SANDIA REPORT

SAND2005-2068
Unlimited Release
Printed April 2005

User Manual for INVICE 0.1-beta: A Computer Code for Inverse Analysis of Isentropic Compression Experiments

Jean-Paul Davis

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2005-XXXX
Unlimited Release
Printed March 2005

User Manual for INVICE 0.1-beta: A Computer Code for Inverse Analysis of Isentropic Compression Experiments

Jean-Paul Davis
Shock and Z-Pinch Physics
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1181

Abstract

INVICE (INVerse analysis of Isentropic Compression Experiments) is a FORTRAN computer code that implements the inverse finite-difference method to analyze velocity data from isentropic compression experiments. This report gives a brief description of the methods used and the options available in the first beta version of the code, as well as instructions for using the code.

This page intentionally left blank.

Contents

Nomenclature

c_L	Lagrangian bulk sound speed, $c_L = (\rho/\rho_0)\,(\partial P/\partial \rho)^{1/2}$
Γ	Grüneisen parameter
Γ_0	Grüneisen parameter at ambient condition
ε	volumetric strain, $\varepsilon = 1 - v/v_0$
P	pressure, or mean stress
P_H	pressure on the Hugoniot reference in Mie-Grüneisen model
ρ	density
σ_x	longitudinal stress
t	time
t_{min}	left-hand boundary of time domain
u_p	particle (material) velocity
u_s	velocity along the principal compression isentrope due to a simple compression wave
u_{shock}	shock velocity used to define Hugoniot reference in Mie-Grüneisen model
v	specific volume
v_0	specific volume at ambient condition
x	Lagrangian position
x_1	Lagrangian position of measurement location in experiment
$()_S$	isentropic derivative (entropy held constant)

Introduction

INVICE (INVerse analysis of Isentropic Compression Experiments) is a Fortran program derived from an original program by Dennis Hayes[1,2] that implements an inverse finite-difference method to analyze velocity-interferometry (VISAR) data from planar-geometry isentropic compression experiments on the Z accelerator. This approach accounts for ramp-wave interactions arising from impedance-mismatched interfaces, and has been applied by Hayes to diverse problems such as measuring the index of refraction of sapphire under isentropic compression,[3] spall in aluminum, [4] and the principal isentrope of aluminum to 185 GPa.[5] The code described in this report was recently used to analyze measurements on Z of the principal isentrope of aluminum to 240 GPa.[6]

The program INVICE applies the downhill simplex minimization method[7] with optional simulated annealing[7] to a routine that integrates isentropic ramp-wave profiles backward in space. Given two or more velocity-time profiles representing different particle velocity measurements resulting from the same loading history (*i.e.*, VISAR results from different locations on a single isentropic compression experiment), INVICE integrates these profiles backward through their respective "stacks" of material layers to obtain a stress-time (loading) history at Lagrangian position $x = 0$. The minimization procedure repeats these calculations while varying selected EOS parameters, time shift corrections, and layer thicknesses to find the values for these parameters which produce the best fit between all the loading profiles at $x = 0$. The selection of $x = 0$ as a reference point is arbitrary; the reference point needs only be far enough away from the nearest interface to remain outside the region of influence of that interface during the time domain of interest.

This manual covers the first version of INVICE intended to be distributed to external users. The version designation of 0.1-beta was chosen to reflect the fact that the code is very much a work in progress. The present source code is not guaranteed to adhere strictly to either the FORTRAN 77 or FORTRAN 90 standards, and does not follow modern FORTRAN programming practice. In addition, all options implemented in the code have not been fully tested. The source code is, however, reasonably well documented internally.

Backward Integration

The backward integration routine uses a two-step Lax-Wendroff differencing scheme[7] to advance the following one-dimensional, planar-coordinate equations in the negative Lagrangian spatial direction:

$$\frac{\partial u_p}{\partial x} = \frac{1}{v_0}\left(\frac{\partial v}{\partial t}\right) \quad \text{(conservation of mass),}$$

$$\frac{\partial \sigma_x}{\partial x} = -\frac{1}{v_0}\left(\frac{\partial u_p}{\partial t}\right) \quad \text{(conservation of momentum), and}$$

$\left(\frac{\partial v}{\partial P}\right)_S$ and/or $\left(\frac{\partial u_s}{\partial P}\right)_S$ = function of P, v, and u_s (equation of state).

Note that while u_s is the local particle (material) velocity, $u_s \neq u_p$ is the velocity obtained along the principal compression isentrope for a simple compression wave; in other words, u_s is considered a thermodynamic variable describing the state along the isentrope. The "equation of state" (EOS) pro-

vides only the compressibility of the material along the compression isentrope; it is not a full EOS. Initial conditions consist of $u_p(x_1,t)$, $P(x_1,t)$, and $v(x_1,t)$ at a fixed value of $x = x_1$. Cell size in t is fixed at a user-specified value (typically corresponding to a VISAR record sample rate). Step size in x is fixed at a user-specified fraction of the Courant stability condition, based on the ambient sound speed in the material, while enforcing an integral number of steps across the material layer. A constant extrapolation boundary condition is applied at $t = t_{min}$ (left-hand boundary) after the first Lax-Wendroff step. The right-hand boundary condition on t consists of a moving boundary that follows the local sound-speed characteristic to ensure the local time domain at any x remains causal to the initial velocity profile. This acts as a realistic extrapolation boundary condition because the ghost cell immediately to the right of the current time domain had its value set during an earlier step of the integration at larger x.

The initial velocity profile can represent a free surface measurement or a window interface measurement; the pressure and volume profiles are automatically initialized, which for a window includes integration of the material and window equations of state along the velocity profile. Note, however, that formation and growth of a shock in the window can invalidate this approach.[8] The backward integration can proceed through any number of material layers before reaching $x = 0$; when a new layer is reached, the volume profile is re-initialized by integrating the new EOS along the current pressure profile. The routine also follows sound-speed characteristics to keep track of two markers that define the desired time range to use in the minimization routine.

The program is currently limited to material models that exhibit unique stress-strain response and do not require integration of the energy conservation equation. No strength model has yet been implemented, so pressure and longitudinal stress are identical. In the computer code, each model is uniquely identified by a number `ieos`. It is a simple matter to add a new model to the code. Following the existing models as a template, one must insert a case for the new model into the `select` structure in the `FUNCTION usvderivp` in the source file `EOS.f` (listed in Appendix 3). The model needs only calculate dP/dv and dP/du_s along the principle compression isentrope. Note that u_s is particle velocity along this isentrope (*i.e.*, a thermodynamic variable), and not the local particle velocity of material in a computational cell. The calculation can use EOS parameters from the array `epar` (note that the assignment statements define the order in which parameter values must be entered in the input file) along with the current values for p, v, and/or u_s. A case must also be inserted into the `select` structure in the `FUNCTION getnpar`, which returns the number of EOS parameters used by the model. Finally, be sure to add comments to the input file template.

A "bootstrapping" option is available for some models to break the EOS into multiple segments applicable over different ranges of pressure. This allows one to hold the EOS fixed below a certain pressure while performing minimization to find EOS parameters above that pressure. The fixed EOS could be defined by a previous run of INVICE using different sets of velocity profiles fit only in the lower-pressure segment; hence the term bootstrapping.

The following models are currently implemented in the code:

`IEOS` = 0 spline interpolation of tabular $P(v)$ or $P(\rho)$ isentrope (cannot be used for minimization)
sets `npar`= 1, `epar(1)`= material number, must supply file name for table

`IEOS` = 1 Polynomial $c_L(u_s)$ isentrope

$$c_L = c_0 + c_1(u_s/c_0) + c_2(u_s/c_0)^2 + c_3(u_s/c_0)^3 + c_4(u_s/c_0)^4$$

5 parameters: c_0, c_1, c_2, c_3, c_4
for bootstrapping option, u_s is replaced by $u_s - u_0$, where u_0 is computed by the code

`IEOS` = 2 Polynomial $c_L(P)$ isentrope

$$c_L = c_0 + c_1\left(P/P_{ref}\right) + c_2\left(P/P_{ref}\right)^2 + c_3\left(P/P_{ref}\right)^3 + c_4\left(P/P_{ref}\right)^4 + c_5\left(P/P_{ref}\right)^5$$

7 parameters: P_{ref}, c_0, c_1, c_2, c_3, c_4, c_5

for bootstrapping option, P is replaced by $P - P_0$, where P_0 = `pseg` of previous segment

`IEOS` = 3 Mie-Grüneisen referenced to $u_{shock} - u_p$ Hugoniot with s quadratic in volumetric strain ε and with $\Gamma/v = \Gamma_0/v_0$ (note that any other choice for Γ, or a non-Hugoniot reference, would require that the equation of energy conservation be added to the system solved during backward integration)

$$u_{shock} = c_0 + (s_1 + s_2\varepsilon + s_3\varepsilon^2)\,u_p\,, \qquad P = P_H\left[1 - \Gamma(v_0 - v)/2v\right] - (\Gamma/v)\int P\,\mathrm{d}v$$

5 parameters: c_0, s_1, s_2, s_3, Γ_0

ε is volumetric strain $1 - v/v_0$, P_H is pressure on the Hugoniot at volume v

bootstrapping option is not implemented for this model

Minimization

The minimization routine minimizes the deviation between pressure profiles at $x = 0$ by use of the multi-dimensional downhill simplex method, which evolves a simplex having $n + 1$ vertices through n-dimensional space, where n is the number of parameters allowed to vary. The first vertex is initialized by the initial guess for parameter values, and each subsequent vertex is initialized by a deviation of one of the parameters from the initial guess, the deviation given by a user-supplied multiplicative factor but alternating direction for each subsequent parameter. If the time shift for a given stack is allowed to vary, but has an initial guess of zero, then its initial deviation is given instead by $10\Delta t$ where Δt is the time step. During an iteration of the minimization process, one or more of the vertices is moved by reflection, expansion, or contraction of the simplex according to the relative values of the minimization function at each vertex. In this way the simplex "oozes" downhill along local gradients until it contracts on a local minimum by criterion of a fractional convergence tolerance on the minimization function, or until the user-specified maximum number of iterations has been reached. To ensure the minimum thus found is not a false minimum due to numerical issues, the minimization process may be restarted using the apparent minimum as a new initial guess, and a new (usually smaller) multiplicative factor to reinitialize the other vertices. The user may specify the number of such cycles to perform.

An option is available to use a simulated annealing modification to the downhill simplex method, which can be useful for problems with many local minima. The technique is analogous to slowly cooling a material so that random motions of its atoms will eventually find the minimum-energy configuration. Random disturbances proportional to a "temperature" (a user-supplied fraction of the function value) are added to and subtracted from the function value at each vertex of the simplex before and after each move; this causes the simplex to remain near a temperature-dependent size and sample random points in that region. The user specifies parameters to control the decrease in temperature for each annealing pass, the maximum number of iterations per annealing pass, and the number of such passes. Each pass begins with the best parameter values (those giving the lowest value for the minimization metric) previously found. A temperature of zero makes the process identical to the standard downhill simplex method.

The code provides three choices for the function to be minimized (the minimization metric). If experimental uncertainties in time and velocity are provided, then the metric consists of the average normalized chi-squared values of all the loading curves with respect to the average loading curve, in both time and stress coordinates. Stress uncertainties are computed from velocity uncertainty parameters. The other two choices are root-mean-square averages of the absolute deviations from the average loading curve in either stress or in time. All metrics are computed over a limited range in stress corresponding to markers propagated from user-supplied time points in the initial velocity profiles. The full fitting range can be split into several sub-ranges, with each profile assigned to only certain sub-ranges. For stress or time values falling within a particular region, only the profiles used in that region contribute to the metric. This approach allows the minimization to be applied only to relevant parts of the VISAR data.

Input File Description

When the executable `INVICE.exe` is run, it asks the user for the name of the input file, which must include the directory path relative to the current directory. This input file contains all the information necessary to run INVICE. Each input line is unformatted, but the lines must be present in a specific order, and line length is limited to 132 characters. Any lines beginning with an asterisk are ignored, allowing for extensive commenting of the input file. Additional comments can be added to the end of any input line that contains only numerical data. A line-by-line description of the input file follows. Units for dimensional quantities are arbitrary but must be self-consistent. Three examples of input files are given in Appendix 1.

1. Number of materials `nmat` (single integer value ≤ `mmax`);

This includes materials that are used only for windows.

2. Material definitions (set of two or more lines for each material);

Line 1: material identification number (≥1, ≤`nmat`), ambient density ρ_0, `ieos`, `istr`, `nseg`

Line i+1: (for `nseg`>1 only) maximum pressure and EOS parameters for ith segment
`pseg(i)`, `eospar(1,i)`, `eospar(2,i)`, …, `eospar(npar,i)`

Line nseg+1: EOS parameters with minimization flags eospar(1,nseg), eosflag(1), eospar(2,nseg), eosflag(2), …, eospar(npar,nseg), eosflag(npar)

Note that `ieos` is the EOS model identification number, and `npar` is the number of parameters (defined by `ieos` and the `FUNCTION getnpar`). Each element of `eosflag` is set to 0 to hold the corresponding parameter constant during minimization, or to 1 if the corresponding parameter is to be varied in the minimization; in the latter case, the value provided in `eospar` is used as the initial guess for that parameter. To use a fixed EOS at lower pressure while varying parameters for the EOS at higher pressure (bootstrapping technique), set `nseg`>1 and insert an additional line for each segment below the final segment, giving the EOS parameters to be used on that segment as well as the upper boundary in pressure `pseg` for that segment. The number of material definitions must match `nmat`, the number of materials. The parameter `istr` is a strength model identification number; there are currently no strength models implemented, so its value should be set to 0. When `ieos` = 0, `npar` is set to 1, and line 2 must be set to `ivor`, `filename`, where `ivor` = 0 for a *p(v)* table and `ivor` = 1 for a $p(\rho)$ table. The filename for the table can have up to 50 characters and can include a relative directory path.

3. Number of stacks `nstk` (single integer value ≤ `nsmax`);

Each initial velocity profile to be considered must correspond to a different "stack" of one or more material layers of specified thicknesses.

4. Number of layers in each stack `nlay(1)`, `nlay(2)`, …, `nlay(nstk)`;

These are given in stack order. Windows are not included in the number of layers.

5. Stack definitions (one line for each layer of each stack);

Line: stack number, layer number, material number, layer thickness `xlayer`, `thkflag`

The lines do not have to be in any order, but there must be one line for each layer of each stack (total number of lines must equal the sum of the elements in `nlayers`). Layers are numbered sequentially starting from the material layer adjacent to the window (or free surface) and going back towards x=0. The material number identifies which material constitutes a given layer. The minimization flag `thkflag` is set to 1 to allow the layer thickness to vary as part of the minimization, and set to 0 otherwise.

6. Window material definitions `mwin(1)`, `mwin(2)`, …, `mwin(nstk)`;

These numbers are given in stack order (so that `mwin(1)` is the material for the window on stack #1, etc.). To indicate that the initial velocity profile for a given stack corresponds to a free surface measurement instead of a window interface measurement, set the appropriate element of `mwin` to –1.

7. Input filenames and parameters for initial velocity profiles (one line per stack);

Line: stack number, `tscale`, `uscale`, `tshft`, `tshftflag`, `tbpad`, `tepad`, *input file name (up to 50 characters, includes relative directory path)*

The ASCII files must be in column format, with time in the first column and velocity in the second column. Any lines beginning with "*" or "!" are considered comments (headers must begin with one of these characters). The time step in the file does not have to be uniform, nor does it have to correspond to the desired time step used in backward integration. The starting time may be different in each file, and the data do not need to be pre-padded. One can use raw velocity data from VISAR analysis without further pre-processing. Each velocity profile read in from a file is first scaled in time and velocity using `tscale` and `uscale` (thus allowing the use of velocity files that are in different units than the INVICE input file), then shifted in time by `tshft`. Set `tshftflag` to 1 to allow that stack's time shift to vary in the minimization. The velocity profile is then padded in front with zero velocity to extend the time domain on the left-hand boundary by a time equal to `tbpad`. This is typically done for experimental velocity data because the initial time domain must contain the time domain occupied by the ramp wave at x=0. In other words, the left hand time boundary must still be at zero velocity after integration back to x=0. Finally, the velocity profile is padded at its end by copying the last value of the input file to extend the right-hand boundary by a time equal to `tepad` (this can be useful for diagnosing whether or not the result is outside the region of influence of the interface).

8. Number of fitting regions `nreg` for minimization

Multiple fitting regions can be defined, each based on only a particular subset of stacks. Set `nreg` = 0 if no minimization is to be performed.

9. Definition of fitting/integration regions for minimization (one line per stack);

Line: stack number, `tfit1`, `tfit2`, `tcut`, `ireg(1)`, ..., `ireg(nreg)`

For each stack, a lower bound `tfit1` and an upper bound `tfit2` define a region in the corresponding input velocity profile that in turn defines a region in the corresponding loading profile at x=0. The backward integration routine tracks the local characteristic at each boundary point in each profile to find the corresponding times in the loading profiles. A particular fitting region `ir` is defined as the smallest region that is common to those stacks having nonzero `ireg(ir)` by taking the maximum lower bound and minimum upper bound. Then minimization metrics are computed over the maximum range available from all fitting regions, using only the stacks "available" at each point. For each stack, an upper bound `tcut` defines the region to be integrated; input data for time greater than `tcut` is discarded in order to improve performance. Setting any of *`tfit1`, `tfit2`, `tcut`* to –1 sets the corresponding bound to the endpoint of the full signal.

10. Integration control parameters `dt`, `stab`, `nsmooth`;

The input velocity profiles are interpolated onto a uniform time grid of cell size `dt` using a cubic spline. `stab` is a multiplicative factor between 0 and 1 applied to the Courant stability condition to obtain a uniform step size in x. Every `nsmooth` steps of the integration, the velocity and stress arrays are smoothed using a three-point weighted average.

11. Experimental uncertainties (one line per stack);

Line: stack number, `sigt, fvpf, ffc, fc`

These uncertainties are used to compute normalized chi-squared metrics for the minimization; `sigt` is the absolute time uncertainty, `fvpf` is the fractional uncertainty in fringe constant (VPF), `ffc` is the fractional uncertainty in fringe count, and `fc` is the fringe constant (VPF) itself. The code calculates local uncertainty in stress based on these last three parameters. Set `sigt` < 0 to use absolute deviations in stress for the minimization metric, and set `fc` < 0 to use absolute deviations in time for the minimization metric.

12. Minimization parameters `fac`, `ncycle`, `frestart`, `ftol`, `tfac`, `teps`, `ntan`, `npan`;

These parameters control the initialization and completion of the minimization procedure, including options for simulated annealing. `fac` is the multiplicative factor used to initialize the simplex vertices for the first minimization cycle. `ncycle` is the number of cycles to complete. `frestart` is the multiplicative factor used to reinitialize the simplex vertices for the second and subsequent cycles. `ftol` is the fractional convergence tolerance on the maximum change in function value between vertices. `fk` is a multiplicative factor determining the strength of randomness in magnitude of simplex moves (set equal to 0 for no randomness). `tfac` is a multiplicative factor to determine the initial "temperature" for simulated annealing (set equal to 0 to turn off annealing). `teps` is the factor by which to reduce the temperature for each annealing pass ($T_{n+1}/T_n = 1-$`teps`). `ntan` is the maximum number of iterations to perform during each annealing pass (or for the entire minimization if annealing is turned off). `npan` is the maximum number of annealing passes. All of these parameters are ignored if no non-zero values were given for `eosflag`, `thkflag`, or `tshftflag`, in which case only a single backward integration is performed for each stack without minimization.

13. Output filenames for stress-loading curves $\sigma_x(t)$ at x=0 (one line for each stack);

Line: stack number, output filename (up to 50 characters, includes relative directory path)

These column-formatted ASCII files (time in the first column, stress in the second column) are output for every stack at the end of each minimization iteration. Files from previous iterations are overwritten. Dimensional units correspond to those used for the input file.

14. Snapshot output times and filenames (one line for each stack);
Line: stack number, `tsnap`, output filename (up to 50 characters including path)
These column-formatted ASCII files give a spatial distribution snapshot of several variables for each stack at a time t = `tsnap`. The output is generated after each backward integration of a stack. The files have four columns; Lagrangian position, velocity, density, and stress. To suppress this output, set `tsnap` equal to a time well outside the expected range of times in the problem (e.g., -1 s).

15. Output filename for minimization log (up to 50 characters including path)
Each line of the log file is also echoed to the screen and lists the following information:

- `iter` = number of steps remaining to be taken by the downhill simplex method during the current annealing pass
- `rtol` = magnitude of the difference in function value between the highest and lowest vertices, relative to the average function value of the same two vertices
- `dev_s` or `dev_t` or `chi2_n` = the minimum value of the minimization metric after the current iteration; units are stress or time or dimensionless, depending on choice of metric
- `low point simp(1,1:nvar)` = nominal values of the EOS parameters at the vertex corresponding to minimum metric value; `nvar` is the number of parameters allowed to vary, equal to the sum of the elements of the arrays `eosflag`, `thkflag`, and `tshftflag`. The parameters are given in order first of increasing material number, then increasing parameter number.

After the last iteration, the final values for the EOS parameters are output along with the corresponding material number and parameter number for ease of identification.

Compilation and Execution Tips

The distribution of INVICE 0.1-beta includes an executable file `INVICE.exe` compiled and linked under Windows 2000 using Compaq Visual Fortran 6.6, a source code file `EOS.f` (which includes only the functions `getnpar`, `getnspar`, `usvderivp`, and `pvderivus`), and pre-compiled object files `INVICE.obj`, `COMPDEV.obj`, `BACKWARD.obj`, `GETPAR.obj`, and `TABDPDV.obj`. The object files contain compiled internal subroutines copyrighted by Numerical Recipes Software (used by permission). If you wish to obtain and work with the full source code of INVICE, you must first have a Numerical Recipes Software license ($50, http://www.nr.com/com/storefront.html). This is currently the only option available for non-Windows platforms. To the author's knowledge, both Los Alamos National Laboratory and Lawrence Livermore National Laboratory own site-wide licenses for Numerical Recipes, but Sandia National Laboratories does not. The list of variables in Appendix 2 may be helpful for working with the full source code.

To recompile under Visual Fortran:

1. create a new "Fortran Console Application" project,
2. add all the `.obj` and `.f` files from the distribution,
3. set the active configuration (under the *Build* menu) to Release,
4. set the Fortran Language option (in *Settings* under the *Project* menu) for Fixed Form Line Length to 132, and
5. *Rebuild All* under the *Build* menu.

This will produce an executable that runs faster than the default debug version. To run INVICE in the background so that your other applications are not slowed, set INVICE's priority to "Below Normal." This can be done using Task Manager after executing INVICE, or if you use a shortcut to launch INVICE, you can set a default priority in the shortcut's properties.

Suggestions for Future Work

The present version of INVICE does not have any strength models and does not allow for hysteretic stress-strain behavior. Including such behavior necessarily makes the equations of motion no longer purely hyperbolic in nature, and thus unstable under backward integration and prone to non-uniqueness. Hayes has had some success, however, performing backward analysis with a quasi-elastic (time-dependent) constitutive model[4] and with a simple chemical reaction model.[9] It remains to be seen if a similar approach will work for materials undergoing structural phase transformations. One difficulty is that the volume collapse due to a phase transformation generally causes a shock to form in the compression wave; waveforms with a shock cannot be analyzed by backward integration unless the shock is weak enough that entropy production is negligible and the shock has only propagated a short distance in the problem.

In addition to strength (and possibly phase transitions), future versions of INVICE could be expanded to allow on-the-fly calculation of window index-of-refraction effects for use with raw (uncorrected) VISAR data, and to couple with forward wave-propagation calculations using the WONDY code[10] in order to include variations in the loading history as part of the minimization; these techniques were used by Hayes to analyze data from an experiment with a large, growing shock in the window. [5]

The backward integration scheme could be updated to include the equation for conservation of energy. This may be needed if backward integration is attempted using more complicated models for phase transitions. Note, however, that for purely mechanical measurements such as velocity, it is generally better to use purely mechanical representations of the isentropic loading curve, such as the four EOS options presently provided in the code. Fitting mechanical measurements to a model that incorporates thermal energy causes the result to be more dependent on the model itself.

Finally, the code could benefit from being rewritten using modern best practices for modularity with the FORTRAN 90/95 standard.[11]

References

[1] D. B. Hayes and C. A. Hall, Correcting free surface effects by integrating the equations of motion backward in space, in *Shock Compression of Condensed Matter—2001* (held in Atlanta, GA, June 24-29, 2001). M. D. Furnish, N. N. Thadhani, and Y. Horie, ed., American Institute of Physics, Melville, NY, pp. 1177-1180, 2002.

[2] D. B. Hayes, *Backward Integration of the Equations of Motion to Correct for Free Surface Perturbations*, Sandia National Laboratories Report SAND2001-1440, May 2001.

[3] D. B. Hayes, C. A. Hall, J. R. Asay, and M. D. Knudson, Continuous index of refraction measurements to 20 GPa in Z-cut sapphire, in *Journal of Applied Physics*, Vol. 94, No. 4, pp. 2331–2336, August 2003.

[4] D. B. Hayes, J. E. Vorthman, and J. N. Fritz, *Backward Integration of a VISAR Record: Free-Surface to the Spall Plane*, Los Alamos National Laboratory Report LA-13830-MS, May 2001.

[5] D. B. Hayes *et al*, Measurement of the compression isentrope for 6061-T6 aluminum to 185 GPa and 46% volumetric strain using pulsed magnetic loading, in *Journal of Applied Physics*, Vol. 96, No. 10, pp. 5520–5527, November 2004.

[6] J.-P. Davis, Experimental measurement of the principal isentrope for aluminum 6061-T6 to 240 GPa. Sandia National Laboratories report SAND-2005-0729J (submitted to *Physical Review Letters*).

[7] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran 77: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, New York, 1992.

[8] D. B. Hayes *et al*, Measurement of the compression isentrope for 6061-T6 aluminum to 185 GPa and 46% volumetric strain using pulsed magnetic loading, in *Journal of Applied Physics*, Vol. 96, No. 10, pp. 5520–5527, November 2004.

[9] D. B. Hayes, private communication.

[10] M. E. Kipp and R. J. Lawrence, *WONDY V — A One-Dimensional Finite-Difference Wave Propagation Code*, Sandia National Laboratories Report SAND81-0930, June 1981.

[11] S. J. Chapman, *Fortran 90/95 for Scientists and Engineers*, WCB McGraw-Hill, Boston, 1998.

Appendix 1: Sample Input Files

One stack, three layers plus window, no minimization

```
* INPUT FILE inpCuSapBi600K.dat FOR INVICE (VERSION 0.1beta)
*    unformatted but line-ordered
*    lines beginning with '*' are ignored
*
** MATERIAL(S) EOS
**** ieos=0 ; tabular p(v) or p(rho) isentrope
**** ieos=1 ; polynomial cL(us) isentrope, eospar=c0,c1,c2,c3,c4
**** ieos=2 ; polynomial cL(p) isentrope, eospar=pref,c0,c1,c2,c3,c4,c5
**** ieos=3 ; c0-s isentrope with s quadratic in strain, gamma/v = g0/v0, eospar=c0,s0,s1,s2,g0
* no. of materials (including windows)
4
* material #, rho0, ieos, istr, nseg
* pseg(i), eospar(1), eospar(2), eospar(3), etc.  (bootstrap data: use only if nseg>1, insert nseg-1 lines)
* eospar(1), eosflag(1), eospar(2), eosflag(2), eospar(3), eosflag(3), etc.
*   (set eosflag=1 to make corresponding parameter adjustable in minimization, value used as initial guess)
* copper (cgs):
1, 8.93, 3, 0, 1
3.9399e5, 0, 1.4884, 0, 0.0, 0, 0.0, 0, 1.96, 0
* sapphire (cgs):
2, 3.985, 3, 0, 1
11.19e5, 0, 1.0, 0, 0.0, 0, 0.0, 0, 1.5, 0
* liquid bismuth at 600K (cgs):
3, 10.04, 3, 0, 1
1.693e5, 0, 1.497, 0, 0.0, 0, 0.0, 0, 1.50, 0
* LiF (cgs):
4, 2.638, 3, 0, 1
5.148e5, 0, 1.353, 0, 0.0, 0, 0.0, 0, 1.63, 0
*
** STACK DEFINITIONS
* no. of stacks
1
* no. of layers (for stacks #1, #2, etc.); excludes windows
2
* stack #, layer #, material #, thickness, thkflag
1, 1, 2, 0.20, 0
1, 2, 1, 0.075, 0
*
** WINDOW DEFINITIONS
**** iwinmat=material number for window, -1 for free surface
* iwinmat1, iwinmat2, ...  (for stacks #1, #2, etc.)
3
** VELOCITY PROFILE(S) INPUT
**** tscale, uscale = scale factors applied to u(t) input from file
**** tshft = time shift applied to u(t) input from file
**** tshftflag = set to 1 for allowing tshft to vary in minimization
**** tbpad = time to add in front of t-array with u=0
**** tepad = time to add at end of t-array with u=u_end
* stack #, tscale, uscale, tshft, tshftflag, tbpad, tepad, input filename for u(t) at measurement interface
1, 1.0, 1.0, 0.e-9, 0, 0.40e-6, 0.15e-6, OptimalLoads\velLiqBi225ns185kbar.dat
*
** FITTING AND INTEGRATION TIME RANGES
* no. of fitting regions
0
**** tfit1, tfit2 = low, high time points in u(t) corresponding to time
****                window in p(t) at x=0 used for minimization
**** tcut = time point in u(t) beyond which integration is not performed (overrides tepad)
**** ireg(i) = flag to include stack in ith fitting region
* stack #, tfit1, tfit2, tcut (set to -1 to use full signal), ireg(1), ireg(2), etc.
1, -1, -1, -1
*
** INTEGRATION CONTROL
**** dt = cell size in t
**** stab = percentage of Courant stability criterion (value from 0 to 1)
**** nsmooth = number of integration steps between smooths (=0 for no smoothing)
* dt, stab, nsmooth
0.20e-9, 0.8, 0
*
** EXPERIMENTAL UNCERTAINTIES
**** sigt = absolute time uncertainty
**** fvpf = fractional uncertainty in VPF (typically 0.001-0.01)
**** ffc = fractional uncertainty in fringe count (typically 0.02)
**** fc = fringe constant (velocity/fringe)
* stack #, sigt, fvpf, ffc, fc
1, 0.e-9, 0., 0., 0.
```

```
*
** MINIMIZATION CONTROL
**** fac = fractional variation in parameters to define initial simplex
**** ncycle = number of minimization cycles to perform
**** frestart = maximum fractional variation in parameters for restart vertices
**** ftol = fractional tolerance for minimization of average deviation
**** fk = strength of randomness in magnitude of simplex moves (0<=fk<=1)
**** tfac = multiplier to compute initial annealing temperature (0=no annealing)
**** teps = factor by which to reduce annealing temperature each pass
**** ntan = number of iterations per annealing pass
**** npan = maximum number of annealing passes
* fac,ncycle,frestart,ftol,fk,tfac,teps,ntan,npan
0.30, 1, 0.10, 1.e-4, 0.1, 0.0, 0.2, 200, 1
*
** OUTPUT FILENAMES
* stack #, output filename for p(t) at x=0
1, OptimalLoads\sout_CuSapBi600K.dat
* stack #, tsnap (outside range to skip snapshot output), output filename for u,v,s(x) at t=tsnap
1, -2., dum
* output filename for minimization log file
OptimalLoads\logCuSapBi600K.dat
```

Three stacks, one layer each (no windows), minimization without annealing, one fit region

```
* INPUT FILE inpZ1190fs234t3.dat FOR INVICE (VERSION 0.1beta)
*    unformatted but line-ordered
*    lines beginning with '*' are ignored
*
** MATERIAL(S) EOS
**** ieos=0 ; tabular p(v) or p(rho) isentrope
**** ieos=1 ; polynomial cL(us) isentrope, eospar=c0,c1,c2,c3,c4
**** ieos=2 ; polynomial cL(p) isentrope, eospar=pref,c0,c1,c2,c3,c4,c5
**** ieos=3 ; c0-s isentrope with s quadratic in strain, gamma/v = g0/v0, eospar=c0,s0,s1,s2,g0
* no. of materials (including windows)
1
* material #, rho0, ieos, istr, nseg
* pseg(i), eospar(1), eospar(2), eospar(3), etc.  (bootstrap data: use only if nseg>1, insert nseg-1 lines)
* eospar(1), eosflag(1), eospar(2), eosflag(2), eospar(3), eosflag(3), etc.
*   (set eosflag=1 to make corresponding parameter adjustable in minimization, given value used as initial
guess)
* aluminum (cgs):
1, 2.703, 1, 0, 1
5.302e5, 1, 14.555e5, 1, 1.538e5, 1, 0.0, 0, 0.0, 0
*
** STACK DEFINITIONS
* no. of stacks
3
* no. of layers (for stacks #1, #2, etc.)
1, 1, 1
* stack #, layer #, material #, thickness, thkflag
1, 1, 1, 0.1219, 0
2, 1, 1, 0.1518, 0
3, 1, 1, 0.1821, 0
*
** WINDOW DEFINITIONS
**** iwinmat=material number for window, -1 for free surface
* iwinmat1, iwinmat2, ...  (for stacks #1, #2, etc.)
-1, -1, -1
*
** VELOCITY PROFILE(S) INPUT
**** tscale, uscale = scale factors applied to u(t) input from file
**** tshft = time shift applied to u(t) input from file
**** tshftflag = set to 1 for allowing tshft to vary in minimization
**** tbpad = time to add in front of t-array with u=0
**** tepad = time to add at end of t-array with u=u_end
* stack #, tscale, uscale, tshft, tshftflag, tbpad, tepad, input filename for u(t) at measurement interface
1, 1.0e-6, 1.0e5, 0.0e-9, 0, 150e-9, 0.0, Z1190\N2_04Smooth_Al1200um.v
2, 1.0e-6, 1.0e5, -0.8e-9, 0, 190e-9, 0.0, Z1190\N3_0304CombinedSmooth_Al1500um.v
3, 1.0e-6, 1.0e5, 0.0e-9, 0, 230e-9, 0.0, Z1190\N4_0304CombinedSmooth_Al1800um.v
*
** FITTING AND INTEGRATION TIME RANGES
* no. of fitting regions
1
**** tfit1, tfit2 = low, high time points in u(t) corresponding to time
****                window in p(t) at x=0 used for minimization
**** tcut = time point in u(t) beyond which integration is not performed (overrides tepad)
**** ireg(i) = flag to include stack in ith fitting region
* stack #, tfit1, tfit2, tcut (set to -1 to use full signal), ireg(1), ireg(2), etc.
1, 219e-9, 294e-9, 314e-9, 1
```

```
2, 246e-9, 304e-9, 324e-9, 1
3, 271e-9, 316e-9, 336e-9, 1
*
** INTEGRATION CONTROL
**** dt = cell size in t
**** stab = percentage of Courant stability criterion (value from 0 to 1)
**** nsmooth = number of integration steps between smooths (=0 for no smoothing)
* dt, stab, nsmooth
0.2e-9, 0.8, 0
*
** EXPERIMENTAL UNCERTAINTIES
**** sigt = absolute time uncertainty
**** fvpf = fractional uncertainty in VPF (typically 0.001-0.01)
**** ffc = fractional uncertainty in fringe count (typically 0.02)
**** fc = fringe constant (velocity/fringe)
* stack #, sigt, fvpf, ffc, fc
1, 0.32e-9, 0.002, 0.02, 0.33e5
2, 0.66e-9, 0.002, 0.02, 1.11e5
3, 0.54e-9, 0.002, 0.02, 1.11e5
*
** MINIMIZATION CONTROL
**** fac = fractional variation in parameters to define initial simplex
**** ncycle = number of minimization cycles to perform
**** frestart = maximum fractional variation in parameters for restart vertices
**** ftol = fractional tolerance for minimization of average deviation
**** fk = strength of randomness in magnitude of simplex moves (0<=fk<=1)
**** tfac = multiplier to compute initial annealing temperature (0=no annealing)
**** teps = factor by which to reduce annealing temperature each pass
**** ntan = number of iterations per annealing pass
**** npan = maximum number of annealing passes
* fac,ncycle,frestart,ftol,fk,tfac,teps,ntan,npan
0.03, 2, 0.01, 1.e-5, 0.0, 0.0, 0.0, 75, 1
*
** OUTPUT FILENAMES
* stack #, output filename for s(t) at x=0
1, Z1190\soutN2fs234t3.dat
2, Z1190\soutN3fs234t3.dat
3, Z1190\soutN4fs234t3.dat
* stack #, tsnap (outside range to skip snapshot output), output filename for u,v,s(x) at t=tsnap
1, -2., dum
2, -2., dum
3, -2., dum
* output filename for minimization log file
Z1190\logZ1190fs234t3.dat
```

Two stacks, minimization without annealing, one fit region, three bootstrap segments

```
* INPUT FILE inpZ1190fs34b.dat FOR INVICE (VERSION 0.1beta)
*    unformatted but line-ordered
*    lines beginning with '*' are ignored
*
** MATERIAL(S) EOS
**** ieos=0 ; tabular p(v) or p(rho) isentrope
**** ieos=1 ; polynomial cL(us) isentrope, eospar=c0,c1,c2,c3,c4
**** ieos=2 ; polynomial cL(p) isentrope, eospar=pref,c0,c1,c2,c3,c4,c5
**** ieos=3 ; c0-s isentrope with s quadratic in strain, gamma/v = g0/v0, eospar=c0,s0,s1,s2,g0
* no. of materials (including windows)
1
* material #, rho0, ieos, istr, nseg
* pseg(i), eospar(1), eospar(2), eospar(3), etc.  (bootstrap data: use only if nseg>1, insert nseg-1 lines)
* eospar(1), eosflag(1), eospar(2), eosflag(2), eospar(3), eosflag(3), etc.
*   (set eosflag=1 to make corresponding parameter adjustable in minimization, given value used as initial
guess)
* aluminum (cgs):
1, 2.703, 1, 0, 3
40.2e10, 5.25e5, 15.307e5, 0.265e5, 0.0, 0.0
104.6e10, 10.716e5, 31.523e5, 1.252e5, 0.0, 0.0
15.999e5, 0, 47.68e5, 1, 2.79e5, 1, 0.0, 0, 0.0, 0
*
** STACK DEFINITIONS
* no. of stacks
2
* no. of layers (for stacks #1, #2, etc.)
1, 1
* stack #, layer #, material #, thickness, thkflag
1, 1, 1, 0.1518, 0
2, 1, 1, 0.1821, 0
*
** WINDOW DEFINITIONS
```

```
**** iwinmat=material number for window, -1 for free surface
* iwinmat1, iwinmat2, ...  (for stacks #1, #2, etc.)
-1, -1
*
** VELOCITY PROFILE(S) INPUT
**** tscale, uscale = scale factors applied to u(t) input from file
**** tshft = time shift applied to u(t) input from file
**** tshftflag = set to 1 for allowing tshft to vary in minimization
**** tbpad = time to add in front of t-array with u=0
**** tepad = time to add at end of t-array with u=u_end
* stack #, tscale, uscale, tshft, tshftflag, tbpad, tepad, input filename for u(t) at measurement interface
1, 1.0e-6, 1.0e5, -0.7e-9, 0, 260e-9, 0.0, Z1190\N3_0304CombinedSmooth_Al1500um.v
2, 1.0e-6, 1.0e5, 0.0e-9, 0, 300e-9, 0.0, Z1190\N4_0304CombinedSmooth_Al1800um.v
*
** FITTING AND INTEGRATION TIME RANGES
* no. of fitting regions
1
**** tfit1, tfit2 = low, high time points in u(t) corresponding to time
****               window in p(t) at x=0 used for minimization
**** tcut = time point in u(t) beyond which integration is not performed (overrides tepad)
**** ireg(i) = flag to include stack in ith fitting region
* stack #, tfit1, tfit2, tcut (set to -1 to use full signal), ireg(1), ireg(2), etc.
1, 291.2e-9, 346.8e-9, 366e-9, 1
2, 304.2e-9, 352.2e-9, 372e-9, 1
*
** INTEGRATION CONTROL
**** dt = cell size in t
**** stab = percentage of Courant stability criterion (value from 0 to 1)
**** nsmooth = number of integration steps between smooths (=0 for no smoothing)
* dt, stab, nsmooth
0.2e-9, 0.8, 0
*
** EXPERIMENTAL UNCERTAINTIES
**** sigt = absolute time uncertainty
**** fvpf = fractional uncertainty in VPF (typically 0.001-0.01)
**** ffc = fractional uncertainty in fringe count (typically 0.02)
**** fc = fringe constant (velocity/fringe)
* stack #, sigt, fvpf, ffc, fc
1, 0.72e-9, 0.002, 0.02, 1.11e5
2, 0.76e-9, 0.002, 0.02, 1.11e5
*
** MINIMIZATION CONTROL
**** fac = fractional variation in parameters to define initial simplex
**** ncycle = number of minimization cycles to perform
**** frestart = maximum fractional variation in parameters for restart vertices
**** ftol = fractional tolerance for minimization of average deviation
**** fk = strength of randomness in magnitude of simplex moves (0<=fk<=1)
**** tfac = multiplier to compute initial annealing temperature (0=no annealing)
**** teps = factor by which to reduce annealing temperature each pass
**** ntan = number of iterations per annealing pass
**** npan = maximum number of annealing passes
* fac,ncycle,frestart,ftol,fk,tfac,teps,ntan,npan
0.20, 1, 0.10, 1.e-5, 0.1, 0.0, 0.0, 100, 1
*
** OUTPUT FILENAMES
* stack #, output filename for s(t) at x=0
1, Z1190\soutN3fs34b.dat
2, Z1190\soutN4fs34b.dat
* stack #, tsnap (outside range to skip snapshot output), output filename for u,v,s(x) at t=tsnap
1, -2., dum
2, -2., dum
* output filename for minimization log file
Z1190\logZ1190fs34b.dat
```

Four stacks, minimization with annealing, three fit regions

```
* INPUT FILE inpZ1190fs1234mct3.dat FOR INVICE (VERSION 0.1beta)
*    unformatted but line-ordered
*    lines beginning with '*' are ignored
*
** MATERIAL(S) EOS
**** ieos=0 ; tabular p(v) or p(rho) isentrope
**** ieos=1 ; polynomial cL(us) isentrope, eospar=c0,c1,c2,c3,c4
**** ieos=2 ; polynomial cL(p) isentrope, eospar=pref,c0,c1,c2,c3,c4,c5
**** ieos=3 ; c0-s isentrope with s quadratic in strain, gamma/v = g0/v0, eospar=c0,s0,s1,s2,g0
* no. of materials (including windows)
1
* material #, rho0, ieos, istr, nseg
* pseg(i), eospar(1), eospar(2), eospar(3), etc.  (bootstrap data: use only if nseg>1, insert nseg-1 lines)
```

```
* eospar(1), eosflag(1), eospar(2), eosflag(2), eospar(3), eosflag(3), etc.
*   (set eosflag=1 to make corresponding parameter adjustable in minimization, given value used as initial
guess)
* aluminum (cgs):
1, 2.703, 1, 0, 1
5.25e5, 0, 15.353e5, 1, 0.329e5, 1, 0.0, 0, 0.0, 0
*
** STACK DEFINITIONS
* no. of stacks
4
* no. of layers (for stacks #1, #2, etc.)
1, 1, 1, 1
* stack #, layer #, material #, thickness, thkflag
1, 1, 1, 0.0920, 0
2, 1, 1, 0.1219, 0
3, 1, 1, 0.1518, 0
4, 1, 1, 0.1821, 0
*
** WINDOW DEFINITIONS
**** iwinmat=material number for window, -1 for free surface
* iwinmat1, iwinmat2, ...  (for stacks #1, #2, etc.)
-1, -1, -1, -1
*
** VELOCITY PROFILE(S) INPUT
**** tscale, uscale = scale factors applied to u(t) input from file
**** tshft = time shift applied to u(t) input from file
**** tshftflag = set to 1 for allowing tshft to vary in minimization
**** tbpad = time to add in front of t-array with u=0
**** tepad = time to add at end of t-array with u=u_end
* stack #, tscale, uscale, tshft, tshftflag, tbpad, tepad, input filename for u(t) at measurement interface
1, 1.0e-6, 1.0e5, 0.0e-9, 0, 125e-9, 0.0, Z1190\N1_0304CombinedSmooth_Al900um.v
2, 1.0e-6, 1.0e5, 0.0e-9, 0, 150e-9, 0.0, Z1190\N2_04Smooth_Al1200um.v
3, 1.0e-6, 1.0e5, -0.5e-9, 1, 190e-9, 0.0, Z1190\N3_0304CombinedSmooth_Al1500um.v
4, 1.0e-6, 1.0e5, 0.0e-9, 0, 230e-9, 0.0, Z1190\N4_0304CombinedSmooth_Al1800um.v
*
** FITTING AND INTEGRATION TIME RANGES
* no. of fitting regions
3
**** tfit1, tfit2 = low, high time points in u(t) corresponding to time
****                window in p(t) at x=0 used for minimization
**** tcut = time point in u(t) beyond which integration is not performed (overrides tepad)
**** ireg(i) = flag to include stack in ith fitting region
* stack #, tfit1, tfit2, tcut (set to -1 to use full signal), ireg(1), ireg(2), etc.
1,  101e-9, 200e-9, 220e-9, 1, 0, 0
2,  153e-9, 294e-9, 314e-9, 1, 1, 0
3,  203e-9, 347e-9, 367e-9, 1, 1, 1
4,  271e-9, 353e-9, 373e-9, 0, 1, 1
*
** INTEGRATION CONTROL
**** dt = cell size in t
**** stab = percentage of Courant stability criterion (value from 0 to 1)
**** nsmooth = number of integration steps between smooths (=0 for no smoothing)
* dt, stab, nsmooth
0.2e-9, 0.8, 0
*
** EXPERIMENTAL UNCERTAINTIES
**** sigt = absolute time uncertainty
**** fvpf = fractional uncertainty in VPF (typically 0.001-0.01)
**** ffc = fractional uncertainty in fringe count (typically 0.02)
**** fc = fringe constant (velocity/fringe)
* stack #, sigt, fvpf, ffc, fc
1, 1.02e-9, 0.002, 0.02, 1.11e5
2, 0.38e-9, 0.002, 0.02, 0.33e5
3, 0.72e-9, 0.002, 0.02, 1.11e5
4, 0.76e-9, 0.002, 0.02, 1.11e5
*
** MINIMIZATION CONTROL
**** fac = fractional variation in parameters to define initial simplex
**** ncycle = number of minimization cycles to perform
**** frestart = maximum fractional variation in parameters for restart vertices
**** ftol = fractional tolerance for minimization of average deviation
**** fk = strength of randomness in magnitude of simplex moves (0<=fk<=1)
**** tfac = multiplier to compute initial annealing temperature (0=no annealing)
**** teps = factor by which to reduce annealing temperature each pass
**** ntan = number of iterations per annealing pass
**** npan = maximum number of annealing passes
* fac,ncycle,frestart,ftol,fk,tfac,teps,ntan,npan
0.30, 1, 0.10, 1.e-5, 0.0, 0.7, 0.3, 50, 10
*
** OUTPUT FILENAMES
* stack #, output filename for s(t) at x=0
```

```
1, Z1190\soutN1fs1234mct3.dat
2, Z1190\soutN2fs1234mct3.dat
3, Z1190\soutN3fs1234mct3.dat
4, Z1190\soutN4fs1234mct3.dat
* stack #, tsnap (outside range to skip snapshot output), output filename for u,v,s(x) at t=tsnap
1, -2., dum
2, -2., dum
3, -2., dum
4, -2., dum
* output filename for minimization log file
Z1190\logZ1190fs1234mct3.dat
```

Appendix 2: List of Variables Used in the Code

Parameters[default]

mmax[4]	maximum number of materials (excluding windows)
nparmax[9]	maximum number of EOS parameters for any one material
nsegmax[3]	maximum number of EOS curve segments for bootstrapping
nsmax[8]	maximum number of stacks (number of VISAR profiles)
lmax[4]	maximum number of layers that may appear in any one stack
nmax[16384]	maximum number of points in any one u(t) input curve

Variables

b(mmax*nparmax)	current values of those parameters being varied in the minimization, corresponding to one vertex of the simplex
c(nmax)	sound speed profile at current x position in current stack
c0(lmax)	ambient sound speed of each layer in the current stack
chalf(nmax), cnew(nmax)	sound speed profile after half and full integration steps
dev	sum over fitting range of average square of deviation at each point in stress or time
devp	sum over stacks of square of deviation at one point
dt	fixed cell size in dimensional time; need not match time step of input velocity profile
dt1, dt2	accumulated negative time from the values at t(it1) and t(it2), corresponding to propagation of characteristics
dpdv, dpdus	derivatives used in EOS routines
dp, ds, du, dudt, dvdt	increments and derivatives used during backward integration
dx(lmax)	spatial integration step size (dimensional) in each layer of the current stack
epar(nparmax)	numerical values of EOS parameters for the material in the current layer of the current stack
eosflag(mmax,nparmax)	nonzero value indicates that corresponding EOS parameter is varied during minimization
eospar(mmax,nsegmax,nparmax)	numerical value for each EOS parameter on each EOS segment of each material (used as initial guess where eosflag=1)
fac	multiplicative factor to initialize simplex vertices

fc(nsmax)	fringe constant (VPF) for each input velocity profile
ffc(nsmax)	fractional uncertainty in the fringe count for each input velocity profile
fk	multiplicative factor for strength of randomness applied to simplex moves (set to 0 for no randomness)
frestart	multiplicative initialization factor for second and subsequent minimization cycles
fvpf(nsmax)	fractional uncertainty in the fringe constant for each input velocity profile
i, j, k, l	general integer counters
iabs	nonzero value indicates use of absolute-deviation minimization function (stress for iabs=1, time for iabs=-1)
ic	identification number of strength model used for the material in the current layer of the current stack
icycle	current cycle number of minimization procedure
id(nsmax)	index shift to synchronize each p(t) to stack is1
ie	identification number of EOS model used for the material in the current layer of the current stack
ieos(mmax)	identification number of EOS model used for each material
istr (mmax)	identification number of strength model used for each material
iline	line number in input file (excludes comment lines)
im, ip, il, is	counters for materials, parameters, layers, and stacks
iminimize	set to zero if no minimization calculation to be performed
ireg(nsmax,nsmax)	nonzero value indicates that a particular stack (1^{st} index) is included in a particular fitting region (2^{nd} index)
is1	stack index of stack having earliest starting time t(is1,1)
it1(nsmax), it2(nsmax)	index of maximum t1 and minimum t2 among stack solutions for each fitting range; also index of current t1 and t2 in SUBROUTINE BACKWARD
itr1(nsmax), itr2(nsmax)	shifted starting index of fit region for each stack
it1s, it2s	shifted indices of fit region
it1shift, it2shift	integral number of time cells encompassed in dt1 and dt2; when it1shift or it2shift > 0, it1 or it2 and dt1 or dt2 are decreased and it1shift or it2shift are reset
iter	number of iterations left in current annealing pass
itsnap	index of snapshot time

ivor	flag to indicate abscissa variable for tabular isentrope: 0 = volume, 1 = density
keep(nsmax)	flag to skip unnecessary integration of non-varying stacks: -1 = keep solution after first integration (reverts to 1), 0 = recompute solution every time, 1 = skip integration
lnum	layer number
mlayer(nsmax,lmax)	index number of material for each layer of each stack
mnum	material number
mtabp(mmax,nmax), mtabv(mmax,nmax)	pressure and volume along tabular isentrope for each material using ieos=0
mtabp2(mmax,nmax)	spline interpolation coefficients for tabular isentropes
mwin(nsmax)	index number of material used as window for each stack -1 = no window (free surface)
n	total number of spatial integration steps for current stack; number of profiles included in average
nbpad(nsmax)	number of time cells added in front of u(t) input curve
ncycle	number of minimization cycles to complete calculation
ndat(nsmax)*	number of points in u(t) input curve for each stack, modified during integration to maintain causal p(t)
ndatcon	number of points in u(t) input curve before padding
ndatkeep(nsmax)	stored values of ndat for each non-varying stack
nepad(nsmax)	number of time cells added in at end of u(t) input curve
nlay	total number of layers from all stacks (excludes windows)
nlayers(nsmax)	number of material layers in each stack (excludes window)
nmat	number of materials used in current problem
nmtab(mmax)	number of points in tabular isentrope for each material
npan	maximum number of annealing passes
npar(mmax)	number of parameters in EOS model for each material
nparwin(nsmax)	number of parameters in EOS model for each window
nreg	number of fitting regions used in minimization
nseg (mmax)	number of segments in EOS model for each material
nshift	integral number of cells encompassed in tshift; if nshift > 0, ndat and tshift are decreased and nshift is reset
nsmth	number of integration steps between smoothing operations
nspar (mmax)	number of parameters in strength model for each material
nstk	number of stacks used in current problem

ntan	maximum number of iterations to perform during each annealing pass
nvar	total number of parameters to be varied in the minimization
nx(lmax)	number of spatial integration steps in each layer of the current stack
p(nmax)	pressure history at current x
p0	reference pressure for current bootstrap segment
phalf(nmax), pnew(nmax)	pressure histories after half and full integration steps
pseg(mmax,nsegmax)	upper pressure boundary of each EOS segment for each material
psync(nmax)	pressure profile spline-interpolated on shifted time series (used when shift is not an integral number of cells)
s(nsmax,nmax)*	longitudinal-stress history for each stack
shalf(nmax), snew(nmax)	stress histories after half and full integration steps
sbar	average stress over all relevant stacks at one time value
sigs(nsmax,nmax)	computed uncertainty in stress for each stack
sigsint(nsmax,nmax)	computed uncertainty in stress for each stack at x=0 from the inverted (tinv, sinv) curves
sigt(nsmax)	uncertainty in time for each input velocity profile
simp(mmax*nparmax+1,mmax*nparmax)	current values of all parameters (including fixed ones) at each vertex of the simplex
simpb(mmax*nparmax)	best set of values for parameters thus far encountered
sint(nsmax,nmax)	longitudinal-stress history for each stack at x=0 from the inverted (tinv, sinv) curves
sinv(nmax)	uniform stress grid for inverted profile t(s)
skeep(nsmax,nmax)	stored stress profile for each non-varying stack
snapfile(nsmax)	name of snapshot file to write s(x), u(x), v(x)
soutfile(nsmax)	name of s(t) output file for each stack
spass(nsmax,2)	final stress endpoints for each fitting region
ssnap(nmax)	spatial distribution of longitudinal stress at time tsnap
stab	scale factor applied to Courant stability condition
t(nsmax,nmax)*	time series (originally from input curve) for each stack
t1(nsmax), t2(nsmax)*	lower and upper time bounds in solution p(t) for each stack, corresponding to fitting region for minimization
tann	current annealing temperature

tau(nmax)	shear stress history at current x (always 0 in version 0.1)
tauhalf(nmax), taunew(nmax)	shear stress histories after half and full integration steps
tinv(nsmax,nmax)	time series interpolated onto uniform stress grid for t(s)
tinvbar	average time over all relevant stacks at one stress value
t1keep(nsmax), t2keep(nsmax)	stored values of t1 and t2 for each non-varying stack
tbpad(nsmax)	length of time added in front of each u(t) input curve
tcon(nmax)	time series interpolated onto constant dt and front-padded
tcut(nsmax)	time value in u(t) input curve for each stack beyond which data is discarded and integration is not performed
tepad(nsmax)	length of time added at end of each u(t) input curve
teps	factor by which to reduce the temperature for each annealing pass (T_{n+1}/T_n = 1−teps)
thkflag(nsmax,lmax)	nonzero value indicates that corresponding layer thickness is varied during the minimization
tkeep(nsmax,nmax)	stored time series for each non-varying stack
tfac	multiplicative factor to determine initial temperature for simulated annealing (set to 0 to turn annealing off)
tfit1(nsmax)	time value in u(t) input curve for each stack corresponding to lower bound of fitting region for minimization
tfit2(nsmax)	as for tfit1, but corresponding to upper bound of region
tpass(nsmax,2)	final time endpoints for each fitting region
tscale(nsmax)	scale factor applied to time in each u(t) input file
tshft(nsmax)	time shift applied to each u(t) input file
tshftflag(nsmax)	nonzero value indicates that corresponding time shift is varied during the minimization
tshift	accumulated negative time from the current value of t(ndat), corresponding to propagation of a characteristic
tsnap(nsmax)	time at which to output a snapshot file
u(nmax)	velocity profile at current x position in current stack
u0	reference velocity for current bootstrap segment
ucon(nmax)	velocity interpolated onto constant dt and front-padded
uhalf(nmax), unew(nmax)	velocity profile after half and full integration steps
uinfile(nsmax)	name of u(t) input file for each stack
us(nmax)	history of principal-isentrope velocity at current x

ushalf(nmax), usnew(nmax)	principal-isentrope velocity histories after half and full integration steps
uscale(nsmax)	scale factor applied to velocity in each u(t) input file
usnap(nmax)	spatial distribution of velocity at time tsnap
v(nmax)	specific volume profile at current x in current stack
v0(mmax)	ambient specific volume of each material
vhalf(nmax), vnew(nmax)	specific volume after half and full integration steps
vnot	ambient specific volume of the material used in the current layer of the current stack
vpass	specific volume of window at current time; used to pass information to window EOS routine for computing dp/du
vsnap(nmax)	spatial distribution of specific volume at time tsnap
vwin(nmax)	specific volume profile at window interface
x	current spatial location during integration
xlayer(nsmax,lmax)	dimensional thickness of each layer of each stack
xsnap(nmax)	Lagrangian positions for snapshot file
y2out(nmax,2)	temporary arrays to hold results from derivative subroutines usvderivp and pderivus
yb	best (smallest) value of dev thus far encountered
yinit(mmax*nparmax+1)	current value of dev at each vertex of the simplex

* In SUBROUTINE BACKWARD, these variables lose their stack index dimension (an array becomes a vector, and a vector becomes a scalar), referring only to the current stack being integrated.

Appendix 3: Source Code Listing of EOS.f

```
c       INVICE
c       ------------- INVerse analysis of ICE waves
c       ----------------------------------------
c       COPYRIGHT 2001-2005 Sandia National Laboratories
c       written by Jean-Paul Davis
c       derived from a program by Dennis Hayes
c       version 0.1-beta
c       last updated 08 MAR 2005
c
c       This program uses the downhill simplex minimization method to find
c       the EOS parameters of one or more materials, given two or more u(t)
c       measurements (from different "stacks" of varying thicknesses and/or materials)
c       that resulted from the same loading history at x=0.  Backward integrations
c       are performed iteratively on the u(t) signals, and the average of the total
c       root mean square deviations from the mean of the solutions at x=0 is minimized.
c
c       This source file contains only those routines intended to be user-
c       customizable.  Once compiled, it must be linked with the object files
c       TABDPDV.obj, COMPDEV.obj, GETPAR.obj, INVICE.obj, and BACKWARD.obj.
c       The object files contain some internally-embedded routines
c       that are copyrighted by Numerical Recipes Software.  The object files
c       may be distributed with this source file and this copyright notice.
c       The object files may not be disassembled or reverse-engineered.
c
c       mmax    = max # of materials (default= 4)
c       nsmax   = max # of stacks (default = 8)
c       lmax    = max # of layers (default = 4)
c       nparmax = max # of EOS parameters (default = 9)
c       nmax     = max # of points in input u(t) curves (default = 16384)
c
c       NOTE: to change any of the above parameters, contact the author for a new
c       object files that have been compiled using the desired values.
c

c=======================================================================
        FUNCTION getnpar(ieos)
c=======================================================================
c       returns number of parameters used in selected eos
        INTEGER ieos,getnpar
        if      (ieos .eq. 0) then        ! tabulated p-v isentrope
                        getnpar=1
        else if (ieos .eq. 1) then        ! polynomial cL(u) isentrope up to 4th order
                        getnpar=5
        else if (ieos .eq. 2) then  ! polynomial cL(p) isentrope up to 5th order
                        getnpar=7
        else if (ieos .eq. 3) then        ! Mie-Grun Us-Up Hugoniot with s quadratic
                        getnpar=5                 ! in strain and gamma/v constant
        else if (ieos .eq. 4) then        ! cL(p) isentrope in three regions
                        getnpar=6     ! (linear, quadratic, linear)
        else
          write(6,*) 'ERROR: INVALID IEOS IN FUNCTION GETNPAR'
          STOP
        end if
        return
        END FUNCTION getnpar


c=======================================================================
        FUNCTION getnspar(istr)
c=======================================================================
c       returns number of parameters used in selected strength model
        INTEGER istr,getnspar
        if (istr .eq. 0) then     ! no strength model
          getnspar=0
        else
          write(6,*) 'ERROR: INVALID ISTR IN FUNCTION GETNSPAR'
          STOP
        end if
        return
        END FUNCTION getnspar
```

```
c=======================================================================
        SUBROUTINE usvderivp(x,y,dydx)
c=======================================================================
c     computes derivative dus/dp and dv/dp according to selected eos
        PARAMETER (nparmax=9)
        REAL v,us,p,dpdv,dpdus,x,y(2),dydx(2)
        REAL pref,cscale,c0,c1,c2,c3,c4,c5,cL,s,s0,s1,s2,ud,pd
        REAL gov,eta,dsdeta,phoeta,ph,dpdeta,pp
        REAL u0,p0,vnot,epar(nparmax)
        INTEGER ie,ic
        COMMON /epass/ ie,ic,vnot,epar
        COMMON /bootstrap/ u0,p0
        p=x
        us=y(1)
        v=y(2)
        select case (ie)
        case (0) ! interpolation of tabular p(v) isentrope
          dpdv=tabdpdv(v)
          dpdus=sqrt(-dpdv)
        case (1) ! polynomial cL(us/c0) isentrope up to 4th order
          c0=epar(1)
          c1=epar(2)
          c2=epar(3)
          c3=epar(4)
          c4=epar(5)
          ud=(us-u0)/c0
          cL=c0+ud*(c1+ud*(c2+ud*(c3+ud*c4)))
          dpdus=cL/vnot
          dpdv=-dpdus**2
        case (2) ! polynomial cL(P) isentrope up to 5th order
          pref=epar(1) ! pressure scale factor to avoid very small coefficients
          cscale=1.0/sqrt(pref)
          c0=epar(2)*cscale
          c1=epar(3)*cscale
          c2=epar(4)*cscale
          c3=epar(5)*cscale
          c4=epar(6)*cscale
          c5=epar(7)*cscale
          pd=(p-p0)/pref
         dpdv=-pref*((c0+pd*(c1+pd*(c2+pd*(c3+pd*(c4+pd*c5)))))/vnot)**2
          dpdus=sqrt(-dpdv)
        case (3) ! c0-s isentrope, s quadratic in strain, gamma/v=constant
          c0=epar(1)
          s0=epar(2)
          s1=epar(3)
          s2=epar(4)
          gov=epar(5)/vnot
          eta=1.0-v/vnot
          s=s0+s1*eta+s2*eta**2
          dsdeta=s1+2.0*s2*eta
          phoeta=(c0**2/vnot)/(1.0-eta*s)**2
          ph=phoeta*eta
          dpdeta=phoeta*(1.0+eta**2*(s**2+2*dsdeta))/(1.0-eta*s)
          dpdv=gov*(0.5*ph-p)-dpdeta*(1.0-0.5*gov*(vnot-v))/vnot
          dpdus=sqrt(-dpdv)
        case default
          write(6,*) 'ERROR: INVALID EOS IN FUNCTION DPDV'
          STOP
        end select
        dydx(1)=1.0/dpdus
        dydx(2)=1.0/dpdv
        return
        END SUBROUTINE usvderivp


c=======================================================================
        SUBROUTINE pvderivus(x,y,dydx)
c=======================================================================
c     computes derivatives dp/dus and dv/dus by calling usvderivp
        PARAMETER (nparmax=9)
        REAL p,v,us,dpderivs(2),x,y(2),dydx(2)
        us=x
        p=y(1)
        v=y(2)
        call usvderivp(p,(/us,v/),dpderivs)
        dydx(1)=1.0/dpderivs(1)   ! dpdusp=1/dusdp
        dydx(2)=-dpderivs(1)      ! dvdusp=-dusdp
        return
        END SUBROUTINE pvderivus
```

Distribution

1 Dennis Hayes
P. O. Box 591
Tijeras, NM 87059

1 Lynn Seaman
SRI International
Poulter Laboratory
333 Ravenswood Ave.
Menlo Park, CA 94025-3493

3 Washington State University
Department of Physics
Institute for Shock Physics
P. O. Box 642816
Pullman, WA 99164-2816
Attn: J. R. Asay
J. Ding
G. R. Hess

8 Los Alamos National Laboratory
Mail Station 5000
P. O. Box 1663
Los Alamos, NM 87545
Attn: W. V. Anderson
C. W. Greeff
R. L. Gustavsen
R. S. Hixson
D. E. Hooks
P. A. Rigg
S. Sheffield
D. G. Tasker

1 John Vorthman
547 N. Todd Loop
Los Alamos, NM 87544

6 Lawrence Livermore National Laboratory
7000 East Ave.
Livermore, CA 94550
Attn: M. Bastea
J. H. Eggert
J. H. Nguyen
D. A. Orlikowski
D. B. Reisman
K. S. Vandersall

1 Stephen Rothman
AWE, Aldermaston
Reading RG7 4PR
UNITED KINGDOM

1 Pierre-Louis Heréil
Centre d'Etudes de Gramat
46500 Gramat
FRANCE

Sandia Internal

1 MS-0836 M. R. Baer

1 MS-1110 J. B. Aidun

2 MS-1168 C. Deeney
1 MS-1168 M. D. Furnish
1 MS-1168 C. A. Hall

1 MS-1181 C. Alexander
2 MS-1181 L. C. Chhabildas
5 MS-1181 J.-P. Davis
1 MS-1181 D. H. Dolan
1 MS-1181 M. D. Knudson
1 MS-1181 T. J. Vogler
1 MS-1181 J. L. Wise

1 MS-9018 Central Technical Files

2 MS-0899 Technical Library