New Developments in the Compression of LS-DYNA Simulation Results using FEMZIP

Authors:

Rodrigo Iza Teran, Clemens-August Thole, Rudolph Lorentz
Fraunhofer Institute for Algorithms and Scientific Computing
SCAI

Correspondence:

Rodrigo Iza Teran
Fraunhofer Institute for Algorithms and Scientific Computing
Phone +49 2241 142712
Fax +49 2241 142181
Email iza-teran@scai.fraunhofer.de

ABSTRACT:

The standard usage of simulation as part of the automotive design process and increased computer power has increased the demand on archiving resources. This data may be reused several times during the development process and reused later for new models. Compression of the simulation results reduces both the archive size and the access times. This is why FEMZIP was developed several years ago. FEMZIP was specially designed for the compression of crash simulation results. We report on the improvements of FEMZIP both in regard to speed and to compression factors. FEMZIP is now also available built into several postprocessors. This means that these postprocessors can read and process compressed files directly. As a consequence of the improvements in FEMZIP, the postprocessors can display the compressed files faster than uncompressed files.

Keywords:

Data compression, Process Optimization, Post processing d3plot files

INTRODUCTION

This paper presents improvements that are now part of the new Femzip generation, a tool that has been used for compressing d3plot crash simulation results. The tool allows a significant compression factor while maintaining the desired precision of the results. Compression has many advantages if used in crash simulation. Many companies have adopted Femzip in their simulation process because of the reduction of archive size and access times. Since our last report [1], new compression methods have been developed that allow a better compression and a faster decompression. A review of the state of the art in this area is presented. An example compressed model is analyzed. Some details are also explained regarding as to how the access to specific parts of a d3plot file has been realized through the decompression inside a post processor. Consequently, visualizing a compressed file directly, i.e., without decompressing it first is faster than visualizing the uncompressed file. Reasons for the improved performance are given. This improved post processing has been observed for the Animator post processing tool into which the decompression tool is fully integrated.

A NEW COMPRESSION METHOD FOR SIMULATION RESULTS

The field of compression of floating point data is, as of the moment, underdeveloped. To our knowledge, there is no commercially available software for this general purpose. If we restrict our consideration to the compression of formatted floating point data, then we know of only two such software packages: FEMzip for LSDyna/Pamcrash data and GRIBzip for the compression of meteorological data in the GRIB format.

On the other hand, the subject itself has been tackled in several papers. Engelson, Fritzson and Fritzson [2], Isenburg, Lindstrom and Snoeyink [3], and Ratanaworabhan, Ke and Burtscher [4] losslessly compress floating point data by treating the IEEE representation of floating point data as integers. Cabrera compresses 3D meteorological data by first applying the Karhunen-Loeve transform and then compressing the 2D-slices with JPEG2000. Finally, Steffen, Wang and Brummer [5,6] have a series of papers on the lossless compression of GRIB data utilising lossy JPEG compression and a difference file.

The algorithms developed by Steffen and Wang show that compression methods can also be successfully applied to simulation results. They cannot be used directly for crash simulation because wavelet transformations cannot be applied to grid functions on unstructured shell element grids. The Fraunhofer FEMzip tool therefore combines a number of new algorithms to compress simulation results[1]. FEMzip implements the three basic steps:

Quantization

Approximation

Coding of the residual

*Quantization***.** Quantization means the representation of the data elements as multiples of a basic increment. Quantization therefore reduces the actual precision of the results. For the quantization, FEMzip requires a basic increment for each of the grid functions contained in the simulation results. Therefore, the simulation results are compressed with some loss in precision. The quantization is specified by the user of FEMzip either as absolute values or in relation to the current size of the grid functions.

Approximation of simulation data. FEMzip combines various methods to approximate the grid functions. The first principle is that the difference between simulation results at adjacent time steps is dominated by rigid body modes. Furthermore, the current velocities can be used to approximate the values at the new time steps. In addition, FEMzip implements a hierarchical interpolation based on AMG coarsening [7].

Encoding of the residual. As a result of the quantization and the approximation, the difference between the results of the quantization and the approximation is of type integer and small. Now one has two options. The first is faster but does not compress as well as the second. It uses the method of "suppression of leading zeros" with adaptive word length, detection of sequences of identical values and special treatment of outliers is used to compress this difference. The second option yields better compression factors but is not as fast. In the following discussion, the second option was used for comparison due to its better compression rate.

[1] The general approach as well as the detailed algorithms is subject of the national patent 103 31 431of Fraunhofer Gesellschaft, Munich. Acceptance of the patent on an international level is pending.

Results of the Compression

Tables 1 and 2 show compression factors and reading time in Animator for a Daimler Chrysler SLK Model with 1006876 nodes, 23716 bars, 1079003 shells, 64776 solids, 76 time steps. Time is measured in seconds. All measurements were done on a 2.13 Ghz PC with a SATA hard disk. Version 1 corresponds to Femzip Release 1.1* and Version 2 corresponds to Release 4.*

Table 1: Compression factor

	Compression Factor
Version 1	8.75
Version 2	13.3

EFFICIENT DECOMPRESSION ON POSTPROCESSORS

For post processing a large model with a visualization tool in a network file system, large amounts of floating point data need to be read and transferred rapidly. On the one hand, our decompression algorithm processes around 40 Mbytes per second on a 3 GHz PC. On the other hand, sending a compressed file over the network saves transfer time. Taking into account the throughput of fast Ethernet networks and of the hard disk, it turns out that even if decompression is necessary, the overall runtime is reduced compared to the one that is needed if the original simulation file is kept uncompressed. If a file is saved locally on the hard disk, the new version has achieved an improvement with respect to the old one, due to the fact the new structure of the compressed data format and the functionality of the API has been improved.

The files compressed with the original version of FEMzip were read substantially slower by Animator than uncompressed files. In Table 2, one can see that this relationship is now just the opposite. Compressed files are read substantially faster than uncompressed files. If the data comes via a remote file transfer, then savings in time are even larger.

Table 2: Reading times in Animator

	Animator reading time [s]
Uncompressed	127
Version 1	-
Version 2	67

The fact that the time to read and visualize remotely a compressed file is less than if an uncompressed file is used has also been observed by our customers in the German automotive industry and is an additional benefit of the compression. If we just decompress a compressed file before reading it into a postprocessor, we do not obtain this performance. Femzip provides a free API interface that decompresses selected parts of the d3plot file to companies with post processors. The information in the compressed file is organized in such a way that selective decompression is possible as explained in the scheme in Figure 1

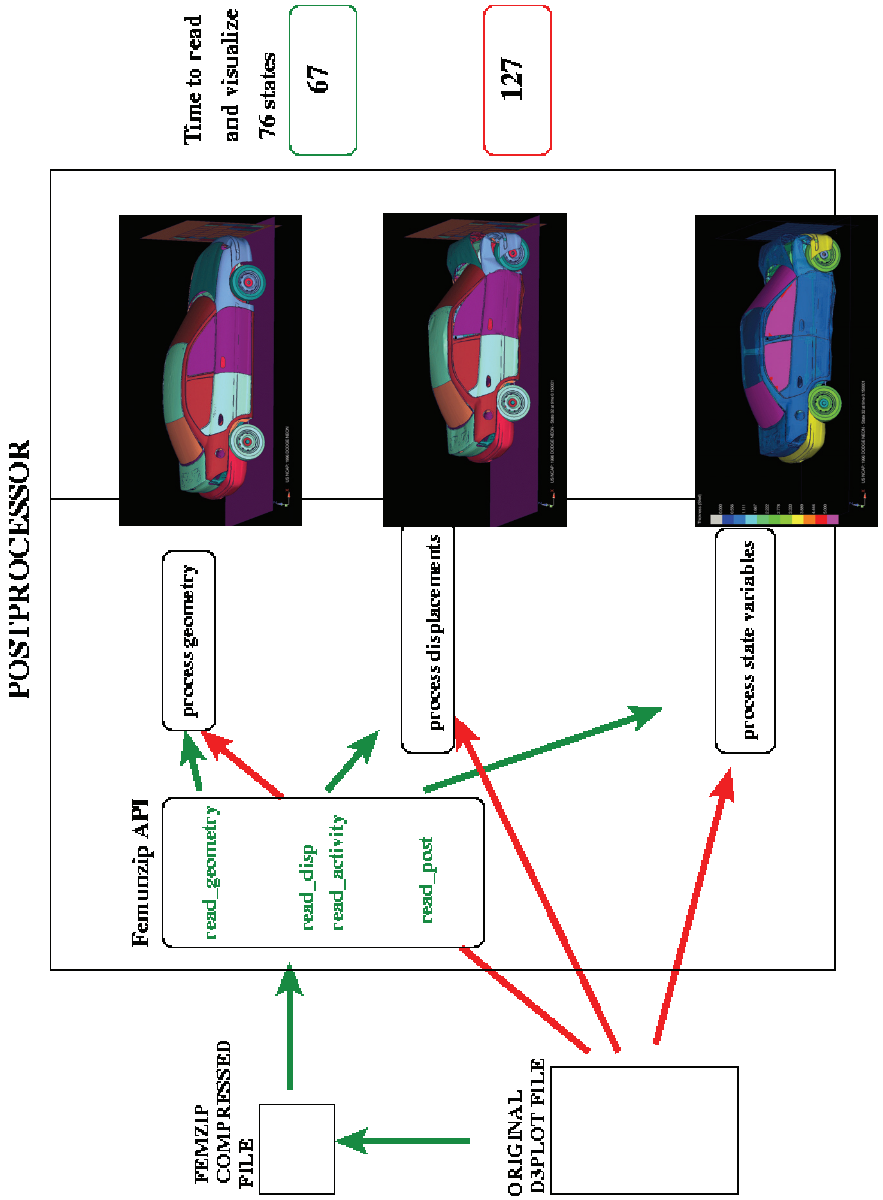

Figure 1: Decompression in a Postprocessor

The post processor reads the compressed file, identifies it as a femzip compressed one and, as a consequence, uses not the normal reading routine but our decompression routines within the API. For the example in Figure 1, the Femunzip routine *read_geometry* will decompress the header and the geometry information of the d3plot file including connectivity, materials, etc. This information is directly transferred to the post processor, to the routines that actually process and visualize the geometry of the model (*process geometry* in the example). Similarly if the user now wants to read the displacements, then two functions of the API are used for each state, namely *read_disp* to decompress the nodal information and *activity_read* to decompress the information regarding failed elements. The information from these routines is then analogously transferred to the post processor function(s) that actually processes and visualizes the displacements (process displacements in the example).

The integration of Femunzip in a post processor has been shown to decrease the time that is needed to load several states. The time to load the displacements and one variable for the Daimler SLK Model has been reduced from 127 seconds to 67 seconds on a 2.13 GHz PC.

SUMMARY AND CONCLUSIONS

In this paper, we have highlighted the essentials of the compression method used for reducing the size of d3plot files. The benefits are observed in several ways. First in the reduction of the archive sizes and the data transfer times. An additional benefit of the compression has been explained for the situation in which the compressed files are used inside a post processing tool directly through an API interface. Significant gains in reading and processing times are observed even if the data first has to be decompressed within the post processor.

REFERENCES

[1] Thole, C. A.: Compression of LS-DYNA^{TM} simulation results using FEMzip. In Proceedings of the 3rd LS-DYNA Anwenderforum, Bamberg 2004. http://www.dynamore.de/download/af04/papers/E-III-4.pdf.
[2] Thole, C. A.: Compression of LS-DYNA^{TM} simulation results using FEMzip. In Proceedings of the 3rd LS-DYNA Anwenderforum, Bamberg 2004.

[2] Engelson, V., Fritzon, D. and Fritzon, P.: Lossless compression of high-volume numerical data from simulations, Data Compression Conference, 2000, 574 – 586.
[3] Isenburg, M., Lindstrom, P. and Snoeyink, J.: Compression of floating point geometry, CAD2004, 495 – 502.
[4] Ratanaworabhan, P., Ke, J. and Burtscher, M.: Fast lossless compression of scientific floating point data, 2006 Data Compression Conference.
[5] Steffen, C. E., Wang, N.: Weather data Compression, Proceedings of the 19th International Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology, American Meteorology Association, 2003.
[6] Wang, N. and Brummer, R.: Experiment of a wavelet-based data compression technique with precision control, Proceedings of the 19th International Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology, American Meteorology Association, 2003.
[7] Ruge, J.W. and Stüben, K.: Algebraic multigrid. In McCormick, S.F., editor, *SIAM Frontiers on Applied Mathematics, Volume 3, Multigrid Methods*, pages 73-130. SIAM 1987.