, R.J., and Fomel, S., 2003, Interpreting multicomponent seismic data in the Gulf of Mexico for shallow sedimentary properties: methodology and case history, Offshore Technology Conference (OTC 15118).
33. Ritzwoller, M. and Levshin, A.L., 2002, Estimating shallow shear velocities with marine multicomponent seismic data, Geophysics **67**, 1991-2004.
34. Schalkwijk, K.M., Wapenaar, C.P.A. and Verschuur, D.J., Decomposition of multicomponent ocean-bottom data: inversion for the sub-bottom parameters, http://www.delphi.tudelft.nl/DELPHI_AP/Abstracts/seg00_schalkwijk.pdf.
35. Stewart, R.R., Gaiser, J.E., Brown, R.J., and Lawton, D.C., 1999, Converted-wave seismic exploration: A tutorial, CREWES Research Report **11**, 21-62.
36. Tatham, R.H. and McCormack, M.D., E.B. Neitzel, and D.F. Winterstein, 1991, Multicomponent seismology in petroleum exploration, Society of Exploration Geophysicists, Tulsa, 248pp.
37. Thomsen, L., The issue of vector fidelity in multicomponent data, http://www.seg.org/meetings/past/srwboise2000/pdf/105.pdf.
38. Wu, Y. and Mugler, D.H., 2004, A robust DSP integrator for accelerometer signals, IEEE Trans. Biomed. Eng., **51**, 385-389. http://ieeexplore.ieee.org/iel5/10/28215/01262117.pdf?arnumber=1262117.

# Appendix 1: Array Configurations and Numbers of Sensors

### Acoustic Vertical Line Array – VLA

| Sensor | No. of sensors |
|---|---|
| Hydrophone | 16 |
| Orientation Sensor | 2 – 16 |
| Thermistor | 8 |

### Horizontal Line Array – HLA (4 Arrays)

| Sensor | No. of sensors per array |
|---|---|
| Hydrophone | 16 |
| 3D Accelerometer | 8 |

### Borehole Vertical Line Array – BLA

| Sensor | No. of sensors |
|---|---|
| Hydrophone | 16 |
| 3D Accelerometer | 8 |
| Thermistor | 16 |

### Benthic Boundary Layer Array – BBLA

| Sensor | No. of sensors |
|---|---|
| ADCP | 1-2 |
| CTD | 1-2 |
| Chlorophyll fluorometer | 1-2 |
| CDOM fluorometer | 1-2 |
| PMT fluorometer | 1-2 |
| Methane sensor (METS) | 1-2 |
| Oxygen sensor | 1-2 |

### Chimney Array – CA

| Sensor | No. of sensors |
|---|---|
| Methane sensor (METS) | 1-2 |
| Conductivity, depth, oxygen sensor | 1 |
| Monochrome camera | 2 |

| | |
|---|---|
| Color camera | 1 |

# Appendix 2: Characterization of "New" Sensors

In BCS's report of January 31, 2005, entitled *Sensor and Data Characterization for the Gulf of Mexico Hydrates Monitoring Station – Version 1.2*, a number of sensors and instruments were identified that were targeted to be included as components of the monitoring station. Since that time, several sensors that were not originally identified have been proposed for inclusion. In this appendix, these "new" sensors are characterized. In addition, some of the previously identified sensors are discussed and/or characterized in greater detail.

## RDI Workhorse Current Meter (ADCP)

| Sensor | Information |
|---|---|
| Consortium contact(s) | Vernon Asper / Paul Higley |
| Product name/number | Deep Water Workhorse 300 kHz |
| Manufacturer | RD Instruments |
| Physical quantity measured | a) Velocity<br>b) Direction<br>c) Tilt<br>d) Temperature |
| Dimensionality of measurements | a) Vector array (units cm/s)<br>b) Vector array (units degrees magnetic)<br>c) Scalar (units degrees)<br>d) Scalar (units degrees Celsius) |
| Range | a) 0 to 500<br>b) 0 to 360<br>c) 0 to 15<br>d) –5 to 45 |
| Resolution | a) 0.1<br>b) 0.01<br>c) 0.01<br>d) 0.01 |
| Type of sampling (continuous or sporadic, variability) | Periodic (specified sample duration every hour)<br>Continuous |
| Typical sampling rate | 1 Hz sampling averaged to mean values for a specified sampling duration (5-60 minutes per hour) |
| Peak sampling rate | 2 Hz |
| Typical data transfer rate (per sensor) | 1 record (a few kilobytes) per sampling event |
| Peak data transfer rate (per sensor) | 50,720 bps |
| Data format(s) (proprietary or public standard) | Variable; formats available in product manuals |
| Required metadata and how | The RDI family of ADCPs has an extensive |

| | |
|---|---|
| they are created | command set for instrument configuration and operation. For metadata, operational modes, dynamic configuration and data formats, please refer to the document “Workhorse Commands and Output Data Format”. |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled<br>Streaming<br>Internally logging |
| Calibration requirements | Calibrations are not typically required for the transducers, although the compass may require swinging. The sensing elements are not particularly susceptible to biofouling. |
| Brochure/description available | Yes |
| Notes | h) Data is typically stored on the instrument in a single binary file. The data acquisition frequency, averaging intervals (if any) and other sampling characteristics are determined in advance by the investigating (scientific) authority and programmed into the instrument’s configuration. Available storage in the instrument (typically on the order of 40 MB) usually constrains the sampling parameters for a given deployment.<br>i) Data formats for post-processing are typically determined by the user after recovery of the information. After an instrument’s data file has been downloaded to a PC, software provided by RDI processes the raw binary data into the data format (typically one of many ASCII formats) as instructed by the user at this point.<br>j) Configuration files are stored in the instrument, but it is important to maintain pre- and post-deployment configuration information for later stages of data processing.<br>k) Compass calibration data may be required to correct directional current information. This calibration (“compass swinging”) is usually done on-site to reflect the local magnetic environment.<br>l) Timestamps are stored along with the data in the instrument. The instrument |

| | |
|---|---|
| | maintains an on-board real-time clock, but it is useful to record the instrument's clock values pre- and post-deployment in order to correct for clock drift.<br>m) Instrument configuration (model, frequency, serial number, and sampling parameters) can be polled interactively and dumped to a configuration file. Note that there is a fairly large set of configurable options that can be set and should be recorded. This metadata should be considered an integral and essential component of the instrument data. A sample of this metadata is provided at the end of this Appendix.<br>n) The standard post-processing methods available via the provided RDI software can be replaced with or augmented by an investigator's own processing methods. Examples of non-standard post-processing include compensation for mooring pull-down in strong currents; compensation for real-time clock drift; correction for beam bias (intensity); correction for temperature variation in the upper water column; and filtering for data quality (e.g., examining error velocity and vertical velocity components). |

## Seabird SBE-37 SI CTD Sensor (and optionally SBE-5 pump)

The SBE 37-SI (MicroCAT) is a high-accuracy conductivity and temperature sensor (pressure optional) without internal batteries or memory. It is designed for moorings and other long-term fixed-site deployments at depths up to 7,000 m (23,000 ft), and is easily integrated with current meters, moored instruments, remotely operated vehicles, AUVs, or Mini-subs. It includes a standard serial interface. Titanium and other non-corroding materials are used to minimize maintenance and ensure long-life.

The SBE 37-SIP combines most of the features of the 37-SI with an integral, internal pump. The pump provides improved conductivity response and anti-foul protection.

The SBE 37-SI series of CTDs does not have its own internal memory for data logging.

The MicroCAT is supplied with SEATERM, a powerful Win 95/98/NT/2000/XP terminal program for easy communication and data retrieval. SEATERM can send commands to the MicroCAT to provide status display, data acquisition setup, data display and capture, and diagnostic tests.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Rich Camilli / Paul Higley |
| Product name/number | SBE-37 SI |
| Manufacturer | Seabird Electronics Inc. |
| Physical quantity measured | a) Temperature<br>b) Conductivity<br>c) Pressure |
| Dimensionality of measurements | a) Scalar (units deg C)<br>b) Scalar (units S/m)<br>c) Scalar (units dbar) |
| Range | a) –5 to +35<br>b) 0 to 7<br>c) 0 to 7,000 |
| Resolution | a) 0.0001 (accuracy 0.002)<br>b) 0.00001 (accuracy 0.0003)<br>c) 0.002% (accuracy 0.1%) |
| Type of sampling (continuous or sporadic, variability) | **Autonomous Sampling** – At pre-programmed intervals, the MicroCAT samples. There are two types of autonomous sampling:<br>• *Continuous sampling* at the fastest rate possible (0.66 second minimum), or<br>• *Interval sampling* at intervals of 10 seconds to 9.1 hours. Jumper positioning determines whether the MicroCAT goes to sleep between samples.<br>**Polled Sampling** – On command from a computer or satellite, radio, or wire telemetry equipment, the MicroCAT takes a sample and transmits the data.<br>**Serial Line Sync** – In response to a pulse on the serial line, the MicroCAT wakes up, samples, transmits the data, and goes to sleep. |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 1 Hz |
| Typical data transfer rate (per sensor) | 768 bps (RS-232 9600,N,8,1) |
| Peak data transfer rate (per sensor) | Approx. 768 bps |
| Data format(s) (proprietary or public standard) | Variable; formats available in product. The SBE 37SI can be configured to output ASCII data in hexadecimal format or in calibrated engineering units.<br>**FORMAT=1** (default)<br>ttt.tttt,cc.ccccc, pppp.ppp, dddd.ddd, sss.ssss, vvvv.vvv, |

| | |
|---|---|
| | rrr.rrrr, dd mmm yyyy, hh:mm:ss<br><br>Leading zeros are suppressed, except for one zero to the left of the decimal point. |
| Required metadata and how they are created | A comprehensive list of instrument-specific metadata is provided in the instrument manual. |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled<br>Streaming |
| Calibration requirements | To resolve fine structure and detect very small changes in property measurements, the sensors must be calibrated periodically. In particular, the conductivity cell is susceptible to biofouling and needs both cleaning and calibration at an interval determined by the rate of algal growth on the sensing elements. |
| Brochure/description available | Yes |
| Other relevant information | Rich Camilli noted that "the CTD will require its own RS232/485 line, I think we should plan on using a SeaBird. I am working on some circuitry to multiplex all of the fluorometers and METS type sensors together. The data from each sensor will be output to a RS232/485, where a single polling command string will yield a data string that will need to be parsed into the appropriate sensor channels. Each of the sensor data channels will be a 12 bit voltage value ranging from 0 to 5 volts (appearing on the serial line as a low and high byte). The conversion from raw voltages to concentrations can be done on the topside. For each sensor node on the mooring we should expect to parse and store at least 14 channels (this could be expanded later to 25 channels with an additional serial line) at up to 1Hz." |

## Seabird SBE-5 Miniature submersible pump

**SBE 5M**

The SBE 5M pump module consists of a centrifugal pump head and a long-life, DC, ball-bearing motor contained in a compact titanium pressure housing useable to 10,500 m (34,400 ft) deep. The pump impeller and electric drive motor are coupled magnetically through the housing, providing high reliability by eliminating moving seals. Motor drive electronics is intrinsically protected against accidental reversed polarity.

The SBE 5M is standard on the 19plus SEACAT Profiler CTD. It is optional on the SBE 16plus and 16plus-IM SEACAT C-T Recorder, and in custom applications. The pump flushes water through the conductivity cell at a constant rate, independent of the CTD's motion, improving dynamic performance. For applications requiring pumping through additional sensors (for example, a dissolved oxygen sensor), the SNE 5T pump should be used instead.

**SBE 5T**

The SBE 5T pump module is a compact unit consisting of a centrifugal pump head and a long-life, brushless, DC, ball-bearing motor contained in a titanium pressure housing useable to 10,500 m (34,400 ft). The pump impeller and electric drive motor are coupled magnetically through the housing, providing high reliability by eliminating moving seals.

The SBE 5T is a primary component in the SBE 9plus CTD Underwater Unit and SBE 25 SEALOGGER CTD. It is also used as optional equipment on the SBE 16plus, 16plus-IM, and 19plus SEACAT CTDs. The pump flushes water through the conductivity cell at a constant rate, independent of the CTD's motion, improving dynamic performance. The pump may also be suitable for custom applications, where pressure heads are less than 300 cm of water and flow rates are less than 100 ml/s.

## Ocean Marine METS Methane Sensor

The CAPSUMS`s METS is a unique underwater methane sensor solving the problems of the in-situ measurements of $CH_4$. Applications range from the treatment of sewage waters and landfill leachates to water quality monitoring, from biogas production to climate change studies and also includes many aspects of the offshore oil, gas and hydrate exploration activities (e.g., installation safety, environmental protection, exploration surveys).

| Sensor | Information |
|---|---|
| Consortium contact(s) | Rich Camilli |
| Product name/number | METS |
| Manufacturer | Capsum Technologie GmbH |
| Physical quantity measured | Methane concentration. The hydrocarbon molecules diffuse out of the liquid through a special silicone membrane into the detector room. The adsorption of hydrocarbon on the active layer leads to electron exchange with oxygen and thus to modification of the resistance, which the electronic transduces into a voltage. |
| Dimensionality of measurements | nmol/l |
| Range | 50 nmol/l to 10 µmol/l |
| Resolution | Up to 24 bit (SDI A/D converter) 0-5V |
| Type of sampling (continuous or sporadic, variability) | Periodic (Polled)<br>Continuous (Polled) |

| | |
|---|---|
| Typical sampling rate | 1 second (sensor has 1-3 sec response time and a t90 time of up to 3 minutes) |
| Peak sampling rate | 1 Hz |
| Typical data transfer rate (per sensor) | 24 bps |
| Peak data transfer rate (per sensor) | 24 bps |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled |
| Brochure/description available | Yes |
| Other relevant information | Transmission protocol: 9600 baud, 8 data bits, 1 stop bit, no parity<br><br>Rich Camilli noted that “the methane and temperature voltages are used to calculate the actual methane concentration using a unique algorithm specific to each sensor. That algorithm is supplied in the documentation that comes with each instrument.”<br><br>Rich Camilli noted with respect to CTDs (assumed that this may be appropriate for METS): “ The data from each sensor will be output to a RS232/485, where a single polling command string will yield a data string that will need to be parsed into the appropriate sensor channels. Each of the sensor data channels will be a 12 bit voltage value ranging from 0 to 5 volts (appearing on the serial line as a low and high byte). The conversion from raw voltages to concentrations can be done on the topside. For each sensor node on the mooring we should expect to parse and store at least 14 channels (this could be expanded later to 25 channels with an additional serial line) at up to 1Hz.” |

## Chlorophyll Fluorometer – Turner Designs SCUFA

The SCUFA (Self-Contained Underwater Fluorescence Apparatus) is an accurate, simple-to-use and versatile submersible fluorometer for chlorophyll and dye tracing applications. The SCUFA has been designed to operate in a wide range of concentrations and environmental conditions. The capability to

program sampling intervals via the Windows Interface Software and the automatic range control enable the user to configure the SCUFA for any type of profiling or moored deployment.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Rich Camilli |
| Product name/number | SCUFA (Self Contained Underwater Fluorescence Apparatus) |
| Manufacturer | Turner Designs |
| Physical quantity measured | a) Fluorescence<br>b) Turbidity |
| Dimensionality of measurements | a) µg/l (chlorophyll) or cells/ml (cyanobacteria)<br>b) NTU |
| Range | a) 0.02 µg/l detection limit; 4 orders of magnitude dynamic range<br>b) 150 cells/ml detection limit; 4 orders of magnitude dynamic range |
| Resolution | 12 bit 0-5V signal (1.22 mV) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 5 Hz |
| Typical data transfer rate (per sensor) | 528 bps (528 bit data string) |
| Peak data transfer rate (per sensor) | 2640 bps |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled |
| Brochure/description available | Yes |

## Chlorophyll Fluorometer – Wetlabs ECO FL Series (Open Path)

Chlorophyll-*a* fluorescence serves as a valuable indicator of active phytoplankton biomass and chlorophyll concentrations in waters. This measurement is used for tracking biological variability and abundance in the water column.

The *Environmental Characterization Optics*, or *ECO* miniature fluorometer allows the user to measure relative chlorophyll, CDOM, uranine, phycocyanin, or phycoerythrin concentrations by directly measuring the amount of fluorescence emission in a sample volume of water. The *ECO* uses an LED to provide the excitation source. An interference filter is used to reject the small amount of out-of-band light emitted by the LED. The light from the source enters the water

volume at an angle of approximately 55–60 degrees with respect to the end face of the unit. Fluoresced light is received by a detector positioned where the acceptance angle forms a 140-degree intersection with the source beam. An interference filter is used to discriminate against the scattered excitation light.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Rich Camilli |
| Product name/number | ECO FL series |
| Manufacturer | Wetlabs |
| Physical quantity measured | Fluorescence (470/695 nm) |
| Dimensionality of measurements | µg/l (chlorophyll) |
| Range | 0.01 µg/l detection limit up to 125 µg/l |
| Resolution | 14 bit 0-5V signal (0.3 mV or 0.01 µg/l) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 8 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |
| Peak data transfer rate (per sensor) | 2560 bps (est.) (19,200 baud communication rate) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## CDOM Fluorometer – Wetlabs ECO FL Series (Open Path)

The CDOM ECO allows measurement of CDOM fluorescence across a wide range of environments, from mangrove swamps to oligotrophic blue water.

The *Environmental Characterization Optics*, or *ECO* miniature fluorometer allows the user to measure relative chlorophyll, CDOM, uranine, phycocyanin, or phycoerythrin concentrations by directly measuring the amount of fluorescence emission in a sample volume of water. The *ECO* uses an LED to provide the excitation source. An interference filter is used to reject the small amount of out-of-band light emitted by the LED. The light from the source enters the water volume at an angle of approximately 55–60 degrees with respect to the end face of the unit. Fluoresced light is received by a detector positioned where the acceptance angle forms a 140-

degree intersection with the source beam. An interference filter is used to discriminate against the scattered excitation light.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Rich Camilli |
| Product name/number | ECO FL series |
| Manufacturer | Wetlabs |
| Physical quantity measured | Fluorescence (370/460 nm) |
| Dimensionality of measurements | ppb (Concentration) |
| Range | 0.25 ppb to 300 ppb |
| Resolution | 14 bit 0-5V signal (0.3 mV or 0.25 ppb) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 8 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |
| Peak data transfer rate (per sensor) | 2560 bps (est.) (19200 baud communication rate) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## Chlorophyll Fluorometer – Wetlabs WETStar Series (Flow-through)

The WETStar fluorometer provides a high level of performance for detecting various types of fluorescence. It uses a central axial optical tube to provide efficient signal output.

The meter has a unique flow tube design that lends itself to both pump-through and flow-through operation. Its power input of 7–15 VDC and 0–5 VDC analog output allow it to easily mate with existing CTD packages.

The WETStar series is compatible with external pumps, such as the Seabird SBE-5.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Rich Camilli |

| | |
|---|---|
| Product name/number | WETStar |
| Manufacturer | Wetlabs |
| Physical quantity measured | Fluorescence (460/695 nm) |
| Dimensionality of measurements | µg/l (chlorophyll) |
| Range | 0.03 µg/l detection limit up to 75 µg/l |
| Resolution | 12 bit 0-5V signal (1.22 mV or 0.03 µg/l) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 8 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |
| Peak data transfer rate (per sensor) | 2560 bps (est.) (9600,N,8,1 communications) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## CDOM Fluorometer – Wetlabs WETStar Series (Flow-through)

The WETStar fluorometer provides a high level of performance for detecting various types of fluorescence. It uses a central axial optical tube to provide efficient signal output.

The meter has a unique flow tube design that lends itself to both pump-through and flow-through operation. Its power input of 7–15 VDC and 0–5 VDC analog output allow it to easily mate with existing CTD packages.

The WETStar series is compatible with external pumps, such as the Seabird SBE-5.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Rich Camilli |
| Product name/number | WETStar series |
| Manufacturer | Wetlabs |
| Physical quantity measured | Fluorescence (370/460 nm) |
| Dimensionality of measurements | ppb (Concentration) |
| Range | 0.100 ppb to 100, 250 or 1000 ppb |
| Resolution | 14 bit 0-5V signal (0.3 mV or 0.1 ppb) |
| Type of sampling (continuous or sporadic, | Periodic (polled)<br>Continuous (polled) |

| | |
|---|---|
| variability) | |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 8 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |
| Peak data transfer rate (per sensor) | 2560 bps (est.) (9600,N,8,1 communications) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## Seapoint Sensors UV CDOM Fluorometer

**A new sensor for measuring chromophoric dissolved organic matter (CDOM), the Seapoint Ultraviolet Fluorometer, is now available. This compact sensor features high performance, low power requirements, selectable ranges, and 6,000 meter depth rating. Applications include ocean color research, wastewater discharge quality monitoring, and DOM cycle studies.**

The Seapoint Ultraviolet Fluorometer (SUVF) is a high-performance, low power instrument for *in situ* measurement of chromophoric dissolved organic matter (CDOM). Its small size, very low power consumption, high sensitivity, wide dynamic range, 6,000 meter depth capability, and open or pump-through sample volume options provide the power and flexibility to measure CDOM in a wide variety of conditions. The SUVF uses modulated ultraviolet LED lamps and excitation filter to excite CDOM. The fluorescent light emitted by the CDOM passes through a blue emission filter and is detected by a silicon photodiode. The low level signal is then processed using synchronous demodulation circuitry which generates an output voltage proportional to CDOM concentration. The SUVF may be operated with or without a pump. The sensing volume may be left open to the surrounding water, or, with the use of the supplied cap, can have water pumped through it. Two control lines allow the user to set the range to one of four options. These lines may be hardwired or microprocessor-controlled to provide a suitable range and resolution for a given application. The sensor is easily interfaced with data acquisition packages; a 5 ft pigtail is supplied. Custom configurations are available.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Rich Camilli |
| Product name/number | Seapoint Ultraviolet Fluorometer |
| Manufacturer | Seapoint Sensors |
| Physical quantity measured | Fluorescence (370/440 nm) |
| Dimensionality of | µg/l |

| | |
|---|---|
| measurements | |
| Range | 0.1 µg/l to 50, 150, 500 or 1500 µg/l |
| Resolution | 14 bit 0-5V signal (0.3 mV or 0.1 µg/l) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 1 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |
| Peak data transfer rate (per sensor) | 2560 bps (est.) (9600,N,8,1 communications) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## Chlorophyll Fluorometer – Chelsea Technologies Group UV AQUA-Tracka

The UV AQUA-Tracka is a submersible fluorometer to monitor the concentration of hydrocarbons (360nm) or Gelbstoff (440nm) in a wide range of oceanographic applications. In support of this, it has been designed to be deployed from undulating vehicles, moored or profiling systems. This robust, compact, lightweight instrument has built-in test (BITE) circuitry which ensures high stability. The instrument is easy to use and gives accurate and repeatable measurements.

| Sensor | Information |
|---|---|
| Consortium contact(s) | Rich Camilli |
| Product name/number | UV AQUA-Tracka |
| Manufacturer | Chelsea Technologies Group |
| Physical quantity measured | Hydrocarbon Fluorescence (239/360 nm) |
| Dimensionality of measurements | µg/l (Carbazole) |
| Range | 0.001 µg/l detection limit up to 10 µg/l |
| Resolution | 12 bit 0-4V or 0-8V signal (0.001 µg/l) |
| Type of sampling (continuous or sporadic, variability) | Periodic (polled)<br>Continuous (polled) |
| Typical sampling rate | 1 Hz |
| Peak sampling rate | 8 Hz |
| Typical data transfer rate (per sensor) | 320 bps (est.) |

| | |
|---|---|
| Peak data transfer rate (per sensor) | 2560 bps (est.) (9600,N,8,1 communications) |
| Data format(s) (proprietary or public standard) | Digitized voltage converted to engineering units using a polynomial calibration equation |
| Operational modes (polled, streaming, internal logging, duty cycles) | Polled, streaming, internal logging |
| Brochure/description available | Yes |

## PixeLINK PL-A741 Monochrome Camera

The PL-A741 is a popular machine vision camera. It is a monochrome 1.3 megapixel digital FireWire camera with global shutter, external trigger and a rich set of on-board features that provides high-quality digital video streams to the computer.

**Color/Mono** : Mono
**Resolution** : 1280 x 1024
**Frame Rate** : 8000 max, 105 @ 640 x 480, 27 @ 1280 x 1027
**Bit Depth** : 8 or 10
**Pixel Pitch** : 6.7 µm
**Lens Mount** : C 2/3
**Interface** : FireWire
**Trigger Type** : H/W or S/W
**Shutter Type** : Global
**Gen Purpose Outputs** : 2 Multi-function

## PiexLINK PL-A742 Color Camera

The PL-A742 is a color 1.3 megapixel digital FireWire camera with global shutter, external trigger and a rich set of on-board features that provides high-quality digital video streams to the computer.

**Color/Mono** : Color
**Resolution** : 1280 x 1024
**Frame Rate** : 8000 max, 105 @ 640 x 480, 27 @ 1280 x 1027
**Bit Depth** : 8 or 10
**Pixel Pitch** : 6.7 µm
**Lens Mount** : C 2/3
**Interface** : FireWire
**Trigger Type** : H/W or S/W
**Shutter Type** : Global
**Gen Purpose Outputs** : 2 Multi-function

## Sample RDI ADCP Configuration File

(commented by user, instrument set up prior to deployment)

```
;WorkHorse-300 s/n 1234
;1-year deployment, starting in mid September 200x
;  operating at 100 m depth in 200 m of water.
;  Memory:  40 Mb
;  Battery: 1x400 watt-hours at 0 Celsius

;Reset WH300 to factory settings
CR1
;Flow Control - automatic ensemble, ping, com/ascii if enabled, com disabled, rcdr enabled
CF11001
;Power level
;CQ Power-level not selectable on WH300

;Heading alignment
EA00000
;Heading bias
EB00000
;Instrument depth specified as 1100 dm
;Target is 110 m, but could be as shallow as 100 m or as deep as 120 m
;Mooring pull-down estimated as 2 m
;WH-0272 has no pressure sensor for depth determination
ED01100
;Salinity specified as 32
ES32
;Coordinate transform, Earth coordinates, use pitch/roll, allow 3-beam, allow bin mapping
EX11111
;Data source: calculate SV, depth, hdg, pitch, roll, specify salinity, measure T
EZ1011101

;ENSEMBLE INTERVAL FOR DEPLOYMENT IS 30 minutes
TE00:30:00.00
;ENSEMBLE INTERVAL CAN BE REDUCED BY 10X FOR BENCH TEST
;TE00:03:00.00

;PING INTERVAL FOR DEPLOYMENT IS 1 second
;Allow sufficient time for both water-column (WPnn) and bottom-track (BPnn) pings
;  30 pings for WC plus 0 pings for BT, at 59 sec ---> 29.5 minutes
TP00:59.00
;PING INTERVAL CAN BE REDUCED BY 10X FOR BENCH TEST
;TP00:05.90

;SELECT START TIME BEFORE DEPLOYMENT (yy/mm/dd)
TF02/09/15,17:00:00
```

```
;Use reduced bandwidth to extend range
WB1
;Record velocity, correlation, intensity, percent good (default)
WD111100000
;Lockout is 176 cm (default for 300 kHz)
WF0176
;Cell size is 800 cm
WS0800
;Number of depth cells is 17
WN017
;Pings per ensemble is 30
WP00030
;Ambiguity velocity change to 050 cm/s radial, 150 cm/s horizontal (default is 170 cm/s radial)
WV050

;Fish rejection threshold set to 30 dB
WA064
;Error velocity rejection threshold set to 200 mm/s as 3 SD's
WE0200
;Correlation rejection threshold at default value (64)
WC64


;Delay 1 ensemble before trying to re-acquire the bottom
;BD command is not available for WH bottom-tracking (BD001, although documented in the
    manual
;No BT <<< Error velocity rejection threshold set to 200 mm/s as 3 SD's
;BE0200
;No BT <<< Set minimum expected range to bottom (ice) at 500 dm
;BF00500  Need to think about this one (conflicting advice)
;No BT <<< Bottom-tracking pings per ensemble is 10 (if BP000000, no bottom tracking)
;BP010  Need to think about this one (needs a lot of power)
;No BT <<< Bottom track mode 4
;BM4
;No BT <<< Limit bottom tracking to 2000 decimeters
;BX2000
;No BT <<<Limit BT pulses to 30% of water depth (default: use smaller values only in very
    shallow water)
;&R30

;Keep parameters as user defaults
CK
;Start pinging now or after delay
CS

;Deployment hours = 17520.00
```

;Temperature = -1.50
;Frequency = 307200

# Appendix 3: Phone Interviews with Sensor Experts

## Interview with Paul Higley (Specialty Devices Inc.)

Location: Barrodale Computing Services Ltd. (BCS), Victoria, BC, Canada
Time: 12:00-13:30, Thursday, November 10, 2005
Present: Cedric Zala, Mike Dunham-Wilkie, Kent Berger-North (BCS)
On Phone: Paul Higley, Larry Higley (Specialty Devices Inc.)

31. Introductions were made; Larry is a software developer/integrator at SDI.
32. Paul indicated that SDI was responsible for the seismic arrays and an oceanographic array data collection platform less sensors. There are others responsible for the benthic boundary array and geochemical experiments.
33. The seismic arrays and oceanographic arrays will operate independently, probably until 2007-2008, and will not be networked and integrated initially.
34. At one point in time there had been a specification for the seafloor Observatory which indicated that sensor packages intended to interface to the Observatory should have data loggers. SDI issued an interface specification that defined serial communications, Ethernet, and available power.
35. Data access will be direct access to the logger hard drives via Ethernet.
36. Specific formats and headers will be used for SDI's data; some are readily available (e.g., the P16 formats for the hydrophones) while others will have to wait until the system is integrated.
37. With respect to data volume, the seismic arrays will account for the majority of the data, followed by camera imagery. The remaining data (primarily oceanographic) will be on the order of kilobytes per week or month.
38. Typical data events may consist of a four-day near-continuous operation of the hydrophone arrays (6 second bursts at 12 second intervals) generating approximately 20GB.
39. Due to high data volumes, the intention is to recover the data from the data loggers on the seafloor to the buoy and physically recover (swap) the hard drives in the surface buoy.
40. The duration and frequency of data sets are expected to be 3-4 days every couple of months (seasonally).
41. Norm Farr at WHOI is responsible for the oceanographic array. Access to the seafloor nodes will initially be via submersible (SSD/ROV).
42. Integration of the seismic and oceanographic arrays will not take place until 2007-2008.
43. The geometry of the oceanographic array will be a single mooring with multiple connected pods that are placed in position by manned vehicle, SSD/ROV.
44. The SDI formats for the accelerometers and orientation sensors are TBD.
45. Examples of the orientation output may be available from a data set collected in Italy (2004). **Paul** will send Cedric an example of this data**.**
46. The thermistor, accelerometer, orientation sensor and hydrophone components of the seismic array will most likely be stored in separate files due to differing sampling regimes (acquisition intervals, sample frequency, and sample duration).

47. All seismic data will be referenced to a shot number (from the acoustic sources), a.k.a. “pop”.
48. Every pop will be stored in a separate file.
49. A log file will maintain the shot times. As timing is critical for the seismic data processing, all time information is relative to the shot “T-zero” and requires <<millisecond or better resolution. T-zero references are generated on the ship and sent down to the instrumentation package. While real-time clock (RTC) information will be recorded, it is not used for reference purposes. Real-time clocks in the logging computers, instruments, and sensors are subject to drift.
50. The temperature data is stored with RTC information.
51. There are a number of important metadata files that will be required for analysis of the acoustic/seismic data sets:
    a. The “Ship Log” data file. This contains navigation data for the ship (GPS positions and time) and gun array, and for the buoy over the gun array. It contains seismic shot file metadata, ship motion data, etc. USBL data are maintained by the vessel. (To be confirmed). Scott Sharpe will have the ship log formats. **Paul** will send Cedric a sample file.
    b. The “Ship Pop” data file. This records the shot number, the GPS position, the time, the delay from the gun, etc. This is maintained by the ship-board acquisition and control system. Larry will have the ship “pop” formats. **Paul** will send Cedric a sample file.
    c. The “Deep Tow” file. This contains data related to the hydrophone far field from guns close to the seafloor. The seismic data file is stored on the ship in real time. The format has yet to be determined, though it will likely be SEG-Y if 24-bit or P16 if 16-bit.
    d. The Sled system. This contains the seismic shear wave source, two orthogonal sources on the seafloor. It will contain 3-axis accelerometer data, orientation (of sled) data, hydrophones, USBL data and shot number. It is placed on the seafloor, triggered, and pulled 1 m at a time across the seafloor. It is also a good way to test the data requirements for the downhole array as the configuration will be similar.
    e. The AUV data. The AUV will likely be on a streamer. A hydrophone is located on the nose and the vehicle is placed close to the seafloor. Deployment is likely in 2006-2007. Data will probably be internally stored, single channel hydrophone. Position and time information is not necessary as the wave will already be well developed; the hydrophone will pick up the signal and subsequent reflection.
52. Graphics and descriptions of these systems may be on the Ole Miss web site:
    http://www.olemiss.edu/depts/mmri/programs/gulf_res.html
    http://www.olemiss.edu/depts/mmri/programs/station_schem_02.jpg
53. The seismic array data sampling regime will likely be as follows:
    a. Hydrophones: 16 channels for 1 to 4 seconds (assume 4) at 10 kHz of 16-bit data continuously for 48-96 hours.
    b. Accelerometers: 8 channels x 3 sensors (3 axes) for 1-4 seconds of 24-bit data at 1 kHz continuously for 48-96 hours.

c. Thermistors: 8 channels for a 12-second record at 4 Hz of 24-bit data every 6 hours for a year.
d. Two sets of 2 orientation sensors (heading, pitch, roll) at 4 Hz for 1-4 seconds.
e. The orientation sensor works as a current meter and may be activated once per hour throughout the year.

54. The deep tow is similar to the accelerometer array; 16-bits at 10 kHz. There is some interest in going faster, up to 100 kHz. The duration will be 1-4 seconds per pop continuously for 48-96 hours.
55. The sled is the same as the other hydroacoustic / seismometer arrays, with only 50 pops (50 m drag, 1 pop per m).
56. The AUV will be the same as the deep tow.
57. The number of arrays will be:
    a. One vertical array with 16 hydrophones.
    b. One downhole array with 8 sets of accelerometers, 16 hydrophones, and thermistors.
    c. Four horizontal arrays (like the downhole), with maybe twice as many accelerometers as the vertical arrays.
58. Jeff Chanton and Laura Lapham data sets representing pore fluids from the sediment will be tabular data.
59. The bottom sediment thermistors (x3) on the bottom hole will collect temperature flux of the near bottom at 30 minute intervals.
60. Two arrays have been installed (already) in the sub-bottom as 10 m gravity entrenchments. One is a pore fluid array and the other is a temperature array.

## Interview with Norm Farr (Woods Hole Oceanographic Institute)

Location: Barrodale Computing Services Ltd., Victoria, BC, Canada
Date: 16:00-16:30, Wednesday, November 16, 2005
Present: Cedric Zala, Mike Dunham-Wilkie, Kent Berger-North (BCS)
On Phone: Norm Farr (Woods Hole Oceanographic Institute - WHOI)

17. Norm is responsible for two oceanographic arrays: the chimney array and the BBLA (Benthic Boundary Layer Array).
18. The chimney array will be situated over a hydrate mass to measure property flux.
19. The BBLA will maintain flotation at the top of the mooring and a cage on the bottom containing the data logger. For the BBLA:
    a. The array will be vertical in orientation.
    b. It is planned to be deployed in Jan/Feb 2006 for 48 hrs and recovered, then redeployed for a week.
    c. Most of the decisions regarding sensor sampling regimes will be determined after inspecting the data collected.
    d. The cage/node at the bottom will have the ability to transport the data over IP.
    e. The data logger may be turned on by event triggers (e.g., METS sensor thresholds).

f. There will be 3, potentially 6, fluorometers on the BBLA. One will be at 50 m and one at 1.5-2.0 m above the bottom.
g. The ADCP will be on top. The orientation (upward-looking, downward-looking) has not been determined. [Note: if downward-looking, the instruments and the mooring may contaminate the acoustic field.]
h. There will be a CTD on the BBLA.
i. There will be an oxygen sensor on the BBLA (possibly Aanderaa?).
j. There may be duplicate sensors of everything on the BBLA (TBD).
k. Data storage, computer, HDD, and batteries will be in the node.
l. The METS is a vulnerable sensor; it will be on both arrays.

20. The chimney array will contain:
    a. The vision package (still being designed): 2 monochrome and 1 color camera simultaneously collecting data, with data reduction schemes where possible.
    b. An Aanderaa sensor for conductivity, depth, and oxygen.
    c. A METS sensor.
    d. All sensors will be connected to the vision package.
21. MBARI has done a lot of work with the MARS Observatory. We should look at their database design. Contact: Mark Chaffey. The mooring system will be similar to their MOOS.
22. All data in the arrays will be centrally logged in their respective nodes, with a common timestamp, if possible.
23. The controller in the node will drive the duty cycles of the sensors.
24. How the event triggering will be realized has yet to be addressed.
25. In late January, Norm will be able to commit to some form of data structure.
26. The AUV is completely independent of the other arrays. The data products will likely be benthic, chemical and photographic maps (lat/long).
27. WHOI is currently logging meteorological and optical data from a buoy, with both raw data and calibration information available to users. Norm has a web-enabled database containing this data and some optical data. **Norm** will send BCS the link to this database.
28. MBARI is running a similar site, without calibration info.
29. The buoy(s) use a “Big S” mooring configuration. See: http://www.mbari.org/news/homepage/2005/mtm3.html
30. Norm is hoping that the ADCP stores data in real units.
31. The mass spectrometer (MS) will go on the AUV, and perhaps on the chimney. Rich Camilli is the contact for the MS, which is possibly manufactured by http://www.monitorinstruments.com.
32. **Norm** will provide BCS with sample data from the arrays when they are available. **BCS** will contact Norm following the Jan/Feb cruise to make arrangements to receive this data, which is proprietary to the Consortium.

# Appendix 4: DMAS Database Design Report

| | |
|---|---|
| Target DBMS: | ODBC Generic Driver |
| Number of tables: | 37 |
| Number of columns: | 180 |
| Number of indexes: | 12 |
| Number of foreign keys: | 44 |

| Tables | Columns | Indexes | Foreign keys | Notes |
|---|---|---|---|---|
| gmh_data_product_permissions | 2 | 0 | 2 | This table relates user roles to the data products that can be requested by users with that role. |
| gmh_user_roles | 2 | 0 | 2 | This table relates GMH users to the access role(s) they have. |
| gmh_standard_data_products | 4 | 0 | 1 | Table of standard data products. Each data product corresponds to a single standard data product type (e.g., "time series of N-minute averages of calibrated data at each of M ordinates"). To accommodate ad-hoc requests, one "standard data product" will be "custom". |
| gmh_calibrated_measurements_usage | 2 | 0 | 2 | Tracks which (calibrated) measurements were used to produce each data product, since a data product can be formed from multiple measurements, and a measurement can be used in more than one data product. |
| gmh_data_product_outputtypes | 2 | 0 | 0 | This table defines the various classes of data products that can be produced from GMH. Examples include images, Excel spreadsheets, Textual reports, etc. |
| gmh_data_products | 7 | 0 | 2 | This table is a catalog of the various data products that have been produced. |
| gmh_roles | 4 | 1 | 0 | This table describes the various database roles that GMH users can have. |
| gmh_users | 5 | 1 | 0 | This table defines the GMH users. |
| gmh_user_annotations | 6 | 0 | 4 | Table of annotations that GMH users have attached to raw and/or calibrated measurements and/or data products. |
| gmh_raw_measurement_ids | 3 | 0 | 0 | |
| gmh_calibrated_measurements | 5 | 0 | 1 | This table is a companion to the various raw |

| | | | | |
|---|---|---|---|---|
| | | | | measurement tables. Post-recovery, the raw measurements can be post-processed by applying calibration parameters to the data. The calibrated data and the calibration parameters are stored in this table.<br><br>In this design, all post-calibration data is stored in this single table, since the "blob" data type is sufficiently general to store any type of measurement. If in the future instrument-specific opaque data types are used, then this table can be replaced by a table hierarchy, with a subtable for each different type of measurement. |
| gmh_fluorometer_measurements | 5 | 0 | 1 | Fluorometers will be stored on the BBLA (Farr-1-3). |
| gmh_sled_configuration_history | 4 | 0 | 0 | This table contains information on how the sled system has been configured, either at present (gmhslh_hist_end_datetime is NULL) or in the past (gmhslh_hist_end_datetime is not NULL). The calibration and configuration history of the sensors on the sled are stored in the gmh_sensor_calibration_history and gmh_sensor_configuration_history tables, respectively. |
| gmh_auv_measurements | 5 | 1 | 2 | Table of AUV data measurements, taken from hydrophone in nose of AUV.<br><br>Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?<br>Reference: Higley-1-23e. |
| gmh_sled_measurements | 10 | 1 | 3 | "Position and time information is not necessary as the wave will already be well-developed. The hydrophone will pick up the signal and subsequent reflection."<br>Table of "Sled System" data |

| | | | | |
|---|---|---|---|---|
| | | | | measurements. Sled contains its own shear wave source; consequently no foreign key to a shot number is needed. Measures accelerometer data, orientation data, hydrophone data. Reference: Higley-1-23d. |
| gmh_deep_tow_measurements | 5 | 1 | 2 | Table of "Deep Tow" measurements. Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL? Reference: Higley-1-23c. |
| gmh_sensor_configuration_history | 5 | 0 | 1 | This table contains information on how a particular sensor has been configured, either at present (gmhscn_hist_end_datetime is NULL) or in the past (gmhscn_hist_end_datetime is not NULL). |
| gmh_sensor_calibration_history | 5 | 0 | 1 | This table contains information on how a particular sensor has been calibrated, either at present (gmhscl_hist_end_datetime is NULL) or in the past (gmhscl_hist_end_datetime is not NULL). |
| gmh_3daccel_measurements | 5 | 1 | 2 | Table of 3D accelerometer measurements. Reference: Higley-1-23b.<br><br>Sampling regime is 8 channels x 3 sensors for 1-4 seconds at 1 kHz of 24 bit data continuously for 48-96 hours. |
| gmh_thermistor_measurements | 5 | 0 | 1 | Table of thermistor measurements. Not tied to shot number; thermistor has own RTC (Higley-1-20). Records every 6 hours for a year. Reference: Higley-1-23c.<br><br>Sampling regime is 8 channels for 12 seconds at 4 Hz of 24 bit data every 6 hours for a year. |
| gmh_orientation_sensor_measurements | 6 | 1 | 2 | Table of orientation sensor measurements. Q: WILL THESE ALL BE |

| | | | | |
|---|---|---|---|---|
| | | | | TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?<br>Reference: Higley-1-23d<br><br>There are 2 sets of 2 orientation sensors measuring heading, pitch, and roll.<br><br>Sampling regime is 4 Hz for 1-4 seconds.<br><br>Higley-1-23e says the orientation sensor works as a current meter and may be activated once per hour throughout the year. |
| gmh_hydrophone_measurements | 11 | 1 | 2 | Table of hydrophone measurements.<br>Reference: Higley-1-23a.<br><br>Sampling regime is 16 channels for 1-4 seconds (assume 4) at 10 kHz of 16 bit data continuously for 48-96 hours. |
| gmh_sensor_types | 4 | 1 | 0 | This table contains all the relevant information related to specific sensor types, e.g., name, type of sensor (hydrophone, ADCP, thermistor, etc), generic properties, etc. |
| gmh_ship_log | 1 | 0 | 0 | |
| gmh_shot_log | 5 | 0 | 0 | Database realization of the "shot log". Also called the "Ship Pop" file. See Higley-1-19 and Higley-1-21b. |
| gmh_sensors | 8 | 1 | 3 | In this context a sensor is either an instrument or an individual sensor on a multi-sensor instrument.<br><br>This table contains all the relevant information related to a specific sensor, e.g., what logical array is it part of? what type of sensor is it? where is it? etc. Note that sensor measurement values are stored in other tables. |
| gmh_sensor_arrays | 4 | 1 | 0 | |
| gmh_array_calibration_history | 5 | 0 | 1 | This table contains information on how a particular physical array has been calibrated, either at present (gmhacl_hist_end_datetime |

| | | | | |
|---|---|---|---|---|
| | | | | is NULL) or in the past (gmhacl_hist_end_datetime is not NULL). |
| gmh_array_configuration_history | 5 | 0 | 1 | This table contains information on how a particular physical array has been configured, either at present (gmhacn_hist_end_datetime is NULL) or in the past (gmhacn_hist_end_datetime is not NULL). |
| gmh_oxygen_sensor_measurements | 5 | 0 | 1 | The BBLA will have oxygen sensors (Farr-1-3). |
| gmh_adcp_measurements | 5 | 0 | 1 | The BBLA will have an ADCP at the top (Farr-1-3). |
| gmh_mets_measurements | 5 | 0 | 1 | The chimney array and the BBLA will both have methane sensors (Farr-1-3 and Farr-1-4). |
| gmh_ctd_measurements | 5 | 0 | 1 | The BBLA will have CTD instruments (Farr-1-3). |
| gmh_mass_spec_measurements | 5 | 0 | 1 | The chimney array may have a mass spectrometer (Farr-1-15). |
| gmh_images | 5 | 0 | 1 | The chimney array will have 3 cameras (Farr-1-4). |
| gmh_cable_arrays | 5 | 1 | 0 | This table contains information describing the individual acoustic and environmental physical arrays (cables) in the observatory. A physical array can contain multiple types of sensors and logical arrays of sensors (e.g., A physical array can contain a logical array of hydrophones + a logical array of thermistors + a selection of individual sensors). |
| gmh_array_connections | 5 | 0 | 2 | This table contains information on which arrays are connected together. From this table and the gmh_sensors and gmh_arrrays tables it will be possible to determine the current topology and 3D positioning of the arrays and sensors. NB There is no provision for storing the connection history. |

# gmh_3daccel_measurements

Physical name: gmh_3daccel_measurements
Notes: Table of 3D accelerometer measurements.
Reference: Higley-1-23b.

Sampling regime is 8 channels x 3 sensors for 1-4 seconds at 1 kHz of 24 bit data continuously for 48-96 hours.

Owner:
Target DB name:
Number of columns: 5
Number of indexes: 1
Number of foreign keys: 2
Primary key: gmh_global_raw_meas_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK,U1) | INT8 | Not allowed | |
| gmhstl_shot_id (FK,U1) | INT8 | Not allowed | |
| gmh3dm_3daccel_meas | BYTE | Not allowed | |
| 3DACCEL_MEAS_METADATA | INT8 | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_3daccel_measurements_AK1 (U1) | gmhstl_shot_id | Ascending |
| | gmhsen_sensor_id | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_3daccel_measurements_FK1 | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |
| gmh_sensors_gmh_3daccel_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmh_global_raw_meas_id
Physical name: gmh_global_raw_meas_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers.

2. gmhsen_sensor_id (FK,U1)
Physical name: gmhsen_sensor_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of 3d accelerometer sensor.

Higley-1-23b says that the sampling regime for accelerometers is 8 channels x 3 sensors (3 axes) for 1-4 seconds of 24-bit data at 1kHz continuously for 48-96 hours.

3. gmhstl_shot_id (FK,U1)
Physical name: gmhstl_shot_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of shot (pop) associated with this measurement.

4. gmh3dm_3daccel_meas
Physical name: gmh3dm_3daccel_meas
Physical data type: BYTE
Allow NULLs: Not allowed
Notes: Blob of measurement data. Higley-1-17 and Higley-1-18 imply that there is one file per shot

(pop).

5. 3DACCEL_MEAS_METADATA

| | |
|---|---|
| Physical name: | 3DACCEL_MEAS_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

**Index details**

gmh_3daccel_measurements_AK1

| | |
|---|---|
| Column(s): | gmhstl_shot_id (Asc)<br>gmhsen_sensor_id (Asc) |
| Unique: | Yes |

**Foreign key details (child)**

gmh_shot_log_gmh_3daccel_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_shot_log_gmh_3daccel_measurements_FK1 |
| Notes: | Each 3d accelerometer measurement is for a single shot. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_3daccel_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | 3d accelerometer |
| Physical name: | gmh_sensors_gmh_3daccel_measurements_FK1 |
| Notes: | Each 3d accelerometer produces 1 or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_adcp_measurements

| | |
|---|---|
| Physical name: | gmh_adcp_measurements |
| Notes: | The BBLA will have an ADCP at the top (Farr-1-3). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK) | INT8 | Not allowed | |
| gmhcpm_adcp_meas | BYTE | Not allowed | |
| gmhcpm_adcp_meas_datetime | DATETIME YEAR TO | Not allowed | |
| ADCP_MEAS_METADATA | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_adcp_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmhsen_sensor_id (FK)

| | |
|---|---|
| Physical name: | gmhsen_sensor_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of the adcp sensor that made this measurement. |

3. gmhcpm_adcp_meas

| | |
|---|---|
| Physical name: | gmhcpm_adcp_meas |
| Physical data type: | BYTE |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

4. gmhcpm_adcp_meas_datetime

| | |
|---|---|
| Physical name: | gmhcpm_adcp_meas_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. ADCP_MEAS_METADATA

| | |
|---|---|
| Physical name: | ADCP_MEAS_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). |

Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

## Foreign key details (child)

gmh_sensors_gmh_adcp_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_adcp_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_array_calibration_history

Physical name: gmh_array_calibration_history
Notes: This table contains information on how a particular physical array has been calibrated, either at present (gmhacl_hist_end_datetime is NULL) or in the past (gmhacl_hist_end_datetime is not NULL).
Owner:
Target DB name:
Number of columns: 5
Number of indexes: 0
Number of foreign keys: 1
Primary key: gmhacl_array_calibration_hist_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhacl_array_calibration_hist_id | SERIAL | Not allowed | |
| gmhcbl_cable_array_id (FK) | INT8 | Not allowed | |
| gmhacl_hist_start_datetime | DATETIME YEAR TO | Not allowed | |
| gmhacl_hist_end_datetime | DATETIME YEAR TO | Allowed | |
| ARRAY_CALIBRATION_PARAMETERS | VARCHAR(100) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_cable_arrays_gmh_array_calibration_history_FK1 | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |

Column details

1. gmhacl_array_calibration_hist_id
Physical name: gmhacl_array_calibration_hist_id
Physical data type: SERIAL
Allow NULLs: Not allowed
Notes: Unique id identifying the physical array calibration history record. Probably a generated key.

2. gmhcbl_cable_array_id (FK)
Physical name: gmhcbl_cable_array_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of the physical (cable) array to which this history pertains.

3. gmhacl_hist_start_datetime
Physical name: gmhacl_hist_start_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Not allowed
Notes: Date and time when this record became effective.

4. gmhacl_hist_end_datetime
Physical name: gmhacl_hist_end_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Allowed
Notes: Date and time when this record ceased to be effective.

5. ARRAY_CALIBRATION_PARAMETERS
Physical name: ARRAY_CALIBRATION_PARAMETERS
Physical data type: VARCHAR(100)

| | |
|---|---|
| Allow NULLs: | Not allowed |
| Notes: | This will be replaced by a set of columns, one column for each calibration parameter, holding the value of that parameter.<br><br>Note that each type of sensor will have different calibration parameters, so either:<br><br>1. The parameters will be stored in subtables, with one subtable for each type of sensor, or<br><br>2. The parameters will be stored in a single table, with the parameters that are not applicable to a particular sensor simply being left blank.<br><br>Option 1 is cleaner, but harder to maintain. It is much easier with Option 2 to add parameters for new types of sensors, or add additional parameters for existing types of sensors. |

**Foreign key details (child)**

gmh_cable_arrays_gmh_array_calibration_history_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_cable_arrays_gmh_array_calibration_history_FK1 |
| Notes: | Each physical array has a calibration history. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_array_configuration_history

Physical name: gmh_array_configuration_history
Notes: This table contains information on how a particular physical array has been configured, either at present (gmhacn_hist_end_datetime is NULL) or in the past (gmhacn_hist_end_datetime is not NULL).
Owner:
Target DB name:
Number of columns: 5
Number of indexes: 0
Number of foreign keys: 1
Primary key: gmhacn_array_config_hist_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhacn_array_config_hist_id | SERIAL | Not allowed | |
| gmhcbl_cable_array_id (FK) | INT8 | Not allowed | |
| gmhacn_hist_start_datetime | DATETIME YEAR TO | Not allowed | |
| gmhacn_hist_end_datetime | DATETIME YEAR TO | Allowed | |
| ARRAY_CONFIG_PARAMETERS | VARCHAR(100) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_cable_arrays_gmh_array_configuration_history_FK1 | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |

Column details

1. gmhacn_array_config_hist_id
Physical name: gmhacn_array_config_hist_id
Physical data type: SERIAL
Allow NULLs: Not allowed
Notes: Unique id identifying the physical array configuration history record. Probably a generated key.

2. gmhcbl_cable_array_id (FK)
Physical name: gmhcbl_cable_array_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of the physical (cable) array to which this history pertains.

3. gmhacn_hist_start_datetime
Physical name: gmhacn_hist_start_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Not allowed
Notes: Date and time when this record became effective.

4. gmhacn_hist_end_datetime
Physical name: gmhacn_hist_end_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Allowed
Notes: Date and time when this record ceased to be effective.

5. ARRAY_CONFIG_PARAMETERS
Physical name: ARRAY_CONFIG_PARAMETERS

| | |
|---|---|
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | This will be replaced by a set of columns, one column for each configuration parameter, holding the value of that parameter.<br><br>Note that each type of sensor will have different configuration parameters, so either:<br><br>1. The parameters will be stored in subtables, with one subtable for each type of sensor, or<br><br>2. The parameters will be stored in a single table, with the parameters that are not applicable to a particular sensor simply being left blank.<br><br>Option 1 is cleaner, but harder to maintain. It is much easier with Option 2 to add parameters for new types of sensors, or add additional parameters for existing types of sensors. |

**Foreign key details (child)**

gmh_cable_arrays_gmh_array_configuration_history_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_cable_arrays_gmh_array_configuration_history_FK1 |
| Notes: | Each physical array has a configuration history. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_array_connections

| | |
|---|---|
| Physical name: | gmh_array_connections |
| Notes: | This table contains information on which arrays are connected together. From this table and the gmh_sensors and gmh_arrrays tables it will be possible to determine the current topology and 3D positioning of the arrays and sensors. NB There is no provision for storing the connection history. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | gmhacn_connection_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhacn_connection_id | INT8 | Not allowed | |
| gmhcbl_cable_array_1_id (FK) | INT8 | Not allowed | |
| gmhacn_array_1_end | CHAR(1) | Allowed | |
| gmhcbl_cable_array_2_id (FK) | INT8 | Not allowed | |
| gmhacn_array_2_end | CHAR(1) | Allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_cable_arrays_gmh_array_connections_FK1 | gmhcbl_cable_array_1_id | gmh_cable_arrays.gmhcbl_cable_array_id |
| gmh_cable_arrays_gmh_array_connections_FK2 | gmhcbl_cable_array_2_id | gmh_cable_arrays.gmhcbl_cable_array_id |

## Column details

1. gmhacn_connection_id

| | |
|---|---|
| Physical name: | gmhacn_connection_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying an individual connection between arrays. Probably a generated key. |

2. gmhcbl_cable_array_1_id (FK)

| | |
|---|---|
| Physical name: | gmhcbl_cable_array_1_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of one of the arrays participating in the connection. |

3. gmhacn_array_1_end

| | |
|---|---|
| Physical name: | gmhacn_array_1_end |
| Physical data type: | CHAR(1) |
| Allow NULLs: | Allowed |
| Notes: | Which end of the array is connected to this connection? |

4. gmhcbl_cable_array_2_id (FK)

| | |
|---|---|
| Physical name: | gmhcbl_cable_array_2_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of another one of the arrays participating in the connection. |

5. gmhacn_array_2_end

| | |
|---|---|
| Physical name: | gmhacn_array_2_end |
| Physical data type: | CHAR(1) |
| Allow NULLs: | Allowed |
| Notes: | Which end of the array is connected to this connection? |

## Foreign key details (child)

gmh_cable_arrays_gmh_array_connections_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhcbl_cable_array_1_id | gmh_cable_arrays.gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | end1 |
| Physical name: | gmh_cable_arrays_gmh_array_connections_FK1 |
| Notes: | Identifies one of the physical arrays (cable) involved in the connection. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_cable_arrays_gmh_array_connections_FK2

| Definition: | Child | Parent |
|---|---|---|
| | gmhcbl_cable_array_2_id | gmh_cable_arrays.gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | end2 |
| Physical name: | gmh_cable_arrays_gmh_array_connections_FK2 |
| Notes: | Identifies one of the physical arrays (cables) involved in the connection. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_auv_measurements

Physical name: gmh_auv_measurements
Notes: Table of AUV data measurements, taken from hydrophone in nose of AUV.

Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?
Reference: Higley-1-23e.

"Position and time information is not necessary as the wave will already be well-developed. The hydrophone will pick up the signal and subsequent reflection."

Owner:
Target DB name:
Number of columns: 5
Number of indexes: 1
Number of foreign keys: 2
Primary key: gmh_global_raw_meas_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK,U1) | INT8 | Not allowed | |
| gmhstl_shot_id (FK,U1) | INT8 | Not allowed | |
| gmhauv_auv_meas | BYTE | Not allowed | |
| AUV_MEASUREMENT_METADATA | INT8 | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_auv_measurements_AK1 (U1) | gmhstl_shot_id | Ascending |
| | gmhsen_sensor_id | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_auv_measurements_FK1 | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |
| gmh_sensors_gmh_auv_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmh_global_raw_meas_id
Physical name: gmh_global_raw_meas_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers.

2. gmhsen_sensor_id (FK,U1)
Physical name: gmhsen_sensor_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of the AUV hydrophone.

3. gmhstl_shot_id (FK,U1)
Physical name: gmhstl_shot_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of shot (pop) associated with this measurement.

4. gmhauv_auv_meas
Physical name: gmhauv_auv_meas
Physical data type: BYTE
Allow NULLs: Not allowed
Notes: Blob of measurement data. Higley-1-17 and Higley-1-18 imply that there is one file per shot (pop).

5. AUV_MEASUREMENT_METADATA

| | |
|---|---|
| Physical name: | AUV_MEASUREMENT_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

**Index details**

gmh_auv_measurements_AK1

| | |
|---|---|
| Column(s): | gmhstl_shot_id (Asc)<br>gmhsen_sensor_id (Asc) |
| Unique: | Yes |

**Foreign key details (child)**

gmh_shot_log_gmh_auv_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_shot_log_gmh_auv_measurements_FK1 |
| Notes: | Each AUV measurement is for a single shot. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_auv_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | auv |
| Physical name: | gmh_sensors_gmh_auv_measurements_FK1 |
| Notes: | Each auv sensor produces one or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_cable_arrays

| | |
|---|---|
| Physical name: | gmh_cable_arrays |
| Notes: | This table contains information describing the individual acoustic and environmental physical arrays (cables) in the observatory. A physical array can contain multiple types of sensors and logical arrays of sensors (e.g., A physical array can contain a logical array of hydrophones + a logical array of thermistors + a selection of individual sensors). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 1 |
| Number of foreign keys: | 0 |
| Primary key: | gmhcbl_cable_array_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhcbl_cable_array_id | SERIAL | Not allowed | |
| gmhcbl_cable_array_code (U1) | CHAR(10) | Not allowed | |
| gmhcbl_cable_array_desc | VARCHAR(255) | Not allowed | |
| gmhcbl_cable_array_linework | BYTE | Allowed | |
| CABLE_ARRAY_INFO | VARCHAR(100) | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmhcbl_cable_array_code_idx (U1) | gmhcbl_cable_array_code | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_cable_arrays_gmh_array_connections_FK1 | gmh_array_connections.gmhcbl_cable_array_1_id | gmhcbl_cable_array_id |
| gmh_cable_arrays_gmh_array_connections_FK2 | gmh_array_connections.gmhcbl_cable_array_2_id | gmhcbl_cable_array_id |
| gmh_cable_arrays_gmh_array_configuration_history_FK1 | gmh_array_configuration_history.gmhcbl_cable_array_id | gmhcbl_cable_array_id |
| gmh_cable_arrays_gmh_array_calibration_history_FK1 | gmh_array_calibration_history.gmhcbl_cable_array_id | gmhcbl_cable_array_id |
| gmh_cable_arrays_gmh_sensors_FK1 | gmh_sensors.gmhcbl_cable_array_id | gmhcbl_cable_array_id |

## Column details

1. gmhcbl_cable_array_id

| | |
|---|---|
| Physical name: | gmhcbl_cable_array_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the cable. Probably a generated key. |

2. gmhcbl_cable_array_code (U1)

| | |
|---|---|
| Physical name: | gmhcbl_cable_array_code |
| Physical data type: | CHAR(10) |
| Allow NULLs: | Not allowed |
| Notes: | A more mnemonic code identifying the cable array. e.g. HLA, VBA, etc. |

3. gmhcbl_cable_array_desc

| | |
|---|---|
| Physical name: | gmhcbl_cable_array_desc |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Not allowed |
| Notes: | A one line description of the array. |

4. gmhcbl_cable_array_linework

| | |
|---|---|
| Physical name: | gmhcbl_cable_array_linework |
| Physical data type: | BYTE |
| Allow NULLs: | Allowed |

Notes: Geometric coordinates (lat,long,depth) describing the shape of the cable array.

5. CABLE_ARRAY_INFO

Physical name: CABLE_ARRAY_INFO
Physical data type: VARCHAR(100)
Allow NULLs: Not allowed
Notes: This column will be replaced by a set of one or more columns, holding other parameters needed to describe this cable array, and any other relevant metadata.

## Index details

gmhcbl_cable_array_code_idx

Column(s): gmhcbl_cable_array_code (Asc)
Unique: Yes

## Foreign key details (parent)

gmh_cable_arrays_gmh_array_connections_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_array_connections.gmhcbl_cable_array_1_id | gmhcbl_cable_array_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Verb phrase: end1
Physical name: gmh_cable_arrays_gmh_array_connections_FK1
Notes: Identifies one of the physical arrays (cable) involved in the connection.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_cable_arrays_gmh_array_connections_FK2

| Definition: | Child | Parent |
|---|---|---|
| | gmh_array_connections.gmhcbl_cable_array_2_id | gmhcbl_cable_array_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Verb phrase: end2
Physical name: gmh_cable_arrays_gmh_array_connections_FK2
Notes: Identifies one of the physical arrays (cables) involved in the connection.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_cable_arrays_gmh_array_configuration_history_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_array_configuration_history.gmhcbl_cable_array_id | gmhcbl_cable_array_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Physical name: gmh_cable_arrays_gmh_array_configuration_history_FK1
Notes: Each physical array has a configuration history.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_cable_arrays_gmh_array_calibration_history_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_array_calibration_history.gmhcbl_cable_array_id | gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_cable_arrays_gmh_array_calibration_history_FK1 |
| Notes: | Each physical array has a calibration history. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_cable_arrays_gmh_sensors_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_sensors.gmhcbl_cable_array_id | gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | sensor situated on physical array |
| Physical name: | gmh_cable_arrays_gmh_sensors_FK1 |
| Notes: | Each sensor is located on a physical array |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_calibrated_measurements

Physical name: gmh_calibrated_measurements
Notes: This table is a companion to the various raw measurement tables. Post-recovery, the raw measurements can be post-processed by applying calibration parameters to the data. The calibrated data and the calibration parameters are stored in this table.

In this design, all post-calibration data is stored in this single table, since the "blob" data type is sufficiently general to store any type of measurement. If in the future instrument-specific opaque data types are used, then this table can be replaced by a table hierarchy, with a subtable for each different type of measurement.

Owner:
Target DB name:
Number of columns: 5
Number of indexes: 0
Number of foreign keys: 1
Primary key: gmh_global_calibrated_meas_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_calibrated_meas_id | SERIAL | Not allowed | |
| gmh_global_raw_meas_id (FK) | INT8 | Not allowed | |
| gmh_calibrated_measurement | BYTE | Not allowed | |
| CALIBRATION_PARAMETERS | VARCHAR(100) | Not allowed | |
| MEASUREMENT_METADATA | VARCHAR(100) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_raw_measurement_ids_gmh_calibrated_measurements_FK1 | gmh_global_raw_meas_id | gmh_raw_measurement_ids.gmh_global_raw_meas_id |
| gmh_calibrated_measurements_gmh_user_annotations_FK1 | gmh_user_annotations.gmh_global_calibrated_meas_id | gmh_global_calibrated_meas_id |
| gmh_calibrated_measurements_gmh_calibrated_measurements_usage_FK1 | gmh_calibrated_measurements_usage.gmh_global_calibrated_meas_id | gmh_global_calibrated_meas_id |

Column details

1. gmh_global_calibrated_meas_id
Physical name: gmh_global_calibrated_meas_id
Physical data type: SERIAL
Allow NULLs: Not allowed
Notes: Unique id identifying the calibrated measurement. Probably a generated key.

2. gmh_global_raw_meas_id (FK)
Physical name: gmh_global_raw_meas_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers.

3. gmh_calibrated_measurement
Physical name: gmh_calibrated_measurement
Physical data type: BYTE
Allow NULLs: Not allowed
Notes: The calibrated measurement itself.

4. CALIBRATION_PARAMETERS
Physical name: CALIBRATION_PARAMETERS
Physical data type: VARCHAR(100)
Allow NULLs: Not allowed
Notes: These columns will define the specific calibration parameters used in producing the calibrated measurement / image from the raw measurement / image.

This design assumes that the combinations of calibration parameter values will be quite dynamic (i.e., the values are quite specific to individual measurements). If instead it is the case that the calibration parameter values are quite static (i.e., a few specific combinations are used for a wide range of measurements) then it might be better to factor these columns out into a separate table and just include a foreign key to that table in this table.

5. MEASUREMENT_METADATA

| | |
|---|---|
| Physical name: | MEASUREMENT_METADATA |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the calibrated measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was produced, and the size of the measurement. Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

## Foreign key details (child)

gmh_raw_measurement_ids_gmh_calibrated_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_raw_meas_id | gmh_raw_measurement_ids.gmh_global_raw_meas_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | produced from |
| Physical name: | gmh_raw_measurement_ids_gmh_calibrated_measurements_FK1 |
| Notes: | Each raw measurement or image (recorded in one of the gmh_*_measurements or the gmh_images table) can have 0 or more corresponding post-deployment calibrated measurements / images. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

## Foreign key details (parent)

gmh_calibrated_measurements_gmh_user_annotations_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_user_annotations.gmh_global_calibrated_meas_id | gmh_global_calibrated_meas_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | annotates a calibrated measurment |
| Physical name: | gmh_calibrated_measurements_gmh_user_annotations_FK1 |
| Notes: | Each calibrated measurement may be annotated zero or more times by one or more users. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_calibrated_measurements_gmh_calibrated_measurements_usage_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_calibrated_measurements_usage.gmh_global_calibrated_meas_id | |

gmh_global_calibrated_meas_id

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | used in data product |
| Physical name: | gmh_calibrated_measurements_gmh_calibrated_measurements_usage_FK1 |
| Notes: | A single calibrated measurement can be used in zero or more data products. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_calibrated_measurements_usage

| | |
|---|---|
| Physical name: | gmh_calibrated_measurements_usage |
| Notes: | Tracks which (calibrated) measurements were used to produce each data product, since a data product can be formed from multiple measurements, and a measurement can be used in more than one data product. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 2 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | 1. gmh_global_data_product_id<br>2. gmh_global_calibrated_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_data_product_id (FK) | INT8 | Not allowed | |
| gmh_global_calibrated_meas_id (FK) | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_calibrated_measurements_gmh_calibrated_measurements_usage_FK1 | gmh_global_calibrated_meas_id | gmh_calibrated_measurements.gmh_global_calibrated_meas_id |
| gmh_data_products_gmh_calibrated_measurements_usage_FK1 | gmh_global_data_product_id | gmh_data_products.gmh_global_data_product_id |

## Column details

1. gmh_global_data_product_id (FK)

| | |
|---|---|
| Physical name: | gmh_global_data_product_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the data product. |

2. gmh_global_calibrated_meas_id (FK)

| | |
|---|---|
| Physical name: | gmh_global_calibrated_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the calibrated measurement. Probably a generated key. |

## Foreign key details (child)

gmh_calibrated_measurements_gmh_calibrated_measurements_usage_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_calibrated_meas_id | gmh_calibrated_measurements.gmh_global_calibrated_meas_id |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | used in data product |
| Physical name: | gmh_calibrated_measurements_gmh_calibrated_measurements_usage_FK1 |
| Notes: | A single calibrated measurement can be used in zero or more data products. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_data_products_gmh_calibrated_measurements_usage_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_data_product_id | |

gmh_data_products.gmh_global_data_product_id

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | based on calibrated measurement |
| Physical name: | gmh_data_products_gmh_calibrated_measurements_usage_FK1 |
| Notes: | A data product is formed from one or more calibrated measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_ctd_measurements

| | |
|---|---|
| Physical name: | gmh_ctd_measurements |
| Notes: | The BBLA will have CTD instruments (Farr-1-3). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK) | INT8 | Not allowed | |
| gmhctd_ctd_meas | BYTE | Not allowed | |
| gmhctd_ctd_meas_datetime | DATETIME YEAR TO | Not allowed | |
| CTD_MEAS_METADATA | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_ctd_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmhsen_sensor_id (FK)

| | |
|---|---|
| Physical name: | gmhsen_sensor_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of the ctd sensor that made this measurement. |

3. gmhctd_ctd_meas

| | |
|---|---|
| Physical name: | gmhctd_ctd_meas |
| Physical data type: | BYTE |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

4. gmhctd_ctd_meas_datetime

| | |
|---|---|
| Physical name: | gmhctd_ctd_meas_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. CTD_MEAS_METADATA

| | |
|---|---|
| Physical name: | CTD_MEAS_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). |

Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

## Foreign key details (child)

gmh_sensors_gmh_ctd_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_ctd_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_data_product_outputtypes

| | |
|---|---|
| Physical name: | gmh_data_product_outputtypes |
| Notes: | This table defines the various classes of data products that can be produced from GMH. Examples include images, Excel spreadsheets, Textual reports, etc. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 2 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmhdap_data_product_outputtype_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhdap_data_product_outputtype_id | SERIAL | Not allowed | |
| gmhdat_data_product_outputtype_desc | VARCHAR(100) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_data_product_outputtypes_gmh_standard_data_products_FK1 | gmh_standard_data_products.gmhdap_data_product_outputtype_id | gmhdap_data_product_outputtype_id |

**Column details**

1. gmhdap_data_product_outputtype_id

| | |
|---|---|
| Physical name: | gmhdap_data_product_outputtype_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID - probably autogenerated. |

2. gmhdat_data_product_outputtype_desc

| | |
|---|---|
| Physical name: | gmhdat_data_product_outputtype_desc |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | Description of the class of data product - e.g., "Excel spreadsheet" |

**Foreign key details (parent)**

gmh_data_product_outputtypes_gmh_standard_data_products_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmh_standard_data_products.gmhdap_data_product_outputtype_id | gmhdap_data_product_outputtype_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | has output type |
| Physical name: | gmh_data_product_outputtypes_gmh_standard_data_products_FK1 |
| Notes: | Each standard data product has a particular output data type (image, spreadsheet, etc.) |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_data_product_permissions

| | |
|---|---|
| Physical name: | gmh_data_product_permissions |
| Notes: | This table relates user roles to the data products that can be requested by users with that role. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 2 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | 1. gmhsdp_standard_data_product_id<br>2. gmhrol_role_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhsdp_standard_data_product_id (FK) | INT8 | Not allowed | |
| gmhrol_role_id (FK) | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_roles_gmh_data_product_permissions_FK1 | gmhrol_role_id | gmh_roles.gmhrol_role_id |
| gmh_standard_data_products_gmh_data_product_permissions_FK1 | gmhsdp_standard_data_product_id | gmh_standard_data_products.gmhsdp_standard_data_product_id |

## Column details

1. gmhsdp_standard_data_product_id (FK)

| | |
|---|---|
| Physical name: | gmhsdp_standard_data_product_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the standard data product description - probably autogenerated |

2. gmhrol_role_id (FK)

| | |
|---|---|
| Physical name: | gmhrol_role_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Alphanumeric identifier of the role. |

## Foreign key details (child)

gmh_roles_gmh_data_product_permissions_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhrol_role_id | gmh_roles.gmhrol_role_id |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_roles_gmh_data_product_permissions_FK1 |
| Notes: | Each role has an associated list of standard data products that users with that role can produce. In order to produce a particular data product, that data product must appear in the list associated with one of the user's roles. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_standard_data_products_gmh_data_product_permissions_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhsdp_standard_data_product_id | gmh_standard_data_products.gmhsdp_standard_data_product_id |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | role has permission to create product |
| Physical name: | gmh_standard_data_products_gmh_data_product_permissions_FK1 |
| Notes: | Each data product has an associated list of user roles. In order to produce a particular data product, a user must have one of the roles in that data product's list. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_data_products

| | |
|---|---|
| Physical name: | gmh_data_products |
| Notes: | This table is a catalog of the various data products that have been produced. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 7 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | gmh_global_data_product_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_data_product_id | SERIAL | Not allowed | |
| gmhsdp_standard_data_product_id (FK) | INT8 | Not allowed | |
| gmhusr_user_id (FK) | INT8 | Not allowed | |
| gmhdat_creation_datetime | DATETIME YEAR TO | Not allowed | |
| gmhdat_creation_sql | TEXT | Not allowed | |
| gmhdat_processing_desc | VARCHAR(255) | Allowed | |
| gmhdat_data_product | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_standard_data_products_gmh_data_products_FK1 | gmhsdp_standard_data_product_id | gmh_standard_data_products.gmhsdp_standard_data_product_id |
| gmh_users_gmh_data_products_FK1 | gmhusr_user_id | gmh_users.gmhusr_user_id |
| gmh_data_products_gmh_calibrated_measurements_usage_FK1 | gmh_calibrated_measurements_usage.gmh_global_data_product_id | gmh_global_data_product_id |
| gmh_data_products_gmh_user_annotations_FK1 | gmh_user_annotations.gmh_global_data_product_id | gmh_global_data_product_id |

## Column details

1. gmh_global_data_product_id

| | |
|---|---|
| Physical name: | gmh_global_data_product_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the data product. |

2. gmhsdp_standard_data_product_id (FK)

| | |
|---|---|
| Physical name: | gmhsdp_standard_data_product_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |

3. gmhusr_user_id (FK)

| | |
|---|---|
| Physical name: | gmhusr_user_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique alphanumeric ID for the GMH user who requested this data product. Not necessarily the same as the PostgreSQL userid that gets used. |

4. gmhdat_creation_datetime

| | |
|---|---|
| Physical name: | gmhdat_creation_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |
| Notes: | When was the data product created? |

5. gmhdat_creation_sql

| | |
|---|---|
| Physical name: | gmhdat_creation_sql |
| Physical data type: | TEXT |
| Allow NULLs: | Not allowed |
| Notes: | What SQL was used to create this data product? |

6. gmhdat_processing_desc

| | |
|---|---|
| Physical name: | gmhdat_processing_desc |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Allowed |
| Notes: | Other information relevant to the processing (e.g., name of external program used). |

7. gmhdat_data_product

| | |
|---|---|
| Physical name: | gmhdat_data_product |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | The data product itself. |

## Foreign key details (child)

gmh_standard_data_products_gmh_data_products_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsdp_standard_data_product_id | gmh_standard_data_products.gmhsdp_standard_data_product_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Verb phrase: | based on standard product | |
| Physical name: | gmh_standard_data_products_gmh_data_products_FK1 | |
| Notes: | Each data product corresponds to a single standard data product type (e.g., "time series of N-minute averages of calibrated data at each of N ordinates") | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_users_gmh_data_products_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhusr_user_id | gmh_users.gmhusr_user_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Verb phrase: | creates product | |
| Physical name: | gmh_users_gmh_data_products_FK1 | |
| Notes: | Each data product was requested / produced by one user. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

## Foreign key details (parent)

gmh_data_products_gmh_calibrated_measurements_usage_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_calibrated_measurements_usage.gmh_global_data_product_id | gmh_global_data_product_id |
| Relationship type: | Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |

| | |
|---|---|
| Verb phrase: | based on calibrated measurement |
| Physical name: | gmh_data_products_gmh_calibrated_measurements_usage_FK1 |
| Notes: | A data product is formed from one or more calibrated measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_data_products_gmh_user_annotations_FK1

| | | | |
|---|---|---|---|
| Definition: | Child | Parent | |
| | gmh_user_annotations.gmh_global_data_product_id | gmh_global_data_product_id | |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | annotates a product |
| Physical name: | gmh_data_products_gmh_user_annotations_FK1 |
| Notes: | Each data product may be annotated zero or more times by one or more users. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_deep_tow_measurements

Physical name: gmh_deep_tow_measurements
Notes: Table of "Deep Tow" measurements.
Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?
Reference: Higley-1-23c.

Owner:
Target DB name:
Number of columns: 5
Number of indexes: 1
Number of foreign keys: 2
Primary key: gmh_global_raw_meas_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK,U1) | INT8 | Not allowed | |
| gmhstl_shot_id (FK,U1) | INT8 | Not allowed | |
| gmhdtm_deep_tow_meas | BYTE | Not allowed | |
| DEEP_TOW_MEAS_METADATA | INT8 | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_deep_tow_measurements_AK1 (U1) | gmhstl_shot_id | Ascending |
| | gmhsen_sensor_id | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_deep_tow_measurements_FK1 | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |
| gmh_sensors_gmh_deep_tow_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmh_global_raw_meas_id
Physical name: gmh_global_raw_meas_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers.

2. gmhsen_sensor_id (FK,U1)
Physical name: gmhsen_sensor_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of the deep tow hydrophone.

3. gmhstl_shot_id (FK,U1)
Physical name: gmhstl_shot_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of shot (pop) associated with this measurement.

4. gmhdtm_deep_tow_meas
Physical name: gmhdtm_deep_tow_meas
Physical data type: BYTE
Allow NULLs: Not allowed
Notes: Blob of measurement data. Higley-1-17 and Higley-1-18 imply that there is one file per shot (pop).

Format yet to be determined, but likely SEG-Y if 24 bit or P16 if 16 bit (Higley 1-21c).

5. DEEP_TOW_MEAS_METADATA

| | |
|---|---|
| Physical name: | DEEP_TOW_MEAS_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

Index details

gmh_deep_tow_measurements_AK1

| | |
|---|---|
| Column(s): | gmhstl_shot_id (Asc)<br>gmhsen_sensor_id (Asc) |
| Unique: | Yes |

Foreign key details (child)

gmh_shot_log_gmh_deep_tow_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_shot_log_gmh_deep_tow_measurements_FK1 |
| Notes: | Each deep tow measurement is for a single shot. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_deep_tow_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | deep tow array |
| Physical name: | gmh_sensors_gmh_deep_tow_measurements_FK1 |
| Notes: | Each deep tow array produces 1 or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_fluorometer_measurements

| | |
|---|---|
| Physical name: | gmh_fluorometer_measurements |
| Notes: | Fluorometers will be stored on the BBLA (Farr-1-3). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK) | INT8 | Not allowed | |
| gmhflm_fluorometer_meas | FLOAT | Not allowed | |
| gmhflm_fluorometer_meas_datetime | DATETIME YEAR TO | Not allowed | |
| FLUOROMETER_MEAS_METADATA | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_fluorometer_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

## Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmhsen_sensor_id (FK)

| | |
|---|---|
| Physical name: | gmhsen_sensor_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of the fluorometer that made this measurement. |

3. gmhflm_fluorometer_meas

| | |
|---|---|
| Physical name: | gmhflm_fluorometer_meas |
| Physical data type: | FLOAT |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself (voltage between 1 and 5 in the prototype). |

4. gmhflm_fluorometer_meas_datetime

| | |
|---|---|
| Physical name: | gmhflm_fluorometer_meas_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. FLUOROMETER_MEAS_METADATA

| | |
|---|---|
| Physical name: | FLUOROMETER_MEAS_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other |

parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

## Foreign key details (child)

gmh_sensors_gmh_fluorometer_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_fluorometer_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_hydrophone_measurements

| | |
|---|---|
| Physical name: | gmh_hydrophone_measurements |
| Notes: | Table of hydrophone measurements.<br>Reference: Higley-1-23a.<br><br>Sampling regime is 16 channels for 1-4 seconds (assume 4) at 10 kHz of 16 bit data continuously for 48-96 hours. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 11 |
| Number of indexes: | 1 |
| Number of foreign keys: | 2 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK,U1) | INT8 | Not allowed | |
| gmhstl_shot_id (FK,U1) | INT8 | Not allowed | |
| gmhhpm_hydrophone_meas_starttime | DATETIME YEAR TO | Not allowed | |
| gmhhpm_hydrophone_meas_endtime | DATETIME YEAR TO | Not allowed | |
| gmhhpm_hydrophone_meas_nummeas | INT8 | Not allowed | |
| gmhhpm_hydrophone_meas_maxamp | INT8 | Allowed | |
| gmhhpm_hydrophone_meas_minamp | INT8 | Allowed | |
| gmhhpm_hydrophone_meas | BYTE | Not allowed | |
| gmhhpm_hydrophone_meas_p16filename | VARCHAR(100) | Allowed | |
| HYDROPHONE_MEAS_METADATA | INT8 | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_hydrophone_measurements_AK1 (U1) | gmhstl_shot_id | Ascending |
| | gmhsen_sensor_id | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_hydrophone_measurements_FK1 | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |
| gmh_sensors_gmh_hydrophone_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

## Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmhsen_sensor_id (FK,U1)

| | |
|---|---|
| Physical name: | gmhsen_sensor_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of the hydrophone sensor. |

3. gmhstl_shot_id (FK,U1)

Physical name: gmhstl_shot_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of shot (pop) associated with this measurement.

4. gmhhpm_hydrophone_meas_starttime
Physical name: gmhhpm_hydrophone_meas_starttime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Not allowed
Notes: Start time for the measurement

5. gmhhpm_hydrophone_meas_endtime
Physical name: gmhhpm_hydrophone_meas_endtime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Not allowed
Notes: End time for the measurement

6. gmhhpm_hydrophone_meas_nummeas
Physical name: gmhhpm_hydrophone_meas_nummeas
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Number of measurements in gmhhpm_hydrophone_meas

7. gmhhpm_hydrophone_meas_maxamp
Physical name: gmhhpm_hydrophone_meas_maxamp
Physical data type: INT8
Allow NULLs: Allowed
Notes: Actual maximum amplitude value stored in measurement.

8. gmhhpm_hydrophone_meas_minamp
Physical name: gmhhpm_hydrophone_meas_minamp
Physical data type: INT8
Allow NULLs: Allowed
Notes: Actual minimum amplitude value stored in measurement.

9. gmhhpm_hydrophone_meas
Physical name: gmhhpm_hydrophone_meas
Physical data type: BYTE
Allow NULLs: Not allowed
Notes: Blob of measurement data. Higley-1-17 and Higley-1-18 imply that there is one file per shot (pop).

10. gmhhpm_hydrophone_meas_p16filename
Physical name: gmhhpm_hydrophone_meas_p16filename
Physical data type: VARCHAR(100)
Allow NULLs: Allowed

11. HYDROPHONE_MEAS_METADATA
Physical name: HYDROPHONE_MEAS_METADATA
Physical data type: INT8
Allow NULLs: Not allowed
Notes: This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

## Index details

gmh_hydrophone_measurements_AK1

| | |
|---|---|
| Column(s): | gmhstl_shot_id (Asc)<br>gmhsen_sensor_id (Asc) |
| Unique: | Yes |

## Foreign key details (child)

gmh_shot_log_gmh_hydrophone_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_shot_log_gmh_hydrophone_measurements_FK1 |
| Notes: | Each hydrophone measurement is for a single shot. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_hydrophone_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | hydrophone |
| Physical name: | gmh_sensors_gmh_hydrophone_measurements_FK1 |
| Notes: | Each hydrophone produces one or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_images

| | |
|---|---|
| Physical name: | gmh_images |
| Notes: | The chimney array will have 3 cameras (Farr-1-4). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK) | INT8 | Not allowed | |
| gmhimg_image | BYTE | Not allowed | |
| gmhimg_image_datetime | DATETIME YEAR TO | Not allowed | |
| IMAGE_METADATA | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_images_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the image. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmhsen_sensor_id (FK)

| | |
|---|---|
| Physical name: | gmhsen_sensor_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of the camera that made this measurement. |

3. gmhimg_image

| | |
|---|---|
| Physical name: | gmhimg_image |
| Physical data type: | BYTE |
| Allow NULLs: | Not allowed |
| Notes: | The image or video itself. |

4. gmhimg_image_datetime

| | |
|---|---|
| Physical name: | gmhimg_image_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the image/video was taken. |

5. IMAGE_METADATA

| | |
|---|---|
| Physical name: | IMAGE_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the image. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored image, the time when it was taken, the size of the image, and any other parameters captured by the camera (other than the image itself). Descriptive metadata is information that can be used in |

discovery and identification (i.e., answering the questions "Find all iamges that ..." and "What is this image?").

## Foreign key details (child)

gmh_sensors_gmh_images_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Physical name: | gmh_sensors_gmh_images_FK1 | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

# gmh_mass_spec_measurements

| | |
|---|---|
| Physical name: | gmh_mass_spec_measurements |
| Notes: | The chimney array may have a mass spectrometer (Farr-1-15). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK) | INT8 | Not allowed | |
| gmhmsm_mass_spec_meas | BYTE | Not allowed | |
| gmhmsm_mass_spec_meas_datetime | DATETIME YEAR TO | Not allowed | |
| MASS_SPEC_MEAS_METADATA | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_mass_spec_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

## Column details

### 1. gmh_global_raw_meas_id

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

### 2. gmhsen_sensor_id (FK)

| | |
|---|---|
| Physical name: | gmhsen_sensor_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of the mass spectrometer that made this measurement. |

### 3. gmhmsm_mass_spec_meas

| | |
|---|---|
| Physical name: | gmhmsm_mass_spec_meas |
| Physical data type: | BYTE |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

### 4. gmhmsm_mass_spec_meas_datetime

| | |
|---|---|
| Physical name: | gmhmsm_mass_spec_meas_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

### 5. MASS_SPEC_MEAS_METADATA

| | |
|---|---|
| Physical name: | MASS_SPEC_MEAS_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other |

parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

## Foreign key details (child)

gmh_sensors_gmh_mass_spec_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_mass_spec_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_mets_measurements

| | |
|---|---|
| Physical name: | gmh_mets_measurements |
| Notes: | The chimney array and the BBLA will both have methane sensors (Farr-1-3 and Farr-1-4). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK) | INT8 | Not allowed | |
| gmhmtm_mets_meas | BYTE | Not allowed | |
| gmhmtm_mets_meas_datetime | DATETIME YEAR TO | Not allowed | |
| METS_MEAS_METADATA | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_mets_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmhsen_sensor_id (FK)

| | |
|---|---|
| Physical name: | gmhsen_sensor_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of the mets sensor that made this measurement. |

3. gmhmtm_mets_meas

| | |
|---|---|
| Physical name: | gmhmtm_mets_meas |
| Physical data type: | BYTE |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

4. gmhmtm_mets_meas_datetime

| | |
|---|---|
| Physical name: | gmhmtm_mets_meas_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. METS_MEAS_METADATA

| | |
|---|---|
| Physical name: | METS_MEAS_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). |

Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

## Foreign key details (child)

gmh_sensors_gmh_mets_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_mets_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_orientation_sensor_measurements

| | |
|---|---|
| Physical name: | gmh_orientation_sensor_measurements |
| Notes: | Table of orientation sensor measurements.<br>Q: WILL THESE ALL BE TIED TO THE SHOT NUMBER? OR DO WE HAVE TO STORE A TIME VALUE FOR THESE AS WELL?<br>Reference: Higley-1-23d<br><br>There are 2 sets of 2 orientation sensors measuring heading, pitch, and roll.<br><br>Sampling regime is 4 Hz for 1-4 seconds.<br><br>Higley-1-23e says the orientation sensor works as a current meter and may be activated once per hour throughout the year. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 6 |
| Number of indexes: | 1 |
| Number of foreign keys: | 2 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK,U1) | INT8 | Not allowed | |
| gmhstl_shot_id (FK,U1) | INT8 | Allowed | |
| gmhosm_orientation_sensor_meas_datetime | DATETIME YEAR TO | Allowed | |
| gmhosm_orientation_sensor_meas | BYTE | Not allowed | |
| ORIENTATION_SENSOR_MEAS_METADATA | INT8 | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_orientation_sensor_measurements_AK1 (U1) | gmhstl_shot_id | Ascending |
| | gmhsen_sensor_id | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_orientation_sensor_measurements_FK1 | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |
| gmh_sensors_gmh_orientation_sensor_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmhsen_sensor_id (FK,U1)

| | |
|---|---|
| Physical name: | gmhsen_sensor_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of lthe orientation sensor. |

3. gmhstl_shot_id (FK,U1)

| | |
|---|---|
| Physical name: | gmhstl_shot_id |
| Physical data type: | INT8 |

| | |
|---|---|
| Allow NULLs: | Allowed |
| Notes: | ID of shot (pop) associated with this measurement. Some measurements will not be associated with a pop (Higley-1-23e). |

4. gmhosm_orientation_sensor_meas_datetime

| | |
|---|---|
| Physical name: | gmhosm_orientation_sensor_meas_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Allowed |
| Notes: | The date and time of the measurement. Higley-1-23e says that the orientation sensor works as a current meter and may be activated once per hour throughout the year. So some measurements will not be tied to a shot number and will need the measurement date and time stored explicitly. |

5. gmhosm_orientation_sensor_meas

| | |
|---|---|
| Physical name: | gmhosm_orientation_sensor_meas |
| Physical data type: | BYTE |
| Allow NULLs: | Not allowed |
| Notes: | Blob of measurement data. Higley-1-17 and Higley-1-18 imply that there is one file per shot (pop). |

6. ORIENTATION_SENSOR_MEAS_METADATA

| | |
|---|---|
| Physical name: | ORIENTATION_SENSOR_MEAS_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

## Index details

gmh_orientation_sensor_measurements_AK1

| | |
|---|---|
| Column(s): | gmhstl_shot_id (Asc)<br>gmhsen_sensor_id (Asc) |
| Unique: | Yes |

## Foreign key details (child)

gmh_shot_log_gmh_orientation_sensor_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmhstl_shot_id | gmh_shot_log.gmhstl_shot_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_shot_log_gmh_orientation_sensor_measurements_FK1 |
| Notes: | Each orientation sensor measurement may be for a single shot (but may not correspond to a shot at all). |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_orientation_sensor_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |

| | | |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | orientation sensor |
| Physical name: | gmh_sensors_gmh_orientation_sensor_measurements_FK1 |
| Notes: | Each orientation sensor produces 1 or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_oxygen_sensor_measurements

| | |
|---|---|
| Physical name: | gmh_oxygen_sensor_measurements |
| Notes: | The BBLA will have oxygen sensors (Farr-1-3). |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 1 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK) | INT8 | Not allowed | |
| gmho2m_oxygen_sensor_meas | BYTE | Not allowed | |
| gmho2m_oxygen_sensor_meas_datetime | DATETIME YEAR TO | Not allowed | |
| OXYGEN_MEAS_METADATA | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_oxygen_sensor_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmhsen_sensor_id (FK)

| | |
|---|---|
| Physical name: | gmhsen_sensor_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of the oxygen sensor that made this measurement. |

3. gmho2m_oxygen_sensor_meas

| | |
|---|---|
| Physical name: | gmho2m_oxygen_sensor_meas |
| Physical data type: | BYTE |
| Allow NULLs: | Not allowed |
| Notes: | The measurement itself. |

4. gmho2m_oxygen_sensor_meas_datetime

| | |
|---|---|
| Physical name: | gmho2m_oxygen_sensor_meas_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the measurement was taken. |

5. OXYGEN_MEAS_METADATA

| | |
|---|---|
| Physical name: | OXYGEN_MEAS_METADATA |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other |

parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

## Foreign key details (child)

gmh_sensors_gmh_oxygen_sensor_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_oxygen_sensor_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_raw_measurement_ids

| | |
|---|---|
| Physical name: | gmh_raw_measurement_ids |
| Owner: | |
| Target DB name: | |
| Number of columns: | 3 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmh_global_raw_meas_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | SERIAL | Not allowed | |
| gmhrmi_raw_measurement_type_code | CHAR(4) | Not allowed | |
| gmhrmi_raw_measurement_datetime | DATETIME YEAR TO | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_raw_measurement_ids_gmh_calibrated_measurements_FK1 | gmh_calibrated_measurements.gmh_global_raw_meas_id | gmh_global_raw_meas_id |
| gmh_raw_measurement_ids_gmh_user_annotations_FK1 | gmh_user_annotations.gmh_global_raw_meas_id | gmh_global_raw_meas_id |

## Column details

1. gmh_global_raw_meas_id

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers. |

2. gmhrmi_raw_measurement_type_code

| | |
|---|---|
| Physical name: | gmhrmi_raw_measurement_type_code |
| Physical data type: | CHAR(4) |
| Allow NULLs: | Not allowed |
| Notes: | See the column descriptions in the appropriate table on diagrams page 3 (Seismic measurements) or page 4 (Environmental measurements). |

3. gmhrmi_raw_measurement_datetime

| | |
|---|---|
| Physical name: | gmhrmi_raw_measurement_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |

## Foreign key details (parent)

gmh_raw_measurement_ids_gmh_calibrated_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmh_calibrated_measurements.gmh_global_raw_meas_id | gmh_global_raw_meas_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Verb phrase: | produced from | |
| Physical name: | gmh_raw_measurement_ids_gmh_calibrated_measurements_FK1 | |
| Notes: | Each raw measurement or image (recorded in one of the gmh_*_measurements or the gmh_images table) can have 0 or more corresponding post-deployment calibrated | |

| | |
|---|---|
| | measurements / images. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_raw_measurement_ids_gmh_user_annotations_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_user_annotations.gmh_global_raw_meas_id | gmh_global_raw_meas_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | annotates a raw measurement |
| Physical name: | gmh_raw_measurement_ids_gmh_user_annotations_FK1 |
| Notes: | Each raw measurement may be annotated zero or more times by one or more users. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_roles

| | |
|---|---|
| Physical name: | gmh_roles |
| Notes: | This table describes the various database roles that GMH users can have. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 4 |
| Number of indexes: | 1 |
| Number of foreign keys: | 0 |
| Primary key: | gmhrol_role_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhrol_role_id | SERIAL | Not allowed | |
| gmhrol_role_name (U1) | VARCHAR(32) | Not allowed | |
| gmhrol_role_description | VARCHAR(255) | Not allowed | |
| gmhrol_database_role_id | VARCHAR(64) | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmhrol_role_name_idx (U1) | gmhrol_role_name | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_roles_gmh_user_roles_FK1 | gmh_user_roles.gmhrol_role_id | gmhrol_role_id |
| gmh_roles_gmh_data_product_permissions_FK1 | gmh_data_product_permissions. gmhrol_role_id | gmhrol_role_id |

## Column details

1. gmhrol_role_id

| | |
|---|---|
| Physical name: | gmhrol_role_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique id for role. |

2. gmhrol_role_name (U1)

| | |
|---|---|
| Physical name: | gmhrol_role_name |
| Physical data type: | VARCHAR(32) |
| Allow NULLs: | Not allowed |
| Notes: | Alphanumeric role name |

3. gmhrol_role_description

| | |
|---|---|
| Physical name: | gmhrol_role_description |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Not allowed |
| Notes: | Description of the role - what basic access rights does it entail? |

4. gmhrol_database_role_id

| | |
|---|---|
| Physical name: | gmhrol_database_role_id |
| Physical data type: | VARCHAR(64) |
| Allow NULLs: | Not allowed |
| Notes: | What is the corresponding database-level role? (i.e., the role according to the underlying PostgreSQL database). |

## Index details

gmhrol_role_name_idx

| | |
|---|---|
| Column(s): | gmhrol_role_name (Asc) |
| Unique: | Yes |

## Foreign key details (parent)

gmh_roles_gmh_user_roles_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_user_roles.gmhrol_role_id | gmhrol_role_id |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | roles are held by users |
| Physical name: | gmh_roles_gmh_user_roles_FK1 |
| Notes: | Each role can be taken on by zero or more GMH users |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_roles_gmh_data_product_permissions_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_data_product_permissions.gmhrol_role_id | gmhrol_role_id |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_roles_gmh_data_product_permissions_FK1 |
| Notes: | Each role has an associated list of standard data products that users with that role can produce. In order to produce a particular data product, that data product must appear in the list associated with one of the user's roles. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_sensor_arrays

| | |
|---|---|
| Physical name: | gmh_sensor_arrays |
| Owner: | |
| Target DB name: | |
| Number of columns: | 4 |
| Number of indexes: | 1 |
| Number of foreign keys: | 0 |
| Primary key: | gmharr_sensor_array_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmharr_sensor_array_id | SERIAL | Not allowed | |
| gmharr_sensor_array_code (U1) | CHAR(10) | Not allowed | |
| gmharr_sensor_array_desc | VARCHAR(255) | Not allowed | |
| ARRAY_INFO | VARCHAR(100) | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmharr_sensor_array_code_idx (U1) | gmharr_sensor_array_code | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensor_arrays_gmh_sensors_FK1 | gmh_sensors.gmharr_sensor_array_id | gmharr_sensor_array_id |

## Column details

1. gmharr_sensor_array_id

| | |
|---|---|
| Physical name: | gmharr_sensor_array_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the array. Probably a generated key. |

2. gmharr_sensor_array_code (U1)

| | |
|---|---|
| Physical name: | gmharr_sensor_array_code |
| Physical data type: | CHAR(10) |
| Allow NULLs: | Not allowed |

3. gmharr_sensor_array_desc

| | |
|---|---|
| Physical name: | gmharr_sensor_array_desc |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Not allowed |
| Notes: | A one line description of the array. |

4. ARRAY_INFO

| | |
|---|---|
| Physical name: | ARRAY_INFO |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of one or more columns holding other parameters needed to describe this array of sensors and any other relevant metadata. |

## Index details

gmharr_sensor_array_code_idx

| | |
|---|---|
| Column(s): | gmharr_sensor_array_code (Asc) |
| Unique: | Yes |

## Foreign key details (parent)

gmh_sensor_arrays_gmh_sensors_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_sensors.gmharr_sensor_array_id | gmharr_sensor_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | logical array is made up of sensors |
| Physical name: | gmh_sensor_arrays_gmh_sensors_FK1 |
| Notes: | A sensor may be part of a logical array, |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_sensor_calibration_history

Physical name: gmh_sensor_calibration_history
Notes: This table contains information on how a particular sensor has been calibrated, either at present (gmhscl_hist_end_datetime is NULL) or in the past (gmhscl_hist_end_datetime is not NULL).
Owner:
Target DB name:
Number of columns: 5
Number of indexes: 0
Number of foreign keys: 1
Primary key: gmhscl_sensor_calibration_hist_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhscl_sensor_calibration_hist_id | SERIAL | Not allowed | |
| gmhsen_sensor_id (FK) | INT8 | Not allowed | |
| gmhscl_hist_start_datetime | DATETIME YEAR TO | Not allowed | |
| gmhscl_hist_end_datetime | DATETIME YEAR TO | Allowed | |
| SENSOR_CALIBRATION_PARAMETERS | VARCHAR(100) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_sensor_calibration_history_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmhscl_sensor_calibration_hist_id
Physical name: gmhscl_sensor_calibration_hist_id
Physical data type: SERIAL
Allow NULLs: Not allowed
Notes: Unique id identifying the sensor calibration history record. Probably a generated key.

2. gmhsen_sensor_id (FK)
Physical name: gmhsen_sensor_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Unique id identifying the individual sensor.

3. gmhscl_hist_start_datetime
Physical name: gmhscl_hist_start_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Not allowed
Notes: Date and time when this record became effective.

4. gmhscl_hist_end_datetime
Physical name: gmhscl_hist_end_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Allowed
Notes: Date and time when this record ceased to be effective.

5. SENSOR_CALIBRATION_PARAMETERS
Physical name: SENSOR_CALIBRATION_PARAMETERS
Physical data type: VARCHAR(100)

| | |
|---|---|
| Allow NULLs: | Not allowed |
| Notes: | This will be replaced by a set of columns, one column for each calibration parameter, holding the value of that parameter.<br><br>Note that each type of sensor will have different calibration parameters, so either:<br><br>1. The parameters will be stored in subtables, with one subtable for each type of sensor, or<br><br>2. The parameters will be stored in a single table, with the parameters that are not applicable to a particular sensor simply being left blank.<br><br>Option 1 is cleaner, but harder to maintain. It is much easier with Option 2 to add parameters for new types of sensors, or add additional parameters for existing types of sensors. |

**Foreign key details (child)**

gmh_sensors_gmh_sensor_calibration_history_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_sensor_calibration_history_FK1 |
| Notes: | Each sensor has a calibration history. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_sensor_configuration_history

Physical name: gmh_sensor_configuration_history
Notes: This table contains information on how a particular sensor has been configured, either at present (gmhscn_hist_end_datetime is NULL) or in the past (gmhscn_hist_end_datetime is not NULL).
Owner:
Target DB name:
Number of columns: 5
Number of indexes: 0
Number of foreign keys: 1
Primary key: gmhscn_sensor_config_hist_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhscn_sensor_config_hist_id | SERIAL | Not allowed | |
| gmhsen_sensor_id (FK) | INT8 | Not allowed | |
| gmhscn_hist_start_datetime | DATETIME YEAR TO | Not allowed | |
| gmhscn_hist_end_datetime | DATETIME YEAR TO | Allowed | |
| SENSOR_CONFIG_PARAMETERS | VARCHAR(100) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_sensor_configuration_history_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmhscn_sensor_config_hist_id
Physical name: gmhscn_sensor_config_hist_id
Physical data type: SERIAL
Allow NULLs: Not allowed
Notes: Unique id identifying the sensor configuration history record. Probably a generated key.

2. gmhsen_sensor_id (FK)
Physical name: gmhsen_sensor_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Unique id identifying the individual sensor.

3. gmhscn_hist_start_datetime
Physical name: gmhscn_hist_start_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Not allowed
Notes: Date and time when this record became effective.

4. gmhscn_hist_end_datetime
Physical name: gmhscn_hist_end_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Allowed
Notes: Date and time when this record ceased to be effective.

5. SENSOR_CONFIG_PARAMETERS
Physical name: SENSOR_CONFIG_PARAMETERS
Physical data type: VARCHAR(100)

| | |
|---|---|
| Allow NULLs: | Not allowed |
| Notes: | This will be replaced by a set of columns, one column for each configuration parameter, holding the value of that parameter.<br><br>Note that each type of sensor will have different configuration parameters, so either:<br><br>1. The parameters will be stored in subtables, with one subtable for each type of sensor, or<br><br>2. The parameters will be stored in a single table, with the parameters that are not applicable to a particular sensor simply being left blank.<br><br>Option 1 is cleaner, but harder to maintain. It is much easier with Option 2 to add parameters for new types of sensors, or add additional parameters for existing types of sensors. |

**Foreign key details (child)**

gmh_sensors_gmh_sensor_configuration_history_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_sensor_configuration_history_FK1 |
| Notes: | Each sensor has a configuration history. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_sensor_types

| | |
|---|---|
| Physical name: | gmh_sensor_types |
| Notes: | This table contains all the relevant information related to specific sensor types, e.g., name, type of sensor (hydrophone, ADCP, thermistor, etc), generic properties, etc. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 4 |
| Number of indexes: | 1 |
| Number of foreign keys: | 0 |
| Primary key: | gmhstp_sensor_type_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhstp_sensor_type_id | SERIAL | Not allowed | |
| gmhstp_sensor_type_code (U1) | CHAR(10) | Not allowed | |
| gmhstp_sensor_type_desc | VARCHAR(255) | Not allowed | |
| SENSOR_TYPE_INFO | VARCHAR(100) | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmhstp_sensor_type_code_idx (U1) | gmhstp_sensor_type_code | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensor_types_gmh_sensors_FK1 | gmh_sensors.gmhstp_sensor_type_id | gmhstp_sensor_type_id |

## Column details

1. gmhstp_sensor_type_id

| | |
|---|---|
| Physical name: | gmhstp_sensor_type_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the sensor type (class). Probably a generated key. |

2. gmhstp_sensor_type_code (U1)

| | |
|---|---|
| Physical name: | gmhstp_sensor_type_code |
| Physical data type: | CHAR(10) |
| Allow NULLs: | Not allowed |
| Notes: | A more mnemonic code identifying the type of sensor. |

3. gmhstp_sensor_type_desc

| | |
|---|---|
| Physical name: | gmhstp_sensor_type_desc |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Not allowed |
| Notes: | A one line description of the type of sensor - e.g. make, model, manufacturer. |

4. SENSOR_TYPE_INFO

| | |
|---|---|
| Physical name: | SENSOR_TYPE_INFO |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of one or more columns holding any other parameters needed to describe this type of sensor. |

## Index details

gmhstp_sensor_type_code_idx

| | |
|---|---|
| Column(s): | gmhstp_sensor_type_code (Asc) |
| Unique: | Yes |

## Foreign key details (parent)

gmh_sensor_types_gmh_sensors_FK1

| | Child | Parent |
|---|---|---|
| Definition: | gmh_sensors.gmhstp_sensor_type_id | gmhstp_sensor_type_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | sensors have types |
| Physical name: | gmh_sensor_types_gmh_sensors_FK1 |
| Notes: | Each sensor has a particular sensor type, or class of sensors. Fields that describe the class as a whole are stored in the gmh_sensor_types table; fields that are specific to an individual sensor are described in the gmh_sensors table. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_sensors

| | |
|---|---|
| Physical name: | gmh_sensors |
| Notes: | In this context a sensor is either an instrument or an individual sensor on a multi-sensor instrument.<br><br>This table contains all the relevant information related to a specific sensor, e.g., what logical array is it part of? what type of sensor is it? where is it? etc. Note that sensor measurement values are stored in other tables. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 8 |
| Number of indexes: | 1 |
| Number of foreign keys: | 3 |
| Primary key: | gmhsen_sensor_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhsen_sensor_id | SERIAL | Not allowed | |
| gmhcbl_cable_array_id (FK) | INT8 | Not allowed | |
| gmharr_sensor_array_id (FK) | INT8 | Allowed | |
| gmhsen_sensor_name (U1) | VARCHAR(100) | Not allowed | |
| gmhstp_sensor_type_id (FK) | INT8 | Not allowed | |
| gmhsen_position_on_array | FLOAT | Allowed | |
| gmhsen_absolute_position | BYTE | Allowed | |
| STATIC_SENSOR_INFO | VARCHAR(100) | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmhsen_sensor_name_idx (U1) | gmhsen_sensor_name | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensor_types_gmh_sensors_FK1 | gmhstp_sensor_type_id | gmh_sensor_types.gmhstp_sensor_type_id |
| gmh_sensor_arrays_gmh_sensors_FK1 | gmharr_sensor_array_id | gmh_sensor_arrays.gmharr_sensor_array_id |
| gmh_cable_arrays_gmh_sensors_FK1 | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |
| gmh_sensors_gmh_sensor_configuration_history_FK1 | gmh_sensor_configuration_history.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_sensor_calibration_history_FK1 | gmh_sensor_calibration_history.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_3daccel_measurements_FK1 | gmh_3daccel_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_orientation_sensor_measurements_FK1 | gmh_orientation_sensor_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_hydrophone_measurements_FK1 | gmh_hydrophone_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_deep_tow_measurements_FK1 | gmh_deep_tow_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_auv_measurements_FK1 | gmh_auv_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_sled_measurements_FK1 | gmh_sled_measurements.gmhsen_orientation_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_sled_measurements_FK2 | gmh_sled_measurements.gmhsen_accelerometer_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_sled_measurements_FK3 | gmh_sled_measurements.gmhsen_hydrophone_sensor_id | gmhsen_sensor_id |

| | | |
|---|---|---|
| gmh_sensors_gmh_thermistor_measurements_FK1 | gmh_thermistor_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_fluorometer_measurements_FK1 | gmh_fluorometer_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_adcp_measurements_FK1 | gmh_adcp_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_mets_measurements_FK1 | gmh_mets_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_mass_spec_measurements_FK1 | gmh_mass_spec_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_images_FK1 | gmh_images.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_oxygen_sensor_measurements_FK1 | gmh_oxygen_sensor_measurements.gmhsen_sensor_id | gmhsen_sensor_id |
| gmh_sensors_gmh_ctd_measurements_FK1 | gmh_ctd_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

Column details

1. gmhsen_sensor_id

| | |
|---|---|
| Physical name: | gmhsen_sensor_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the individual sensor. Probably a generated key. |

2. gmhcbl_cable_array_id (FK)

| | |
|---|---|
| Physical name: | gmhcbl_cable_array_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID of the cable array upon which this sensor resides. |

3. gmharr_sensor_array_id (FK)

| | |
|---|---|
| Physical name: | gmharr_sensor_array_id |
| Physical data type: | INT8 |
| Allow NULLs: | Allowed |
| Notes: | ID of the sensor array to which this sensor belongs, if any. |

4. gmhsen_sensor_name (U1)

| | |
|---|---|
| Physical name: | gmhsen_sensor_name |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | Name of the sensor. |

5. gmhstp_sensor_type_id (FK)

| | |
|---|---|
| Physical name: | gmhstp_sensor_type_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | ID identifying the class of sensor to which this sensor belongs. |

6. gmhsen_position_on_array

| | |
|---|---|
| Physical name: | gmhsen_position_on_array |
| Physical data type: | FLOAT |
| Allow NULLs: | Allowed |
| Notes: | Measuring from a designated end of the array, where is this sensor located? |

7. gmhsen_absolute_position

| | |
|---|---|
| Physical name: | gmhsen_absolute_position |
| Physical data type: | BYTE |
| Allow NULLs: | Allowed |
| Notes: | What is the absolute location of this sensor? (geographic coordinates and depth). |

8. STATIC_SENSOR_INFO

| | |
|---|---|
| Physical name: | STATIC_SENSOR_INFO |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of one or more columns holding other relevant information about the sensor, other than time-dependent properties like configuration, calibration, and measurements, which are stored in other tables. |

## Index details

gmhsen_sensor_name_idx

| | |
|---|---|
| Column(s): | gmhsen_sensor_name (Asc) |
| Unique: | Yes |

## Foreign key details (child)

gmh_sensor_types_gmh_sensors_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhstp_sensor_type_id | gmh_sensor_types.gmhstp_sensor_type_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | sensors have types |
| Physical name: | gmh_sensor_types_gmh_sensors_FK1 |
| Notes: | Each sensor has a particular sensor type, or class of sensors. Fields that describe the class as a whole are stored in the gmh_sensor_types table; fields that are specific to an individual sensor are described in the gmh_sensors table. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensor_arrays_gmh_sensors_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmharr_sensor_array_id | gmh_sensor_arrays.gmharr_sensor_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | logical array is made up of sensors |
| Physical name: | gmh_sensor_arrays_gmh_sensors_FK1 |
| Notes: | A sensor may be part of a logical array, |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_cable_arrays_gmh_sensors_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhcbl_cable_array_id | gmh_cable_arrays.gmhcbl_cable_array_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | sensor situated on physical array |
| Physical name: | gmh_cable_arrays_gmh_sensors_FK1 |
| Notes: | Each sensor is located on a physical array |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

## Foreign key details (parent)

gmh_sensors_gmh_sensor_configuration_history_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmh_sensor_configuration_history.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_sensor_configuration_history_FK1 |
| Notes: | Each sensor has a configuration history. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_sensor_calibration_history_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmh_sensor_calibration_history.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_sensor_calibration_history_FK1 |
| Notes: | Each sensor has a calibration history. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_3daccel_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmh_3daccel_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | 3d accelerometer |
| Physical name: | gmh_sensors_gmh_3daccel_measurements_FK1 |
| Notes: | Each 3d accelerometer produces 1 or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_orientation_sensor_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmh_orientation_sensor_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | orientation sensor |
| Physical name: | gmh_sensors_gmh_orientation_sensor_measurements_FK1 |
| Notes: | Each orientation sensor produces 1 or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_hydrophone_measurements_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmh_hydrophone_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | hydrophone |

Physical name: gmh_sensors_gmh_hydrophone_measurements_FK1
Notes: Each hydrophone produces one or more measurements.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_sensors_gmh_deep_tow_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_deep_tow_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Verb phrase: deep tow array
Physical name: gmh_sensors_gmh_deep_tow_measurements_FK1
Notes: Each deep tow array produces 1 or more measurements.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_sensors_gmh_auv_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_auv_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Verb phrase: auv
Physical name: gmh_sensors_gmh_auv_measurements_FK1
Notes: Each auv sensor produces one or more measurements.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_sensors_gmh_sled_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_sled_measurements.gmhsen_orientation_sensor_id | gmhsen_sensor_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Verb phrase: sled orientation sensor
Physical name: gmh_sensors_gmh_sled_measurements_FK1
Notes: Each sled orientation sensor produces one or more measurements.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_sensors_gmh_sled_measurements_FK2

| Definition: | Child | Parent |
|---|---|---|
| | gmh_sled_measurements.gmhsen_accelerometer_sensor_id | gmhsen_sensor_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Verb phrase: sled accelerometer
Physical name: gmh_sensors_gmh_sled_measurements_FK2
Notes: Each sled accelerometer sensor produces one or more measurements.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_sensors_gmh_sled_measurements_FK3

| Definition: | Child | Parent |
|---|---|---|
| | gmh_sled_measurements.gmhsen_hydrophone_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | sled hydrophone |
| Physical name: | gmh_sensors_gmh_sled_measurements_FK3 |
| Notes: | Each sled hydrophone produces one or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_thermistor_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_thermistor_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | thermistor |
| Physical name: | gmh_sensors_gmh_thermistor_measurements_FK1 |
| Notes: | Each thermistor produces 1 or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_fluorometer_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_fluorometer_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_fluorometer_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_adcp_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_adcp_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_adcp_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_mets_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_mets_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_mets_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_mass_spec_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_mass_spec_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_mass_spec_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_images_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_images.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_images_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_oxygen_sensor_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_oxygen_sensor_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_oxygen_sensor_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_ctd_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_ctd_measurements.gmhsen_sensor_id | gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_sensors_gmh_ctd_measurements_FK1 |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_ship_log

| | |
|---|---|
| Physical name: | gmh_ship_log |
| Owner: | |
| Target DB name: | |
| Number of columns: | 1 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmhspl_log_entry_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhspl_log_entry_id | INT8 | Not allowed | |

Column details

1. gmhspl_log_entry_id

| | |
|---|---|
| Physical name: | gmhspl_log_entry_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |

# gmh_shot_log

| | |
|---|---|
| Physical name: | gmh_shot_log |
| Notes: | Database realization of the "shot log". Also called the "Ship Pop" file. See Higley-1-19 and Higley-1-21b. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 0 |
| Number of foreign keys: | 0 |
| Primary key: | gmhstl_shot_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhstl_shot_id | SERIAL | Not allowed | |
| gmhstl_shot_datetime | DATETIME YEAR TO | Not allowed | |
| gmhstl_shot_delay_time | DATETIME YEAR TO | Not allowed | |
| gmhstl_shot_position | BYTE | Not allowed | |
| SHOT_LOG_INFO | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_shot_log_gmh_hydrophone_measurements_FK1 | gmh_hydrophone_measurements.gmhstl_shot_id | gmhstl_shot_id |
| gmh_shot_log_gmh_3daccel_measurements_FK1 | gmh_3daccel_measurements.gmhstl_shot_id | gmhstl_shot_id |
| gmh_shot_log_gmh_orientation_sensor_measurements_FK1 | gmh_orientation_sensor_measurements.gmhstl_shot_id | gmhstl_shot_id |
| gmh_shot_log_gmh_deep_tow_measurements_FK1 | gmh_deep_tow_measurements.gmhstl_shot_id | gmhstl_shot_id |
| gmh_shot_log_gmh_auv_measurements_FK1 | gmh_auv_measurements.gmhstl_shot_id | gmhstl_shot_id |

## Column details

1. gmhstl_shot_id

| | |
|---|---|
| Physical name: | gmhstl_shot_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique id identifying the shot. This will likely be the value assigned at the site. |

2. gmhstl_shot_datetime

| | |
|---|---|
| Physical name: | gmhstl_shot_datetime |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |
| Notes: | Date and time when the shot was fired. See Higley-1-21b. |

3. gmhstl_shot_delay_time

| | |
|---|---|
| Physical name: | gmhstl_shot_delay_time |
| Physical data type: | DATETIME YEAR TO YEAR |
| Allow NULLs: | Not allowed |
| Notes: | Delay time for the shot. See Higley-1-21b. |

4. gmhstl_shot_position

| | |
|---|---|
| Physical name: | gmhstl_shot_position |

Physical data type: BYTE
Allow NULLs: Not allowed
Notes: Position of the shot when fired. See Higley-1-21b.

5. SHOT_LOG_INFO
Physical name: SHOT_LOG_INFO
Physical data type: INT8
Allow NULLs: Not allowed
Notes: This column will be replaced by a set of columns each representing one of the other fields in the shot log.

## Foreign key details (parent)

gmh_shot_log_gmh_hydrophone_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_hydrophone_measurements.gmhstl_shot_id | gmhstl_shot_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Physical name: gmh_shot_log_gmh_hydrophone_measurements_FK1
Notes: Each hydrophone measurement is for a single shot.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_shot_log_gmh_3daccel_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_3daccel_measurements.gmhstl_shot_id | gmhstl_shot_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Physical name: gmh_shot_log_gmh_3daccel_measurements_FK1
Notes: Each 3d accelerometer measurement is for a single shot.
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_shot_log_gmh_orientation_sensor_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_orientation_sensor_measurements.gmhstl_shot_id | gmhstl_shot_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Physical name: gmh_shot_log_gmh_orientation_sensor_measurements_FK1
Notes: Each orientation sensor measurement may be for a single shot (but may not correspond to a shot at all).
Ref. Integrity on update: No Action
Ref. Integrity on delete: No Action

gmh_shot_log_gmh_deep_tow_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_deep_tow_measurements.gmhstl_shot_id | gmhstl_shot_id |

Relationship type: Non-Identifying
Cardinality: One -to- Zero-or-More
Allow NULLs: Not allowed
Physical name: gmh_shot_log_gmh_deep_tow_measurements_FK1

| | |
|---|---|
| Notes: | Each deep tow measurement is for a single shot. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_shot_log_gmh_auv_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_auv_measurements.gmhstl_shot_id | gmhstl_shot_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Physical name: | gmh_shot_log_gmh_auv_measurements_FK1 |
| Notes: | Each AUV measurement is for a single shot. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_sled_configuration_history

Physical name: gmh_sled_configuration_history
Notes: This table contains information on how the sled system has been configured, either at present (gmhslh_hist_end_datetime is NULL) or in the past (gmhslh_hist_end_datetime is not NULL). The calibration and configuration history of the sensors on the sled are stored in the gmh_sensor_calibration_history and gmh_sensor_configuration_history tables, respectively.
Owner:
Target DB name:
Number of columns: 4
Number of indexes: 0
Number of foreign keys: 0
Primary key: gmhslh_sled_config_hist_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhslh_sled_config_hist_id | SERIAL | Not allowed | |
| gmhslh_hist_start_datetime | DATETIME YEAR TO | Not allowed | |
| gmhslh_hist_end_datetime | DATETIME YEAR TO | Allowed | |
| SLED_CONFIG_PARAMETERS | VARCHAR(100) | Not allowed | |

Column details

1. gmhslh_sled_config_hist_id
Physical name: gmhslh_sled_config_hist_id
Physical data type: SERIAL
Allow NULLs: Not allowed
Notes: Unique id identifying the sled configuration history record. Probably a generated key.

2. gmhslh_hist_start_datetime
Physical name: gmhslh_hist_start_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Not allowed
Notes: Date and time when this record became effective.

3. gmhslh_hist_end_datetime
Physical name: gmhslh_hist_end_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Allowed
Notes: Date and time when this record ceased to be effective.

4. SLED_CONFIG_PARAMETERS
Physical name: SLED_CONFIG_PARAMETERS
Physical data type: VARCHAR(100)
Allow NULLs: Not allowed
Notes: This will be replaced by a set of columns, one column for each configuration parameter, holding the value of that parameter.

# gmh_sled_measurements

Physical name: gmh_sled_measurements
Notes: Table of "Sled System" data measurements. Sled contains its own shear wave source; consequently no foreign key to a shot number is needed. Measures accelerometer data, orientation data, hydrophone data.
Reference: Higley-1-23d.
Owner:
Target DB name:
Number of columns: 10
Number of indexes: 1
Number of foreign keys: 3
Primary key: gmh_global_raw_meas_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_orientation_sensor_id (FK) | INT8 | Not allowed | |
| gmhsen_accelerometer_sensor_id (FK) | INT8 | Not allowed | |
| gmhsen_hydrophone_sensor_id (FK) | INT8 | Not allowed | |
| ACCELEROMETER_MEAS | INT8 | Not allowed | |
| ORIENTATION_MEAS | INT8 | Not allowed | |
| HYDROPHONE_MEAS | BYTE | Not allowed | |
| gmhslm_sled_shot_position | BYTE | Not allowed | |
| gmhslm_sled_shot_datetime (I1) | DATETIME YEAR TO | Not allowed | |
| SLED_MEASUREMENT_METADATA | INT8 | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmh_sled_measurements_IDX1 (I1) | gmhslm_sled_shot_datetime | Descending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_sled_measurements_FK1 | gmhsen_orientation_sensor_id | gmh_sensors.gmhsen_sensor_id |
| gmh_sensors_gmh_sled_measurements_FK2 | gmhsen_accelerometer_sensor_id | gmh_sensors.gmhsen_sensor_id |
| gmh_sensors_gmh_sled_measurements_FK3 | gmhsen_hydrophone_sensor_id | gmh_sensors.gmhsen_sensor_id |

## Column details

1. gmh_global_raw_meas_id
Physical name: gmh_global_raw_meas_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers.

2. gmhsen_orientation_sensor_id (FK)
Physical name: gmhsen_orientation_sensor_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of the sled's orientation sensor.

3. gmhsen_accelerometer_sensor_id (FK)
Physical name: gmhsen_accelerometer_sensor_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of the sled's accelerometer sensor.

4. gmhsen_hydrophone_sensor_id (FK)
Physical name: gmhsen_hydrophone_sensor_id

Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of the sled's hydrophone.

5. ACCELEROMETER_MEAS
Physical name: ACCELEROMETER_MEAS
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Measurement from the inboard accelerometers.

6. ORIENTATION_MEAS
Physical name: ORIENTATION_MEAS
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Measurement from the inboard orientation sensors.

7. HYDROPHONE_MEAS
Physical name: HYDROPHONE_MEAS
Physical data type: BYTE
Allow NULLs: Not allowed
Notes: Measurement from the inboard hydrophones.

8. gmhslm_sled_shot_position
Physical name: gmhslm_sled_shot_position
Physical data type: BYTE
Allow NULLs: Not allowed
Notes: Position of the sled (Lat,long,depth) when source fired.

9. gmhslm_sled_shot_datetime (I1)
Physical name: gmhslm_sled_shot_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Not allowed
Notes: Date and time when source fired.

10. SLED_MEASUREMENT_METADATA
Physical name: SLED_MEASUREMENT_METADATA
Physical data type: INT8
Allow NULLs: Not allowed
Notes: This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...".

## Index details

gmh_sled_measurements_IDX1
Column(s): gmhslm_sled_shot_datetime (Desc)
Unique: No

## Foreign key details (child)

gmh_sensors_gmh_sled_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_orientation_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | sled orientation sensor |
| Physical name: | gmh_sensors_gmh_sled_measurements_FK1 |
| Notes: | Each sled orientation sensor produces one or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_sled_measurements_FK2

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_accelerometer_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | sled accelerometer |
| Physical name: | gmh_sensors_gmh_sled_measurements_FK2 |
| Notes: | Each sled accelerometer sensor produces one or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_sensors_gmh_sled_measurements_FK3

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_hydrophone_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | sled hydrophone |
| Physical name: | gmh_sensors_gmh_sled_measurements_FK3 |
| Notes: | Each sled hydrophone produces one or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_standard_data_products

Physical name: gmh_standard_data_products
Notes: Table of standard data products. Each data product corresponds to a single standard data product type (e.g., "time series of N-minute averages of calibrated data at each of M ordinates"). To accommodate ad-hoc requests, one "standard data product" will be "custom".
Owner:
Target DB name:
Number of columns: 4
Number of indexes: 0
Number of foreign keys: 1
Primary key: gmhsdp_standard_data_product_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhsdp_standard_data_product_id | SERIAL | Not allowed | |
| gmhdap_data_product_outputtype_id (FK) | INT8 | Not allowed | |
| gmhsdp_standard_data_product_desc | VARCHAR(255) | Not allowed | |
| gmhsdp_standard_data_product_params | VARCHAR(255) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_data_product_outputtypes_gmh_standard_data_products_FK1 | gmhdap_data_product_outputtype_id | gmh_data_product_outputtypes.gmhdap_data_product_outputtype_id |
| gmh_standard_data_products_gmh_data_products_FK1 | gmh_data_products.gmhsdp_standard_data_product_id | gmhsdp_standard_data_product_id |
| gmh_standard_data_products_gmh_data_product_permissions_FK1 | gmh_data_product_permissions.gmhsdp_standard_data_product_id | gmhsdp_standard_data_product_id |

## Column details

1. gmhsdp_standard_data_product_id
Physical name: gmhsdp_standard_data_product_id
Physical data type: SERIAL
Allow NULLs: Not allowed
Notes: Unique ID - probably autogenerated

2. gmhdap_data_product_outputtype_id (FK)
Physical name: gmhdap_data_product_outputtype_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of the output type for this type of standard data product (e.g., image, spreadsheet, text report, etc.)

3. gmhsdp_standard_data_product_desc
Physical name: gmhsdp_standard_data_product_desc
Physical data type: VARCHAR(255)
Allow NULLs: Not allowed
Notes: Description of the standard data product. Examples:

"time series of N-minute averages of calibrated data at each of M ordinates"

"custom ad-hoc request"

4. gmhsdp_standard_data_product_params
Physical name: gmhsdp_standard_data_product_params
Physical data type: VARCHAR(255)
Allow NULLs: Not allowed
Notes: Description of any parameter values that the user must specify in requesting this data product.

## Foreign key details (child)

gmh_data_product_outputtypes_gmh_standard_data_products_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhdap_data_product_outputtype_id | |
| | | gmh_data_product_outputtypes.gmhdap_data_product_outputtype_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Verb phrase: | has output type | |
| Physical name: | gmh_data_product_outputtypes_gmh_standard_data_products_FK1 | |
| Notes: | Each standard data product has a particular output data type (image, spreadsheet, etc.) | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

## Foreign key details (parent)

gmh_standard_data_products_gmh_data_products_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_data_products.gmhsdp_standard_data_product_id | |
| | | gmhsdp_standard_data_product_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Verb phrase: | based on standard product | |
| Physical name: | gmh_standard_data_products_gmh_data_products_FK1 | |
| Notes: | Each data product corresponds to a single standard data product type (e.g., "time series of N-minute averages of calibrated data at each of N ordinates") | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_standard_data_products_gmh_data_product_permissions_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_data_product_permissions.gmhsdp_standard_data_product_id | |
| | | gmhsdp_standard_data_product_id |
| Relationship type: | Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Verb phrase: | role has permission to create product | |
| Physical name: | gmh_standard_data_products_gmh_data_product_permissions_FK1 | |
| Notes: | Each data product has an associated list of user roles. In order to produce a particular data product, a user must have one of the roles in that data product's list. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

# gmh_thermistor_measurements

Physical name: gmh_thermistor_measurements
Notes: Table of thermistor measurements.
Not tied to shot number; thermistor has own RTC (Higley-1-20).
Records every 6 hours for a year.
Reference: Higley-1-23c.

Sampling regime is 8 channels for 12 seconds at 4 Hz of 24 bit data every 6 hours for a year.

Owner:
Target DB name:
Number of columns: 5
Number of indexes: 0
Number of foreign keys: 1
Primary key: gmh_global_raw_meas_id

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmh_global_raw_meas_id | INT8 | Not allowed | |
| gmhsen_sensor_id (FK) | INT8 | Not allowed | |
| gmhsen_thermistor_meas_datetime | DATETIME YEAR TO | Not allowed | |
| gmhthm_thermistor_meas | BYTE | Not allowed | |
| THERMISTOR_MEAS_METADATA | VARCHAR(100) | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_sensors_gmh_thermistor_measurements_FK1 | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

Column details

1. gmh_global_raw_meas_id
Physical name: gmh_global_raw_meas_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: Unique id identifying the measurement. Probably a generated key. Primary IDs for all measurement tables come from the same pool of numbers.

2. gmhsen_sensor_id (FK)
Physical name: gmhsen_sensor_id
Physical data type: INT8
Allow NULLs: Not allowed
Notes: ID of the thermistor sensor.

Higley-1-23c says that the sampling regime for thermistors is 8 channels for 12 second record of 24-bit data at 4Hz every 6 hours for a year.

3. gmhsen_thermistor_meas_datetime
Physical name: gmhsen_thermistor_meas_datetime
Physical data type: DATETIME YEAR TO YEAR
Allow NULLs: Not allowed
Notes: The date and time of the measurement. Higley-1-23c says that the sampling regime is 8 channels for a 12-second record at 4 Hz of 24-bit data every 6 hours for a year. Therefore these records do not necessarily correspond to shots.

IF ANY MEASUREMENTS DO CORRESPOND TO SHOTS, WE NEED TO ADD A GMHSTL_SHOT_ID COLUMN.

4. gmhthm_thermistor_meas

| | |
|---|---|
| Physical name: | gmhthm_thermistor_meas |
| Physical data type: | BYTE |
| Allow NULLs: | Not allowed |
| Notes: | Blob of measurement data. |

5. THERMISTOR_MEAS_METADATA

| | |
|---|---|
| Physical name: | THERMISTOR_MEAS_METADATA |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | This column will be replaced by a set of columns storing the relevant metadata for the measurement. There are two basic types of metadata: administrative metadata and descriptive metadata. Administrative metadata will include such things as the format of the stored measurement, the time when it was taken, the size of the measurement, and any other parameters captured by the measuring device (other than the measurement itself). Descriptive metadata is information that can be used in discovery and identification (i.e., answering the questions "Find all measurements that ..." and "What is this measurement?"). Descriptive metadata will include information deduced from the stored measurement itself (e.g., maximum value, minimum value), enabling queries such as "Find all measurements where the maximum value was less than ..." or "Find all measurements where the values were between ... and ...". |

## Foreign key details (child)

gmh_sensors_gmh_thermistor_measurements_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhsen_sensor_id | gmh_sensors.gmhsen_sensor_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | thermistor |
| Physical name: | gmh_sensors_gmh_thermistor_measurements_FK1 |
| Notes: | Each thermistor produces 1 or more measurements. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_user_annotations

| | |
|---|---|
| Physical name: | gmh_user_annotations |
| Notes: | Table of annotations that GMH users have attached to raw and/or calibrated measurements and/or data products. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 6 |
| Number of indexes: | 0 |
| Number of foreign keys: | 4 |
| Primary key: | gmhuan_user_annotation_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhuan_user_annotation_id | SERIAL | Not allowed | |
| gmh_global_raw_meas_id (FK) | INT8 | Allowed | |
| gmh_global_calibrated_meas_id (FK) | INT8 | Allowed | |
| gmh_global_data_product_id (FK) | INT8 | Allowed | |
| gmhusr_user_id (FK) | INT8 | Not allowed | |
| gmhuan_user_annotation | TEXT | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_users_gmh_user_annotations_FK1 | gmhusr_user_id | gmh_users.gmhusr_user_id |
| gmh_raw_measurement_ids_gmh_user_annotations_FK1 | gmh_global_raw_meas_id | gmh_raw_measurement_ids.gmh_global_raw_meas_id |
| gmh_calibrated_measurements_gmh_user_annotations_FK1 | gmh_global_calibrated_meas_id | gmh_calibrated_measurements.gmh_global_calibrated_meas_id |
| gmh_data_products_gmh_user_annotations_FK1 | gmh_global_data_product_id | gmh_data_products.gmh_global_data_product_id |

Column details

1. gmhuan_user_annotation_id

| | |
|---|---|
| Physical name: | gmhuan_user_annotation_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID - probably autogenerated. |

2. gmh_global_raw_meas_id (FK)

| | |
|---|---|
| Physical name: | gmh_global_raw_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Allowed |
| Notes: | The ID of the raw measurement associated with this annotation (if any). At least one of gmh_global_raw_meas_id, gmh_global_calibrated_meas_id, or gmh_global_data_product_id must be specified. |

3. gmh_global_calibrated_meas_id (FK)

| | |
|---|---|
| Physical name: | gmh_global_calibrated_meas_id |
| Physical data type: | INT8 |
| Allow NULLs: | Allowed |
| Notes: | The ID of the calibrated measurement associated with this annotation (if any). At least one of gmh_global_raw_meas_id, gmh_global_calibrated_meas_id, or gmh_global_data_product_id must be specified. |

4. gmh_global_data_product_id (FK)

| | |
|---|---|
| Physical name: | gmh_global_data_product_id |
| Physical data type: | INT8 |
| Allow NULLs: | Allowed |
| Notes: | The ID of the data product associated with this annotation (if any). At least one of gmh_global_raw_meas_id, gmh_global_calibrated_meas_id, or gmh_global_data_product_id must be specified. |

5. gmhusr_user_id (FK)

| | |
|---|---|
| Physical name: | gmhusr_user_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID of the GMH user making this annotation. |

6. gmhuan_user_annotation

| | |
|---|---|
| Physical name: | gmhuan_user_annotation |
| Physical data type: | TEXT |
| Allow NULLs: | Not allowed |
| Notes: | The annotation text itself. |

Foreign key details (child)

gmh_users_gmh_user_annotations_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmhusr_user_id | gmh_users.gmhusr_user_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | creates an annotation |
| Physical name: | gmh_users_gmh_user_annotations_FK1 |
| Notes: | Each annotation is written by a GMH user. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_raw_measurement_ids_gmh_user_annotations_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_raw_meas_id | gmh_raw_measurement_ids.gmh_global_raw_meas_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | annotates a raw measurement |
| Physical name: | gmh_raw_measurement_ids_gmh_user_annotations_FK1 |
| Notes: | Each raw measurement may be annotated zero or more times by one or more users. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_calibrated_measurements_gmh_user_annotations_FK1

| Definition: | Child | Parent |
|---|---|---|
| | gmh_global_calibrated_meas_id | gmh_calibrated_measurements.gmh_global_calibrated_meas_id |

| | |
|---|---|
| Relationship type: | Non-Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | annotates a calibrated measurment |
| Physical name: | gmh_calibrated_measurements_gmh_user_annotations_FK1 |
| Notes: | Each calibrated measurement may be annotated zero or more times by one or more users. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_data_products_gmh_user_annotations_FK1

| Definition: | Child | Parent |
| --- | --- | --- |
| | gmh_global_data_product_id | gmh_data_products.gmh_global_data_product_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Verb phrase: | annotates a product | |
| Physical name: | gmh_data_products_gmh_user_annotations_FK1 | |
| Notes: | Each data product may be annotated zero or more times by one or more users. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

# gmh_user_roles

| | |
|---|---|
| Physical name: | gmh_user_roles |
| Notes: | This table relates GMH users to the access role(s) they have. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 2 |
| Number of indexes: | 0 |
| Number of foreign keys: | 2 |
| Primary key: | 1. gmhrol_role_id<br>2. gmhusr_user_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhrol_role_id (FK) | INT8 | Not allowed | |
| gmhusr_user_id (FK) | INT8 | Not allowed | |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_users_gmh_user_roles_FK1 | gmhusr_user_id | gmh_users.gmhusr_user_id |
| gmh_roles_gmh_user_roles_FK1 | gmhrol_role_id | gmh_roles.gmhrol_role_id |

## Column details

1. gmhrol_role_id (FK)

| | |
|---|---|
| Physical name: | gmhrol_role_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Id of the role |

2. gmhusr_user_id (FK)

| | |
|---|---|
| Physical name: | gmhusr_user_id |
| Physical data type: | INT8 |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID for the GMH user. |

## Foreign key details (child)

gmh_users_gmh_user_roles_FK1

| | Child | Parent |
|---|---|---|
| Definition: | gmhusr_user_id | gmh_users.gmhusr_user_id |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | users hold roles |
| Physical name: | gmh_users_gmh_user_roles_FK1 |
| Notes: | Each user can take on one or more roles. |
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

gmh_roles_gmh_user_roles_FK1

| | Child | Parent |
|---|---|---|
| Definition: | gmhrol_role_id | gmh_roles.gmhrol_role_id |

| | |
|---|---|
| Relationship type: | Identifying |
| Cardinality: | One -to- Zero-or-More |
| Allow NULLs: | Not allowed |
| Verb phrase: | roles are held by users |
| Physical name: | gmh_roles_gmh_user_roles_FK1 |
| Notes: | Each role can be taken on by zero or more GMH users |

| | |
|---|---|
| Ref. Integrity on update: | No Action |
| Ref. Integrity on delete: | No Action |

# gmh_users

| | |
|---|---|
| Physical name: | gmh_users |
| Notes: | This table defines the GMH users. |
| Owner: | |
| Target DB name: | |
| Number of columns: | 5 |
| Number of indexes: | 1 |
| Number of foreign keys: | 0 |
| Primary key: | gmhusr_user_id |

| Columns | Data type | Allow NULLs | Value/Range |
|---|---|---|---|
| gmhusr_user_id | SERIAL | Not allowed | |
| gmhusr_user_shortname (U1) | VARCHAR(100) | Not allowed | |
| gmhusr_user_name | VARCHAR(100) | Not allowed | |
| gmhusr_user_description | VARCHAR(255) | Allowed | |
| gmhusr_database_user_id | VARCHAR(64) | Not allowed | |

| Indexes | Columns | Sort order |
|---|---|---|
| gmhusr_user_shortname_idx (U1) | gmhusr_user_shortname | Ascending |

| Foreign keys | Child | Parent |
|---|---|---|
| gmh_users_gmh_user_annotations_FK1 | gmh_user_annotations.gmhusr_user_id | gmhusr_user_id |
| gmh_users_gmh_user_roles_FK1 | gmh_user_roles.gmhusr_user_id | gmhusr_user_id |
| gmh_users_gmh_data_products_FK1 | gmh_data_products.gmhusr_user_id | gmhusr_user_id |

Column details

1. gmhusr_user_id

| | |
|---|---|
| Physical name: | gmhusr_user_id |
| Physical data type: | SERIAL |
| Allow NULLs: | Not allowed |
| Notes: | Unique ID for the GMH user. |

2. gmhusr_user_shortname (U1)

| | |
|---|---|
| Physical name: | gmhusr_user_shortname |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | Shortname (alphanumeric user id) |

3. gmhusr_user_name

| | |
|---|---|
| Physical name: | gmhusr_user_name |
| Physical data type: | VARCHAR(100) |
| Allow NULLs: | Not allowed |
| Notes: | Name of the GMH user. |

4. gmhusr_user_description

| | |
|---|---|
| Physical name: | gmhusr_user_description |
| Physical data type: | VARCHAR(255) |
| Allow NULLs: | Allowed |
| Notes: | Other descriptive information about the GMH user. E.g., Organization, office location, phone number, etc. |

5. gmhusr_database_user_id

| | |
|---|---|
| Physical name: | gmhusr_database_user_id |
| Physical data type: | VARCHAR(64) |
| Allow NULLs: | Not allowed |
| Notes: | The PostgreSQL userid corresponding to this GMH user. |

## Index details

gmhusr_user_shortname_idx

| | |
|---|---|
| Column(s): | gmhusr_user_shortname (Asc) |
| Unique: | Yes |

## Foreign key details (parent)

gmh_users_gmh_user_annotations_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmh_user_annotations.gmhusr_user_id | gmhusr_user_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Verb phrase: | creates an annotation | |
| Physical name: | gmh_users_gmh_user_annotations_FK1 | |
| Notes: | Each annotation is written by a GMH user. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_users_gmh_user_roles_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmh_user_roles.gmhusr_user_id | gmhusr_user_id |
| Relationship type: | Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Verb phrase: | users hold roles | |
| Physical name: | gmh_users_gmh_user_roles_FK1 | |
| Notes: | Each user can take on one or more roles. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

gmh_users_gmh_data_products_FK1

| | | |
|---|---|---|
| Definition: | Child | Parent |
| | gmh_data_products.gmhusr_user_id | gmhusr_user_id |
| Relationship type: | Non-Identifying | |
| Cardinality: | One -to- Zero-or-More | |
| Allow NULLs: | Not allowed | |
| Verb phrase: | creates product | |
| Physical name: | gmh_users_gmh_data_products_FK1 | |
| Notes: | Each data product was requested / produced by one user. | |
| Ref. Integrity on update: | No Action | |
| Ref. Integrity on delete: | No Action | |

# Appendix 5: PostgreSQL Database Creation SQL

The following SQL was used to create objects in the “gmh” PostgreSQL database:

```
--    This SQL DDL script was generated by VisioModeler 3.1 (Release Date:
10/02/1998).

--    Driver Used : VisioModeler 3.1 - ODBC Generic Driver Driver.
--    Document    : Z:\doc\OleMiss\GMH.IMD.
--    Time Created: February 01, 2006 3:26 PM.
--    User Action : From VisioModeler GenerateWizard.
--    Connected Data source : No connection.
--    Connected Server      : No connection.
--    Connected Database    : Not applicable.

drop table gmh_array_connections cascade ;
drop table gmh_cable_arrays cascade ;
drop table gmh_images cascade ;
drop table gmh_mass_spec_measurements cascade ;
drop table gmh_ctd_measurements cascade ;
drop table gmh_mets_measurements cascade ;
drop table gmh_adcp_measurements cascade ;
drop table gmh_oxygen_sensor_measurements cascade ;
drop table gmh_array_configuration_history cascade ;
drop table gmh_array_calibration_history cascade ;
drop table gmh_sensor_arrays cascade ;
drop table gmh_sensors cascade ;
drop table gmh_shot_log cascade ;
drop table gmh_ship_log cascade ;
drop table gmh_sensor_types cascade ;
drop table gmh_hydrophone_measurements cascade ;
drop table gmh_orientation_sensor_measurements cascade ;
drop table gmh_thermistor_measurements cascade ;
drop table gmh_3daccel_measurements cascade ;
drop table gmh_sensor_calibration_history cascade ;
drop table gmh_sensor_configuration_history cascade ;
drop table gmh_deep_tow_measurements cascade ;
drop table gmh_sled_measurements cascade ;
drop table gmh_auv_measurements cascade ;
drop table gmh_sled_configuration_history cascade ;
drop table gmh_fluorometer_measurements cascade ;
drop table gmh_calibrated_measurements cascade ;
drop table gmh_raw_measurement_ids cascade ;
drop table gmh_user_annotations cascade ;
drop table gmh_users cascade ;
drop table gmh_roles cascade ;
drop table gmh_data_products cascade ;
drop table gmh_data_product_outputtypes cascade ;
drop table gmh_calibrated_measurements_usage cascade ;
drop table gmh_standard_data_products cascade ;
drop table gmh_user_roles cascade ;
drop table gmh_data_product_permissions cascade ;
drop  sequence gmh_global_raw_meas_seq;
```

```
create  sequence gmh_global_raw_meas_seq;

create table gmh_array_connections (
      gmhacn_connection_id INTEGER not null,
      gmhcbl_cable_array_1_id INTEGER not null,
      gmhacn_array_1_end CHAR(1) null,
      gmhcbl_cable_array_2_id INTEGER not null,
      gmhacn_array_2_end CHAR(1) null, constraint gmh_array_connections_PK
primary key (gmhacn_connection_id) );

create table gmh_cable_arrays (
      gmhcbl_cable_array_id serial not null,
      gmhcbl_cable_array_code CHAR(10) not null,
      gmhcbl_cable_array_desc VARCHAR(255) not null,
      gmhcbl_cable_array_linework oid null,
      CABLE_ARRAY_INFO VARCHAR(100) not null, constraint gmh_cable_arrays_PK
primary key (gmhcbl_cable_array_id) );

create table gmh_images (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhimg_image oid not null,
      gmhimg_image_datetime timestamp not null,
      IMAGE_METADATA INTEGER not null, constraint gmh_images_PK primary key
(gmh_global_raw_meas_id) );

create table gmh_mass_spec_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhmsm_mass_spec_meas oid not null,
      gmhmsm_mass_spec_meas_datetime timestamp not null,
      MASS_SPEC_MEAS_METADATA INTEGER not null, constraint
gmh_mass_spec_measurements_PK primary key (gmh_global_raw_meas_id) );

create table gmh_ctd_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhctd_ctd_meas oid not null,
      gmhctd_ctd_meas_datetime timestamp not null,
      CTD_MEAS_METADATA INTEGER not null, constraint gmh_ctd_measurements_PK
primary key (gmh_global_raw_meas_id) );

create table gmh_mets_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhmtm_mets_meas oid not null,
      gmhmtm_mets_meas_datetime timestamp not null,
      METS_MEAS_METADATA INTEGER not null, constraint
gmh_mets_measurements_PK primary key (gmh_global_raw_meas_id) );
```

```
create table gmh_adcp_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhcpm_adcp_meas oid not null,
      gmhcpm_adcp_meas_datetime timestamp not null,
      ADCP_MEAS_METADATA INTEGER not null, constraint
gmh_adcp_measurements_PK primary key (gmh_global_raw_meas_id) );

create table gmh_oxygen_sensor_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmho2m_oxygen_sensor_meas oid not null,
      gmho2m_oxygen_sensor_meas_datetime timestamp not null,
      OXYGEN_MEAS_METADATA INTEGER not null, constraint
gmh_oxygen_sensor_measurements_PK primary key (gmh_global_raw_meas_id) );

create table gmh_array_configuration_history (
      gmhacn_array_config_hist_id serial not null,
      gmhcbl_cable_array_id INTEGER not null,
      gmhacn_hist_start_datetime timestamp not null,
      gmhacn_hist_end_datetime timestamp null,
      ARRAY_CONFIG_PARAMETERS VARCHAR(100) not null, constraint
gmh_array_configuration_history_PK primary key (gmhacn_array_config_hist_id)
);

create table gmh_array_calibration_history (
      gmhacl_array_calibration_hist_id serial not null,
      gmhcbl_cable_array_id INTEGER not null,
      gmhacl_hist_start_datetime timestamp not null,
      gmhacl_hist_end_datetime timestamp null,
      ARRAY_CALIBRATION_PARAMETERS VARCHAR(100) not null, constraint
gmh_array_calibration_history_PK primary key
(gmhacl_array_calibration_hist_id) );

create table gmh_sensor_arrays (
      gmharr_sensor_array_id serial not null,
      gmharr_sensor_array_desc VARCHAR(255) not null,
        gmharr_sensor_array_code CHAR(10) not null,
      ARRAY_INFO VARCHAR(100) not null, constraint gmh_sensor_arrays_PK
primary key (gmharr_sensor_array_id) );

create table gmh_sensors (
      gmhsen_sensor_id serial not null,
      gmhcbl_cable_array_id INTEGER not null,
      gmharr_sensor_array_id INTEGER null,
      gmhsen_sensor_name VARCHAR(100) not null,
      gmhstp_sensor_type_id INTEGER not null,
      gmhsen_position_on_array FLOAT null,
      gmhsen_absolute_position integer null,
      STATIC_SENSOR_INFO VARCHAR(100) not null, constraint gmh_sensors_PK
primary key (gmhsen_sensor_id) );
```

```
create table gmh_shot_log (
      gmhstl_shot_id serial not null,
      gmhstl_shot_datetime timestamp not null,
      gmhstl_shot_delay_time interval not null,
      gmhstl_shot_position integer not null,
      SHOT_LOG_INFO varchar(100) not null, constraint gmh_shot_log_PK primary
key (gmhstl_shot_id) );

create table gmh_ship_log (
      gmhspl_log_entry_id INTEGER not null, constraint gmh_ship_log_PK
primary key (gmhspl_log_entry_id) );

create table gmh_sensor_types (
      gmhstp_sensor_type_id serial not null,
      gmhstp_sensor_type_code CHAR(10) not null,
      gmhstp_sensor_type_desc VARCHAR(255) not null,
      SENSOR_TYPE_INFO VARCHAR(100) not null, constraint gmh_sensor_types_PK
primary key (gmhstp_sensor_type_id) );

create table gmh_hydrophone_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhstl_shot_id INTEGER not null,
      gmhhpm_hydrophone_meas_starttime timestamp not null,
      gmhhpm_hydrophone_meas_endtime timestamp not null,
      gmhhpm_hydrophone_meas_nummeas INTEGER not null,
      gmhhpm_hydrophone_meas oid not null,
        gmhhpm_hydrophone_meas_maxamp integer,
        gmhhpm_hydrophone_meas_minamp integer,
      gmhhpm_hydrophone_meas_p16filename VARCHAR(100) null,
      HYDROPHONE_MEAS_METADATA varchar(100) not null, constraint
gmh_hydrophone_measurements_PK primary key (gmh_global_raw_meas_id) );

create table gmh_orientation_sensor_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhstl_shot_id INTEGER null,
      gmhosm_orientation_sensor_meas_datetime timestamp null,
      gmhosm_orientation_sensor_meas oid not null,
      ORIENTATION_SENSOR_MEAS_METADATA varchar(100) not null, constraint
gmh_orientation_sensor_measurements_PK primary key (gmh_global_raw_meas_id)
);

create table gmh_thermistor_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhsen_thermistor_meas_datetime timestamp not null,
      gmhthm_thermistor_meas oid not null,
      THERMISTOR_MEAS_METADATA VARCHAR(100) not null, constraint
gmh_thermistor_measurements_PK primary key (gmh_global_raw_meas_id) );
```

```
create table gmh_3daccel_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhstl_shot_id INTEGER not null,
      gmh3dm_3daccel_meas oid not null,
      THREEDACCEL_MEAS_METADATA varchar(100) not null, constraint
gmh_3daccel_measurements_PK primary key (gmh_global_raw_meas_id) );

create table gmh_sensor_calibration_history (
      gmhscl_sensor_calibration_hist_id serial not null,
      gmhsen_sensor_id INTEGER not null,
      gmhscl_hist_start_datetime timestamp not null,
      gmhscl_hist_end_datetime timestamp null,
      SENSOR_CALIBRATION_PARAMETERS VARCHAR(100) not null, constraint
gmh_sensor_calibration_history_PK primary key
(gmhscl_sensor_calibration_hist_id) );

create table gmh_sensor_configuration_history (
      gmhscn_sensor_config_hist_id serial not null,
      gmhsen_sensor_id INTEGER not null,
      gmhscn_hist_start_datetime timestamp not null,
      gmhscn_hist_end_datetime timestamp null,
      SENSOR_CONFIG_PARAMETERS VARCHAR(100) not null, constraint
gmh_sensor_configuration_history_PK primary key
(gmhscn_sensor_config_hist_id) );

create table gmh_deep_tow_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhstl_shot_id INTEGER not null,
      gmhdtm_deep_tow_meas oid not null,
      DEEP_TOW_MEAS_METADATA varchar(100) not null, constraint
gmh_deep_tow_measurements_PK primary key (gmh_global_raw_meas_id) );

create table gmh_sled_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_orientation_sensor_id INTEGER not null,
      gmhsen_accelerometer_sensor_id INTEGER not null,
      gmhsen_hydrophone_sensor_id INTEGER not null,
      ACCELEROMETER_MEAS varchar(100) not null,
      ORIENTATION_MEAS varchar(100) not null,
      HYDROPHONE_MEAS oid not null,
      gmhslm_sled_shot_position integer not null,
      gmhslm_sled_shot_datetime timestamp not null,
      SLED_MEASUREMENT_METADATA varchar(100) not null, constraint
gmh_sled_measurements_PK primary key (gmh_global_raw_meas_id) );

create table gmh_auv_measurements (
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhstl_shot_id INTEGER not null,
      gmhauv_auv_meas oid not null,
      AUV_MEASUREMENT_METADATA varchar(100) not null, constraint
```

```
gmh_auv_measurements_PK primary key (gmh_global_raw_meas_id) );
```

```
create table gmh_sled_configuration_history (
      gmhslh_sled_config_hist_id serial not null,
      gmhslh_hist_start_datetime timestamp not null,
      gmhslh_hist_end_datetime timestamp null,
      SLED_CONFIG_PARAMETERS VARCHAR(100) not null, constraint
gmh_sled_configuration_history_PK primary key (gmhslh_sled_config_hist_id) );

create table gmh_fluorometer_measurements (
      gmh_global_raw_meas_id varchar(100) not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmhsen_sensor_id INTEGER not null,
      gmhflm_fluorometer_meas float not null,
      gmhflm_fluorometer_meas_datetime timestamp not null,
      FLUOROMETER_MEAS_METADATA varchar(100) not null, constraint
gmh_fluorometer_measurements_PK primary key (gmh_global_raw_meas_id) );

create table gmh_calibrated_measurements (
      gmh_global_calibrated_meas_id serial not null,
      gmh_global_raw_meas_id INTEGER not null default
nextval('public.gmh_global_raw_meas_seq'::text),
      gmh_calibrated_measurement oid not null,
      CALIBRATION_PARAMETERS VARCHAR(100) not null,
      MEASUREMENT_METADATA VARCHAR(100) not null, constraint
gmh_calibrated_measurements_PK primary key (gmh_global_calibrated_meas_id) );

create table gmh_raw_measurement_ids (
      gmh_global_raw_meas_id serial not null,
      gmhrmi_raw_measurement_type_code CHAR(4) not null,
      gmhrmi_raw_measurement_datetime timestamp not null, constraint
gmh_raw_measurement_ids_PK primary key (gmh_global_raw_meas_id) );

create table gmh_user_annotations (
      gmhuan_user_annotation_id serial not null,
      gmh_global_raw_meas_id INTEGER null,
      gmh_global_calibrated_meas_id INTEGER null,
      gmh_global_data_product_id INTEGER null,
      gmhusr_user_id INTEGER not null,
      gmhuan_user_annotation oid not null, constraint gmh_user_annotations_PK
primary key (gmhuan_user_annotation_id) );

create table gmh_users (
      gmhusr_user_id serial not null,
      gmhusr_user_shortname VARCHAR(100) not null,
      gmhusr_user_name VARCHAR(100) not null,
      gmhusr_user_description VARCHAR(255) null,
      gmhusr_database_user_id VARCHAR(64) not null, constraint gmh_users_PK
primary key (gmhusr_user_id) );

create table gmh_roles (
      gmhrol_role_id serial not null,
      gmhrol_role_name VARCHAR(32) not null,
      gmhrol_role_description VARCHAR(255) not null,
      gmhrol_database_role_id VARCHAR(64) not null, constraint gmh_roles_PK
primary key (gmhrol_role_id) );
```

```
create table gmh_data_products (
      gmh_global_data_product_id serial not null,
      gmhsdp_standard_data_product_id INTEGER not null,
      gmhusr_user_id INTEGER not null,
      gmhdat_creation_datetime timestamp not null,
      gmhdat_creation_sql oid not null,
      gmhdat_processing_desc VARCHAR(255) null,
      gmhdat_data_product INTEGER not null, constraint gmh_data_products_PK
primary key (gmh_global_data_product_id) );

create table gmh_data_product_outputtypes (
      gmhdap_data_product_outputtype_id serial not null,
      gmhdat_data_product_outputtype_desc VARCHAR(100) not null, constraint
gmh_data_product_outputtypes_PK primary key
(gmhdap_data_product_outputtype_id) );

create table gmh_calibrated_measurements_usage (
      gmh_global_data_product_id INTEGER not null,
      gmh_global_calibrated_meas_id INTEGER not null, constraint
gmh_calibrated_measurements_usage_PK primary key (gmh_global_data_product_id,
gmh_global_calibrated_meas_id) );

create table gmh_standard_data_products (
      gmhsdp_standard_data_product_id serial not null,
      gmhdap_data_product_outputtype_id INTEGER not null,
      gmhsdp_standard_data_product_desc VARCHAR(255) not null,
      gmhsdp_standard_data_product_params VARCHAR(255) not null, constraint
gmh_standard_data_products_PK primary key (gmhsdp_standard_data_product_id)
);

create table gmh_user_roles (
      gmhrol_role_id INTEGER not null,
      gmhusr_user_id INTEGER not null, constraint gmh_user_roles_PK primary
key (gmhrol_role_id, gmhusr_user_id) );

create table gmh_data_product_permissions (
      gmhsdp_standard_data_product_id INTEGER not null,
      gmhrol_role_id INTEGER not null, constraint
gmh_data_product_permissions_PK primary key (gmhsdp_standard_data_product_id,
gmhrol_role_id) );
```

```
create unique index gmhcbl_cable_array_code_idx on gmh_cable_arrays (
      gmhcbl_cable_array_code);

alter table gmh_cable_arrays add constraint
        gmhcbl_cable_array_code_id_uc1 unique (gmhcbl_cable_array_code);

create unique index gmharr_sensor_array_code_idx on gmh_sensor_arrays (
      gmharr_sensor_array_code);

alter table gmh_sensor_arrays add constraint
     gmharr_sensor_array_code_i_uc2 unique (gmharr_sensor_array_code);

create unique index gmhsen_sensor_name_idx on gmh_sensors (
      gmhsen_sensor_name);


alter table gmh_sensors add constraint gmhsen_sensor_name_idx_uc1 unique (
      gmhsen_sensor_name);


create unique index gmhstp_sensor_type_code_idx on gmh_sensor_types (
      gmhstp_sensor_type_code);


alter table gmh_sensor_types add constraint gmhstp_sensor_type_code_id_uc2
unique (
      gmhstp_sensor_type_code);


create unique index gmh_hydrophone_measurements_AK1 on
gmh_hydrophone_measurements (
      gmhstl_shot_id,
      gmhsen_sensor_id);


alter table gmh_hydrophone_measurements add constraint
gmh_hydrophone_measurement_uc3 unique (
      gmhstl_shot_id,
      gmhsen_sensor_id);


create unique index gmh_orientation_sensor_measurements_AK1 on
gmh_orientation_sensor_measurements (
      gmhstl_shot_id,
      gmhsen_sensor_id);


alter table gmh_orientation_sensor_measurements add constraint
gmh_orientation_sensor_mea_uc4 unique (
      gmhstl_shot_id,
      gmhsen_sensor_id);


create unique index gmh_3daccel_measurements_AK1 on gmh_3daccel_measurements
(
      gmhstl_shot_id,
      gmhsen_sensor_id);
```

```
alter table gmh_3daccel_measurements add constraint
gmh_3daccel_measurements_A_uc5 unique (
      gmhstl_shot_id,
      gmhsen_sensor_id);


create unique index gmh_deep_tow_measurements_AK1 on
gmh_deep_tow_measurements (
      gmhstl_shot_id,
      gmhsen_sensor_id);


alter table gmh_deep_tow_measurements add constraint
gmh_deep_tow_measurements__uc6 unique (
      gmhstl_shot_id,
      gmhsen_sensor_id);


create index gmh_sled_measurements_IDX1 on gmh_sled_measurements (
      gmhslm_sled_shot_datetime );


create unique index gmh_auv_measurements_AK1 on gmh_auv_measurements (
      gmhstl_shot_id,
      gmhsen_sensor_id);


alter table gmh_auv_measurements add constraint gmh_auv_measurements_AK1_uc7
unique (
      gmhstl_shot_id,
      gmhsen_sensor_id);


create unique index gmhusr_user_shortname_idx on gmh_users (
      gmhusr_user_shortname);


alter table gmh_users add constraint gmhusr_user_shortname_idx_uc8 unique (
      gmhusr_user_shortname);


create unique index gmhrol_role_name_idx on gmh_roles (
      gmhrol_role_name);


alter table gmh_roles add constraint gmhrol_role_name_idx_uc9 unique (
      gmhrol_role_name);


alter table gmh_array_connections
      add constraint gmh_cable_arrays_gmh_array_connections_FK1 foreign key (
            gmhcbl_cable_array_1_id)
       references gmh_cable_arrays (
            gmhcbl_cable_array_id);
```

```
alter table gmh_array_connections
      add constraint gmh_cable_arrays_gmh_array_connections_FK2 foreign key (
            gmhcbl_cable_array_2_id)
       references gmh_cable_arrays (
            gmhcbl_cable_array_id);

alter table gmh_images
      add constraint gmh_sensors_gmh_images_FK1 foreign key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_mass_spec_measurements
      add constraint gmh_sensors_gmh_mass_spec_measurements_FK1 foreign key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_ctd_measurements
      add constraint gmh_sensors_gmh_ctd_measurements_FK1 foreign key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_mets_measurements
      add constraint gmh_sensors_gmh_mets_measurements_FK1 foreign key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_adcp_measurements
      add constraint gmh_sensors_gmh_adcp_measurements_FK1 foreign key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_oxygen_sensor_measurements
      add constraint gmh_sensors_gmh_oxygen_sensor_measurements_FK1 foreign
key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_array_configuration_history
      add constraint gmh_cable_arrays_gmh_array_configuration_history_FK1
foreign key (
            gmhcbl_cable_array_id)
       references gmh_cable_arrays (
            gmhcbl_cable_array_id);

alter table gmh_array_calibration_history
      add constraint gmh_cable_arrays_gmh_array_calibration_history_FK1
foreign key (
            gmhcbl_cable_array_id)
       references gmh_cable_arrays (
            gmhcbl_cable_array_id);
```

```
alter table gmh_sensors
      add constraint gmh_sensor_types_gmh_sensors_FK1 foreign key (
            gmhstp_sensor_type_id)
       references gmh_sensor_types (
            gmhstp_sensor_type_id);

alter table gmh_sensors
      add constraint gmh_sensor_arrays_gmh_sensors_FK1 foreign key (
            gmharr_sensor_array_id)
       references gmh_sensor_arrays (
            gmharr_sensor_array_id);

alter table gmh_sensors
      add constraint gmh_cable_arrays_gmh_sensors_FK1 foreign key (
            gmhcbl_cable_array_id)
       references gmh_cable_arrays (
            gmhcbl_cable_array_id);

alter table gmh_hydrophone_measurements
      add constraint gmh_shot_log_gmh_hydrophone_measurements_FK1 foreign key
(
            gmhstl_shot_id)
       references gmh_shot_log (
            gmhstl_shot_id);

alter table gmh_hydrophone_measurements
      add constraint gmh_sensors_gmh_hydrophone_measurements_FK1 foreign key
(
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_orientation_sensor_measurements
      add constraint gmh_shot_log_gmh_orientation_sensor_measurements_FK1
foreign key (
            gmhstl_shot_id)
       references gmh_shot_log (
            gmhstl_shot_id);

alter table gmh_orientation_sensor_measurements
      add constraint gmh_sensors_gmh_orientation_sensor_measurements_FK1
foreign key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_thermistor_measurements
      add constraint gmh_sensors_gmh_thermistor_measurements_FK1 foreign key
(
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_3daccel_measurements
      add constraint gmh_shot_log_gmh_3daccel_measurements_FK1 foreign key (
            gmhstl_shot_id)
       references gmh_shot_log (
```

```
            gmhstl_shot_id);

alter table gmh_3daccel_measurements
      add constraint gmh_sensors_gmh_3daccel_measurements_FK1 foreign key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_sensor_calibration_history
      add constraint gmh_sensors_gmh_sensor_calibration_history_FK1 foreign
key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_sensor_configuration_history
      add constraint gmh_sensors_gmh_sensor_configuration_history_FK1 foreign
key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_deep_tow_measurements
      add constraint gmh_shot_log_gmh_deep_tow_measurements_FK1 foreign key (
            gmhstl_shot_id)
       references gmh_shot_log (
            gmhstl_shot_id);

alter table gmh_deep_tow_measurements
      add constraint gmh_sensors_gmh_deep_tow_measurements_FK1 foreign key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_sled_measurements
      add constraint gmh_sensors_gmh_sled_measurements_FK1 foreign key (
            gmhsen_orientation_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_sled_measurements
      add constraint gmh_sensors_gmh_sled_measurements_FK2 foreign key (
            gmhsen_accelerometer_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_sled_measurements
      add constraint gmh_sensors_gmh_sled_measurements_FK3 foreign key (
            gmhsen_hydrophone_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);
```

```
alter table gmh_auv_measurements
      add constraint gmh_shot_log_gmh_auv_measurements_FK1 foreign key (
            gmhstl_shot_id)
       references gmh_shot_log (
            gmhstl_shot_id);

alter table gmh_auv_measurements
      add constraint gmh_sensors_gmh_auv_measurements_FK1 foreign key (
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_fluorometer_measurements
      add constraint gmh_sensors_gmh_fluorometer_measurements_FK1 foreign key
(
            gmhsen_sensor_id)
       references gmh_sensors (
            gmhsen_sensor_id);

alter table gmh_calibrated_measurements
      add constraint gmh_raw_measurement_ids_gmh_calibrated_measurements_FK1
foreign key (
            gmh_global_raw_meas_id)
       references gmh_raw_measurement_ids (
            gmh_global_raw_meas_id);

alter table gmh_user_annotations
      add constraint gmh_users_gmh_user_annotations_FK1 foreign key (
            gmhusr_user_id)
       references gmh_users (
            gmhusr_user_id);

alter table gmh_user_annotations
      add constraint gmh_raw_measurement_ids_gmh_user_annotations_FK1 foreign
key (
            gmh_global_raw_meas_id)
       references gmh_raw_measurement_ids (
            gmh_global_raw_meas_id);

alter table gmh_user_annotations
      add constraint gmh_calibrated_measurements_gmh_user_annotations_FK1
foreign key (
            gmh_global_calibrated_meas_id)
       references gmh_calibrated_measurements (
            gmh_global_calibrated_meas_id);

alter table gmh_user_annotations
      add constraint gmh_data_products_gmh_user_annotations_FK1 foreign key (
            gmh_global_data_product_id)
       references gmh_data_products (
            gmh_global_data_product_id);

alter table gmh_data_products
      add constraint gmh_standard_data_products_gmh_data_products_FK1 foreign
key (
            gmhsdp_standard_data_product_id)
       references gmh_standard_data_products (
```

```
            gmhsdp_standard_data_product_id);

alter table gmh_data_products
      add constraint gmh_users_gmh_data_products_FK1 foreign key (
            gmhusr_user_id)
       references gmh_users (
            gmhusr_user_id);

alter table gmh_calibrated_measurements_usage
      add constraint
gmh_calibrated_measurements_gmh_calibrated_measurements_usage_FK1 foreign key
(
            gmh_global_calibrated_meas_id)
       references gmh_calibrated_measurements (
            gmh_global_calibrated_meas_id);

alter table gmh_calibrated_measurements_usage
      add constraint gmh_data_products_gmh_calibrated_measurements_usage_FK1
foreign key (
            gmh_global_data_product_id)
       references gmh_data_products (
            gmh_global_data_product_id);

alter table gmh_standard_data_products
      add constraint
gmh_data_product_outputtypes_gmh_standard_data_products_FK1 foreign key (
            gmhdap_data_product_outputtype_id)
       references gmh_data_product_outputtypes (
            gmhdap_data_product_outputtype_id);

alter table gmh_user_roles
      add constraint gmh_users_gmh_user_roles_FK1 foreign key (
            gmhusr_user_id)
       references gmh_users (
            gmhusr_user_id);

alter table gmh_user_roles
      add constraint gmh_roles_gmh_user_roles_FK1 foreign key (
            gmhrol_role_id)
       references gmh_roles (
            gmhrol_role_id);

alter table gmh_data_product_permissions
      add constraint gmh_roles_gmh_data_product_permissions_FK1 foreign key (
            gmhrol_role_id)
       references gmh_roles (
            gmhrol_role_id);

alter table gmh_data_product_permissions
      add constraint
gmh_standard_data_products_gmh_data_product_permissions_FK1 foreign key (
            gmhsdp_standard_data_product_id)
       references gmh_standard_data_products (
            gmhsdp_standard_data_product_id);


-- This is the end of the VisioModeler 3.1 generated SQL DDL script.
```

***Support of Gulf of Mexico Hydrate Research Consortium: Activities to Support Establishment of Seafloor Monitoring Station***

# TECHNICAL PROGRESS REPORT

Reporting Period Start Date: October 1, 2005

Reporting Period End Date: March 31, 2006

Principal Investigator (Author): Bob A. Hardage

Date Issued: May 15, 2006

**DOE Cooperative Agreement No. DE-FC26-02NT41628**

Task 6: Seismo-acoustic characterization of seafloor properties and processes at the hydrate-monitoring station

Submitting Organization:
Bureau of Economic Geology
John A. and Katherine G. Jackson School of Geosciences
The University of Texas at Austin
University Station, Box X
Austin, TX 78713-8924

## Abstract

We have developed software that implements a new theory to create higher-resolution P-SV images of near-seafloor geology from large volumes of 4C OBC seismic data. Our previous software was experimental code that was structured to analyze only small data sets. This report summarizes this software development and illustrates that the quality of the P-SV image that can be generated is identical to that achieved with the initial experimental code.

## Introduction

Work subcontracted to the Bureau of Economic Geology (Bureau) has been expanded to allow the Exploration Geophysics Laboratory (EGL) at the Bureau to process and interpret all horizontal-array and vertical-array seismic data acquired at the seafloor observatory constructed across Block MC118. Work during this quarter has focused on developing software that will generate optimal-resolution P-SV images of data acquired with multicomponent seismic sensors positioned on the seafloor.

## Executive Summary

Researchers at the EGL have been subcontracted to create deliverables for Work Task 6, *Seismo-acoustic characterization of seafloor properties and processes at the hydrate-monitoring station*. Investigations have shown that standard P-SV imaging of data acquired using 4C seafloor sensors does not produce the resolution of near-seafloor geology that is desired for this project. EGL scientists have developed a new concept for processing deep-water 4-OBC data that yields a significant improvement in the spatial resolution of P-SV images. The fundamental concept of this new theory is to abandon the conventional approach of common-conversion-point binning and to focus on simple common-receiver gathers to construct P-SV images. An example of our first-generation P-SV data processing was illustrated in our report of May 2005. We have now modified that test code so that it can process large volumes of horizontal-array data. We show here that this modified software produces the same quality image as that which excited us when we processed data using our initial algorithm.

## Experimental

Experimental activity during this period focused on developing and testing software that creates high-resolution P-SV images of near-seafloor geology from deep-water 4C OBC seismic data.

## Results and Discussion

For better resolution of geologic targets using seismic data, it is necessary to acquire data that have shorter wavelengths. The wavelength **λ** of a propagating seismic wave is given by

$$\lambda = V/f,$$

where **V** is propagation velocity and **f** is frequency. This equation shows that there are two ways to reduce an imaging wavelength **λ**: either (1) increase **f** or (2) reduce **V**.

### Option 1: Increasing the frequency

If deep-water strata are illuminated with conventional air-gun seismic sources towed at the sea surface, there is really no way to cause a significant increase in the frequency content of the illuminating wavefield that reaches the seafloor. A different data-acquisition strategy has to be used to acquire shorter-wavelength marine P-P data. An approach now used for acquiring deep-water, short-wavelength P-P data is to use an Autonomous Underwater Vehicle (AUV) system. An AUV travels only 50 meters or so above the seafloor and illuminates seafloor strata, with chirp-sonar pulses having frequency bandwidths of 2 to10 kHz. This increase in signal frequency shortens P-P wavelengths by about a factor of 100, as compared with wavelengths of an air-gun signal. The result is an illuminating wavefield having wavelengths of less than 1 meter, when P-wave velocity $V_P$ is 1,500 to 1,600 m/s, a common range of $V_P$ for deep-water, near-seafloor sediments across the Gulf of Mexico (GOM). An example of an AUV chirp-sonar image acquired in water depths of approximately 900 meters in the Green Canyon area of the GOM is shown in Figure 1a. These high-frequency P-P signals penetrate only 40 or 50 meters into the seafloor, but they image bedding and fault throws of meter-scale dimensions across this image space.

### Option 2: Reducing the velocity

It is not possible to acquire shorter-wavelength P-P data by reducing $V_P$ in a seismic propagation medium. The value of $V_P$ within a system of targeted strata is fixed and cannot be altered. However, a seismic imaging effort can switch from the conventional approach of using the P-P seismic mode and focus on using another wave mode that does have reduced velocity within a targeted interval. That logic has great benefit for imaging deep-water, near-seafloor geology when the imaging focuses on P-SV rather than P-P data. Across most deep-water areas, S-wave velocity $V_S$ in near-seafloor sediments tends to be 20 to 50 times less than P-wave velocity $V_P$. Therefore, if P-P and P-SV data have equivalent frequency content, which they do for shallow penetration distances of an illuminating P-P wavefield into the seafloor, P-SV data will have wavelengths much shorter than P-P wavelengths.

Shown as Figure 1b is a P-SV image constructed from 4-C data acquired with seafloor sensors deployed along the same profile as the AUV data in Figure 1a. The illuminating wavefield that created these P-SV data was a 10- to 100-Hz P-P wavefield produced by a conventional air-gun array positioned at the sea surface. Because $V_S$ in near-seafloor sediment along this profile is less than 100 m/s, the P-SV data have many wavelengths less than 1-meter in length, just as do the high-frequency chirp-sonar data. Visual inspection of the images in Figure 1 shows the spatial resolutions of kilohertz-range P-P data, and low-frequency P-SV data are equivalent in deep-water, near-seafloor geology. The same data are shown again in Figure 2, with depth-equivalent horizons superimposed to emphasize the amazing resolution of the low-frequency P-SV data. Horizon **A** shown on the AUV image is not easily seen on this particular P-SV image display, so no P-SV equivalent horizon is labeled. Note the large magnitudes of interval values of the $V_P/V_S$ velocity ratio. Also note how easy it is to identify where stratigraphy first becomes unconformable to the seafloor in these seafloor-flattened data (Horizon **B**).

Unfortunately these high-resolution P-SV images cannot be extended to great subseafloor depths. P-SV wavelengths increase and P-SV resolution then decreases with increasing depth below the seafloor because

1. $V_S$ increases with depth, and

2. Higher frequencies attenuate more rapidly with depth for P-SV wavefields than for their companion P-P wavefields.

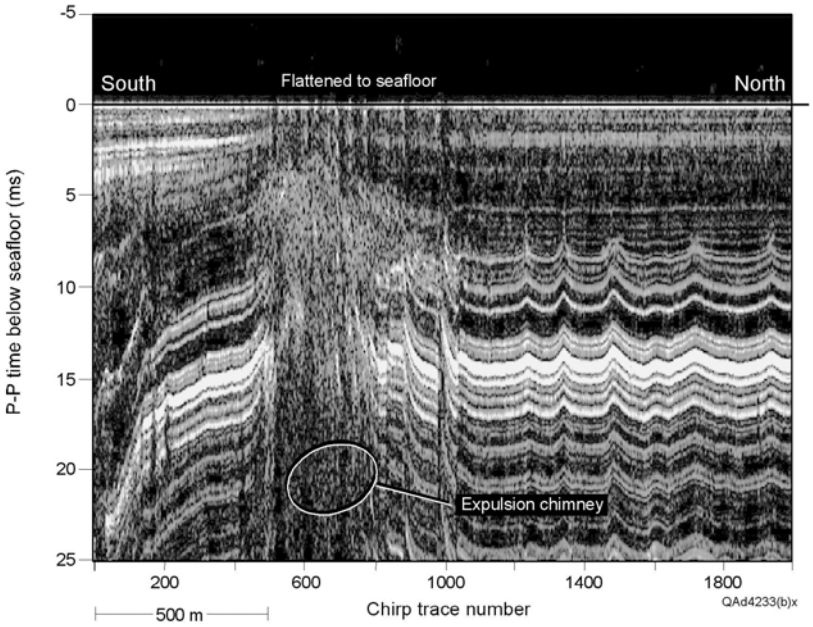
At subseafloor depths of several kilometers, P-P and P-SV data have approximately the same resolution. However, for deep-water strata close to the seafloor, the spatial resolution of P-SV data is most impressive (Figures 1b and 2b).

## Conclusions

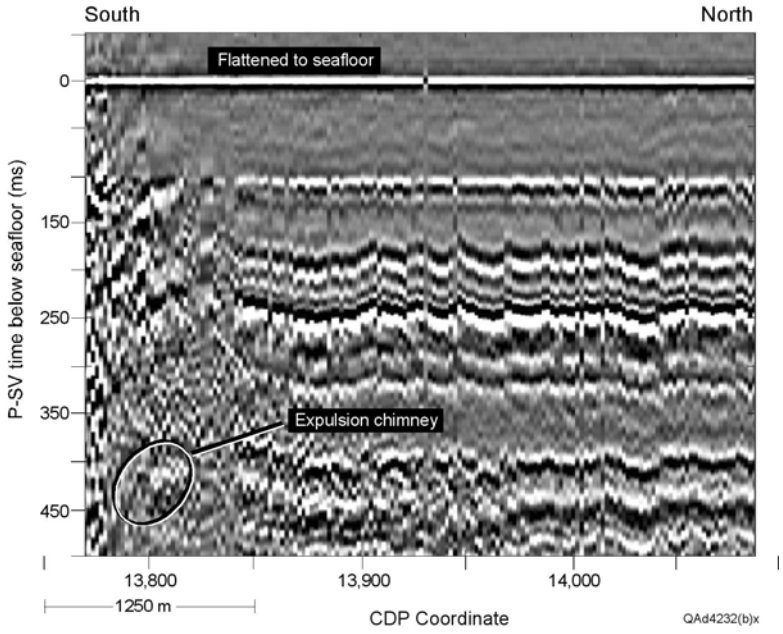
Important software needed for ongoing research at the hydrate-monitoring station has been developed and tested using data similar to those expected to be acquired across Block MC 118. Test results indicate that the software that has been developed is robust and creates higher-resolution P-SV images of near-seafloor geology than what can be provided by commercial contractors.

## References

None.

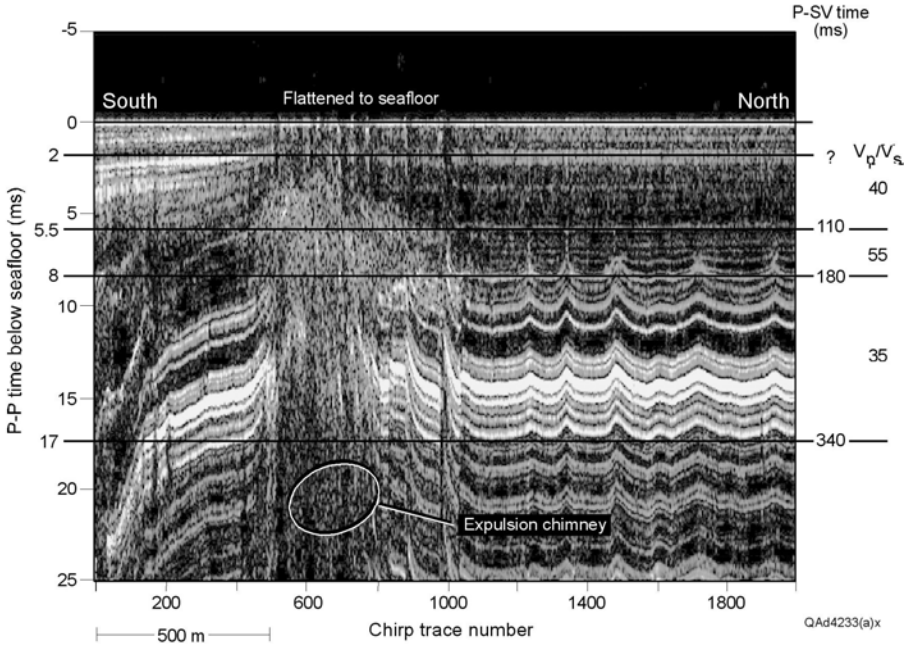



(a)

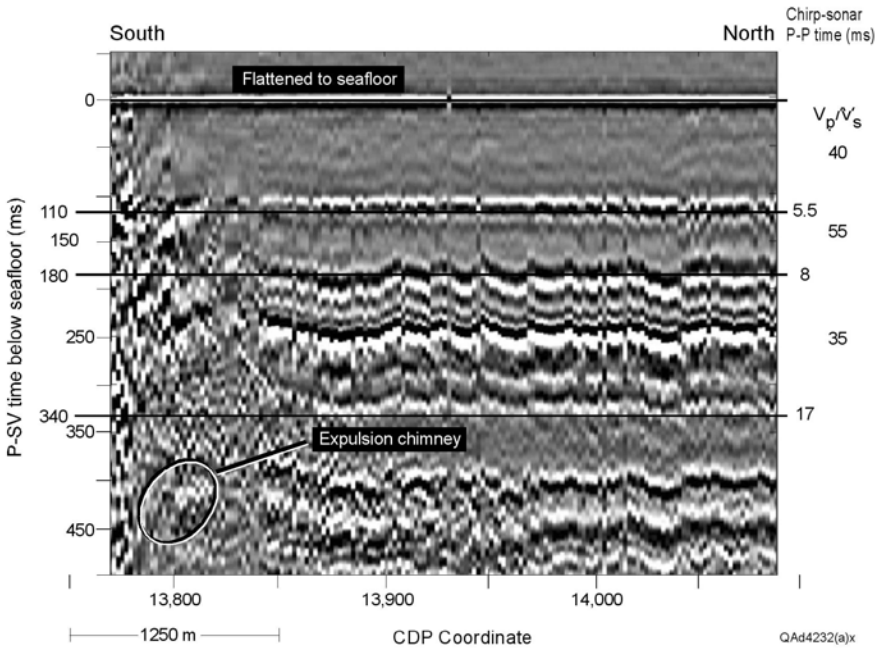



(b)

Figure 1. (a) High-frequency (2- to 10-kHz) AUV P-P image of near-seafloor strata across a fluid expulsion chimney. (b) Low-frequency (10- to 100-Hz) P-SV image produced by our new code along the same profile. Visual comparisons show that the images have equivalent spatial resolutions and, thus, equivalent wavelength spectra. The south end of the P-SV profile starts at about AUV chirp-trace number 700. These images have been flattened to the seafloor, which causes small-throw faults (throws of 1 meter and less) to appear as chevron-shaped patterns.




(a)




(b)

Figure 2. The same images as in Figure 1 with depth-equivalent horizons defined. A P-SV horizon equivalent to the AUV horizon at 2 ms is not labeled because the P-SV reflection event is quite faint in this P-SV display format. Interval values of $V_P/V_S$ velocity ratio between P-SV horizons are labeled in the right-hand margins. P-P and P-SV image times are labeled on opposing sides of the images.

# Coupling of Continuous Geochemical and Sea-floor Acoustic Measurements

## Progress report for the period

## October 1, 2005 – March 31, 2006

## May 15, 2006

*Jeffrey Chanton[1] and Laura Lapham[2]*

*[1] Department of Oceanography, Florida State University, Tallahassee, FL*

*[2] Department of Marine Sciences, University of North Carolina, Chapel Hill, NC*

**Abstract**
Several acoustic wipe-out zones were found within a 1km diameter region at Mississippi Canyon Lease Block 118, Gulf of Mexico, 100 miles offshore of Louisiana. These zones are thought to be indicative of active methane venting and therefore these sediments potentially contain gas hydrate. To test this, we collected thirty cores within this region both outside and within the wipe-out zones and measured for dissolved methane concentrations. We also addressed methane source and microbial controls on methane venting to the overlying water by determining the characteristic biogeochemical depth zonation of sulfate, methane and bicarbonate concentrations and stable carbon isotopes. As expected, acoustic and geochemical data outside the wipe-out zones were consistent with no venting activity. Within the wipe-outs, three distinct vent types could be characterized that suggest an episodic venting history at MC 118, adapted from Roberts et al., 1997. A “mud prone” vent had wipe-out zones with sharp edges and a flat, consistently deep layer above it. Although methane concentrations at this vent were low, its source was purely thermogenic and suggested a new vent whose microbial community has not yet established. A “transitional” vent had similar wipe-out zone characteristics with high concentrations of mixed methane sources (both biogenic and thermogenic source) suggesting active microbial communities at shallow sediment depths. Sulfate was depleted and methane began to increase at 50cmbsf. At this depth, the methane isotopic signature was depleted in 13C, around -90‰, indicating a biogenic methane source through the $CO_2$ reduction pathway. Deeper in the core, the isotopic composition shifted to -49.73 ± 1.11‰ (n=10), indicating a constant, thermogenic methane source. The third vent was characterized as “mineral prone” where the wipe-out zone had very rough seafloor topography, parabolic shaped features, and overlain by deeper sediment layers. Cores collected in this type of vent contained carbonate nodules and biogenic methane with no down-core variation in sulfate and methane. The presence of the carbonate and biogenic methane suggests previous microbial activity that has since been exhausted. This novel data set shows that acoustic wipe-out zones at MC 118 are not indicative of homogenous active methane venting at the sediment water interface. The combined geophysical and geochemical techniques direct sampling is needed to identify the current state of venting at a particular site.
Progress on Equipment Construction.

A second long PFA (8 m) has been constructed.

A third short PFA (0.5m, for high resolution sampling near a hard ground) has been constructed and will be deployed in the fall of 2006.

All three PFAs are designed for replacement of sampling coil units and pumps to allow sample recovery and pump recharge without disturbing the undersea floor portion of the probe. This design allows continuous geochemical monitoring.

We have 2 designs for pressurized pore water “peepers”. One design is undergoing laboratory testing and the second is under construction.

**Introduction**

Oceanic gas hydrate deposits form and reside in seafloor sediments where active methane venting or methane saturated sediments is occurring or has previously

occurred. These vents are typically identified on the seafloor as acoustic anomalies in geophysical records due to the occurrence of hydrate, gas bubbles, or carbonate that either scatter or absorb the acoustic signal. However, the presence of carbonate suggests an older vent that may no longer contain saturated gas needed to form or stabilize hydrate. Therefore, acoustic anomalies such as wipe-out zones may not indicate methane venting that is active. A combined approach of geophysics and geochemistry will help elucidate venting features at MC 118 to aid in prediction of hydrate occurrence.

Clathrates or gas hydrates are ice-like structures composed of light hydrocarbon gases, mainly methane, enclosed in frozen water molecules. They naturally occur in continental shelf sediments and arctic permafrost where saturated gas concentrations are found in areas of high pressure and low temperature. Based on several independent models, seismic surveys and direct measurements, worldwide hydrate deposits are estimated to contain around $10^{16}$ $m^3$ carbon, about twice the amount of carbon stored in fossil fuel reserves (KVENVOLDEN, 1988). Although this large methane reservoir is included in the global methane budget, they are assumed to be a constant small source of methane to the atmosphere. However, hydrates may be more dynamic than originally thought and act as a microbially mediated capacitor that releases large amounts of methane over short time scales (DICKENS, 2003). Therefore, techniques are needed to locate sediments conducive for hydrate formation and microbial processes controlling hydrate stability must be evaluated.

Gas hydrate deposits may be more dynamic than originally thought due to temporal changes in fluid flux rates. Since microbial processes may control the release of methane to the overlying water column in a moderately fluxing site, it is important to understand how these changes affect these microbial processes. Furthermore, the acoustic signatures of active vents need to be better constrained. Therefore, we sought to characterize the dissolved gas distribution, sediment structure, and carbonates in surficial sediments overlying acoustically anomalous zones at MC 118 and to determine the methane source and biogeochemical processes affecting it. We tested the hypothesis that acoustic wipe-out zones are methane-rich indicative of surficial sediment which host microbial processes such as sulfate reduction, AMO and MP. To this end, we collected sediment gravity cores within and outside several acoustic wipe-out zones at Mississippi Canyon 118, Gulf of Mexico and analyzed for methane concentrations and characteristic down-core biogeochemical zonation of sulfate, methane and bicarbonate. This is the first study to approach methane venting with combined geophysical and geochemical techniques at MC 118 and will elucidate the geochemical expression at the sediment surface of acoustic wipe-out zones as well as possible microbial controls on methane venting.

## Methods

### *Core collection*

In May and October 2005, twenty-nine gravity cores (<4.5m in length) were collected off the Research Vessel (R/V) Pelican (operated by Louisiana University Consortium) at MC 118 (Appendix 6.1). Chirp sonar subbottom profile data was used to target core positions. However, exact core positions were determined by triangulating three surface positions during the May deployment. In October, an ultra short baseline

transponder was connected to core wires to target exact locations of cores.

Once shipboard, gravity cores were sliced lengthwise. One half was used for core descriptions and the other sampled for geochemistry. Geochemical samples were collected by using 3 mL cut-off plastic syringes to collect sediment plugs (6-9 mL) at ~50 cm intervals. Plugs were placed in 30 mL glass serum vials, capped, and frozen for future analysis of methane concentrations, stable carbon isotopes and porosity. Remaining sediment from each interval was then collected into 2-50 mL centrifuge tubes and frozen for later analysis of pore-water ion (sulfate, chloride, nitrate, phosphate, and ammonium) concentrations and dissolved inorganic carbon (DIC) isotopic compositions. Immediately back at the laboratory (<3 days after cruise), tubes were thawed and centrifuged at 3000xg for 5 minutes. Resultant supernatant from one set of tubes was transferred to 2mL o-ring sealed plastic microcentrifuge tubes and frozen for future analysis for ion concentrations. The supernatant from the other set of tubes was injected into evacuated ~15 mL glass serum vials and measured at the Florida State University isotopic laboratory for determination of DIC stable carbon isotopic compositions. Isotopic compositions of DIC were only measured on push core 4414 and October 2005 cruise samples.

In August 2002, as part of a Gulf of Mexico Hydrate Research Consortium cruise, the Johnson Sea-Link (JSL) submersible visited MC 118 on JSL dive 4414. The dive site contained outcropping hydrate, bacterial mats, and clam shells. A sediment push core (PC4414) was collected ~2 meters from hydrate outcrop using 30cm long, 5cm diameter Lexan core barrels (28º51.1319N, 88º29.5502W; core position known by submersible navigation on seafloor). The push core was sectioned every 2-3 cm and pore-waters were expressed by pressure filtration (REEBURGH, 1967). Two-milliliters of pore-water was stored at 4ºC in 2mL o-ring sealed plastic microcentrifuge tubes for later analysis of dissolved sulfate and chloride by ion chromatography (CRILL and MARTENS, 1983) while the remaining volume was placed in evacuated glass vials for determination of dissolved inorganic carbon (DIC) carbon isotopic composition. Remaining sediment patties from the pressure filtration were frozen in plastic bags for future determination of total organic carbon (TOC) concentrations and isotopic composition.

## *Analytical methods*

Core pore-waters were measured for dissolved methane concentrations on a Shimadzu Mini II Gas Chromatograph (GC). A 5 mL gas aliquot was needed for each GC injection. To ensure no isotopic fractionation while collecting sample aliquots, each sample vial was injected with 10 mL of methane-free deionized water to displace 10 mL headspace needed for two GC injections. Dissolved gas was extracted via agitation of the sample vials. Five milliliter headspace aliquots were then taken and injected into an external sample loop (1 mL) of the GC. Constant volume samples were then carried into the GC with a helium stream. Peak areas were integrated and compared to Scotty methane standards (101.6 ppm $CH_4$).

Aliquots of the remaining methane and previously prepared dissolved inorganic carbon (DIC) samples were then analyzed for carbon isotopic signatures using a gas chromatograph-isotope ratio mass spectrometer (GC-IRMS) with a Hewlett-Packard 5890 GC equipped with a 6 m Poropaq Q column set at 30°C and a Finnigan Mat 252 IRMS at Florida State University. DIC samples were prepared by adding $H_3PO_4$ to

release the DIC from solution. Microliter volumes of headspace gas were directly injected onto the GC column and introduced to the mass spectrometer. Methane and carbon dioxide standards were run to calibrate the GC-IRMS.

Sulfate and chloride concentrations were measured by diluting 100 µL of sample with 10 mL eluent (carbonate buffer) and injecting 1 mL into a Dionex Ion Chromatograph. Standards ranging from 10-100% seawater were run for calibration curves each day and sample concentrations were determined from this curve.

Sediment patties were freeze-dried and ground for determination of total organic carbon concentrations and carbon isotopic composition. Approximately 30 mg of sample was introduced by flash combustion into a Carlo Erba Elemental Analyzer and resultant carbon dioxide flushed into the Finnigan Mat 252 IRMS system as described above. Nutrients (nitrate/nitrite, phosphate, and ammonium) were measured on a Lachet QuickChem 8000 autoanalyzer using colourmetric techniques. Dilutions of 48µM ammonium, 8µM phosphate, and 4µM nitrate+nitrite were run as the standard curve. Nutrient concentrations were only measured on the May 2005 cruise samples (cores 1-10).

Visually observed carbonate nodules and shell fragments were hand-picked from cores and stored frozen for stable carbon isotopic ratio analysis. At the Florida State University Isotope Lab, the nodules and shells were rinsed with tap water and then dried at 60ºC for 12 hours. Samples were then ground and about 0.5 grams placed into a 20 mL glass serum vial, stoppered and capped. Then, each vial was flushed with nitrogen for a few minutes and ~1 mL of 85% phosphoric acid was added to fully dissolve the sample as $CO_2$ gas. Five microliters of the evolved $CO_2/N_2$ gas mixture was then directly injected into the mass spectrometer for isotopic analysis. Samples were run in duplicate.

**Results**

The MC 118 site had been shown from a previous geophysical acoustic survey to contain several wipe-out zones (Fig. 3, (SLEEPER et al., 2006)). These wipe-out zones were targeted to assess their biogeochemical processes by collecting thirty sediment cores. The cores were mapped to the GPS coordinates of the survey and show that fifteen cores were collected outside and fifteen cores were collected within the wipe-out zones. Core pore-waters were measured for methane, sulfate, chloride and nutrient concentrations, methane isotopes, dissolved inorganic carbon isotopes, as well as total organic carbon content and isotopic composition.

The cores collected outside the wipe-out zones resulted in low methane concentrations and high sulfate concentrations with little vertical depth variation (Fig. 4). Sulfate concentrations ranged between 32mM at the sediment water interface and 22.5mM at 200 centimeters below seafloor (cmbsf). For all fifteen cores, sulfate concentrations averaged 27.42±2.52mM over all depths. Methane concentrations ranged 0.2-1.5uM for all cores except core 32 whose methane concentration reached 10uM at 300cmbsf. Methane and dissolved inorganic carbon (DIC) isotopes were not measured because concentrations were too low.

Within the wipe-out zones, the core geochemistry varied greatly. However, the data was organized into three vent stages: early, middle, and late.

An early stage vent was defined by low concentrations of thermogenic methane that lacked an active microbial community. Core 30 represented such a vent (Fig. 5a).

Sulfate concentrations were 28.2mM at the sediment water interface and decreased to an average of 26.3±0.6mM from 16 to 121 cmbsf. Methane concentrations were <15uM throughout the core but exhibited a concave down profile, indicating upward fluid advection. The isotopic signature of methane was extremely enriched in $^{13}C$ and consistent with depth at a value of -38±4‰ (n=7). The isotopic trend in dissolved inorganic carbon (DIC) was decoupled from methane; it became steadily more depleted in $^{13}C$ from -18 to -32‰ with depth.

Middle stage vents were characterized by those cores containing high methane concentrations and active microbial communities. This definition corresponded to cores 9, 26, 31, and PC4414 (Fig. 5b). Generally, each core exhibited the same geochemical down-core trend where sulfate decreased and methane increased. These four cores have sulfate concentrations around 28mM at the sediment water interface and between 0.5 and 4 mM at the bottom of each core (~50 to 100 cmbsf). The region where sulfate and methane are both low is known as the sulfate-methane interface (SMI). At the SMI for each core, methane concentrations began to increase to a maximum of 4300uM. Gradients of these profiles are high, between 100-930uM/cm for sulfate and 180-800uM/cm for methane (Table 1). Although the SMI depth varies between cores, it coincides with a minimum in $\delta^{13}C\text{-}CH_4$ values for both methane and dissolved inorganic carbon (DIC), above and below this depth, the isotopic signature becomes enriched in $^{13}C$. These concentration and isotopic trends indicate active microbial sulfate reduction, anaerobic methane oxidation and methane production.

Specifically, for core 9 the SMI for core 9 was around 50cmbsf. It should be noted that methane concentrations reached equilibrium at 1atm of ~2000uM below 150 cmbsf due to gas dissolution during core retrieval. The SMI coincided with the subsurface isotopic minimum in methane of -70‰. Below this depth, the isotopic signature became more enriched in $^{13}C$ to a value of -49.73 ± 1.11‰ (n=10), indicating a thermogenic source gas from deep below. Above the minimum value, the isotopic composition became enriched in $^{13}C$ to -60.66 ± 0.20‰, suggesting anaerobic methane oxidation. For core PC 4414, the SMI depth of 10cmbsf was estimated based on the minimum value of -25‰ for the isotopic composition of dissolved inorganic carbon (DIC). Above and below this depth, isotopic values become enriched in $^{13}C$ to 0‰. In core 31, the SMI was around 100cmbsf where the methane isotope minimum was -100‰. Above and below this depth, the methane was enriched in $^{13}C$ to -80‰. The isotopic signature of DIC also followed the same trend with the minimum of -50‰ at the SMI. Above this depth, the $\delta^{13}C$-DIC reached -25‰ at the sediment water interface and below this depth, the value reached -37‰. The SMI in core 26 was at 27cmbsf. This depth corresponded to a methane isotope minimum of -73‰. Above and below this depth, the value became enriched in 13C to ~65‰ near the sediment water interface and -40‰ at 50 cmbsf, respectively. The isotopic signature of DIC also followed the same trend with the minimum -34‰ at the SMI. Above this depth, the $\delta^{13}C$-DIC reached -12‰ at the sediment water interface and below this depth, the value reached -21‰, at the bottom of the core.

Late stage vents were characterized by those cores containing low concentrations of biogenic methane that lacked an active microbial community. This definition corresponded to cores 2, 7, 8, 21, 22, 28, and 29 (Fig. 6a). Lack of microbial activity was inferred from high sulfate concentrations, low methane concentrations, and

concentration gradients more than an order of magnitude lower than those found at the active sites (Table 1). This pattern was seen in seven cores where sulfate concentrations were ~28mM at the sediment water interface and decreased to ~20mM by 150 cmbsf (centimeters below sea floor). Methane concentrations showed the inverse pattern with depth, concentrations were near zero at the surface and increased to ~20uM at 100 cmbsf. Cores 21, 22, and 29 were analyzed for methane and DIC stable carbon isotopes. These cores were collected ~200meters from each other and still exhibited similar isotopic values. $\delta^{13}C\text{-}CH_4$ ranged from -61 to -76‰ with an average of -68.6±4.1‰ (n=17). $\delta^{13}C$-DIC were decoupled from the methane isotope values but were similar to each other. At the sediment water interface, isotopic values were around -10‰ and become more depleted in $^{13}C$ with depth to -25‰ at 100cmbsf.

Within this late stage vent category, we include the cores 25 and 37 which were collected that returned with little or no core. The cores hit near the center of the acoustic feature on line 120 (Fig. 3b). The core catcher in core 25 was bent and contained carbonate rocks in it, suggesting it hit carbonate hard bottom. Core 37 also hit hard bottom and the core catcher was caked in dark mud, possibly because it hit hydrate instead of carbonate. Neither sample was analyzed further due to lack of material.

The final core we also categorized as coming from a late stage vent, however it suggests a possible transition between middle and late stage vents. Core 24 penetrated 15cmbsf and contained oil and carbonate rock on the bottom. Sulfate was depleted by the end of the core yet methane concentrations were low (Fig. 6b). The sulfate gradient was more like the active microbial communities and the methane gradient was between an active and exhausted community (Table 1). The isotopic composition of methane was ~-53‰ and DIC was ~-25‰ near the sediment water interface, no clear trend with depth was seen because the core was too short. The isotopic composition of the carbonate was -25.01±0.04‰, similar to the DIC pool and suggests that a moderately active microbial community.

To evaluate the source of the organic matter, total organic carbon (TOC) concentration and isotopes and carbon to nitrogen ratios were analyzed (Fig. 7). There was little difference between samples collected within the wipe-outs zones and those collected outside. For all cores analyzed, the TOC ranged between 0.4 and 1.8% (Fig. 7a). The exception to this was PC4414 where %TOC increased from ~2.5% at the surface to 5% at the 30 cmbsf. $\delta^{13}C$-TOC varied between -33 and -22‰ (Fig. 7b) and the C:N varied between 10 and 20 (Fig. 7c).

There was no indication of brine fluid advecting from below (Fig. 8). Chloride concentrations averaged 543.71±24.7mM for all cores. This value is similar to seawater chloride concentration of 550mM.

Ammonia, phosphate, and nitrate/nitrite were measured in cores 1-10 (Fig. 9). Overall patterns of nutrient concentrations and wipe-out zones were not as obvious as methane concentration and isotope profiles. However, individual patterns within the data set were seen. For ammonia, sediment seawater interface concentrations had a large range between 26 to 123uM (Fig. 9a). Overlying water was not directly measured but typically, ammonia concentrations are low. Concentrations increased down-core for most cores but core 9 was consistently ~53±14uM until 300cmbsf where concentrations increased to 350uM within 150cmbsf. Phosphate concentrations for cores 1-8 were all

<5uM, yet concentrations were shifted to ~10uM for cores 9 and 10 at the seawater interface (Fig. 9b). Core 10 remained at this value down-core but core 9 increased to 25uM by 350 cmbsf, although there is scatter in the data. Nitrate/nitrite concentrations in all cores were between 0-30uM (Fig. 9c). Cores 7, 8, and 9 exhibited subsurface maxima of 15uM around 50 cmbsf.

The data presented in this study support the ideas that wipe-out zones are evidence of both active and past methane venting, combined techniques are needed to determine the vent stage, and active microbial processes may control the methane flux from vents. The detailed spatial survey of gas concentrations and microbial processes within the wipe-out zones suggested different possible causes of wipe-outs at MC 118.

**Figure 18: Location of Mississippi Canyon 118, Gulf of Mexico**

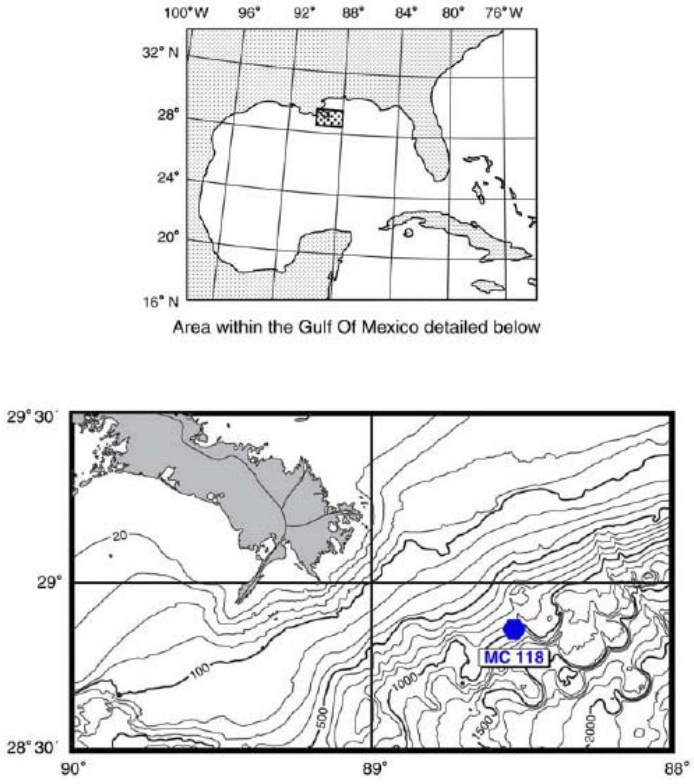


**Figure 19: Seafloor picture taken at MC 118 from submersible JSL dive 4414 in 2002. Bottom right hand shows clam shells, diagonally in the middle shows yellow outcropping hydrate with white bacterial mat surrounding it and the top left of the picture is sediment**

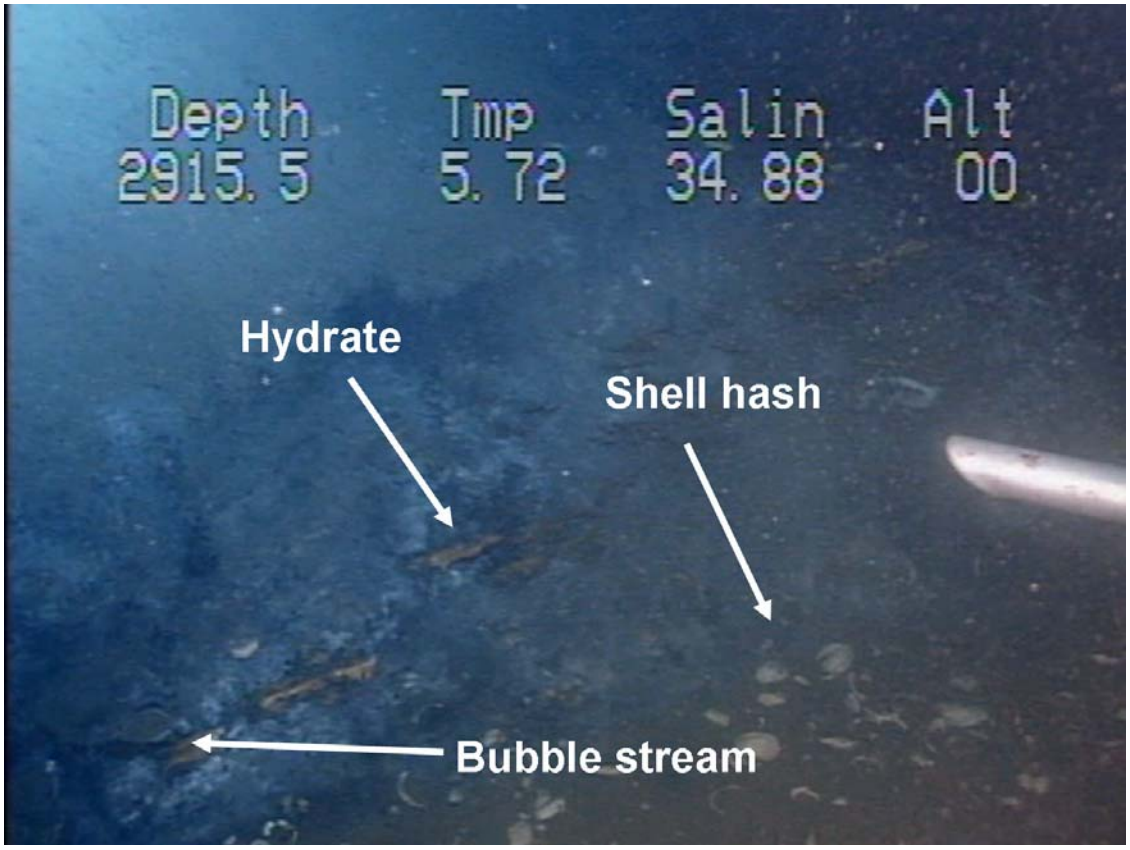
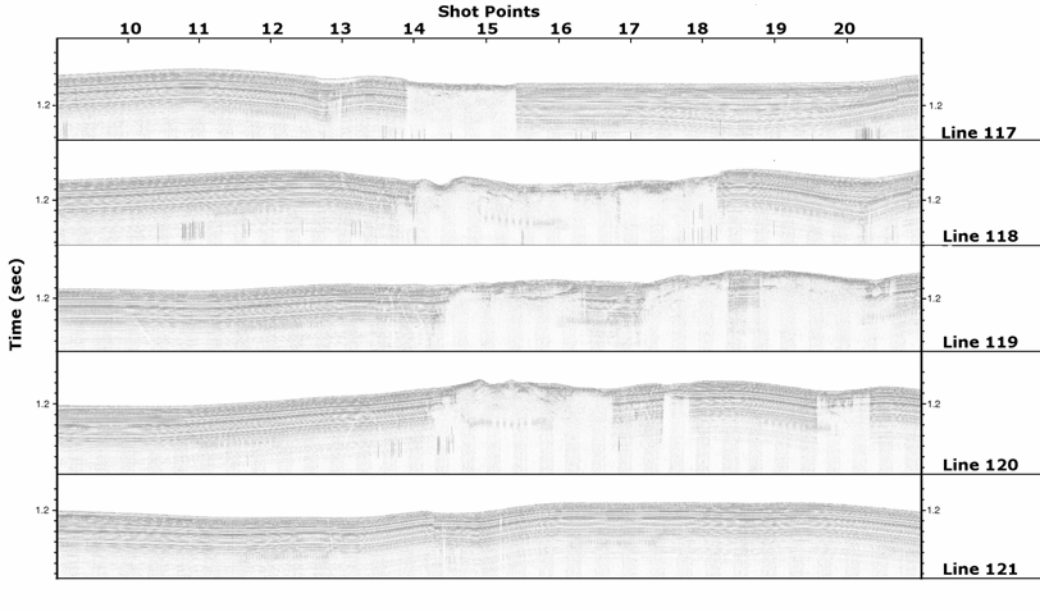


**Figure 20: Geophysical maps of MC 118. A) Chirp sub-bottom profiles for acoustic lines 117-121 with shot points for reference. The wipe-out zones can be seen as the areas missing visible stratified sediments, such as the region on line 117 between shot point 13.8 and 15.4. Notice that line 119 and 120 each have 3 wipe-out zones. B) Side scan sonar map of the region shown in part A. Superimposed are acoustic lines 117-121 with shot points 10-20 (white circles) for reference. White segments along acoustic lines represent the wipe-out zone regions as seen in A. White numbers are cores collected outside a wipe-out zone. The red, green, and yellow numbered cores are respectively the early, middle, and late stage vents.**



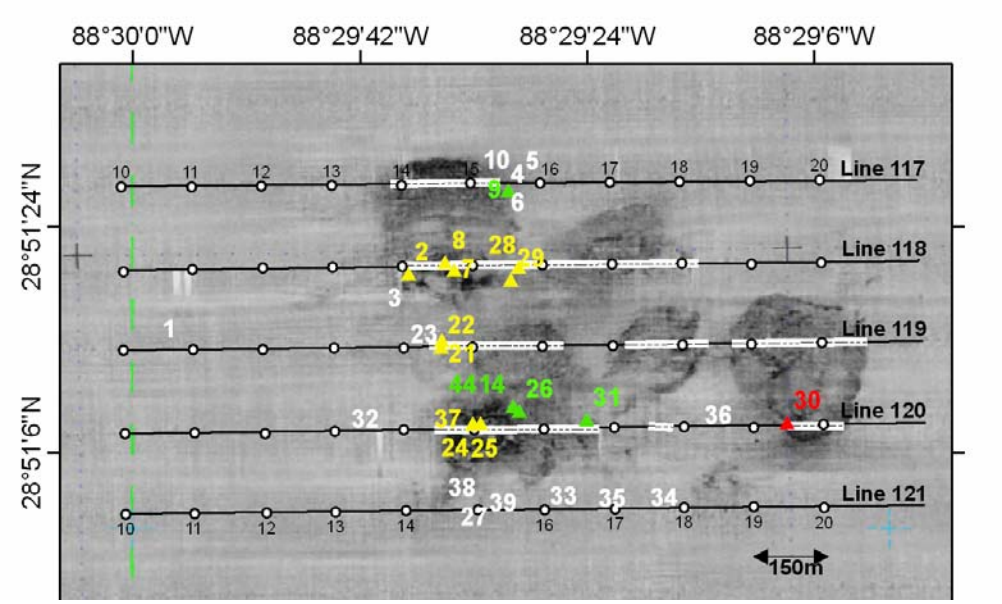



**Figure 21:Geochemical profiles of the 15 sediment cores collected outside the wipe-out zones. Thirteen cores with nearly identical profiles are shown as filled black squares, while cores 3 and 32 are shown in open symbols.**

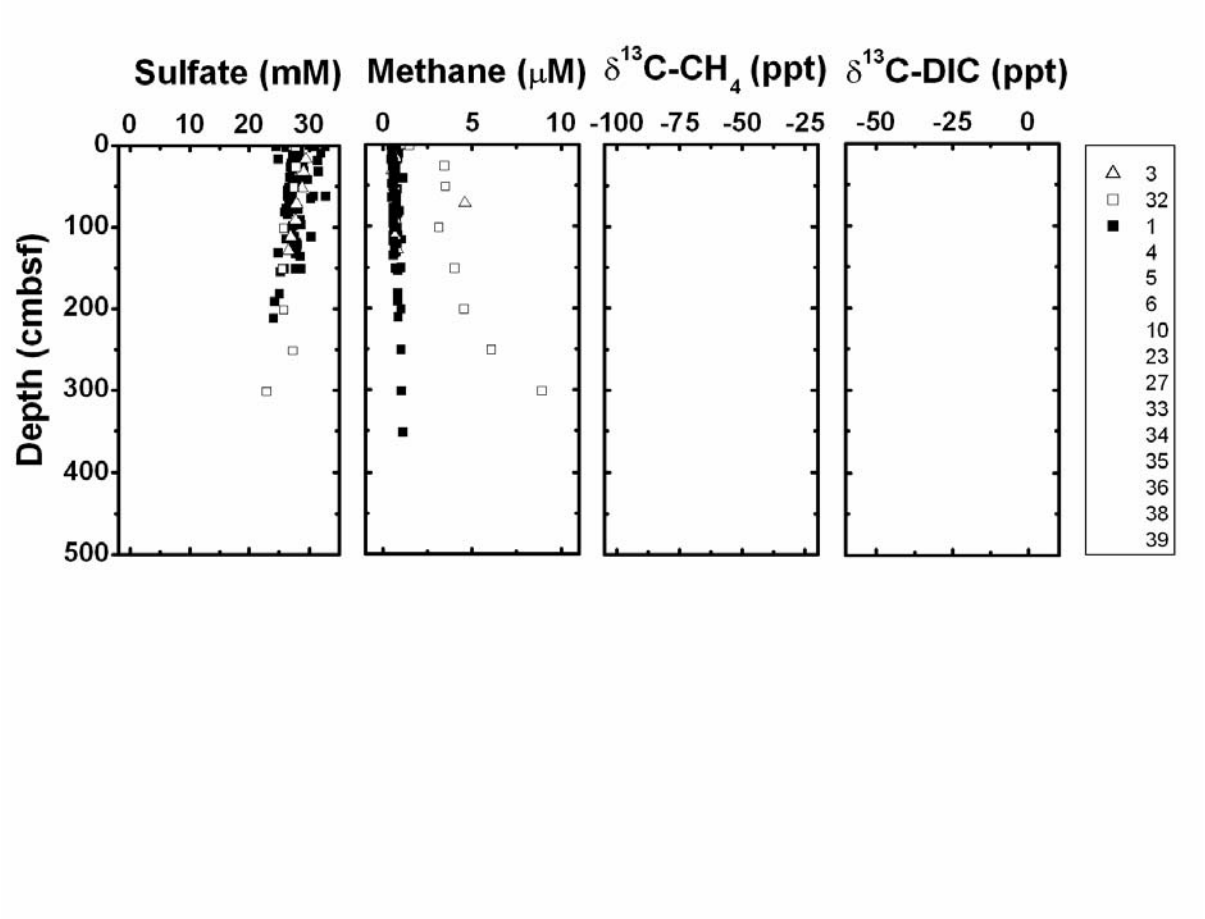


**Figure 22: Geochemical profiles from the wipe-out zones. A) Early stage vent represented by core 30. B) Middle stage vents were represented by four cores and contained evidence of active microbial communities. Please note methane concentration scale change.**

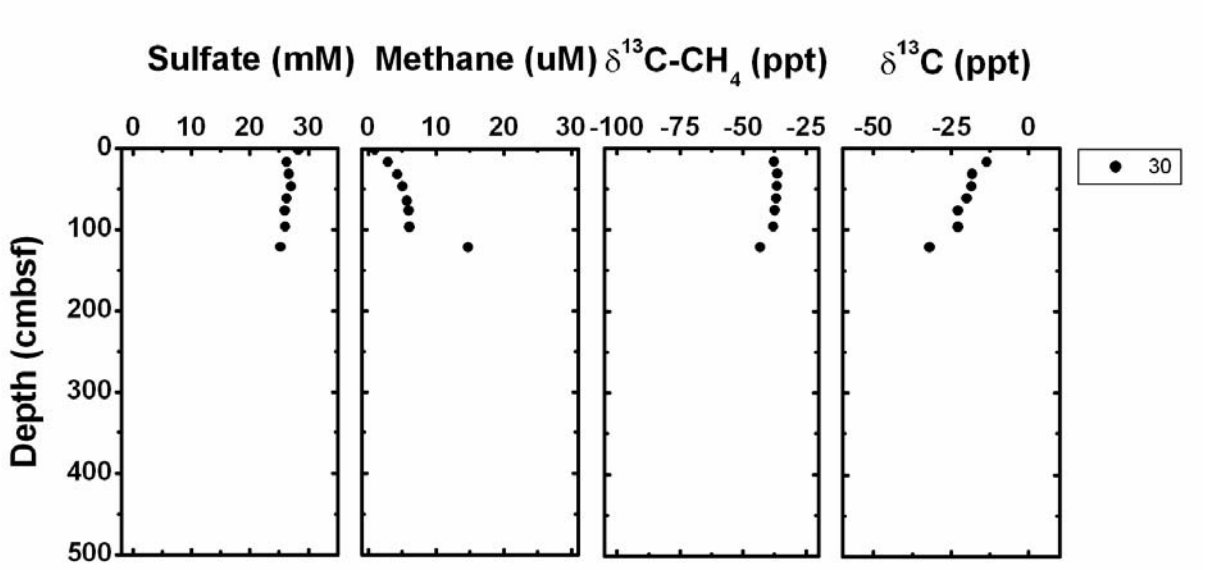


**Figure 23: Geochemical profiles from the wipe-out zones for late stage vents. While ten cores fall into this classification, cores 25 and 37 are not shown due to hitting hard carbonate near the surface. A) Seven cores were found with data which suggests limited microbial activity. B) One core may represent a transition to a late stage vent. Please note the depth and methane concentration scale changes from figure 5.**

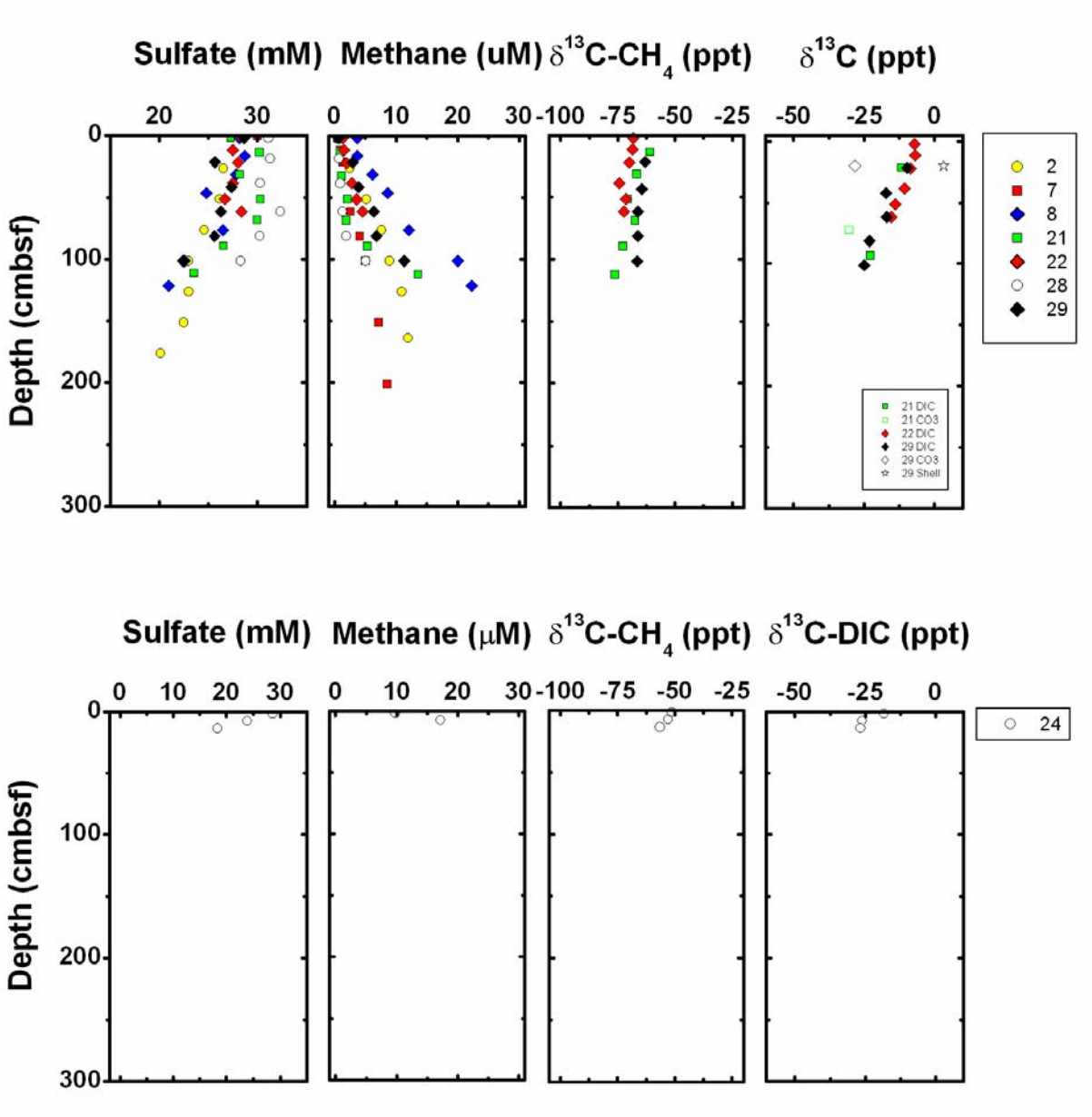


**Figure 24: Total organic carbon a) percentage, b) $\delta^{13}C$ and c) carbon to nitrogen ratios.**

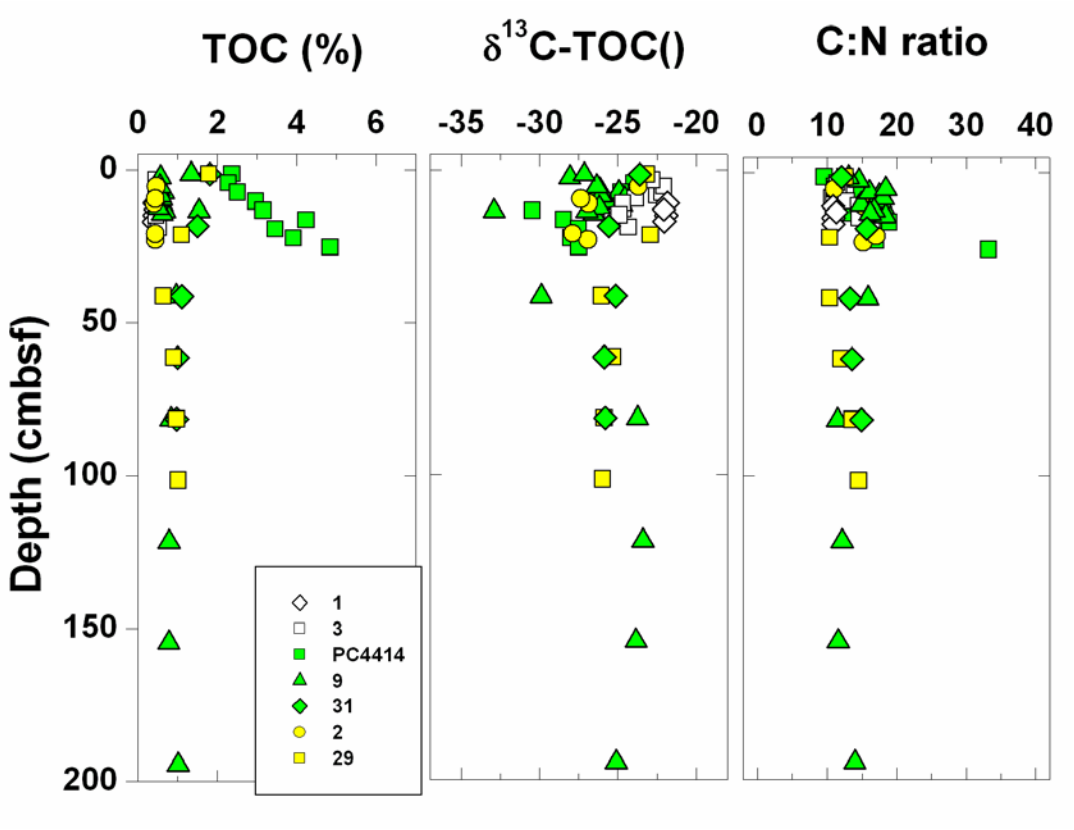



**Figure 25: Chloride concentrations for most cores. Average value is 543.71±24.7mM. For reference, seawater is 550mM.**

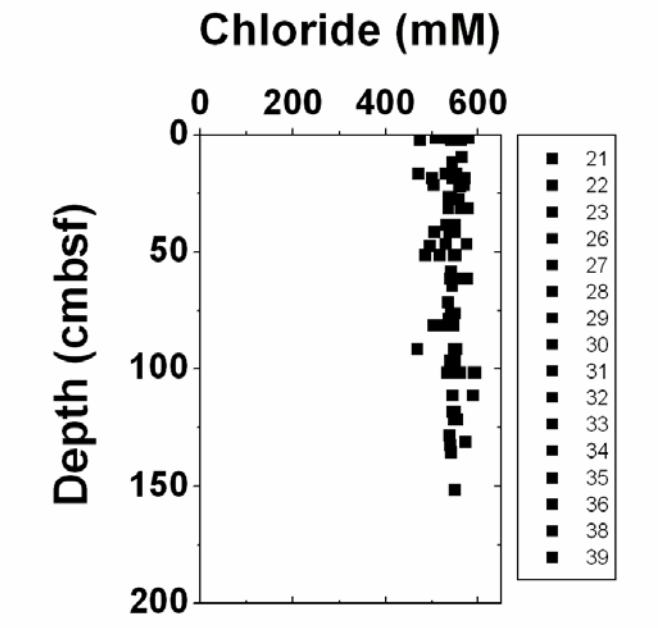
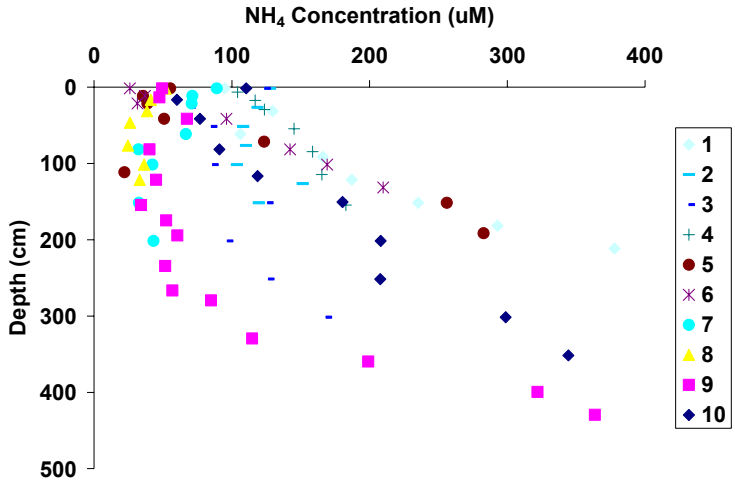
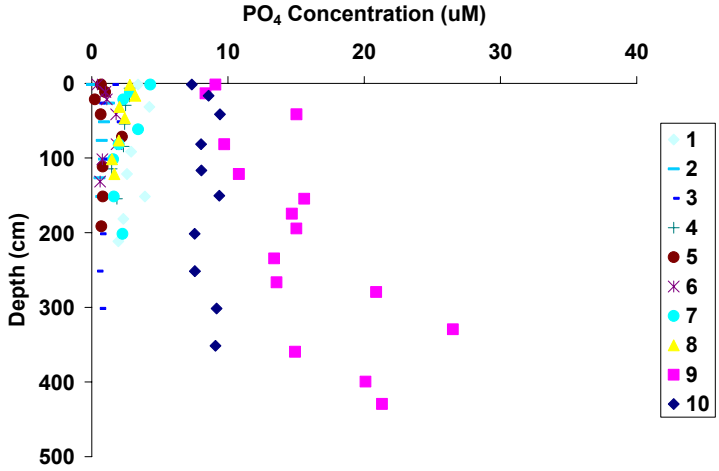
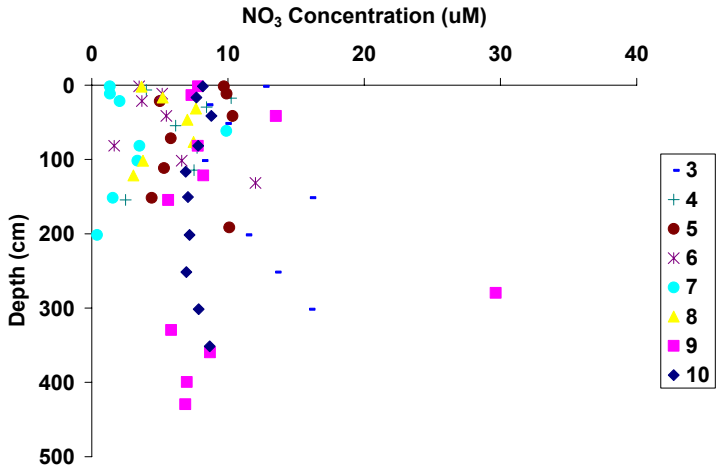


**Figure 26: a) Ammonia, b) nitrate/nitrite and c) phosphate concentrations for cores 1-10.**



# Laboratory Analyses and Comparison of Cores from MC118 and MC798

**Progress report for the period**

**October 1, 2005 – March 31, 2006**

**May 30, 2006**

*R.E. Rogers, J.L. Dearman*

*Swalm School of Chemical Engineering*

*P.O. Box 9595, Mississippi State, MS 39762*

## ABSTRACT

Sediments in zones of the Gulf of Mexico where hydrates occur may exhibit hydrate-forming properties that differ from non-hydrate-bearing sediments. If so, an analysis could not only be valuable in establishing gas-hydrate formation mechanisms in ocean sediments, but such data could provide guides in locating gas-hydrate deposits worthy of production. To evaluate this possibility, sediments from two locations in the Mississippi Canyon, MC-118 and MC 798, were analyzed for propensity to form gas hydrates. Hydrate formation rates and crystal initiation times were measured in the laboratory as a function of depth below seafloor and as a function of lateral displacement. Trends in these parameters are reported for 8 different cores. Interpretations suggest sulfate zone depth, bioactivity, and pore-water salinity are important in determining hydrate ease of formation in near-surface sediments.

## INTRODUCTION

The preceding quarterly report, 3rd quarter report 2005, detailed the hydrate-forming propensity of sediments from Core 11, MC-118, and Core 04, MC-118. Hydrate formations were determined from replicated laboratory tests to determine any trend with depth.

In the absence of cores as deep as the 30 m from the Dufresne cruise (1$^{st}$ quarter 2005), investigations were made of cores in the upper 6 m of various locations in the Mississippi Canyon. Specifically, tests on sediments of 8 cores from MC 118 and MC 798 are reported this quarterly to determine hydrate-formation variability near-surface as a function of lateral displacement.

It is helpful to remember approximate depths of the hydrate zone as compared to the depth of near-surface cores being analyzed. Typically, the hydrate zone depth would be about 200 mbsf in the Mississippi Canyon, whereas the Dufresne cores extended to 30 mbsf and available push cores reach to about 6 mbsf. In fact, most of the limited hydrate-formation data reported in the literature for GOM are restricted to the first few meters below seafloor, which is the case in this report.

## EXPERIMENTAL PROCEDURE

A few important features of the laboratory procedure for analyzing cores are listed below.

1. Twenty grams of mud samples are evenly dispersed through 60 grams of coarse Ottawa sand that has been cleaned. The reasons for doing this are the following: (a) to give maximum access of natural gas to the sediment samples, thus exposing maximum surface areas of the sediment particles to the reacting gas, (b) to provide larger porosities in which hydrates are allowed to form and expand, (c) to be able to utilize the small mud sample sizes.
2. The samples are tested for hydrate formation in Teflon containers. Teflon provides a hydrophobic surface that does not affect hydrate formation.
3. For testing, the samples are placed in annular spaces of concentric Teflon cylinders. Both cylinders have many gas-access holes spaced over their entire areas. This allows maximum contact of gas with sediments and prevents mass transfer from limiting hydrate kinetics.

4. Only original seawater removed with the cores is present in the tests.
5. Teflon containers with samples are placed in stainless steel Parr reactors, sealed, pressurized with natural gas and submerged in constant temperature baths maintained at 0.5°C.
6. Pressures and temperatures are recorded continuously during the tests.

## RESULTS AND DISCUSSION

As stated in the preceding quarterly report, formation rates and induction times determined in laboratory measurements of Gulf of Mexico (GOM) sediments (Dufresne MD02-2570 cores) show distinct trends to 30-m depths. In the current report, shallower cores from different locations in the Mississippi Canyon are compared.

Generally, when induction times decrease, hydrates begin forming more quickly. This can be important when gases percolate through sediments and have a limited residence time. After the hydrates are initiated, higher formation rates are indicative of favorable hydrates occurring in the sediments. The laboratory-generated hydrate curves were interpreted with help from the stratigraphic analyses of Dr. C. Brunner as well as the comments on the mud logs during core retrieval.

Patterns of Hydrate Formation with Sediment Depth

*Norm Pattern*. Patterns differ widely of laboratory-determined hydrate formation rates and hydrate induction times as a function of depth in sediments from GOM cores. However, patterns repeat in those cores composed primarily of homogeneous mud. (The lithostratigraphy was established by Dr. Charlotte Brunner, University of Southern Mississippi, during analysis of sediments from the same core.) See Figure 1, Figure 2, and Figure 3.

In the three cores composed of homogeneous mud, hydrate formation rates increased with depth until a maximum was reached as compared to surface values. The induction times show decreases with depth until a minimum is reached.

In an attempt to relate hydrate-forming propensity to sediment characteristics, it is useful to take these sediments where homogeneous mud is the dominant characteristic as a norm by which to evaluate other cores.

*High Salinity.* Compare Fig. 4 with the 'norm pattern' of Figure 1 through Figure 3. It is evident that formation rates decline and induction times increase with depth below the surface of Core 05 from MC 798 of Figure 4 (28° 4.0000 N, 89° 42.0003 W). These formation rate and induction time behaviors denote hydrates increasingly more difficult to form as depth increases. Dr. Brunner's analyses of samples from the same core found "high pore water salinity 200 cm to bottom of core…" The hydrate tests of these sediments, therefore, seem to verify the known retarding action of saline waters on hydrate formation.

*Sulfate Zone.* It is hypothesized that one of the most important factors in hydrate formation in near-surface sediments is the depth of the sulfate zone. The shallower the sulfate zone depth, the greater is the methane flux through the zone. The microbial communities working in the sulfate zone differ from those below the zone. Bioproducts differ in the zone, and biopolymers/biosurfactants have been shown in our laboratory to have a significant effect on hydrate formation. Because of the anaerobic oxidation of

methane in the sulfate zone and the reduction of sulfate, carbonate nodules as well as hydrogen sulfide occur there.

In Figure 5 (MC 118, Core 13, 28° 52.55' N; 88° 28.7' W) is the plot of formation rates and induction times as measured in the laboratory for sediments from Core 13 that reached a depth of 300 cm bsf. In the Mud Log at the time of coring was noted that between 56 to 130 cm disseminated shells occurred. The laboratory tests of hydrate formation in these sediments indicated a minimum was reached in both rate and induction time over the depth increment where the disseminated shells occurred. These carbonate nodules may typify the bottom of the sulfate zone, or where the bottom of the sulfate zone was at one time.

In Figure 6 (MC 118, Core 29; 28° 51.3293' N and 88° 29.4996' W) are the plots of formation rates and induction times for sediments from Core 29 that reached a depth of 120 cm. In the Mud Log at the time of coring was noted that shells occurred at 21 cm depth. Figure 6 indicates a maximum in both formation rate and induction time at this 21 cm depth. The Mud Log also noted a sulfur smell at 80 cm and 115 cm. Minimums occur in formation rates and induction times at these points.

In Figure 4 (MC 798, Core 05; 28° 4.0000 N and 89° 42.0003 W) are the plots of formation rates and induction times for sediments from Core 05 that reached a depth of 600 cm. Carbonate nodules were observed at 240 and 260 cm. At these depths, Fig. 4 indicates minima and maxima in formation rates and induction times.

Again, in Figure 7 with Core 06, MC 798 (28° 85.3904 N and 89° 39.4997 W) a maximum is reached on hydrate formation rate and a minimum on induction time at the point where fine-grained carbonates occur from a 140 cm to 240 cm depth interval according to Dr. Brunner's lithostratigraphy.

More data are needed for a conclusive interpretation, but these multiple cores indicate some substantial alterations in hydrate-formation propensity at depths where carbonate nodules or sulfur smell (probably $H_2S$) occur.

Microbial Activity

In Figure 8 for MC 118, Core 12 (28° 52.45' N and 88° 29.2' W), the Mud Log states the depth interval from 20 to 70 cm was bioturbated. The plots of hydrate formation rates and induction times from laboratory data show in this region an increase of formation rate and a decrease in induction time, consistent with the expectation that bioproducts cause the same sort of hydrate promotion as they do in the laboratory.

Similarly, in Cores 08A and 08B of Figure 2 and Figure 3 the laboratory hydrate formation rates and induction times from the sediments show an increase of formation rate and a decrease in induction time over the interval of 200 to 300 cm depth where Dr. Brunner's lithostratigraphy shows microbial action occurring.

## CONCLUSIONS

Variations in ease of gas-hydrate formation occur as one moves laterally or vertically at near-surface depths of the seafloor in Mississippi Canyon. Near-surface variations in laboratory hydrate formation within GOM sediments are possibly caused by multiple parameters. A major cause of the variations may be the different microbial activities in near-surface sediments dependent on magnitude of methane flux and the consequent depth of the sulfate zone. The work reported in the current quarter sought to characterize the sediments according to multiple parameter effects on gas hydrate formation. To do so, evaluations of gas hydrate formation properties were made in laboratory tests, and resultant curves of formation rate and induction time vs depth were established. In an attempt to explain trends, the curves were compared to comments from Mud Logs and from the stratigraphy analysis of Dr. C. Brunner at the University of Southern Mississippi.

In Table I is summarized the laboratory data of eight cores taken from MC 118 and MC 798 for their propensity to form gas hydrates.

**TABLE I.** Effects of six parameters on hydrate-formation vs depth for near-surface cores. (From hydrate experiments of J. Dearman, stratigraphy analysis of C. Brunner, and comments recorded in Mud Logs during core retrieval.)

| Core | $F_r$ $T_i$ | Bio-activity | Red Color | Homogeneous Mud | Carbonates | Sulfur | High Salinity |
|---|---|---|---|---|---|---|---|
| MC118 #12 | Fr<br>Ti | Increase<br>Decrease | | | | | |
| MC118 #13 | Fr<br>Ti | | Increase<br>Increase | | Minimum<br>Minimum | | |
| MC118 #29 | Fr<br>Ti | | | | Maximum<br>Maximum | Minimum<br>Minimum | |
| MC798 #04 | Fr<br>Ti | | | Norm<br>Norm | | | |
| MC798 #05 | Fr<br>Ti | | | | Min/Max<br>Min/Max | | <1<br><1 |
| MC798 #06 | Fr<br>Ti | | | | Maximum<br>Minimum | | |
| MC798 #08A | Fr<br>Ti | Increase<br>Decrease | | Norm<br>Norm | | | |
| MC798 #08B | Fr<br>Ti | Increase<br>Decrease | | Norm<br>Norm | | | |

It is concluded that near-surface sediments of primarily homogeneous muds exhibited a pattern of hydrate formation rates and induction times versus depth that were taken as a ‘norm’. Changes to the normal curves were interpreted in terms of the six parameters in Table I. Carbonate nodules are taken as indicative of the bottom of the sulfate zone, and hydrogen sulfide (sulfur smell) is taken as indicative of the transition region of the sulfate zone. Where these occurred in the sediments, a minimum and/or maximum occurred in the hydrate formation and induction-time curves,

possibly suggesting a significant change in hydrate-forming ease at that locale. Wherever bioactivity had been indicated, the formation rates increased and the induction times decreased, suggesting a boost to hydrate formation. Wherever high salinity of the pore waters was indicated, a hindrance to hydrate formation was indicated in the curves.

Analyses will continue to substantiate and possibly expand the interpretation of the hydrate curves.

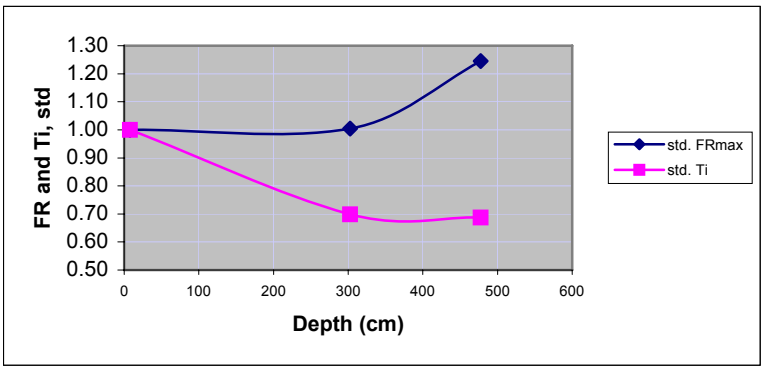



**Fig. 1.** Hydrate formation in Core 04 from MC-798 (28° 8.1176 N, 89° 39.6689 W)

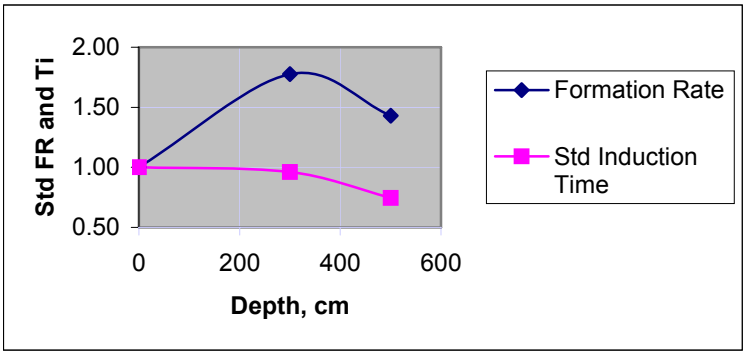



**Fig. 2.** Hydrate formation in Core 08A from MC-798 (28° 2.8914 N, 89° 44.4297 W)

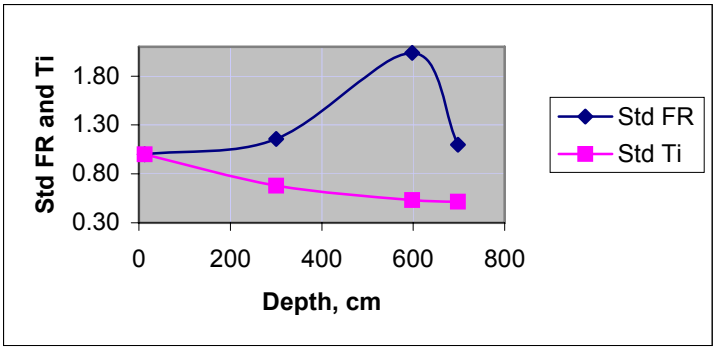



**Fig. 3.** Hydrate formation in Core 08B from MC-798 (28° 2.8183 N, 89° 44.4321 W)

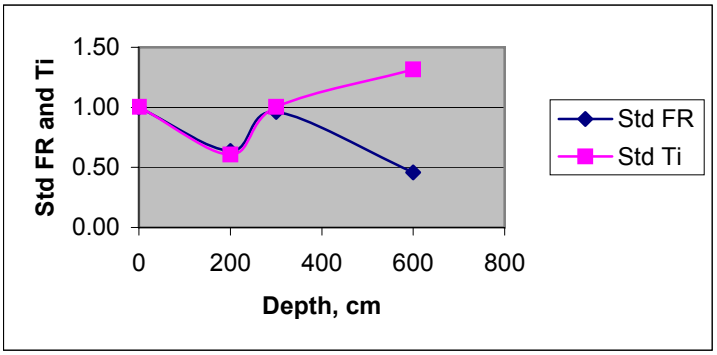



**Fig. 4.** Hydrate formation in Core 05 from MC 798 (28° 4.0000 N, 89° 42.0003 W)

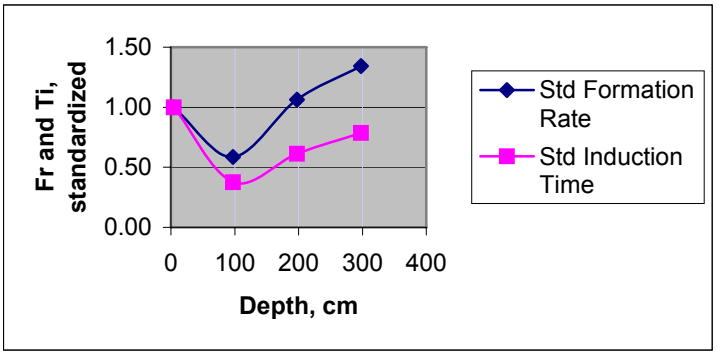



**Fig. 5.** Hydrate formation in Core 13 from MC 118 (28° 52.55' N, 88° 28.7' W)

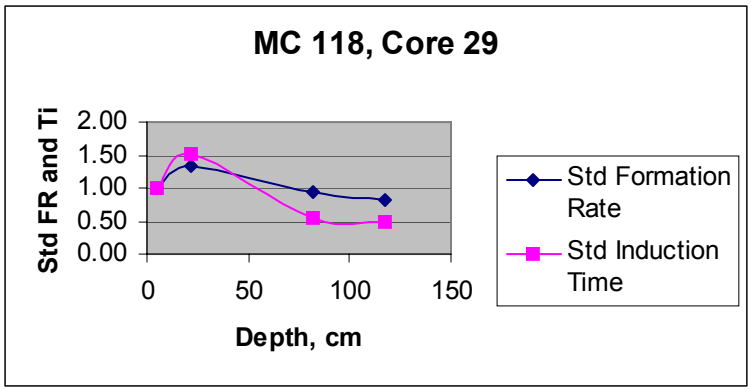



**Fig. 6.** Hydrate formation in Core 29 from MC 118 (28° 51.3293' N and 88° 29.4996' W)

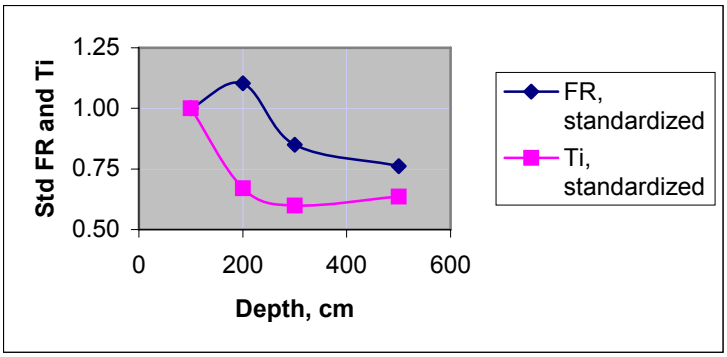



**Fig. 7.** Hydrate formation in Core 06, MC 798 (28° 85.3904 N and 89° 39.4997 W)

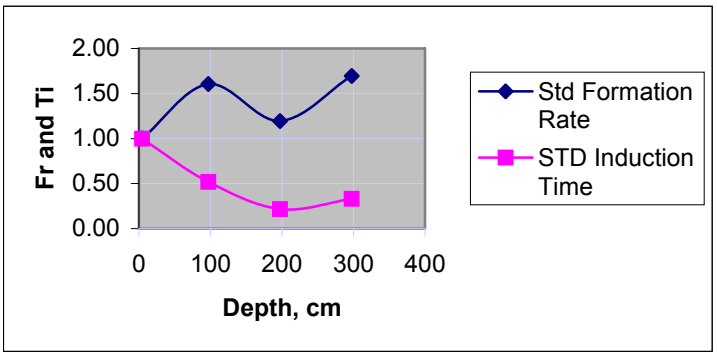



**Fig. 8.** Hydrate formation in Core 12, MC 118 (28° 52.45 N and 89° 29.2 W)

# EXPERIMENT TO GENERATE SHEAR WAVES IN THE SEA-FLOOR AND RECORD THEM WITH A HORIZONTAL LINE ARRAY

INITIAL SEMIANNUAL REPORT COVERING THE PERIOD

October 1, 2005 to March 31, 2006

Specialty Devices, Inc. 2905 Capital Street, Wylie, TX 75098

June, 2006

# EXPERIMENT TO GENERATE SHEAR WAVES IN THE SEA FLOOR AND RECORD THEM WITH A HORIZONTAL LINE ARRAY


## Abstract

The Gas Hydrate Sea Floor Observatory program includes the potential installation of P wave and S wave sensors on the sea floor in approximately 1,000m depths. A testing program is planned to install and test both the shear wave source and the shear wave receivers.


## Introduction

This experiment includes sea floor deployment of the line array constructed for eventual installation in a JIP borehole and to test it with the shear sled. Deployment will be near the observatory site selected in Mississippi Canyon Block 118 and will serve as a test of the horizontal line configuration as well as of the line array itself prior to its installation in a borehole. Major funding for this array is already in place so funding is only requested to ready the array and to conduct the test.

The function of the HLAs is to complete the three-dimensional seismo-acoustic framework that will make it possible to determine the locations of the energy sources of the seismo-acoustic data. The completed MS/SFO will include four 400m HLAs arranged into two orthogonal 800m lines interfaced with the IDP unit. Included in the design is a Data Acquisition and Telemetry System with the capability to acquire and store the high-volume digital data stream generated by the HLAs.

**Subtask 2.1: Deployment and testing of an array in the horizontal configuration**

The primary objective is to field test the devices, especially the array intended for the borehole, in an environment from which they can be recovered and altered, if necessary.

**Subtask 2.2: Shear wave experiment**

A secondary objective is to record and process shear waves generated directly into the sea floor and thereby acquire more data concerning the eventual design and construction of the four 400m HLAs to be incorporated into the sea floor observatory.

## Summary

The field tests have not been performed during this time frame and are delayed by storm related and software related issues. New tests could occur in the summer of 2006.

## Experimental

There have been several sources causing delay in the implementation and testing of bore hole 3C shear wave sensing equipment on the MMRI SLED.

The 3C accelerometer sensors were designed by Input Output Corp. to integrate into large land seismic systems. The sensors were designed with the communications and timing part of a large multiple failsafe ring communications net using multiple

communications packages set out on the land site. These land seismic arrays typically included many hundreds to thousand of sensors and costs for the complete system is typically more than $10M. For application to a small sea floor array, SDI has been allowed access to the technology used to communicate with the sensors through the I-O arrays and has received some support in efforts to make this system useable in a deep ocean environment. However in recent testing of the sensor communications system to be used on these sensors a Software and circuitry problem has become apparent and SDI is working with I-O to develop a solution to allow testing to proceed.

The weather problems of last year have removed from service the vessel planned for testing of this system. The R/V Kit Jones is soon to be back in operation and can again be available to use in this testing program.

## Results and Discussion

As this experiment has not been conducted this section will be completed when there are results to be presented.

## Conclusion

Lab testing will resume when the software communications problem is cured. A field exercise will be scheduled following completion of further lab testing.

**ACRONYMS**

| | |
|---|---|
| **1D** | one-dimensional |
| **2D** | two-dimensional |
| **3D** | three-dimensional |
| **4-C** | four-component |
| **ACDP** | acoustic-doppler current profiler |
| **AJAX** | Asynchronous Javascript with XML (interface) |
| **ALA (=VLA)** | acoustic line array |
| **Am1d** | structure used in ambiguity function generation |
| **Am2d** | structure used in ambiguity function generation |
| **Am3d** | structure used in ambiguity function generation |
| **AMO** | anaerobic methane oxidation |
| **Aray** | structure that holds the data for the array |
| **ASCII** | American Standard Code for Information Interchange |
| **AUV** | autonomous underwater vehicle |
| **BA** | bottom accelerometer |
| **BBLA** | benthic boundary layer array |
| **BCOMFI** | Barrodale Computing Matched Field Inversion |
| **BCS** | Barrodale Computing Services, Ltd. |
| **BEG** | Bureau of Economic Geology (University of Texas) |
| **BHA** (=BLA) | borehole array |
| **BLA** (=BHA) | borehole line array |
| **BS** | Battery System |
| **C&C** | Chance and Chance |
| **CDOM** | chromophoric dissolved organic matter |
| **CH4 (=$CH_4$)** | methane |
| **cmbsf** | centimeters below sea floor |
| **CMRET** | Center for Marine Resources and Environmental Technology |
| **CO2 (=$CO_2$)** | carbon dioxide |
| **CSA** | chimney sampler array |
| **CTD** | conductivity, temperature, depth (sensors) |
| **DATS** | Data Acquisition and Telemetry System |
| **DE** | development environment |
| **DEM** | digital elevation model |
| **DIC** | Dissolved Inorganic Carbon |
| **DMAS** | Data Management and Archive System |
| **DOC** | Department of Commerce |
| **DOE** | Department of Energy |
| **DOI** | Department of the Interior |
| **DRS** | Data Recovery System |
| **DS** | Docking Station |
| **ECO** | Environmental Characterization Optics |
| **EGL** | Exploration Geophysics Laboratory |
| **FD** | frequency domain |
| **FFT** | fast Fourier transform |

| | |
|---|---|
| **FORTRAN** | formula translating system |
| **$F_r$ (=FR)** | formation rate |
| **FY** | Fiscal Year |
| **GC** | gas chromatograph |
| **GLA** | geophysical line array |
| **GOM** | Gulf of Mexico |
| **GOM-HRC** | Gulf of Mexico-Hydrates Research Consortium |
| **GPS** | global positioning systems |
| **GUI** | graphical user interface |
| **HLA** | horizontal line array |
| **Hpdt** | structure that holds simulated and real FD data input for MFI |
| **HRC** | Hydrates Research Consortium |
| **$H_2S$** | hydrogen sulfide |
| **HSZ** | Hydrate Stability Zone |
| **ID** | identification |
| **IDL** | Interactive Data Language |
| **IDLVM** | Interactive Data Language Virtual Machine |
| **IDP** | Integrated Data Power Unit |
| **I-O** | Input-Output Corporation |
| **IP** | Internet Protocol |
| **IRMS** | isotope ratio mass spectrometer |
| **JIP** | Joint Industries Program |
| **JSL** | Johnson SeaLink |
| **LUMCON** | Louisiana Universities Marine Consortium |
| **MBARI** | Monterey Bay Aquarium Research Institute |
| **mbsf** | meters below sea floor |
| **MC** | Mississippi Canyon |
| **MD** | Marion Dufresne |
| **ME** | microbial experiments |
| **METS** | methane sensor |
| **MFI** | matched-field inversion |
| **Mfop** | structure holds the conditions for, and the results of, an MFI run |
| **MFP** | matched field processing |
| **MMRI** | Mississippi Mineral Resources Institute |
| **MMS** | Minerals Management Service |
| **Moni** | structure containing the setup conditions and results for a monitoring run |
| **MP** | methane production |
| **MPC** | multi-purpose cable |
| **MS/SFO** | monitoring station/sea-floor observatory |
| **M/V** | Merchant Vessel |
| **NETL** | National Energy Technology Laboratory |
| **NIUST** | National Institute for Undersea Science and Technology |
| **NOAA** | National Oceanographic and Atmospheric Administration |
| **NURP** | National Undersea Research Program |
| **OBC** | ocean-bottom cable |

| | |
|---|---|
| **OLA (=OVA)** | Oceanographic Line Array |
| **ORCA** | a range-independent acoustic model (Microsoft) code |
| **ORDBMS** | object relational database management systems |
| **OVA (=OLA)** | Oceanographic Vertical Line Array |
| **PC** | personal computer |
| **PCB** | pressure-compensated battery |
| **PCA (=PFA)** | pore-fluid array |
| **PFA (=PCA)** | pore-fluid array |
| **PFP** | pore-fluid probe |
| **P-P** | standard P-wave seismic data |
| **P-SV** | converted-shear mode (P-wave to SV-shear wave conversion) |
| **P-wave** | compressional wave |
| **RAM** | Range-dependent Acoustic Model |
| **RDBMS** | relational database management systems |
| **RDI** | RD Instruments |
| **Repl** | structure that holds parameters that vary |
| **ROV** | remotely operated vehicle |
| **RSI** | Research Systems, Inc. |
| **RTC** | real-time clock |
| **R/V** | Research Vessel |
| **SCUFA** | self-contained underwater fluorescence apparatus |
| **SDI** | Specialty Devices, Inc. |
| **SFO** | Sea Floor Observatory |
| **SFP** | Sea Floor Probe |
| **Ship** | structure that holds the traces for a single shot |
| **Shot** | structure that holds the traces for a single shot |
| **SMI** | sulfate-methane interface |
| **Sopt** | search/optimization file |
| **SSD** | Station Service Device |
| **SS/DR** | shallow-source/deep-receiver |
| **SSP** | sound speed profile |
| **S-wave** | shear wave |
| **TD** | time domain |
| **Ti** | induction time |
| **TOC** | total organic carbon |
| **TP** | thermistor probe |
| **UNC** | University of North Carolina at Chapel Hill |
| **US** | United States |
| **USBL** | ultra-short base-line (locating system) |
| **USGS** | United States Geological Survey |
| **VLA** | vertical line array |
| **VM** | virtual machine |
| **VSP** | vertical seismic profile |
| **Wave** | structure that holds an acoustic wavelet |
| **WHOI** | Woods Hole Oceanographic Institution |
| **Wssp** | water sound speed profile (file) |

## APPENDIX

## GULF OF MEXICO HYDRATE RESEARCH CONSORTIUM: ESTABLISHMENT OF A SEA FLOOR MONITORING STATION, AN UPDATE

### Introduction

Since the Gulf of Mexico Gas Hydrates Research Consortium (GOM-HRC) was organized in 1999, considerable progress has been made toward establishing a monitoring station or sea-floor observatory (MS/SFO) to monitor and investigate the hydrocarbon system within the hydrate stability zone of the northern Gulf of Mexico. The intention has been to equip the MS/SFO with a variety of sensors designed to determine a steady-state description of physical, chemical, thermal and, most recently, microbiological conditions in its local environment as well as to detect temporal changes of those conditions.

In the original design, the heart of the MS/SFO was a network of five vertical line arrays (VLAs), each consisting of 16 channels of hydrophones spaced over the lower 200m of the water column. Each VLA would be suspended from glass floats and would have been anchored to the sea floor. Since water currents would cause the VLAs to deviate from vertical, each would also include inclinometers and compasses for determining the location of each hydrophone within the water column.

The intention was to use standard surveying techniques to determine the configuration of sub-bottom strata and to monitor that configuration by applying Matched Field Processing (MFP) to the acoustic energy received by the VLAs. The source of the energy could be either the intentional firing of conventional seismic devices or the opportunistic noise of passing ships.

In either case, MFP would require knowledge of the source location. In the former, the location would be measured directly. In the latter, it would be estimated relative to the known location of the VLAs by triangulation. The net of five VLAs would provide 20 independent estimations that would be analyzed statistically to minimize error in the final determination.

Significant disagreement between the MFP results and the sub-bottom configuration determined previously would indicate that a change had occurred within the sea floor. A new survey could then be carried out to determine the structural nature of the change and the output of other sensors examined to determine chemical and thermal changes. This original strategy came under question during 2003, however, due to a number of external factors that surfaced. Discussions arose among some Consortium members as to whether or not the design of the MS/SFO could be modified to accommodate, and perhaps even to capitalize on, those factors. There was agreement to explore a number of modifications but not to alter the original intention or basic mission of the MS/SFO. This update documents that exploration and other developments.

**Modifications**

Modifications to the design of the monitoring station/sea floor observatory are described below and are illustrated in Figure A1.

*CHANGE 1: ARRAY TYPE*

One external factor affecting the establishment of the station is the development of an ocean acoustics technique by which the sound of waves at the sea surface can be used to image the sea-floor. The method requires that at least two horizontal line arrays (HLAs) be deployed on the sea-floor perpendicular to each other. Each HLA should be as long as the water is deep and contain as many hydrophones as is feasible. If each hydrophone comprises a separate data channel, the cross of HLAs will also be capable of triangulating on ship noise. One VLA would still be required to separate the up-going and down-going wave-fields, but the sound of waves could be utilized as an energy source by redeploying the other four VLAs as two HLAs. This would allow the sound of wind-driven waves to be used without forfeiting the use of either intentional seismic sources or ship noise.

A second external factor is the opportunity to deploy an array of sensors in a borehole that will be drilled by the Department of Energy/Joint Industry Program (DOE/JIP) Consortium. The borehole array (BHA) will consist of hydrophones, three-component accelerometers and temperature sensors that would remain in the hole after the drill stem is recovered, letting the hole collapse and making the installation permanent. It would provide long-term monitoring from within the hydrate stability zone. If located at a site appropriate to the other requirements of the monitoring station, it would comprise a valuable addition to the MS/SFO.

If both these array modifications were to be incorporated, the seismo-acoustic components of the MS/SFO would comprise three mutually perpendicular axes of a Cartesian coordinate system. One VLA would be the vertical axis in the water column and the horizontal axes would consist of the other four VLAs deployed horizontally. The BHA would comprise the sub-bottom portion of the vertical axis.

A second VLA has been constructed to accommodate geochemical sensors: off-the-shelf thermistors, CTDs, fluorometers and transmissometers. This array will provide the capability of studying hydrate-related hydrocarbon fluids in the water column. It will be possible to deploy this array either in an autonomous mode or as a component of the MS/SFO.

The original design of the MS/SFO calls for each of the VLAs to be equipped with a sea-floor data-logger. The five data loggers were to be connected to a central integrated data/power (IDP) module that would collect data from, and supply power to, the individual loggers. The change to using HLAs would not affect this arrangement.

The BHA has been funded separately by DOE/JIP and it would not represent a cost increase to the SFO. The only cost increase would be associated with increasing the length of the four VLAs so they could be re-deployed as two HLAs with lengths equivalent to the water depth. This could be a factor in whether or not the BHA becomes an integral part of the MS/SFO.

Since the Consortium's break from the JIP plan, it appears likely that the placement of a BHA will not happen in the near future. For this reason and because the BHA concept adds so much to the overall station capability, the idea of emplacing

shorter arrays via the Sea Floor Probe has been revived. Ten meter arrays, both geochemical and geophysical have been added to the plan for the station. Although these arrays are temporary, they will provide much valuable data at a fraction of the cost of a borehole array. These are the arrays that were emplaced on the sea-floor in May, 2005. Data from these arrays will be retrieved at the earliest opportunity.

*CHANGE 2: DATA RECOVERY*

External factors have also impacted the way MS/SFO data will be recovered. For some time it was thought that a commercial service would be available in 2004 which would allow the IDP to stream data onto an optic-fiber link for near-to-real time transmission to shore. It was learned in the autumn of 2003, however, that the service would not become available until 2006 or later. It now appears that "later" will be the case though CMRET maintains contact with companies involved in development and deployment of fiber-optic systems.

The use of a remotely operated vehicle (ROV) to download data directly from the MS/SFO's data-loggers was found to be prohibitively expensive due to the depth of water and the weight of the battery packs that would need to be exchanged. Therefore, until such a link becomes available, the IDP module will stream data onto an optic-fiber data recovery system (DRS) which will be connected via optic fiber to an access connector. Whenever downloading is required, a system of buoys will bring the DRS access connector to the surface so that the data can be downloaded onto computer in a boat. The system has been used successfully before and involves far less expense than repeated use of a deep-water ROV. The system has been dubbed the "Big M" and is illustrated in Fig.A1.

*CHANGE 3: POSITIVE SYNCHRONIZATION OF TEST SIGNALS*

The DRS will serve yet another need. While surveying to determine the configuration of sub-bottom strata in the vicinity of the MS/SFO, the towed sea-floor sled will be used to generate shear waves for recording by the MS/SFO's arrays. During the course of that survey, an access connector will be brought to the surface and connected to a radio telemetry buoy that will synchronize the firing and receiving of signals.

*CHANGE 4: ELECTRICAL POWER FOR THE SFO*

The Gulf of Mexico Hydrates Research Consortium funds the development of microbial batteries but it will be some time before they can provide electrical power to the MS/SFO. In the meantime, the IDP module will supply electricity to the MS/SFO by exchanging the pressure compensated battery (PCB) component about once a year. This will involve unplugging the depleted PCB from the IDP and plugging in a fresh one. The emplacement and exchange of PCBs will be accomplished by a station service device (SSD) especially designed for the task.

A docking station will be incorporated into the IDP module to facilitate changing the PCB. The SSD will carry the recharged PCB unit to the sea floor and return with the depleted unit. In addition, the SSD will be capable of recovering pore-fluid samples at *in situ* pressures. Perhaps most significantly, the SSD will be the means by which all station systems are connected to the IDP for data recovery and electrical power.

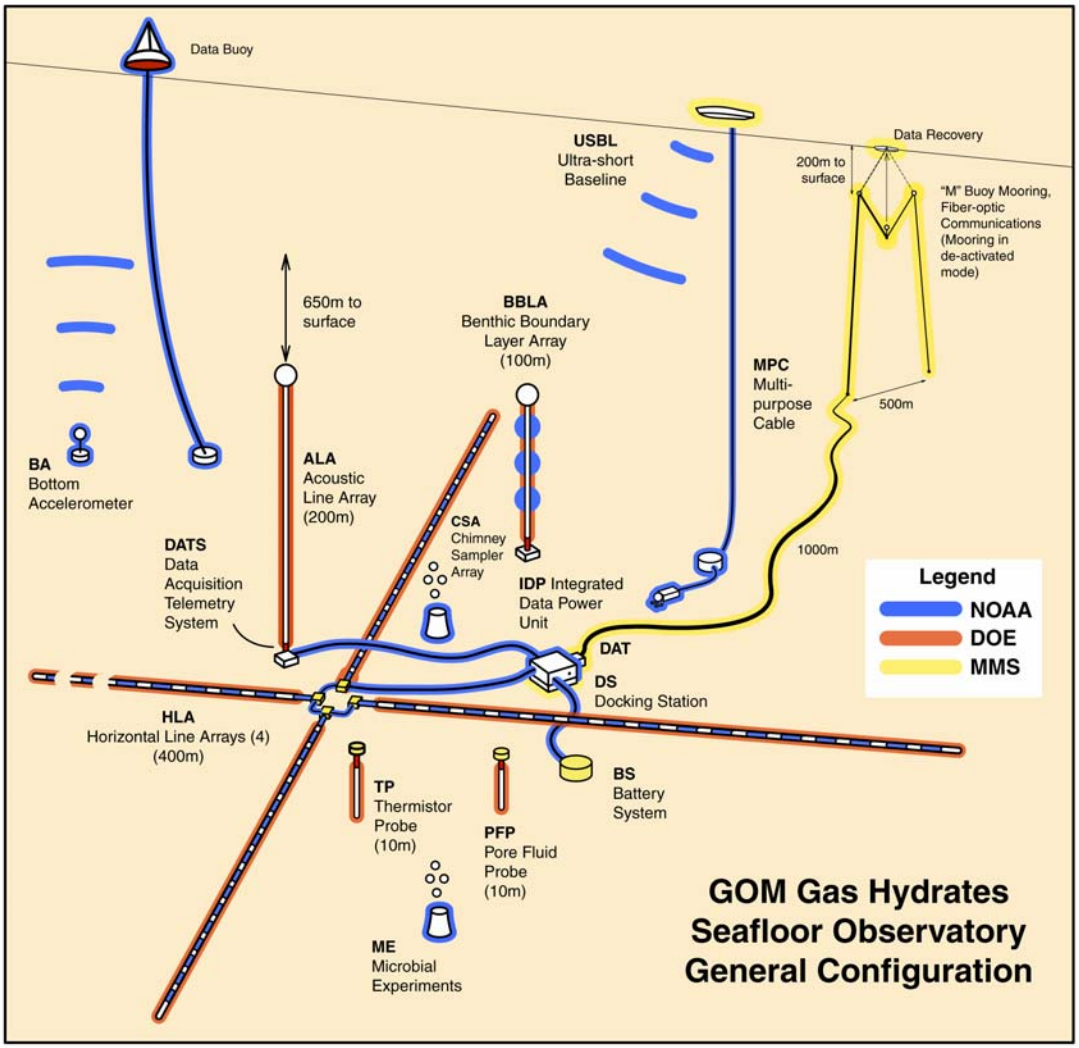



Figure A1. Diagram of the monitoring station/sea floor observatory.

**Conclusion**

Modifications discussed herein are not intended to change the basic concepts, overall plans and mission for the MS/SFO. Instead, they are expected to enhance the accomplishment of that mission.

Funding has been requested for the supply of components and construction of the new systems in order to adapt to the changing circumstances, as well as for the continuation of the all-important, on-going, studies and systems development projects. Data are anticipated in 2006 and tests of the station's operation and software systems should follow. A fully operational station is anticipated in 2007.