SANDIA REPORT

SAND2008-4590
Unlimited Release
Printed July 2008

Two-way Coupling of Presto v2.8 and CTH v8.1

H. Carter Edwards, David A. Crawford, Christopher W. S. Bruner, Joseph E. Bishop

Sandia National Laboratories

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

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

SAND2008-4590
Unlimited Release
Printed July 2008

Two-way Coupling of Presto v2.8 and CTH v8.1

H. Carter Edwards
Computational Thermal and Fluid Mechanics
hcedwar@sandia.gov

David A. Crawford
Computational Thermal and Fluid Mechanics
dacrawf@sandia.gov

Christopher W. S. Bruner
Aerosciences
cwbrune@sandia.gov

Joseph E. Bishop
Solid Mechanics / Structural Dynamics
jebisho@sandia.gov

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185

Abstract

A loose two-way coupling of SNL's Presto v2.8 and CTH v8.1 analysis code has been developed to support the analysis of explosive loading of structures. Presto is a Lagrangian, three-dimensional explicit, transient dynamics code in the SIERRA mechanics suite for the analysis of structures subjected to impact-like loads. CTH is a hydro code for modeling complex multi-dimensional, multi-material problems that are characterized by large deformations and/or strong shocks. A fundamental assumption in this loose coupling is that the compliance of the structure modeled with Presto is significantly smaller than the compliance of the surrounding medium (e.g. air) modeled with CTH. A current limitation of the coupled code is that the interaction between CTH and thin structures modeled in Presto (e.g. shells) is not supported. Research is in progress to relax this thin-structure limitation.

Contents

Figures

1 Introduction

A new SIERRA Mechanics capability, labeled Fortissimo, provides a loose two-way coupling of SNL's Presto and CTH analysis codes to support the analysis of explosive blast loading of structures. This new single code includes all of Presto v2.8 [5] for modeling the structure, all of CTH v8.1 [3] for modeling the explosive blast in a medium (e.g. air) surrounding the structures, and a new interface component that exchanges data between Presto and CTH. The coupling between Presto and CTH is loose in that each code performs its own independent time step solution using boundary data from other code. For CTH this boundary data consists of a time-interpolated surface mesh (position and velocity) from Presto's finite element mesh. For Presto this boundary data consists of a time-averaged surface pressure from CTH. The Fortissimo interface code negotiates the time-interpolation, time-averaging, and exchange of this surface data between the codes during coupled execution.

1.1 Fortissimo contains Presto

Fortissimo couples Presto and CTH by linking both codes into a single executable named "`fortissimo`". The coupled analysis code is run just like Presto with a standard Presto input file; however, the name 'fortissimo' is substituted for the name 'presto' on a command line, e.g. "`sierra fortissimo ...`" instead of "`sierra presto ...`" If this Presto input file does not include a command block to activate CTH coupling then it will simply run Presto. The standard Presto input file is extended with the command block shown in Figure 1.

```
begin pressure
  surface = skin
  object type = face
  field variable = pressure
  cth input = <cth_input_file_name> \#
      idrun = h                     \#
      coordinate scaling = 2.54     \#
      pressure scaling = 14.5e-6
# coordinate scaling is Presto's inches to CTH's centimeters
# pressure scaling is CTH's dyne/cm^2 to Presto's psi
end
```

Figure 1. Presto input file extension for CTH pressure loading

The "`surface = skin`" command line in Figure 1 denotes that the entire surface skin of the structure is to be involved in the interaction. The two "`object type = face`" and “`field variable = pressure`” command lines informs Presto that that face pressure loads will be applied through the `pressure` variable. The “`cth input =`” command line provides the CTH input file name, run identifier (defaults to `h` when CTH is run standalone), how to scale coordinates when passing Presto data to CTH (e.g. inches to centimeters), and how to scale pressure when passing CTH data to Presto (e.g. $dyne/cm^2$ to *psi*). The `<cth_input_file_name>` is the name of a standard CTH input file that is passed to CTH as if it had been placed on the command line of a standalone CTH execution.

1.2 Fortissimo contains CTH

The CTH code is fully contained within the Fortissimo executable. The coupled code utilizes the immersed **rigid material** option within the DIATOM capability of CTH. It is necessary to invoke the rigid model with a **rigid** keyword in the CTH **control** section. The Presto mesh will have first-in priority in the CTH mesh as a rigid material. Other traditional CTH materials will be inserted via DIATOM afterwards. The current version of Fortissimo requires that all rigid materials inserted in this manner have at least three CTH cells through their thickness in order to prevent bleed-through of CTH material from one side of the Presto mesh to the other.

For testing purposes the CTH component can be run from within Fortissimo without running Presto. This CTH-only execution mode is activated by providing Fortissimo with the three line input file illustrated in Figure 2, where the `<cth_input_file_name>` denoted in Figure 2 is the name of a standard CTH text input file.

```
begin sierra fortissimo
  cth input = <cth_input_file_name> idrun = h
end
```

Figure 2. Fortissimo input file for CTH-only execution mode

1.3 CTH AMR Tracking of the Structure

CTH can be requested to adaptively refine its mesh in the vicinity of the Presto’s surface. This capability is supported by Fortissimo passing a characteristic length for each surface element from Presto to CTH. In CTH this is the database variable known as RGDL for visualization and adaptive meshing purposes. The RGDL variable is used to control CTH adaptive mesh refinement (AMR) as shown by example in Figure 3. This particular example input will direct CTH mesh refinement to occur in the vicinity of Presto surface mesh that has a characteristic element length less than 2 cm.

```
indicator
  val RGDL
  refbelow 2.0
endi
```

Figure 3. Example CTH input file (fragment) for CTH AMR in the vicinity of a Presto surface

1.4 Decoupled “Leap Frog” Time Stepping

The time steps performed by Presto and CTH are decoupled such that no attempt is made to align their respective simulation times. Thus each code is allowed to run at what it determines to be its own optimal time step. While Presto’s and CTH’s time steps are independent, Fortissimo manages *when* each code performs its time step. Presto and CTH time step such that the code that is behind in simulation time executes until its simulation time overtakes the other code’s simulation time. Thus the code that currently has a smaller time step will usually perform several time steps (i.e. subcycle) until it catches up to the code with the larger time step.

In this “leap frog” coupled time stepping algorithm the surface data exchanged between the codes will not be aligned with respect to the codes’ simulation time. The Fortissimo interface manages this time-misalignment by interpolating Presto’s moving surface mesh to CTH’s time step and time-averaging CTH’s surface pressure load given to Presto. In this algorithm the surface pressure load is time-lagged such that Presto is responding to pressure loads generated with respect to the motion of the surface during Presto’s previous time step. For the problems of interest and anticipated time step sizes we have assumed that this loose coupling strategy of time-averaging and time-lagging of pressure loads will not have a significant impact on the accuracy of the solution.

1.5 Coupled Parallel Execution

The exchange of surface data between Presto and CTH is complicated by Presto and CTH having completely independent parallel domain decompositions. In a parallel execution of Fortissimo it is highly unlikely that the surface data extracted from Presto on a given processor will align with the surface data that CTH requires on that processor. This situation is further complicated in that both Presto and CTH are capable of changing their domain decomposition via dynamic load balancing.

This parallel execution complexity is currently addressed by simply replicating the entire surface mesh from Presto on every processor before passing it to CTH. For the size of the problems of interest the surface mesh data structure is assumed to be sufficiently small so that it will not be a performance concern, both in the amount of memory consumed and in the time required to process the surface. The initial scaling study of the coupled code, presented in Section 3.1, indicates that this assumption will hold for the problems of interest.

If replication of surface data becomes a performance concern then the Fortissimo interface will have to be enhanced to query CTH's domain decomposition and redistribute the surface mesh extracted from Presto accordingly. Such an enhancement would significantly increase the complexity of the interface and increase communication overhead to determine the mapping between Presto's and CTH's domain decomposition. It was decided to not introduce this complexity and communication overhead unless it is shown to be necessary for scalability.

1.6 Coupled Restart

Coupled restart of Presto and CTH requires writing and reading of coordinated restart images to each code's respective restart files. The writing of restart files is scheduled through restart commands in Presto's input file (i.e., Presto's "`begin restart`" command block. When Presto performs a time step in which a restart image is written the coupling code instructs CTH to also write its own restart image. These two restart images are written into standard Presto and CTH restart files, no additional restart data is needed for the coupling code.

The Presto and CTH restart images will not have the same simulation time due to the "leap frog" loose coupling algorithm. In order to properly restart the coupled code Presto and CTH must each read restart images that were written as a pair. The simulation time-tags of each written Presto and CTH restart images are noted in the Presto log file as fillows.

```
COUPLED Presto + CTH WRITING RESTART
  Presto RESTART TIME TAG = <time for presto>
  CTH    RESTART TIME TAG = <time for cth>
```

In order to restart coupled code the respective read-restart commands must be entered into the respective input files for the analysis. These restart commands must specify each code's restart simulation times as noted in Presto's log file. For Presto this is "`RESTART TIME = <time for presto>`" and for CTH this is

```
restart
  time = <time for cth>
endr
```

2 Loose Coupling Algorithm

Fortissimo loosely couples Presto and CTH through a “leap frog” algorithm where Presto and CTH are allowed to take their own “best” time step, unconstrained by the other code’s time step. Fortissimo matches the Presto-to-CTH mesh and CTH-to-Presto pressure data exchanges by interpolating Presto mesh data to CTH’s time step and time-averaging CTH pressure data to Presto’s time step.

2.1 Algorithm Steps

A simple illustration of this “leap frog” algorithm is given in Figure 4.

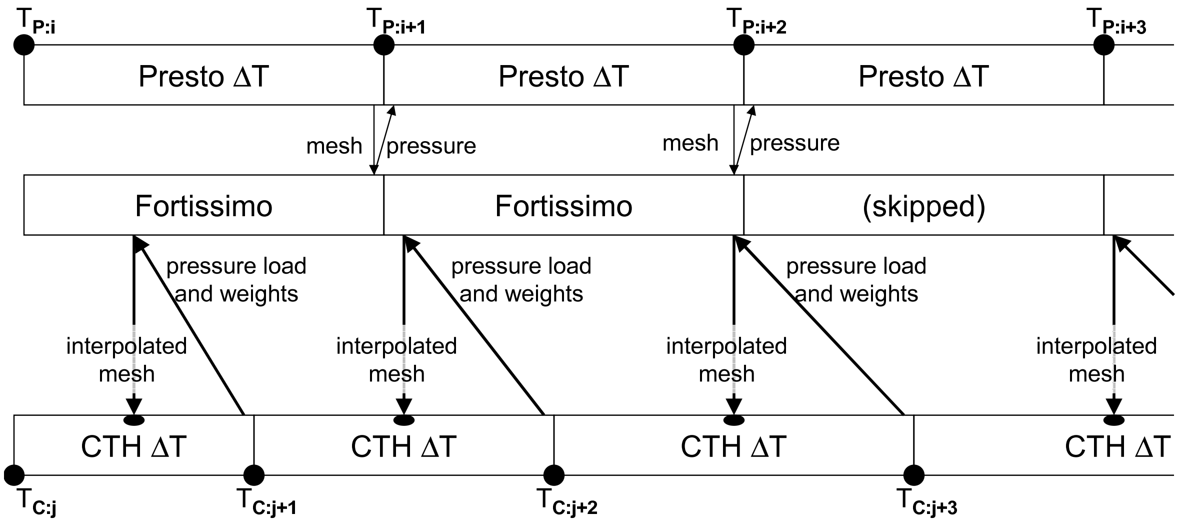

Figure 4. Fortissimo loose coupling “leap frog” time stepping

The sequencing steps of this algorithm are labeled with a number (1-8) and the first letter of the code (**P**resto, **F**ortissimo, **C**TH) to indicate which code has control of that step.

1[P] Presto performs an explicit dynamics time step $T_{P:i} \rightarrow T_{P:i+1}$ and calls the Fortissimo interface routine at the end of its time step.

2[F] Fortissimo queries the CTH status and *planned* CTH time step, $T_{Cj} \rightarrow T_{Cj+1}$. If the CTH status is not terminating and the midpoint of the planned CTH time step occurs before the end of the Presto time step then Fortissimo will call CTH. Otherwise Fortissimo returns execution control to Presto to take another time step.

let $T_{C:\frac{1}{2}} = \frac{1}{2}\left(T_{C:j} + T_{C:j+1}\right)$
if $T_{C:\frac{1}{2}} < T_{P:i+1}$ then go to **3[F]**
else go back to **1[P]**

3[F] Fortissimo extracts the current ($T_{P:i+1}$) surface mesh from Presto's solid structure(s) and replicates it on all processors. This surface mesh consists of triangular facet connectivity, vertex coordinates X and average velocity $\overline{V}$, and a characteristic length of the element underlying each facet (to support CTH AMR if enabled).

4[F] Fortissimo interpolates the vertex coordinates to the current CTH half time step, $T_{C:\frac{1}{2}}$.

$$X\left(T_{C:\frac{1}{2}}\right) = X(T_{P:i+1}) - \left(T_{P:i+1} - T_{C:\frac{1}{2}}\right)\overline{V}$$

The surface facet normals are computed from these interpolated coordinates and vertex normals are computed as the angle-weighted average of the attached facets' normals. Fortissimo calls CTH to perform its time step with the updated surface data.

5[C] CTH accepts the triangular faceted surface data and inserts it into CTH's data structures. CTH performs its planned time step $T_{C:j+n} \rightarrow T_{C:j+n+1}$, where if CTH is subcycling then n is the current subcycle counter. For this time step CTH defines a boundary condition for the embedded surface such that reaction loads are computed and accumulated on each facet. Each facet load consists of an load and weight contribution $\left(L_f, w_f\right)$ accumulated for each CTH cell that intersects a facet on that processor and during the time step. The mean CTH pressure loading for a facet is given by $P_f = L_f/w_f$ for all $\left(L_f, w_f\right)$ contributions to that facet.

6[F] Fortissimo checks the CTH return status and next planned CTH time step. If the CTH return status is not terminating and the CTH half time step is still less than the end of the Presto time step then Fortissimo loops back to step **4[F]** to subcycle CTH. Otherwise CTH subcycling is complete and Fortissimo continues on to step **7[F]**.

let $T_{C:\frac{1}{2}} = \frac{1}{2}\left(T_{C:j+n} + T_{C:j+n+1}\right)$
if $T_{C:\frac{1}{2}} < T_{P:i+1}$ then go back to **4[F]**

7[F] Fortissimo sums the accumulated per-processor CTH facet loading contributions from each processor onto the processors from which the facets originated. The mean pressure loading for each facet, $P_f = L_f/w_f$, is then input to the Presto data structures. Execution control is then returned to Presto (go to step **1[P]**) to perform its next time step with the current CTH-determined surface loads.

2.2 Interface Mesh

The interface mesh between Fortissimo and CTH consists of a set of triangular facets with normal vectors and element thicknesses. If the Presto mesh defines quadrilateral facets then Fortissimo splits these facets along a diagonal into two triangles. A surface defined by triangle facets is used for interface simplicity and processing efficiency within CTH. The interface data structure is a single 'C' language 'struct' consisting of simple flat input arrays for the triangle-to-vertex connectivity, vertex coordinates, facet normals, vertex normals, and facet-element thickness; and output arrays for the accumulated loads and weights.

```
struct Triangle_Faceted_Surface {
  unsigned num_dim ;      /* 3D (someday could be 2D) */
  unsigned num_loading ;  /* Per-facet loading coefficients */
  unsigned num_nodes ;    /* Total number of nodes  on this processor */
  unsigned num_facets ;   /* Total number of facets on this processor */
  double * node_coord ;   /* Node coordinates  [ num_nodes * num_dim ]  */
  float  * node_velocity; /* Node velocities   [ num_nodes * num_dim ]  */
  float  * node_normal ;  /* Node unit normal  [ num_nodes * num_dim ]  */
  float  * facet_normal ; /* Facet unit normal [ num_facets * num_dim ] */
  float  * facet_length ; /* Facet characteristic length [ num_facets ] */

  /*  Node-connectivity of the triangular facets.
   *  The connectivity values are zero-based ordinals into the
   *  'node_coord', 'node_velocity', and 'node_normal'  arrays.
   */
  unsigned * facet_node_connectivity ;  /* [ num_facets * num_dim ] */

  /*  The last coefficient is the accumulation weight. */
  double * facet_loading ; /* [ num_facets * num_loading ] */
};
```

Figure 5. Simple 'C' language interface for interface surface data

2.3 Parallel Gather / Reduce_Scatter

Presto's distributed surface is gathered to all processors into the data structure given in Figure 5 which is identically duplicated on all processors. For this gathering operation Fortissimo extracts Presto's surface data that is available on each processor according to Presto's domain decomposition. This decomposed surface data is replicated on all processors using the Message Passing Interface (MPI) [4] function `MPI_Allgatherv` function.

CTH computes loads for facets which intersect cells in CTH's domain decomposition. If a given facet intersects CTH cells on two different processors then CTH will compute a contribution for that facet, but only on the cell's processor. These facet load contributions (partial sums) must be summed among processors to determine the complete load on a given facet. The facet arrays (Fig. 5) are partitioned such that each processor "owns" a contiguous span of an array. This partitioning allows the final parallel assembly of the facet load data to be performed with a single call to the MPI function `MPI_Reduce_scatter`.

The parallel gather and reduce_scatter operations are implemented in the described manner for simplicity and performance. It is simpler to utilize a single MPI collective communication function that to orchestrate a set of point-to-point communications. It is also our experience that a MPI collective function will be faster than an application-orchestrated set of point-to-point communications that accomplish the same operation.

2.4 Quadrilateral Splitting

Quadrilateral surface facets extracted from Presto's mesh are split along a diagonal into two triangles when loaded into the interface data structure. The splitting diagonal is selected such that the resulting two triangles' normals are the least aligned. Given a quadrilateral with vertex coordinates $\{X_1, X_2, X_3, X_4\}$ the quadrilateral is split as follows.

$$\begin{array}{ll} n_1 = (X_2 - X_1) \times (X_4 - X_1) & n_2 = (X_3 - X_2) \times (X_1 - X_2) \\ n_3 = (X_4 - X_3) \times (X_2 - X_3) & n_4 = (X_1 - X_4) \times (X_3 - X_4) \\ d_{13} = (n_1/\|n_1\|) \bullet (n_3/\|n_3\|) & d_{24} = (n_2/\|n_2\|) \bullet (n_4/\|n_4\|) \end{array}$$

If $d_{13} < d_{24}$ then the quadrilateral is split along the 2-4 diagonal to place vertices #1 and #3 in separate triangles, otherwise it is split along the 1-3 diagonal to place vertices #2 and #4 in separate triangles.

3 Initial Testing

3.1 Scaling Study

The runtime of Fortissimo is a sum of the Presto runtime, CTH runtime, and the transfer time between the two codes orchestrated by Fortissimo. For most problems the Fortissimo transfer time is expected to be a small percentage of the Presto and CTH runtime. Thus, the total runtime should be approximately equal to Presto runtime plus the CTH runtime. For problems dominated by CTH the cpu scaling rate of Fortissimo should be comparable to the scaling rate of CTH. Likewise, for problems dominated by Presto the cpu scaling rate of Fortissimo should be comparable to the scaling rate of Presto. Either extreme could be realized by using a relatively fine mesh in either CTH or Presto, respectively.

For a first scaling study an example problem dominated by CTH was chosen. The problem consists of an initially stress free elastic spherical shell containing an inert gas at an initially uniform pressure of 100 atm. The inside radius of the sphere is 24 inches and the thickness is 4 inches. Two unstructured finite-element meshes (**m1** and **m2**) were generated for this problem using the Cubit meshing tool, as shown in Figure 6. Both meshes have four elements through the sphere thickness. However, the second mesh (**m2**) has approximately

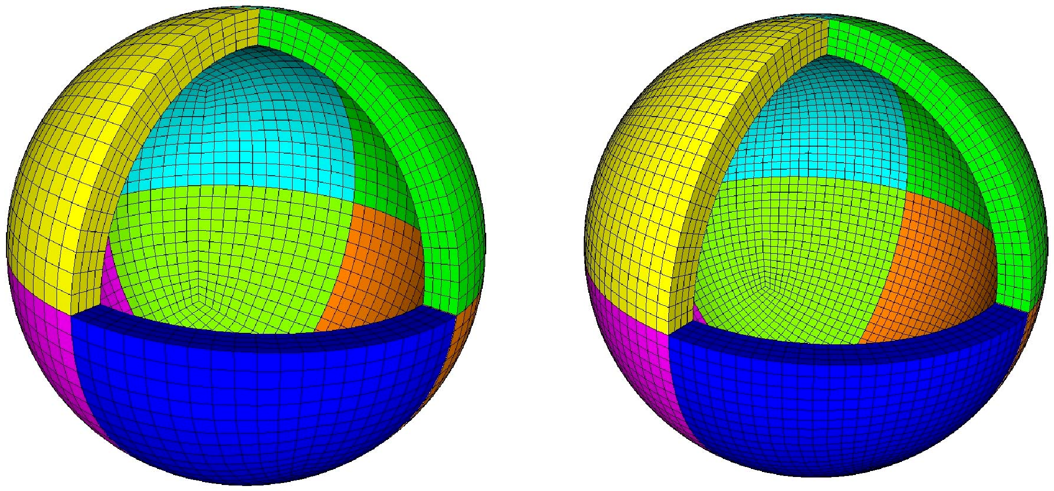

Figure 6. Scaling study spherical shell meshes **m1** and **m2** with removed octant

twice as many elements around the circumference of the sphere as the first mesh (**m1**). These two baseline meshes are referred to as resolution **r0**. For this scaling study each mesh (**m1-r0** and **m2-r0**) was refined by uniformly dividing each hexahedral element into eight elements resulting in two meshes at resolution **r1** (**m1-r1** and **m2-r1**). A further refinement was performed to obtain two meshes at resolution **r2**. A summary of number of elements for each mesh and resolution is given in Table 1.

For the structured CTH mesh a uniform rectangular grid was used. The baseline CTH grids (**r0**) were chosen such that were at least three CTH cells through the sphere thickness to avoid bleed through of the internal gas to the exterior of the sphere. The two baseline CTH grids were then refined by decreasing the cell size by factor of two for each refinement level. The total number of CTH cells used for each finite-element mesh type and refinement level is given in Table 1.

Table 1. Scaling Presto element and CTH cell counts

	Mesh **m1**		Mesh **m2**	
refinement	Presto	CTH	Presto	CTH
r0	10.7 K	216 K	20.5 K	422 K
r1	86 K	1.73 M	164 K	3.38 M
r2	688 K	13.8 M	1.13 M	27 M

The scaling study was performed on SNL's Thunderbird platform with CTH performing 100 time steps and Presto performing 575 time steps. The simulation for the mesh-grid combination **m1-r0** used four cpus, and the Fortissimo simulation for the mesh-grid combination **m2-r0** used eight cpus as list in Table 2. The next finer mesh resolution used a factor of eight more processors. Thus, the number of elements and cells per processor was kept constant for each of the six mesh-grid combinations. The resulting run times are given in Table 2 and graphed in Figure 7. A linear regression of the runtime data, also shown in Figure 7, has a slope of 0.26, or approximately 74% scaling efficiency for this problem. This scaling result is comparable to that which would be expected from a similar, pure CTH calculation, confirming that the runtime of this calculation is dominated by CTH.

Table 2. Scaling study results on SNL's Thunderbird

mesh refinement	processor count	total run time (seconds)	interface memory
m1-r0	4	1248	0.7 Mb
m2-r0	8	1017	1.4 Mb
m1-r1	32	1757	2.9 Mb
m2-r1	64	1660	5.6 Mb
m1-r2	256	2711	11.7 Mb
m2-r2	512	4276	22.3 Mb

The interface memory listed in Table 2 is the memory required by the interface mesh data structure described in Section 2.2. Assuming that this scaling problem is representative of the proportions of CTH cells, Presto elements, and interface facets that will be appear in an actual coupled analysis, then the current strategy of replicating the interface mesh data on all processors should *not* impose a constraint on the scalability of the coupled code.

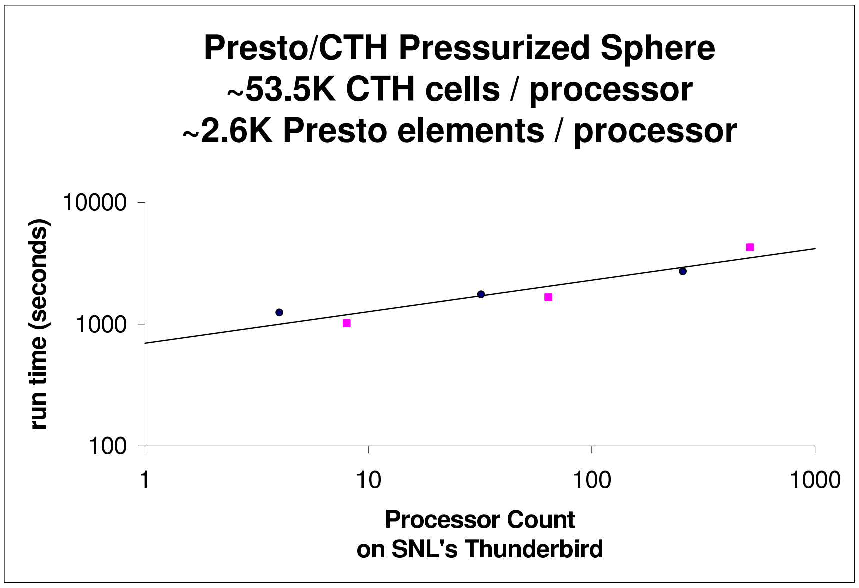

Figure 7. Scaling results for problem size scaled with processor count

3.2 Hydrobulge Cylinder Experiment

A standard validation test for fluid-structure interaction codes as well as standalone CTH is the Hydrobuldge Cylinder experiment [2]. A schematic of the test setup is shown in Figure 8. In this test a small explosive charge is placed in the center of a water filled 5086 aluminum tube. In this test the explosive charge was 2.8g of PETN. The tube was 17.8 cm long with a 10.1 cm outer diameter and a 0.635cm wall thickness. The tube deforms plastically but does not rupture. Measurements are made of the water-aluminum interface pressure and the outside surface velocity of the aluminum tube, both at the midplane.

The finite element mesh of the aluminum tube used in the Presto portion of the simulation is shown in Figure 9. The mesh consisted of 110K hexahedral elements with aspect ratios of approximately one. Four elements through the thickness of the tube where adequate to capture the deformation of the tube as further mush refinements showed no significant change in results. The base CTH grid contained 11.7M cells with approximately eight cells through the tube thickness. AMR was used only to resolve the explosive. No attempt was made to model the plastic enclosures. The BCJ viscoplastic constitutive model [1] was used in Presto to model the aluminum tube. The BCJ model is a rate dependent plasticity model with the ability to model adiabatic heating and damage due to void growth. The equivalent plastic strain field is shown in Figure 10. The maximum plastic strain is roughly 12%. Parameters studies of the DCJ material model showed that the simulation results were insensitive to both adiabatic heating and damage effects. A comparison of the experimental and simulation surface velocities (outer) at the midplane are shown in Figure 11. The simulation captures the peak surface velocity but underpredicts the post peak response Validation studies using CTH alone confirm that the matching of peak surface velocity is

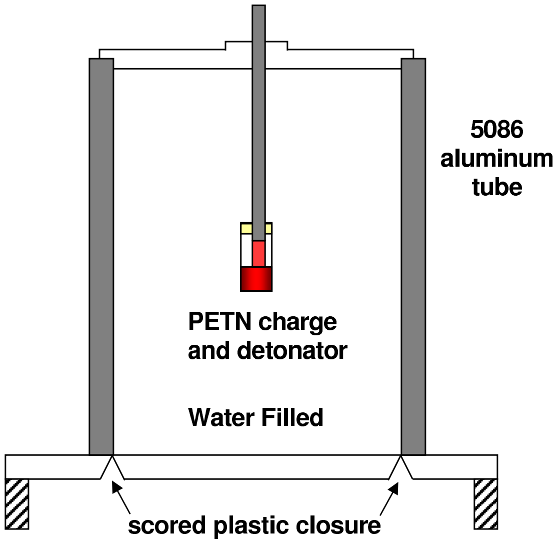

Figure 8. Schematic of the Hydrobulge Cylinder Experiment

a critical test of the coupling algorithm. An almost identical post-peak underestimate occurs in the CTH-only studies. The reason is probably due to incomplete understanding of the material response under these conditions and not do to any deficiencies in the coupling algorithm.

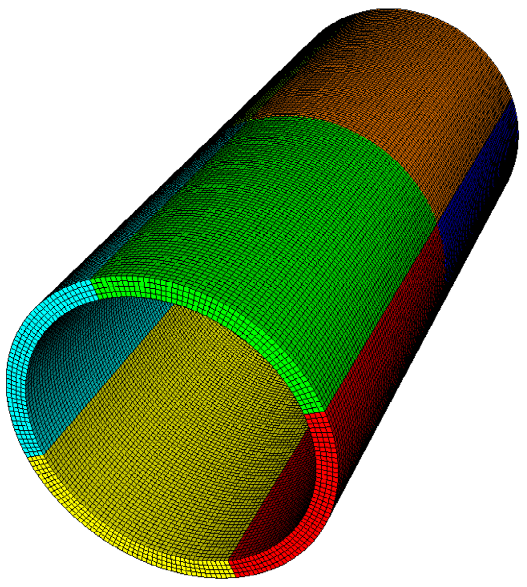

Figure 9. Presto Mesh for the hydrobulge cylinder experiment

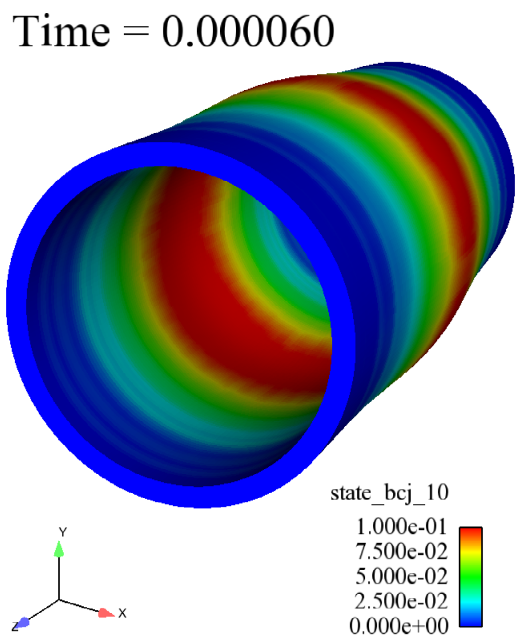

Figure 10. Equivalent plastic strain in the aluminum tube

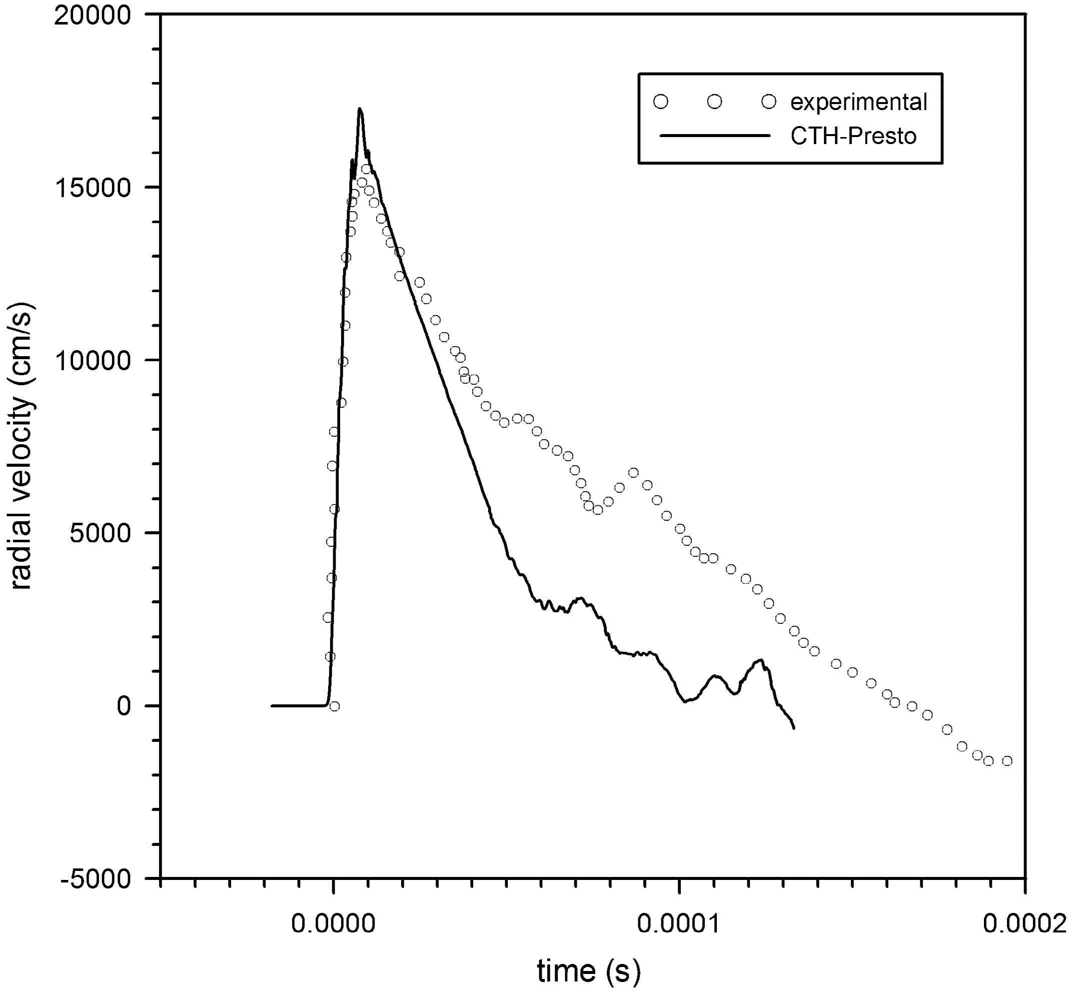

Figure 11. Comparison of experimental and simulatian surface velocities

4 Plans

Two-way coupling strategies between CTH and Presto are still evolving and should be considered experimental in the present version. Work is ongoing to implement coupling between Presto shell elements with CTH. The promise of this capability is to remove the three-CTH-cell-through-the-thickness requirement, producing a significant performance improvement. We intend for the next Fortissimo release to provide an experimental shell coupling capability. Other areas that we will improve as required include on-the-fly visualization of interfacial quantities via Spymaster, improvements to restart functionality and more scalable approaches to load balancing of the CTH/Presto interface across processors.

References

[1] D. Bammann. Modeling temperature and strain dependent large deformations in metals. *Appl. Mech. Rev.*, 5:312–319, 1990.

[2] G. Chambers, H. Sandusky, F. Zerilli, K. Rye, and R. Tussing. Pressure measurements on a deforming surface in response to an underwater explosion. In Schmidt/Dandekar/Forbes, editor, *Shock Compression of Condensed Matter*, number CP429. The American Institute of Physics, 1997.

[3] D. A. Crawford and et al. *CTH User's Manual and Input Instructions: Version 8.0*. Sandia National Laboratories, March 2007. CTH Development Project.

[4] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*, March 1994.

[5] James Richard Koteras, Arne S. Gullerud, Nathan Carl Crane, and Jason Dean Hales. Presto user's guide version 2.6. Technical report SAND2006-6093, Sandia National Laboratories, Albuquerque, New Mexico 87185, October 2006.

DISTRIBUTION:

1	MS 0382	Basil Hassan, 01541
1	MS 0380	Joe Jung, 01542
1	MS 0832	Gene S. Hertel, 01545
1	MS 0382	H. Carter Edwards, 01541
1	MS 0382	Greg D. Sjaardema, 01541
1	MS 0382	Michael W. Glass, 01541
1	MS 0380	Arne S. Gullerud, 01542
1	MS 0380	Martin W. Heinstein, 01542
1	MS 0380	Sam W. Key, 01542
1	MS 0372	Jeff D. Gruda, 01524
1	MS 0372	John Pott, 01524
1	MS 0836	David A. Crawford, 01516
1	MS 0825	Chris W. Bruner, 01515
1	MS 0346	Joe E. Bishop, 01525
1	MS 0847	Stephen W. Attaway, 01534
1	MS 0372	Jonathan S. Rath, 01524
1	MS 0899	Technical Library, 9536

Sandia National Laboratories