LLNL-TR-826004

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Implementation of Plot File Testing in the DYNA3D/ParaDyn Software Quality Assurance Suite

E. Zywicz, R. Hathaway

August 23, 2021

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Implementation of Plot File Testing in the DYNA3D/ParaDyn Software Quality Assurance Suite

Edward Zywicz and Ryan Hathaway
Methods Development Group
Defense Technologies Engineering Division

March 8, 2021

1 Summary

Automated testing of DYNA3D/ParaDyn plot files was added to the DYNA3D/ParaDyn software quality assurance (SQA) test suite. The new capability extracts select data from the plot files generated during each verification run and compares it to the same baseline answers used to verify the problem. Deviations between baseline answers and plot file values are reported in the same manner as solution discrepancies, and differences in precision levels between the baseline answers and plot file results are accounted for. The new testing leverages the existing SQA test suite framework and test problems and the Python Mili reader and minimally increases the overall run time (< 5%) of the SQA test suite. This new capability provides *incremental* end-to-end testing of the most common DYNA3D/ParaDyn simulation workflows.

2 Background

The typical DYNA3D (Zywicz *et al*., 2020) analysis workflows involve three main steps performed sequentially: 1) mesh generation, 2) simulation and 3) post processing, i.e., result extraction and visualization. ParaDyn (DeGroot *et al*., 2020) workflows includes two additional steps - a partitioning step performed prior to simulation and, depending upon how results are post processed, a plot file combining step performed after the simulation. Each step involves one or more software packages and accompanying libraries. While it is highly desirable to verify the entire workflow as a single process, continuous software development by different teams makes it extremely challenging; the primary difficultly is determining causality and accountability when final results change. As an alternative, each tool involved in the workflow independently verifies its functionality and output with its own SQA test suite. Since the output from mesh generation, a DYNA3D input deck, is the input to simulation and the output from simulation, a plot file database, is the input for post processing, end-to-end testing of the entire analysis workflow is verified in an *incremental* manner.

DynaPart, a component of the ParaDyn suite, is used to partition ParaDyn problems. It uses DYNA3D to parse the input deck, performs a series of optimization calculations and then generates a partition file that ParaDyn reads in along with the input deck. While the problem partition impacts ParaDyn's computational performance, it does not impact the numerical results, beyond numerical roundoff, in well posed problems provided that all elements and nodes are properly included in the overall partition.

Three primary post-processing tools are used in the typical DYNA3D/ParaDyn analysis workflows at LLNL: 1) Griz (Speck, 2001), 2) VisIt (Whitlock, 2006) and 3) the Python Mili Reader (Leglar and Durrenberger, 2020). While DYNA3D generates a single plot-file database, ParaDyn generates one plot-file database per processor. VisIt and the Python Mili Reader can read one or more plot-file databases and render them concurrently as if they were a single database, but Griz can only visualize a single plot-file database. The tool XMiliCS, which is part of the Mili library (Durrenberger *et al*., 2017), is used to combine a parallel plot-file database into a single plot-file database. Consequently, XMiliCS may reside in the post-processing workflow. Unlike the other software, DynaPart employs various manually initiated verification testing and lacks its own automated verification test suite.

Historically, DYNA3D/ParaDyn developers inferred the accuracy of plot file databases by manually confirming the simulation results visualized correctly in the post-processing tools. Consequently, incremental end-to-end testing did not truly exist. The inclusion of automated plot-file testing rectifies this deficiency.

3 SQA Testing

DYNA3D and ParaDyn optionally generate a binary time-history database and output numerous text files that contain specific results. The former contains a limited set of mechanics variables typically written more frequently than in the regular plot file database. Alternatively, the time-history data can be included in the high-speed printer (HSP) file as text. The DYNA3D/ParaDyn SQA test suite directs time-history data to the HSP file and extracts data from the HSP file and other ASCII files to perform its verifications.

The DYNA3D/ParaDyn SQA test suite consists currently of 691 individual problems divided into roughly twenty-six functional categories. Each problem has an input deck, a partition file for eight processors, one or more Awk scripts to extract answers from the HSP file and other ASCII output files and create an answer file, and two baseline answer files – one contains results from a run with eight-processors and the other from a serial run. For each problem, one to several hundred mechanics quantities are extracted from the various HSP and text files at different times in the analysis, typically ten to twenty times, and written to a temporary answer file. The mechanics quantities include nodal positions and velocities, element stresses, contact forces, nodal reaction forces, etc., and vary from problem to problem. The main test script compares the current results with the appropriate baseline values to sixteen digits and flags discrepancies. The script summarizes by suite the number of problems that pass and reports any differences that arise. The suite is run twice – once using serial DYNA3D and once using ParaDyn with eight processors.

The new plot file testing leverages the existing SQA framework and required minimal modifications to incorporate. Each test-problem answer file contains a character string that describes the associated mechanics quantity and is printed out when differences arise. To facilitate plot file testing, a second string is added that contains the Mili short-name and additional descriptors to uniquely specify the mechanics quantity in the plot file. (The amended test-problem answer file syntax is summarized in Appendix A.) An additional Python script decodes the second string, extracts the values from the plot file via the Python Mili reader, calculates select derived values from the extracted quantities, e.g., the pressure from the stress components and the displacement magnitude from the positions, and stores them in an analogous answer file. The main script compares the values in the current plot-file answer file and the baseline file to seven significant digits (plot file results are written as single precision values), and records its findings in a log file. When the mechanics quantity is known not to exist in the plot file, the second string is left blank in the answer file and the script gracefully deals with it. Plot file testing is only performed if the problem passes its verification test, and parallel databases are read without being combined into a single database.

The plot-file test script optionally identifies variables in the answer files whose values are not verified during plot file testing. Currently, 156 of the 691 test problems contain one or more such quantities, and the slide surface suite accounts for 140 of these problems. The unchecked slide surface quantities are sums of nodal reaction forces originally included to verify the problem setup; unfortunately, these quantities cannot be included in the plot files in a general manner. Of the remaining sixteen tests with unchecked plot file values, four tests do not create plot files, four tests intentionally don't check select variables due to precision issues arising from single precision data, and the remaining eight problems involve seldom used features whose output has never been included in the plot file database.

The inclusion of plot file testing in the DYNA3D/ParaDyn SQA suite revealed one defect. The number of integration points included for material model 28 for variable integration point output was not written to the plot file. As neither Griz nor VisIt currently support variable integration point rendering, it did not impact users.

4 Potential Future Work

While the inclusion of plot file verification has addressed a long-standing gap in *incremental* end-to-end testing of DYNA3D and ParaDyn, several areas for potential improvement remain. Foremost, an automated SQA test suite needs to be developed for DynaPart and utilized routinely. Although DynaPart development utilizes most of the DYNA3D/ParaDyn SQA test suite problems for its SQA, the process is not yet fully automated. Checks should be included in ParaDyn to ensure all nodes and elements are accounted for by the partition file*. Additional variables could be added to the plot files and checked in the test suite. Finally, an assessment, similar to a code coverage analysis for a SQA suite, that determines the percent of different plot file variables examined in the plot file testing would be insightful.

*- This functionality was added to ParaDyn in June 2021.

References

DeGroot, A.J., Durrenberger, J.K., Giffin, B., Pham, E., Zywicz, E., "ParaDyn: A Parallel Nonlinear Explicit, Three-Dimensional Finite-Element Code for Solid and Structural Mechanics User Manual – Version 20.1", Lawrence Livermore National Laboratory, Report LLNL-SM-816810, 2020.

Durrenberger, J.K., Corey, I., Pierce, E., Speck, D., "Mili I/O Library Programmer's Reference Manual", Lawrence Livermore National Laboratory, Report UCRL-SM-743317, 2017.

Legler, P.M., Durrenberger, J.K., "Mili Python Interface: Users Guide V0.1", Lawrence Livermore National Laboratory, Report LLNL-SM-811236, 2020.

Speck, D.E., "Griz Finite Element Analysis Results Visualization for Unstructured Grid User Manual", Lawrence Livermore National Laboratory, Report UCRL-MA-115696-REV-2, 2001.

Whitlock, B.J., "VisIt User's Manual", Lawrence Livermore National Laboratory, Report UCRL-SM-220449, Revision b7c2a67f, 2006.

Zywicz, E., Giffin, B., DeGroot, A.J., Durrenberger, J.K., "DYNA3D: A Nonlinear, Explicit, Three-Dimensional Finite-Element Code for Solid and Structural Mechanics User Manual – Version 20", Lawrence Livermore National Laboratory, Report LLNL-SM-816811, 2020.

Appendix A - <u>Answer File Syntax</u>

The DYNA3D/ParaDyn SQA baseline answer files are written with a standard format to facilitate automated processing. Each file contains one or more result blocks separated by one or more blank lines. One block is used for each mechanics result. Each block starts with a title line whose format is "# *string1 string2*" and is followed by *N* data lines ordered in increasing analysis time. A data line has the format "*time value*" or "*time value # comment*". Here *time* denotes the analysis time at which the result is sampled, and *value* is the associated numerical value at that time. The second form accommodates an optional inline comment, used to supplement *string1*, that is printed out when discrepancies arise. Blocks can contain an arbitrary number of comment lines, i.e., a line whose first character is "#", before and after the primary section. Within an answer file, each block must have *N* data lines and use the same times (*time*). If the analysis has both a dynamic relaxation phase and a transient phase, it is permissible for the time value to jump backwards in time at the end of the dynamic relaxation phase.

In the title line *string1* provides a succinct description of the variable, e.g., node_12_x-velocity. Space are not allowed in either *string1* nor *string2*. *String2* defines the variable in a manner that allows the script to extract it from the plot file. It has three general forms:

1. *classname_variable*,
2. *classname_label_variable*
3. *classname_label_ipt_point_variable*

Here *classname* is the Mili class name that DYNA3D associates with a specific model entity when it writes the plot files, e.g., brick, node, shell, tshell, global, etc., *label* and *point* are integer numbers, and *variable* is the Mili short name that DYNA3D assigns that quantity, e.g., sx (x-stress), xv (x-velocity), eps (equivalent plastic strain), etc., or the name of a supported derived quantity such as seff (equivalent stress) or dispx (x-displacement of a node). The first form is used to select global and other similar data that requires no additional specifiers to uniquely identify it. The second form is used for bricks, nodes, load curves, etc. where *label* is the desired mesh entity number. The third form is for elements with data outputted at multiple integration points. Here *point* is the integration point number desired. The following examples show valid *string2* descriptors:

- brick_571_seff
- node_667_dispx
- node_1080_vy
- shell_22_ipt_5_sz
- beam_45_ipt_1_sx
- lcurve_3_lcfunc
- global_ke

The appropriate Mili short names can be obtained from the DYNA3D source or by looking at the plot file with Griz and examining the “primal” tab and its sub-tabs. The currently supported derived quantities in the plot file testing script are:

- For Nodes:
    - dispx, dispy, dispz, dispmag (x-, y-, and z-displacement)
    - dispmag (displacement magnitude)
- For Materials:
    - matcg-dispx, matcg-dispy, matcg-dispz (x-, y-, and z-displacement of the CG)
- For Elements:
    - seff (equivalent stress)
    - press (pressure)

Additional quantities will be added as needed.