COMPRESSION OF NVH SIMULATION RESULTS

Josua Lidzba[1], Matthias Rettenmeier[2], Dennis Hahn[2], Clemens-August Thole[2], Rodrigo Iza-Teran[2]

THEME

Simulation Data Management

KEYWORDS

Data compression, NASTRAN, NVH, OP2 file

SUMMARY

Crash specific data compression has become a standard technology, which is widely used in automotive industry. This paper describes lossy data compression of NASTRAN-OP2 files as they emerge from NVH simulations in the automotive sector. The largest part of these files contains real or complex eigenmodes but also element energies and stress or strain tables. During the automotive design process a large number of simulations is performed. As a result huge amounts of data have to be stored. Using data compression the size of the files can be significantly reduced with less storage space required. The reduction in file size resulting from the compression also leads consequently to faster file transfer and I/O-times of post processing tools, which are able to read compressed files via a specific library. Therefore, efficient compression of OP2-files means a large benefit for NASTRAN users. To achieve very high compression ratios a lossy data compression scheme was chosen. During compression floating point data is quantized using a user controlled precision. This allows the user to meet his requirements as a tradeoff between the magnitude of errors and the compression factor. To achieve the best compression ratio at a given precision the entropy of the data is reduced further by using specially designed prediction algorithms. The technologies involved exploit the shell and solid element connectivity information derived from the OP2 files. To accomplish seamless workflow integration the compressed data

1 University of Cologne, Germany
2 Fraunhofer Institute for Algorithms and Scientific Computing (SCAI) , Germany

must be fast and easily accessible. Therefore the HDF5 format was chosen as output file format. The advantages of the HDF5 file format will be exploited during decompression. Easy and fast data access facilitates the parallelization of the decompression using OpenMP. This is especially interesting for post processing tools, which directly read OP2 data from the compressed files.

The software FEMZIP-N which makes use of all these features showed very satisfying compression results on NVH simulations conducted by different automotive companies. Compression ratios between 5.0 and 40.0 have been achieved.

1: INTRODUCTION

Today in almost every area of research large amounts of data are produced. In many cases it must be stored for long periods of time. Even though storage space is getting cheaper everyday, there are some good reasons for reducing the data instead of increasing storage space. One major problem with huge amounts of data or – even worse – huge files is the data or file transfer from one machine to another e.g. via local networks or the internet. Another problem, often depending on disk speed or transfer rates, is loading huge files into programs.

All of this applies also to the situation within the simulation sector, especially in the automotive industry - no matter which discipline we look at: crash-, CFD- or NVH-simulations. Each of these disciplines produces a great amount of large files that often have to be re-used several times or must be archived (e.g. for postprocessing purposes). To avoid these problems an obvious approach is to reduce file sizes using data compression. Usually general purpose compression tools do not lead to satisfying results. In most cases they only reduce file sizes by less then 50 percent. Therefore a much better approach is the use of specialized tools that are adapted to different file formats and contents.

At the Frauhofer Institut for Algorithms and Scientific Computing (SCAI) the software FEMZIP was specially designed as a lossy compression tool to compress crash simulation results. It allows a significant reduction in file size while maintaining the desired precision of the results. The compression of NASTRAN OP2 output files from NVH simulations[3] was recently chosen as the first additional member of the FEMZIP tool box.

From a compression point of view the main difference between crash and NVH simulation results is that crash simulations are time-dependent and many NVH

3 The FEMZIP version to compress NASTRAN OP2 files is called FEMZIP-N.

simulations (such as eigenvalue analysis, frequency response) are static. This results in different requirements for the compression methods. The new methods involved and some new features to distinguish FEMZIP for crash data from FEMZIP for NVH data are described in this paper. Further it will illustrate the advantages of using HDF5 (Hierarchical Data Format) as output format for compressed data, the multi processor architecture of the decompression, the Rice encoder and the integrated security option. First we give a short overview of data compression in general and lossy compression of simulation results in particular.

2: DATA COMPRESSION AT A GLANCE

Compression involves changing the form of data in a file, so that the compressed file takes up less space than the original file. There are two types of compression: lossless and lossy.

If lossless compression is used, the original file can be retrieved exactly from the compressed file. Lossless compression is typical for text files, but also for files containing sensitive numerical data, e.g. medical or meteorological data. All Zip utilities perform lossless compression, with compression factors of around 1.5 to 3.

On the other hand, if the compression is lossy, one cannot retrieve the original file from the compressed file. The advantage of this approach is that the data can be compressed to a much greater degree. Lossy compression is typically used to compress graphic and video files, especially in the context of the Internet. Typical lossy compression programs are JPEG and MPEG, which can reduce the size of a file by factors of ten to fifty [1].

The main principle behind lossy compression methods is to interpret input data in its original meaning (i.e. as physical quantity or as pixel in a picture). This kind of information is usually stored as binary floating point data, a fact which allows a rounding process while compressing data. This method is called quantization. The quantization causes a loss of information that cannot be retrieved during decompression. Therefore quantization should be carried out carefully. If executed correctly the user will not even recognize any difference between the original and the uncompressed file. Good examples for this technique can be found in audio-, video- or 3D-graphics-compression. In this area many (commercial) applications exist.

In our approach we concentrate on lossy compression schemes using the procedures described before. Whatever kind of simulation discipline the files emerge from, the main part of every file consists of floating point data. However, a great fraction of each floating point number appears to be noise.

Because the output files are normally used for visualization only, this part of each number can be truncated without loosing user-visible information. This is carried out by the quantization that maps the floating point numbers into the integer domain. To avoid any unwanted effects the user is in control of the rounding precision for each floating point quantity. Quantization is the first of three steps used in a wide range of lossy compression algorithms. The other two steps are approximation and coding of the residual.

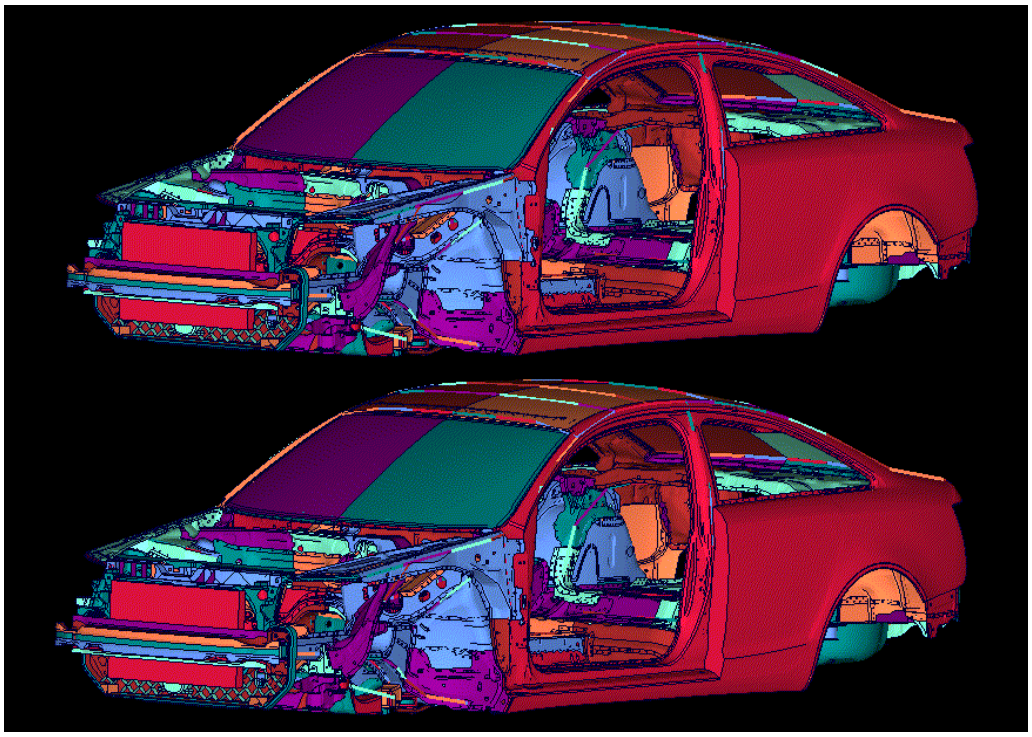

Figure 1: Visualization of an original (at the top) and a decompressed (at the bottom) OP2 file (property of AUDI AG as provided for the European SimDAT[4] project).

The approximation contributes most to the quality of the compression algo-rithm. The main principle is to reduce entropy by prediction methods: For every value a prediction is made and instead of storing the original number, the difference between original and predicted number is stored. This operation serves as pre-conditioning for subsequent coding. If the prediction method has worked correctly, we expect lower entropy within remaining data. That leads to better compression results using standard entropy encoding approaches, such as Huffman coding or arithmetic encoding.

4 www.simdat.org

As mentioned before, the prediction method used (together with quantization precision) is decisive for compression quality. Therefore one of the biggest challenges designing new compression methods is to find appropriate predictors. In FEMZIP for crash other prediction methods are used than in FEMZIP-N as crash simulations are time dependent and hence powerful prediction schemes can easily be adopted. The following paragraph describes the approach we have chosen for FEMZIP-N and some new developments which were used here.

3: DATA COMPRESSION OF NASTRAN NVH DATA

First, we want to point out, that FEMZIP-N was not developed to compress OP2 files in general but to compress OP2 files emerging from NVH simulations, which foremost means eigenvalue analysis and frequency response simulations as they result from NASTRAN solutions 103, 108 and 111 (see [2] for more information). However, FEMZIP-N should be able to deal with every other NASTRAN simulation result, whereas best compression results will be reached in upper cases. In the near future we plan to improve the efficiency of compressing other solution types. In the following we will briefly describe the way prediction was handled in FEMZIP-N and the special features established in this release.

3.1: Prediction scheme and handling

In FEMZIP-N some different prediction schemes are used. The most important improvement is the way we predict the values of vertex based variables such as coordinates or displacements. The scheme exploits mesh connectivity information without maintaining complicated data management structures.

Almost every prediction scheme can be interpreted as linear operator. Thus we can specify a matrix, which represents this method. These matrices often are sparse, depending on the predictor they implement. This allows to perform a matrix vector multiplication during compression and another matrix vector multiplication during decompression (to retrieve the prediction step) with sparse matrices involved. During decompression the inverse matrix is used.

The second innovation in FEMZIP-N prediction methods is the possibility to compute these matrices while compressing the file and store the inverse matrix in the compressed file. At the first glance, this seems to be counterproductive because we store data into compressed file, which is not explicitly necessary. By storing prediction methods into compressed file we achieve two things: maximum flexibility and speed. A mechanism, which assigns a matrix to every compressed data set in the compressed file, allows to retrieve prediction through a simple matrix vector multiplication with an arbitrary matrix. I.e. we can now change a method in compression software without changing the decompression tool. More important, it is possible to realize prediction algorithms, which could not be retrieved without additional information. This provides possibilities for very powerful algorithms.

The second advantage is speed: a matrix vector multiplication with a sparse matrix engaged implements prediction twice as fast than a standard algorithm. If many data sets are included, speed could be increased not only for decompression but also for compression. The prediction matrix must be found only once for every similar data set (vertex dependent variables, shell dependent variables etc.).

But how about the loss in compression ratio through storing additional data? As stated before we are dealing with sparse matrices. A standard approach to store those matrices is Compressed Row Storage [3]. We combined this approach with standard entropy encoding methods using Zlib[5]. The compressed matrices occupy only 1-2 % of the compressed file for industrial use cases, i.e. the compression ratio stays nearly the same.

3.2: New features in FEMZIP-N

HDF5 output data format: The Hierarchical Data Format HDF5 is a unique technology suite that makes possible the management of extremely large and complex data collections.[6] The high flexibility and user friendliness was the key reason to use HDF5 in FEMZIP-N. The way HDF5 was constructed allows to map the OP2 file format onto an HDF5 file. This makes simple data access possible. Additional data could easily be added. As illustrated, it proved to be very helpful in storing the compressed matrices used to specify the prediction algorithms and making them easily available during decompression. The special structure of an HDF5 file and the existing API marked integral requirements to realize the second new feature in FEMZIP-N: the ability to decompress files using several cores.

OpenMP parallelization of the decompression: Decompression speed is one of the most important features of every compression tool. We decided to increase decompression speed through an OpenMP parallelization of our code. The special characteristic of data compression tools in general and FEMZIP-N in particular is that the proportion between arithmetic operations and data input/output size is typically small, which means a challenge for parallelization. We succeeded in speeding up the decompression using several cores (see 4: “Numerical experiments”). To achieve this significant speedup the algorithm works on various vector based quantities in parallel, for example

5 Zlib is an open source entropy encoding tool, see www.zlib.net for details.
6 For more information see www.hdfgroup.org.

on the displacements of x-, y- and z-coordinate. Regarding a typical model with about one million vertices each of these vectors has a size of several megabytes. This is enough to make parallelization profitable, even if the number of arithmetic operations for each entry is small.

Rice coding and Zlib integration: During compression, the floating point part of input data will be mapped into integer domain. Afterwards the entropy of all data should be reduced through prediction algorithms. The residuals that emerge from this process are subsequently coded, normally by using an entropy encoding scheme like Huffmann coding or arithmetic coding. Instead of using such a coder, in FEMZIP-N most data is coded by the Rice encoder [4]. We chose this coding scheme because of its advantages in speed and compression ratio in combination with the compression of simulation data [5]. In addition a standard entropy encoder - the Zlib - was used to encode some special parts of the input data like headers, or parts not interpreted[7]. On these parts Zlib obtains compression ratios between 1.5 and 4.

Security Option: The original simulation results in practice are often deleted after they were compressed. To ensure that everything has been executed correctly and no files have been damaged during compression FEMZIP-N provides an option that allows the user to validate the compressed output. If this “Security Option” is activated, the tool automatically decompresses the file temporarily after compression has finished and compares the result to the original model. Only if no error occurs, the original model can be deleted without risk.

4: NUMERICAL EXPERIMENTS

In this paragraph we present the results of some numerical experiments concerning FEMZIP-N’s properties. The database we used for benchmarking purposes was provided by automotive companies. It consists of five models, which originate from the NVH sector. In Table 1 the properties of each of these models are listed. Additionally the NASTRAN solution type and the included out-put data are shown.

We will look at two different main aspects of data compression, first the compression ratio and second the compression and decompression speed. We estimate the strength of compression by means of the compression ratio c_r. It is the ratio between the size of the original file and the size of the compressed

7 Every OP2 file contains data, which will not be covered by the special compression algorithms we developed. We call this data “not interpreted”. Regarding OP2 files from the NVH sector its percentage should be small. Therefore the compression ratio of the whole file does not change significantly, whether this part will be compressed or not, although it can be reasonable to reduce its size using general compression approaches.

file. We will see that in most cases a compression ratio around 10 can be achieved using a reasonable quantization precision. That means we usually achieve a file size reduction of 90 percent.

Name	Size (MB)	#Nodes	Solution	OUGV1	ONRGY1	ONRGY2	OES*/ OSTR*
sim1.op2	3435	1428719	103	yes	yes	yes	no
sim2.op2	353	528132	103	yes	no	no	no
sim3.op2	391	1432541	108	yes	no	no	no
sim4.op2	545	1432541	111	yes	no	no	no
sim5.op2	524	844520	111	yes	no	no	no

Table 1: Properties of the models used for testing

The compression or decompression speed will be specified as throughput. Regarding the compression it evaluates the ratio between the original file size and the absolute time needed to compress the file. Regarding decompression it is the ratio between the size of the uncompressed file and the absolute time measured for decompression. Both values are given in MB/sec. The speed benchmarks were conducted on a SUN Fire X4600 with eight 2.6 GHz dual cores and 64 GB RAM and a SUSE LINUX OS. We want to point out that the speed measurements include I/O-times. In this case it is not without risk because there are several aspects which could lead to unreliable results. Sometimes more time is used to write the decompressed file to disk than to compute it during decompression. This effect is becoming more problematic if multi core benchmarks should be performed. The graph presented in Figure 2 shows directly the effect of our OpenMP parallelization. These benchmarks were conducted without measuring the time needed to write output to disk. On the other hand read-in times often could not be appropriately taken into account because the file has been already cached by the operating system. This leads to wrong results because reading from cache is much faster than reading from disk. We tried to avoid this effect within our measurements.

4.1: Compression ratio

Table 2 and Figure 2 show the results of our benchmarks regarding the compression factors FEMZIP-N achieves. For each of the five models two different quantization precisions were chosen and the results with or without rotations[8] were measured respectively. The quantization error depends on the relative precision p_r which the user chooses for every data object that will be quantized and its absolute maximum M_a over all of its values. The deviation between each original value and its equivalent after decompression is smaller than the chosen precision which computes as $p = (M_a \cdot p_r)/100$. The compression ratios shown in the first and second row of Table 2 were computed with $p_r = 0.01$ for the coordinates (GEOM1 block) and $p_r = 0.1$ for displacements and rotations (OUGV1 blocks). In row three and four of Table 2 we chose $p_r = 0.001$ for the coordinates and $p_r = 0.01$ for displacements and rotations. The results in row two and four correspond to skipped rotations. In both cases we chose $p_r = 0.0001$ for all energies in ONRGY* blocks. Therefore the compression ratios of sim1.op2 do not differ as much as the ratios of the other models. We did not change p_r for ONRGY blocks to a coarser level because the values are distributed very unbalanced over their domain. If a large maximum appears together with many small values, all the small values could be zero after de-quantization if a high precision is used.

Table 2 shows that an average compression ratio of about 10 is achieved using a reasonable quantization precision. We also see that it can be useful to delete information from the file during decompression (as seen here for rotations). However, this option should be used very carefully because this data cannot be reconstructed during decompression. If we allow a visible loss of information much higher compression ratios can be achieved.

Name	Low prec	Low prec/rot skipped	High prec	High prec/rot skipped	Comp. (MB/s)	Decomp Proc. 1 (MB/s)	Decomp Proc. 2 (MB/s)	Decomp Proc. 4 (MB/s)
sim1.op2	10.41	10.45	9.05	9.28	38.6	42.4	49.8	70.1
sim2.op2	22.20	25.45	12.89	16.59	39.2	44.1	50.4	58.8
sim3.op2	17.06	17.74	12.40	13.66	35.6	43.5	55.7	65.2
sim4.op2	18.14	19.48	11.71	13.72	36.3	41.9	54.5	60.6
sim5.op2	12.69	12.80	9.59	10.01	12.8	43.7	47.6	65.5

Table 2: Benchmark results: On the left side we see the compression ratio using different precisions, on the right side we see the throughput

8 The OUG1/OUGV1blocks in NASTRAN OP2 file format include not only displacements for every vertex but also rotations. These rotations often are not used in practice. FEMZIP-N provides the possibility to skip every floating point data object that must be quantized during compression. If a data object was skipped it cannot be reconstructed during decompression.

4.2: Compression speed

Table 2 also illustrates the speed benchmark results. In row five we see the compression throughput measured in megabytes per second. The average of the first four models adds up to the satisfying result of 37.4 MB/sec. For sim5.op2 this value is much smaller due to the special nature of two of the three included OUGV1 blocks. These blocks contain several thousand different modes with only a few vertices per mode. This leads to suboptimal results while analyzing the file for compression purposes.

In most cases it is more important to have a faster decompression than a fast compression algorithm. Therefore we concentrated on the acceleration of the decompression. Normally decompression is automatically somewhat faster than compression due to fewer analysis the algorithms have to accomplish (we can see this effect if we compare row five and six of Table 2). To accelerate decompression further we decided to make use of multi core CPUs (see "*OpenMP parallelization of the decompression")*. If we look at the last three rows of Table 2 we see that throughput increases by approximately 10-20 percent if we use two cores for decompression and up to 65 percent on four cores. That is no amazing speed up. But we could not expect perfect scalability due to the writing time overhead mentioned before. If we take into account that a modern system can write only 80 MB per second to disk, it is obvious why parallelization does not scale better: On four cores almost the whole time is used to write data to disk, the calculating time hides behind this amount.

We investigated the effect of our parallelization approach more precisely. We compiled a special decompression executable, which does not write any output. Figure 2 presents the benchmarks made with this tool. We decompressed sim1.op2 several times and took the average for each number of processors (1, 2, 4, 8 and 16) over all measurements. As we can see this application accelerates better on more cores. But it also does not perfectly scale: It only works twice as fast if we use four cores. On the one hand it depends on a basic problem of data compression (regarding parallelization approaches): Much data is moved but only a few calculations are executed. On the other hand one part of the program was not parallelized. It reconstructs the original OP2 file format. This part is not easy to parallelize because there is only one stream to build. All of the threads should be working on it at the same time.

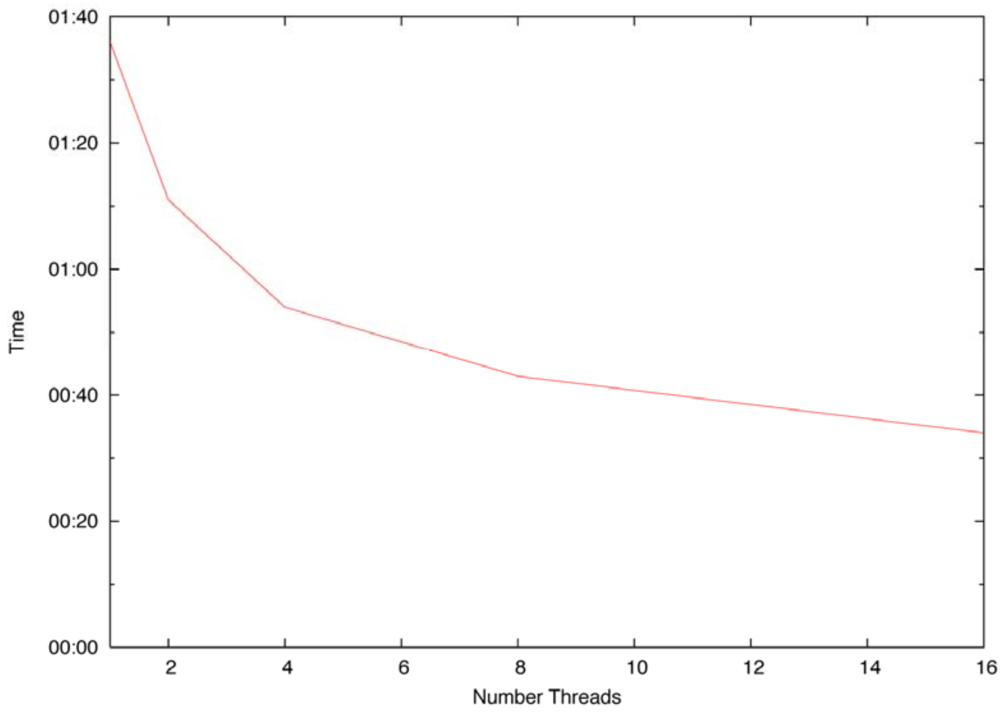

Figure 2: Effect of the OpenMP parallelization of FEMZIP-N on decompression speed

One of the main purposes of our decompression tool (as a library version) will be the decompression of compressed NASTRAN OP2 simulation results on the fly, during the read-in process of a postprocessing software. For this purpose the parallelization effort exemplarily shown in Figure 2 will be fully taken into account because this kind of software does not write output data to disk.

5: CONCLUSION AND OUTLOOK

In this paper we described the advantages of data compression and introduced the FEMZIP-N software that was developed for a lossy compression of NASTRAN OP2 simulation results out of the NVH sector. Further more we saw that the use of this tool can save up to 90 percent storage space and that it does not need a significant amount of time for decompression and even compression. The throughput of both applications is approximately located at 40 MB/s. The decompression time can be increased further using multi core machines. This is particularly interesting for postprocessing applications where no output must be written to disk. In this case reading in compressed models can be faster than reading the original data.

In FEMZIP-N several future-oriented technologies were introduced. These features give us the possibility to further improve its performance. First, there is the possibility of new prediction schemes that could easily be integrated due to our matrix approach and might increase compression ratio or could be applied on OP2 data blocks not supported yet. This could lead to a FEMZIP-N version that would not be specialized on data from NVH simulations. Second, the use of HDF5 could be further exploited. HDF5 is a very modern data management approach, which currently captures the area of simulation applications. We plan to use HDF5 filter technology to compose a self inflating file format based on the HDF5-API.

Finally, there are other file formats waiting to be compressed. Almost every profession in the simulation area struggles with huge amounts of data. Especially in the CFD disciplines huge amounts of data are produced. Therefore the next augmentation of the FEMZIP family will probably be a tool to compress STAR-CD output data.

REFERENCES

[1] R.IZA-TERAN, R. LORENTZ - Lossless Compression of Meteorological Data, ERCIM News, No. 61, p. 41, April 2005

[2] MSC, Basis Dynamic Analysis User's Guide, MSC, 2004

[3] J. BAI, J. DONGARRA, A. RUHE, H. VAN DER VORST, Templates for the solution of algebraic eigenvalue problems, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.

[4] P.-S. YEH, The CCSDS Lossless Data Compression Recommendation for Space Applications, Lossless Compression Handbook, K. Sayood, Academic Press, pp. 311, 2003

[5] M. RETTENMEIER, Zwei Strategien zur verlustfreien Kompression von Simulationsergebnissen, Diploma Thesis at the Mathematical Institute of the University of Cologne, 2007