7 Reference Manual

7.1 Quick Start

This section provides a minimum of information about getting PHAML up and running the example programs. For many people, this will be sufficient to get started using PHAML with your application, as many applications may be implemented by modifying one of the examples. If you encounter problems, refer to the appropriate section of the User's Guide for further details. Also read the remainder of the User's Guide for a better understanding of how PHAML works and what options may be useful to you.

The four steps in getting started with PHAML are

1. obtaining the software,
2. compiling the PHAML library,
3. compiling an example, and
4. running the example.

For this brief tutorial, we will build and run the `simple` example as a master/slave message passing MPI program with the master spawning the slave processes, and without graphics.

7.1.1 Obtaining the software

PHAML can be obtained from the PHAML web page `http://math.nist.gov/phaml` by following the Software link. It comes as a gzipped tar file for Unix-like systems. (It has not yet been tested on MS Windows systems, but since it is written in standard Fortran 90, the adventurous may find that it works on MS Windows, too.) When unpacked, it will place everything in a new directory called `phaml-x.x.x`, where `x.x.x` is the current version number.

PHAML requires the BLAS and LAPACK libraries. You will probably find these are already installed on your computer, but if not, see Section 2.1.3.

For parallelism, you need an MPI library. You will probably find that LAM, MPICH, or a commercial MPI library is already installed on your computer, but if not see Section 2.1.4.

7.1.2 Compiling the PHAML library

The first step in compiling the PHAML library is to create the `Makefile`. This is done with the shell script `mkmkfile.sh` in the top PHAML directory. Since PHAML allows so many options in terms of what compilers and libraries to use, it would be difficult to auto-locate these files. So, you must modify `mkmkfile.sh` to specify some paths, command names, and defaults for your computer system. Instructions for modifying it can be found within the file.

Now create the `Makefile` with

`./mkmkfile.sh PARALLEL messpass_spawn PARLIB mpi GRAPHICS none`

You should replace `mpi` with `mpich`, `mpich2`, or `lam` if you use an MPICH or LAM library. You may omit some of the arguments if your defaults are already set to these values; defaults can be determined with `mkmkfiles.sh help`.

`make` should now compile the library and place it in the `lib` subdirectory.

7.1.3 Compiling an Example

Go to the directory `examples/simple` and type `make`. (The `Makefile`s for the examples were also created by `mkmkfile.sh`.) This should create the executables `phaml` and `phaml_slave`.

7.1.4 Running the Example

The details of running an MPI program vary with the different MPI libraries. You may need to check your MPI documentation to find the correct command(s). It may also require starting some daemon before running the execution command.

Note that you should specify *one* process, because you are running the master processes which will spawn the slave processes. The number of slaves is specified in the main program, `master.f90`.

If you are using LAM, try

```
lamboot
mpirun -np 1 phaml
```

If you are using MPICH, try

```
mpirun -np 1 phaml
```

If you are using MPICH2, try

```
mpiexec -n 1 phaml
```

7.1.5 Now what?

If you have successfully run the first example, you are ready to install the graphics and any other optional software you desire (Section 2.1), run the other examples, and begin working on your own application!

7.2 Public Entities in PHAML

The statement `use phaml` in a program unit provides access to the public entities in PHAML. These consist of a derived type, variables for the user to use, symbolic constants, and procedures. The procedures are described in Section 7.4. The other entities are described in this section.

7.2.1 phaml_solution_type

`phaml_solution_type` is a type that contains all the data used for solving the PDE (grid, etc.). The type is public, but the contents are private. You can