ARMY RESEARCH LABORATORY

Extending Post-Processing and Run Time Capabilities of the CTH Shock Physics Code

by Jerry A. Clarke and Eric R. Mark

ARL-TR-3576 **August 2005**

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-TR-3576 **August 2005**

Extending Post-Processing and Run Time Capabilities of the CTH Shock Physics Code

Jerry A. Clarke and Eric R. Mark
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* August 2005	**2. REPORT TYPE**	**3. DATES COVERED (From - To)** July 2004 to July 2005
4. TITLE AND SUBTITLE Extending Post-Processing and Run Time Capabilities of the CTH Shock Physics Code		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Jerry A. Clarke and Eric R. Mark (both of ARL)		**5d. PROJECT NUMBER** 5UH7FC
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory, Computational and Informational Sciences Directorate, Aberdeen Proving Ground, MD 21005-5066		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-3576
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

CTH, which is not an acronym, is a multi-material, large deformation, strong shock wave, solid mechanics code that runs on most UNIX workstations and massively parallel processing supercomputers. CTH is one of the most heavily used computational structural mechanics codes on Department of Defense high performance computing platforms. Although CTH includes some internal graphics capabilities, it is preferable to take advantage of widely used scientific visualization packages such as EnSight[1] and ParaView[2] to analyze the results of calculations. A new method has been devised that extends the capabilities of CTH to allow three-dimensional polygonal models to be written directly from a running calculation in a format compatible to both EnSight and ParaView. Additionally, an interpreter for the scripting language Python has been embedded into CTH and its post-processor Spymaster. Embedded Python allows for almost limitless, parallel capabilities to be added, which do not require a recompilation or re-linking of the CTH executable. Examples of these capabilities include one- and two-way code coupling and behind-armor debris applications.

[1]EnSight is a registered trademark of Computational Engineering International, Inc.
[2]ParaView is a registered trademark of Kitware, Inc.

15. SUBJECT TERMS
CTH, Python, isosurface

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Jerry A. Clarke
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	15	**19b. TELEPHONE NUMBER** *(Include area code)* 410-278-9279

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

INTENTIONALLY LEFT BLANK

1. CTH on High Performance Computing (HPC) Platforms

CTH is a multi-material, large deformation, strong shock wave, solid mechanics code and is one of the most heavily used computational structural mechanics codes on Department of Defense (DoD) HPC platforms. Typical applications of CTH on DoD platforms employ a three-dimensional rectilinear structured mesh. Values such as material volume fraction, pressure, and stress are calculated for the cells of this mesh and are written to files at regular intervals in the calculation's simulation time. CTH has added an adaptive mesh refinement (AMR) capability which dynamically provides greater resolution in areas of interest.

AMR and non-AMR (flat mesh) data can be visualized via the Spymaster graphics package that is included with CTH. Spymaster graphics can be produced during run time or as a post-processing step. To enable post processing via Spymaster, CTH can produce data files in a special "*spyplt*" format. On parallel platforms, CTH will typically save one spyplt file per processor used in the calculation. Since production CTH calculations regularly require tens or hundreds of millions of cells in the computational mesh, the amount of data saved to disk can be enormous.

Although Spymaster supports many common visualization operations, such as cutting planes and isosurfaces, it is not an interactive graphical user interface-based application and does not present the user with a viewing window. This can make it particularly cumbersome to use for adjusting the viewing angle when one is generating images or movies. Because of this limitation, it is preferable to take advantage of widely used scientific visualization packages such as EnSight and ParaView to analyze the results of calculations. EnSight is a widely used commercial visualization package and ParaView is a popular open source visualization package built on top of the Visualization Toolkit (vtk).

2. Extending Spymaster

In addition to visualization of the data for analysis, the results of calculations can be used to couple simulations. For example, we can accomplish blast loading on a structure by producing pressure-time history curves from a CTH simulation and applying them to a finite element structural dynamics code such as LS-DYNA[1].

The eXtensible Data Model and Format (XDMF) can be used to accomplish visualization and code coupling. For visualization, EnSight and ParaView have XDMF readers. Coupling tools

[1]LS-DYNA, which is not an acronym, is a trademark of Livermore Software Technology Corporation.

that use the XDMF format are provided within the interdisciplinary computing environment (ICE).

Producing XDMF data from CTH requires the addition of commands to Spymaster. We accomplish this by interfacing with the C-like language interpreter (S-Lang) which Spymaster uses to parse input commands. Since this requires additional "C" code to be added to Spymaster and linkage with the XDMF library, a new executable (IceSpy) is produced that understands all normal Spymaster commands plus additional commands used to produce XDMF.

The most notable of these additional commands is

XdmfIsoAllMaterial (*scalar, mirror*)

This command employs the Spymaster internal isosurface generator to produce a surface for each material in the calculation where the material volume fraction is 0.51. The *scalar* is used to interpolate a value such as pressure on the surface, while *mirror* will reflect the surface across planes of symmetry. Instead of the isosurface generator producing a two-dimensional image, however, this new command produces an XDMF file with the polygons for the surfaces which can then be imported into EnSight or ParaView.

Additional commands allow finer control such as producing surfaces of other scalars than material volume fraction. While useful by themselves, adding more commands requires recompiling and re-linking of the IceSpy executable. *An important limitation is that the internal Spymaster isosurface generator will only interpolate one scalar value onto the surface at a time.* This means that to color the surface by more than one scalar, we must take another approach.

3. Embedding an Interpreter

Python is a heavily used interactive, object-oriented programming scripting language. Projects such as SPaSM (Scalable Parallel Short-range Molecular-dynamics) and VTF (Virtual Test Facility) have shown that a Python interpreter can be embedded in a parallel high performance computing code to provide a flexible interface to a wide variety of functionality. Using this concept, we embedded a Python interpreter in the IceSpy executable. Two more commands have been added to IceSpy to access python from CTH:

XdmfPythonExecFile (*filename*)

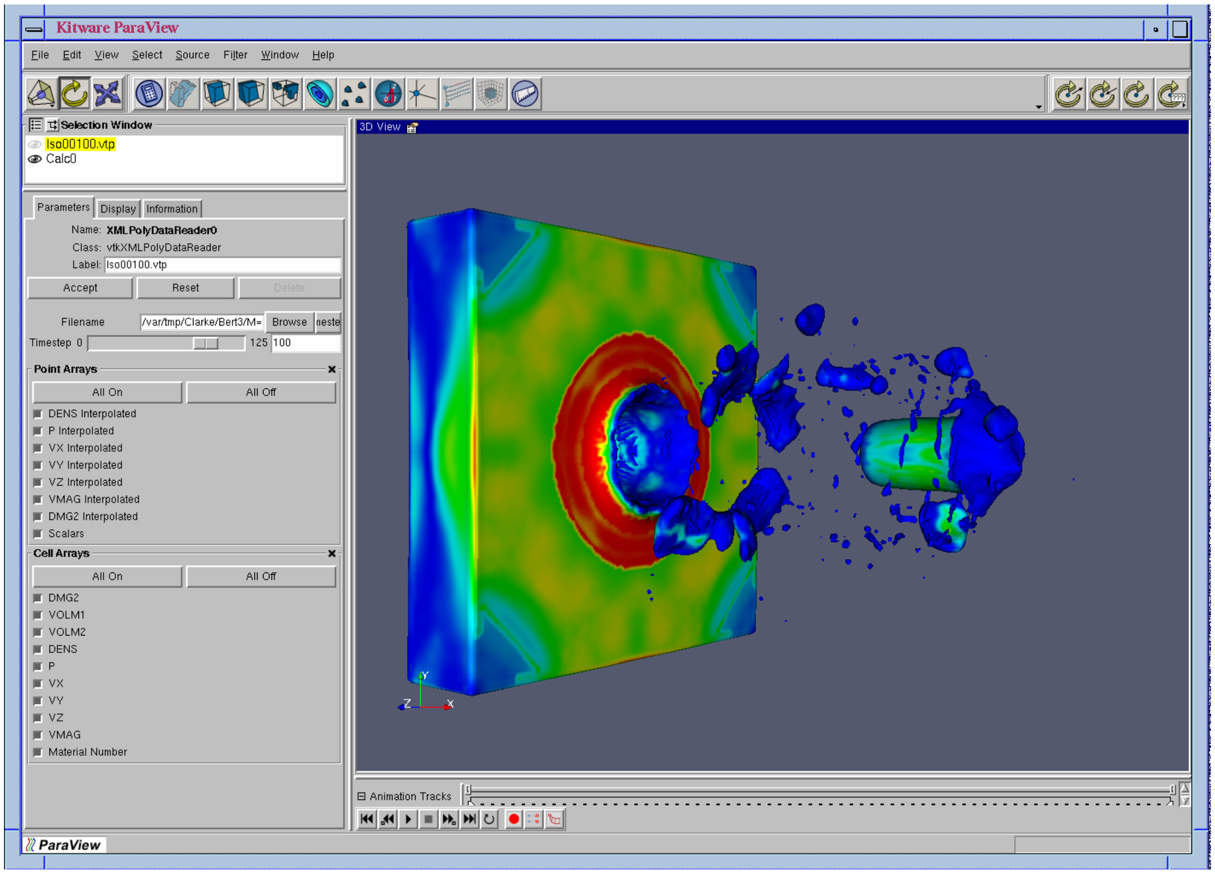
XdmfPythonExec (*string*)

These commands initialize the Python interpreter and execute Python commands from a file or a string.

A Python interpreter itself is fairly thin and has limited functionality beyond the basic language constructs. We obtain additional functionality by *importing* external modules. For example, importing the xml.dom module allows Python scripts to easily parse XML documents. The vtk has been *wrapped* to allow a Python script to access its functionality. Once imported, the vtk module allows the user to create vtk objects and call methods from python; no code needs to be recompiled or linked.

The same has been done for the CTH data. Once in the Python interpreter, the script can access CTH data via classes and methods that are loaded from a module. Methods to retrieve saved variable names and values are provided. One of the most important of these methods generates a vtk rectilinear grid from a CTH block (AMR or flat). By importing vtk functionality, the script can then use the full power of the vtk system to perform myriad visualization functions on the CTH data. Since the computer-intensive portion of this processing is accomplished in the underlying "C" or "C++" code, Python actually adds very little computational overhead while providing enormous flexibility.

Figure 1 shows ParaView being used to visualize armor penetration data from CTH. The spyplt data were post-processed with a Python script that runs as a parallel message-passing interface program to generate isosurfaces and write them as XDMF data sets.



Iso00100.vtp
Calc0

Name: XMLPolyDataReader0
Class: vtkXMLPolyDataReader
Label: Iso00100.vtp

Filename /var/tmp/Clarke/Bert3/M=
Timestep 0 125 100

Point Arrays
- DENS Interpolated
- P Interpolated
- VX Interpolated
- VY Interpolated
- VZ Interpolated
- VMAG Interpolated
- DMG2 Interpolated
- Scalars

Cell Arrays
- DMG2
- VOLM1
- VOLM2
- DENS
- P
- VX
- VY
- VZ
- VMAG
- Material Number

Figure 1. CTH data in ParaView.

With careful coding, the Python script itself can be used as input to IceSpy. The Spymaster interpreter looks for the word "spy" and "endspy" in the input, ignoring all other lines. Usually, these other lines are CTH input (i.e., the Spymaster input is embedded in the CTH input file), but we can use this to provide input to the embedded interpreter. One of the simplest ways to demonstrate this is by example:

```
#!/usr/bin/env python

# Hide the SpyPlt Input from Python
SpyInput = """
spy

% Call every usec in simulation time
PlotTime(0, 1.0e-6);
% Parse this file
XdmfPythonExecFile(XdmfGetCommandInputFileName());

define Ice()
{
% Call the Execute() method of the object with the cycle number and time
XdmfPythonExec("IceSpy.Execute(" + string(CYCLE) + "," + string(TIME) + ")");
}

define main()
{
pprintf(" in Spy PLOT: Cycle=%d, Time=%e\n",CYCLE,TIME);
Ice();
}

endspy
"""

# It's all Python from here
from Cth.IceSpy import *

class _IceSpy :
    def __init__(self) :
            self.Spy = IceSpyAbstract()

    def Execute(self, cycle, time):
            print "IceSpy Called at cycle=", cycle, " time = ", time

# Create an instance
IceSpy = _IceSpy()
```

This file is first parsed by the Spymaster interpreter which begins parsing when it encounters "spy" (highlighted in red) and stops when it encounters "endspy". This makes a call to the XdmfPythonExecFile() command to re-parse the file in the Python interpreter. The Python interpreter treats the Spymaster input as just another string of data assigned to the variable "SpyInput". For every microsecond of simulation time, the Spymaster interpreter will call the

Execute() method (highlighted in blue) of the IceSpy object which then prints to the standard output of the process.

4. Using the Tools for Visualization

With this concept, several useful Python scripts have been developed that allow users to post-process CTH data in parallel. These scripts have been used to post-process large CTH calculations and visualize them with ParaView via image compositing and with EnSight in the "server of servers" mode.

The scripts are presented in the /share/CTH directory of the ICE distribution. The following is a short description of each of the scripts.

> AllMaterials.in - uses the internal Spy isosurface generator to produce material volume fraction isosurfaces and allows for the inclusion of one scalar value. It generates one .h5 and one .xmf file for each time step.
>
> LargeIsosurface.py – generates material volume fraction isosurfaces in parallel while including all the original scalar values. This method is recommended when one is using fewer processors than the number used in the original calculation. It generates one .h5 file and one .xmf file for every processor used to post process and an additional .xmf file to tie the individual "per processor" files together for each time step.
>
> ParallelIsosurface.py - generates material volume fraction isosurfaces in parallel while including all the original scalar values. In this case, all the individual processors write their data back through node 0 to generate one .xmf and one .h5 file per material per time step.
>
> WriteGrid.py – generates one unstructured data set for all the data in an AMR or flat mesh. This does not create isosurfaces but an entire volume containing all the cell and scalar data of the original calculation.

As discussed earlier, IceSpy is the executable that links Spymaster commands and the commands to generate XDMF output. There are currently two versions of the IceSpy command, IceSpyInterim03 and IceSpy7, each corresponding with the version of CTH used to generate the spcth.x files. To use the scripts with a particular data set, very little needs to be edited. For example, to use AllMaterials.in, the following lines may require editing:

> Iteration = 0; "Sets the beginning for the output files. This is changed only to start numbering with something other than 0. This is useful when one is processing

calculations that have been restarted and have multiple groups of spcth.x files, i.e., spcth, spcth_a, etc.

PlotTime(0, 1.0e-6); Just as in Spy, the first number is the "Start Time" and the second is the time when each subsequent isosurface should be generated.

XdmfIsoAllMaterial(P, MIR_FLAG); Here, "P" is selecting pressure as the single scalar to map to the isosurface. Any other available scalar could replace this. For example, "DENS" could replace "P" to map density instead of pressure.

To use LargeIsosurface.py or ParallelIsosurface, the user may only need to edit the PlotTime, as described before. These two scripts also provide pre-set execution flags that can be edited as needed. Listed below is that portion of the file:

```
##################################
## Execution Flags
##################################
        Self.WriteVtkXml = 0  # Write vtk XML Polygons instead of XDMF
        Self.CapIso = 0   # Cap Isosurface if it crosses boundary
        Self.Iteration = 0  # Number Isosurfaces starting with this number
        Self.CompressData = 1  # Compress data with zlib
```

Finally, the command line to run IceSpy at the U.S. Army Research Laboratory's major shared resource center is listed. Installation at other sites may be in a different location and involve queuing systems, etc.

```
/usr/cta/unsupported/ICE/ice5 IceSpy7 i=AllMaterials.in  mv=spcth
```

or

```
/usr/cta/unsupported/ICE/ice5 IceSpy=7 i=LargeIsosurface.py  mv=spcth
```

5. What's Next

ParaView recently added a native spyplt reader, and EnSight is developing one as well. Generating isosurfaces directly from CTH data, however, is a time-consuming process and thus well suited for batch processing, so the parallel Python scripts to generate surfaces in batch will continue to be used. Visualization aside, the ability to access CTH AMR data in a scripting language is particularly useful for developing coupling of CTH calculations with finite element structural mechanics codes; data can be easily manipulated and formatted in the scripting language and passed to the finite element code.

Finally, developing custom post-processing applications will greatly benefit from this method. One application currently in development attempts to quantify the behind-armor debris field. In this application, the CTH data are analyzed to produce a table of mass and velocity for each piece of debris created during armor penetration. This table is then used as input to a survivability code to determine the impact on functionality of various vehicles. Such applications would be virtually impossible to develop from scratch without a flexible method for accessing the results of CTH calculations.

6. References

1. Beazley, David M.; Lomdahl Peter S. Feeding a large-scale physics application to Python. *Proceedings of the 6th International Python Conference*, pages 21–29, San Jose, CA, October 1997.
2. Cummings, J.; Aivazis, M.; Samtaney, R.; Radovitzky, R.; Mauch, S.; Meiron, D. A Virtual Test Facility for the Simulation of Dynamic Response in Materials. *The journal of Supercomputing* **August 2002**, *23* (1), 39–50(12)
3. Littlefield, David L. A Brief Description of New Algorithms Incorporated into CTH: A model for Rigid Obstacles and Interface for Coupling with Structural Codes. Texas Institute for Computational and Applied Mathematics, The University of Texas at Austin, Technical report November, 2001
4. Clarke, Jerry A.; Namburu, Raju R. A distributed computing environment for interdisciplinary applications. *Concurrency and Computation: Practice and Experience* **November-December 2002**, *14* (13–15), 1161–1174
5. Clarke, Jerry Emulating Shared Memory to Simplify Distributed memory Programming. *IEEE Computational Science and Engineering* **1997**, *4* (1), 55–62
6. Web site: http://www.python.org, valid as of June 11, 2005.

NO. OF COPIES	ORGANIZATION
*	ADMINISTRATOR DEFENSE TECHNICAL INFO CTR ATTN DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218 *pdf file only
1	DIRECTOR US ARMY RSCH LABORATORY ATTN IMNE ALC IMS MAIL & REC MGMT 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL CI OK TL TECH LIB 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	AERONAUTICAL SYSTEMS CTR ATTN RHONDA VICKERY HIGH PERFORMANCE COMPUTING BR ASC/HP BLDG 676 2435 FIFTH ST WRIGHT PATTERSON AFB OH 45433-7802
2	RAYTHEON SYSTEMS ATTN M A BOLSTAD J C RENTERIA 939 I BEARDS HILL RD PMB 191 ABERDEEN MD 21001
	<u>ABERDEEN PROVING GROUND</u>
1	DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL CI OK (TECH LIB) BLDG 4600
1	DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL CI C NIETIUBICZ BLDG 328
3	DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL CI HC J CLARKE R NAMBURU R ANGELINI BLDG 394