SANDIA REPORT

Tabular water properties interface for Hydra-TH: CASL THM.CFD.P6.03 Milestone Report

John H. Carpenter and Noel Belcourt

Sandia National Laboratories

Tabular water properties interface for Hydra-TH: CASL THM.CFD.P6.03 Milestone Report

John H. Carpenter[1] and Noel Belcourt[2]
[1]Computational Shock and Multiphysics,
[2]Severe Accident Analysis
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185

Abstract

Completion of the CASL L3 milestone THM.CFD.P6.03 provides a tabular material properties capability to the Hydra code. A tabular interpolation package used in Sandia codes was modified to support the needs of multi-phase solvers in Hydra. Use of the interface is described. The package was released to Hydra under a government use license. A dummy physics was created in Hydra to prototype use of the interpolation routines. Finally, a test using the dummy physics verifies the correct behavior of the interpolation for a test water table.

Contents

Chapter 1

Interpolation package

The interpolation package consists of a set of routines written in C that interpolate n-tuples of data defined on an unstructured triangular grid. A data file in NetCDF format containing the tabular information must be provided as input to the routines. This data file defines the topology of the triangular grid and also provides n-tuples of data at each node of the grid. More information on the format and generation of these tables will be provided in a report for a follow-on milestone. Herein the focus will be on the interface presented to host codes by the interpolation package.

There are two sets of routines that define the interface requirements, the core interface routines and the host implemented routines. The former provide the actual interpolation of material data, while the latter are support routines provided by the host for various I/O processes, including message output and data broadcasts for parallel support.

1.1 Core interface routines

The four core interface routines are prototyped in the **utri_eos_mig.h** header and must be wrapped with the **LAMBDA_FORTRAN** macro to support FORTRAN linkage requirements. The memory for all arrays passed to these routines must be managed by the host code. There are two arrays (ui and dc) that are passed to all the interface routines. They carry the model parameters and constants and should not be modified after initialization.

The initialization routine has the form:

> **utri_eos_init**(double *ui, double *gc, double *dc, const char *uc,
>   int *mdc, int *ndc, double *vi)

This routine initializes the model based upon the contents of the ui and uc arrays. The ui array must be of size **NUM_UI_INDEX_NAMES**. All entries should be set to zero except for ui[IVAR] which specifies the independent variable space, ui[DVAR] which specifies the desired dependent variables, ui[SR]=1., and ui[TYP] = 2. The dc array must be of size **NUM_DC_INDEX_NAMES**. On input, the first seven entries contain conversion factors for the host units into SI. These correspond respectively to the units for length, mass, time, temperature, discrete amount, electric current, and luminosity. For example, if the host is using cgs units then on would set dc[0] = 100.

On output the dc array contains model constants. Any modification of them will invalidate the model and possibly lead to fatal errors. The uc array is a character string containing the name of the desired input data table. Note that the init code will automatically append a prefix '-xx.nc' to this string, where the 'xx' depends upon the independent variable space specified in ui[IVAR]. The integer mdc must be the size of the dc array. On return ndc gives the amount of space used in the dc array. The vi array contains initial model values and must be of length 5. The initial values are currently not useful for the models used in Hydra. The gc variable is unused.

The cleanup routine has the form:

utri_eos_delete(double *ui, double *gc, double *dc)

Here the ui, gc, and dc arrays must be from a valid model. That is, they must have been returned from a call to the init function. The delete function frees all the resources associated with the model, invalidating the passed ui and dc arrays.

The interpolation routines have the form:

utri_eos_pxdrv(int *mc, int *nc, double *ui, double *dc,
                     double *pressure, double *xvar, int *compute_vars,
                     double *scratch, int *int_scratch)

utri_eos_ptsat(int *mc, int *nc, double *ui, double *dc, double *ptvar,
                     int *pttype, int *boundary, int *compute_vars,
                     double *sat1, double *sat2, double *scratch, int *int_scratch)

Both functions are vectorized entry points controlled by the two integers mc and nc. The value of mc is the vectorization stride and nc is the number of points to evaluate in that vector, starting at the first value. The ui and dc arrays are the model parameters and constants as returned from a prior call to the **utri_eos_init** function. The compute_vars integer gives the set of dependent variables upon which to interpolate.

The **utri_eos_pxdrv** method returns values of the dependent variables interpolated at the given independent variable location on the table mesh. The pressure and xvar arrays must be of length mc and contain the values of the independent variables at which to interpolate. The interpolated dependent variables are returned in the scratch array as described in the next section. The scratch array must be of length 17*mc and the int_scratch array must be of length 2*mc.

The **utri_eos_ptsat** method returns values of the dependent variables interpolated along the phase boundary specified by the boundary variable. The independent variable is specified in the array ptvar of length mc. The independent variable can either be pressure or temperature, which is specified by the integer pttype using the utri_dvar_names enum. Here the interpolated values are returned in the sat1 and sat2 arrays which must be of size 14*mc. The scratch array must be of size 5*mc and int_scratch of size 2*mc.

1.2   Host implemented routines

The three following routines must be implemented by the host to support the function of the inter-polation package.

void **reportUTriError**(const int code, const char * message)

int **is_master_node**()

void **broadcast_utri_data**(void *data, int n, int *error)

The **reportUtriError** method passes informational messages from the library to the host code for the host to present to the user in its normal manner. The input code denotes the type of message and may be 0 for an informational message, 1 for a non-fatal warning, and 2 for a fatal error. The message is a character string with the intended output. This routine must not return upon a fatal error.

The remaining two host implemented routines allow the host to control the reading of data from disk and subsequent communication of that data in a parallel computing environment. The **is_master_node** method must return a non-zero value for the node on which to perform I/O operations and zero otherwise. The **broadcast_utri_data** method must implement a broadcast method that transfers the data array of size n from the I/O node to all other nodes. An error code may be returned if this function does not otherwise cause termination on error. Example implementations of these function for Hydra are given in the dummy physics.

1.3   Specifying Model Variables

During initialization, as well as interpolation calls, the host must inform the library as to the model variables that are of interest. This allows the interpolation code to minimize extraneous computation. The set of independent variables on which to operate is specified to the **utri_eos_init** function in the ui[IVAR] parameter. It may be one of pressure–enthalpy (PH), pressure–entropy (PS), or pressure–internal energy (PE).

The desired dependent variables are specified using a bit field defined by the **utri_dvar_names** enum. All desired dependent variables must be specified in the ui[DVAR] parameter during the initialization call. Later calls to the interpolation routines may either request all these same variables, or only a subset of them. In either case, the interpolated variables are returned in a strided array with the stride determined by the vector size. The variables are placed in the order shown in Tab. 1.1. Note that the first two entries change depending on the input variable space.

9

Table 1.1. Index for Dependent Variables

Index	Dependent Variable
0	internal energy (PH, PS) or enthalpy (PE)
1	entropy (PH, PE) or enthalpy (PS)
2	density
3	temperature
4	Gibbs free energy
5	Helmholtz free energy
6	adiabatic sound speed
7	isothermal bulk modulus
8	isobaric heat capacity
9	isothermal heat capacity
10	material phase
11	dynamic viscosity
12	thermal conductivity
13	surface tension

1.4   Inclusion in Hydra

The interpolation package has been integrated in Hydra as a TPL, distributed as a tarball containing the library source files. The key source file for Hydra developers is the **utri_eos_mig.h** header. Including this file in a source will give access to all the function prototypes and enum definitions needed to access the interface described above. A test data table for water is also included in the distribution. However, the location to place data files will need to be examined more closely in the future, so as to provide better access for analysts as well as the testing infrastructure.

Chapter 2

Interface testing

The interpolation package installation was verified through a simple regression test using a dummy physics method in Hydra. This dummy physics also serves as an example implementation of the routines described in the previous chapter.

Given the number of elements sent for evaluation, a rectangular grid is created with them in pressure—internal energy space. If the number of elements does not make a perfect square, those above the highest perfect square are ignored. The **utri_eos_pxdrv** routine is then called for each element. The resulting interpolated values are output for plotting. The results are shown in Fig. 2.1 for the density and adiabatic sound speed. The blue lines in Fig. 2.1 show triangles drawn on top of the interpolated surface using the explicitly tabulated nodes of the mesh. Visually, the surface appears to line up exactly with the triangles, demonstrating the correct interpolation of the table. Due to the use of linear interpolation, the triangles should be flat planes and appear as such in the plots.

The table used in the test approximates the IAPWS-IF97 standard for the thermodynamic behavior of water [1]. It was tabulated in the pressure—internal energy space. The table bounds are rectangular from 5×10^7 to 10^8 Pa in pressure and from 273.16 to 1073.15 K in temperature. The table mesh was refined to a relative tolerance of 1, resulting in a very coarse approximation. As such, the table should not be used for real calculations, although it is sufficient to verify the interface implementation. The dummy physics in the test sampled this table in a rectangular 105x105 linear grid with the pressure range 5×10^7 to 10^8 Pa and the internal energy range 10^3 to 10^6 J/kg.

An additional test is also coded in the dummy physics to quantitatively verify the properly interpolated values at a given point on the input table. The results for all variables agree within a relative tolerance of 10^{-14} with values calculated with a separate implementation of the interpolation. Both tests were run successfully in serial and parallel, the latter testing the broadcast of the table data in the init routine. Thus, the installation of the interpolation package in Hydra appears to be correct.
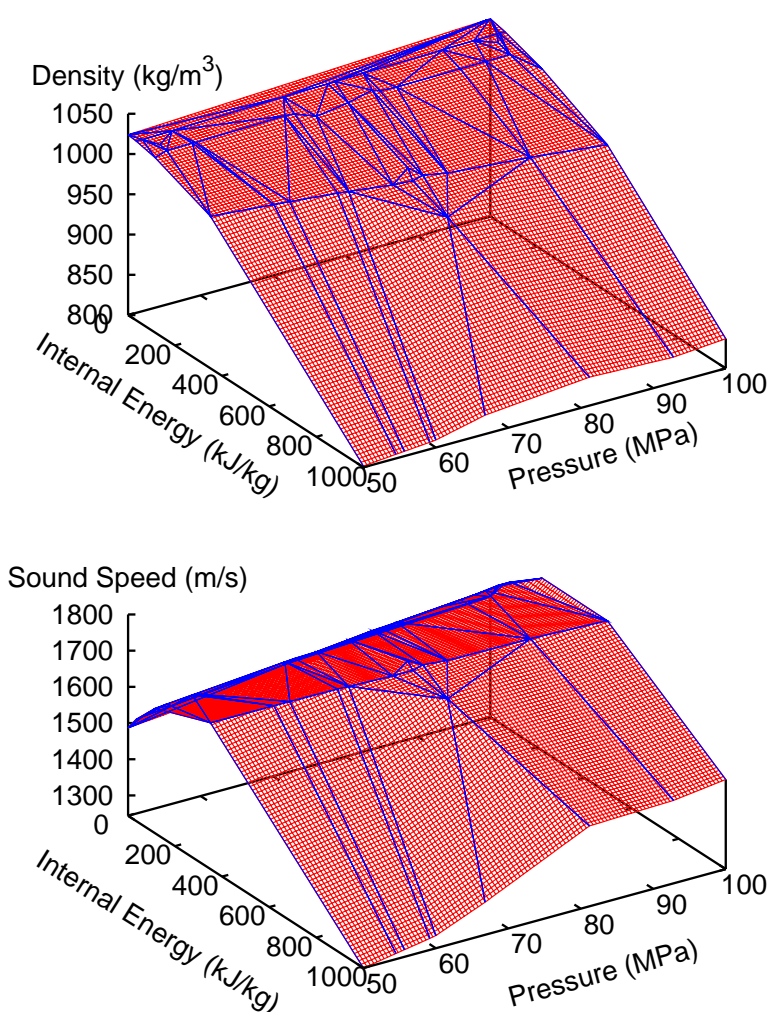
Figure 2.1. Interpolated density and sound speed for a coarse tabulation of the IAPWS-IF97 water standard. The tabulation is evaluated in the pressure—internal energy space. The red surface shows a uniform grid of samples evaluated by the dummy physics. The blue lines show the location of triangles in the table.

Chapter 3

Summary

The interpolation package, dummy physics implementation, and test water property table have all been distributed to Hydra. A government use notice has been approved for LANL and ORNL and a similar notice will be arranged for any other CASL partner that desires to use the code. In the longer term, two to three months, the interpolation package is planned to be released as open source software, thereby eliminating the need for the government use notices.

Documentation for the tabular interface has been provided in a number of forms, in addition to the beginning of this report. In particular, a set of view graphs describing the interface design has been provided for inclusion in the Hydra documents. The interface is also documented in the main header for the library.

Further work on tabular water property support will be performed for the follow on milestone THM.CFD.P7.04. This will include delivery of a production quality water table, the tabulation software and interpolation library as open source, and also integration with the Hydra material model interface.

References

[1] J. R. Cooper and R. B. Dooley, *Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam*, the International Association for the Properties of Water and Steam (2007).

DISTRIBUTION:

Sandia National Laboratories