Advances in Data Compression Technique for Three-Dimensional Numerical Simulations

Mitsue Den
Hiraiso Solar Terrestrial Research Center Communications Research Laboratory,
3601 Isozaki, Hitachinaka, Ibaraki 311-1202
029-265-9729
029-265-9728
den@crl.go.jp

Kazuyuki Yamashita
Information Processing Center, Chiba University, 1-33 Yayoi, Inage, Chiba 263-8522
043-290-3536
043-290-3544
yamasita@ipc.chiba-u.ac.jp

Ryoji Matsumoto
Department of Physics, Faculty of Science, Chiba University, 1-33 Yayoi, Inage, Chiba 263-8522
043-290-3724
043-290-3720
matumoto@c.chiba-u.ac.jp

Abstract

We propose an algorithm of data compression for the purpose of effective output of multi-dimensional numerical simulation data with a huge number of mesh cells during simulation runs, supposed to use the compressed data including necessary and sufficient information in visualization or analyses after the simulations. This algorithm consists of classification or quantization of single or double precision real data at each mesh point and subsequent compression by using conventional compression algorithm. In the previous paper, we described detailed method of data compression, obtained the compression efficiency for several typical simulations and showed that this method gave good results for those simulations. However, our method was limited to the variables which have positive or negative definite sign such as density. In this paper, we extend this algorithm to all variables which have any signs such as velocity variables. We show that our proposed compression algorithm can attain as high compression efficiencies for the magnetic fields as our previous method. Furthermore we develop another version of our proposed algorithm in which is adopted for the second stage compression method as the next step. We perform test calculations of the compression efficiencies calculated by using this method for the same models as those in the previous paper and show that this compression method is also very useful as well as the previous one.

keywords:

Data compression, Image processing, Information processing, Numerical methods

1 Introduction

According to the increase of processing speed of operations and memory size of super computers, we can perform multi-dimensional numerical simulations and can simulate the 'virtual world' in the computer. Actually we perform three-dimensional (3D) numerical simulations in most cases, where any specific symmetries cannot be applied. However, 3D numerical simulations require outputting a huge amount of data during the runs if analyses of the data is done separately from the runs and the spatial resolution of the output data is imposed to be kept high, as usual. When debugging the simulation code or analyzing the data repetitiously, it is inconvenient to handle huge data. A huge amount of numerical data cause the other problems such that they occupy large disk space and need unnegligible I/O processing time. In general, super computers are shared by many users, so both disk space and CPU time available for a user are limited. Furthermore, people who treat dynamically evolving system must store the time sequential data at certain intervals. The shorter the intervals are set, the higher temporal resolution can be attained. However, the disk space is needed in proportion to the degree of the resolution. Namely the temporal resolution is limited by disk space and CPU time for data I/O. The limited number of snapshots may lead them to miss some important phenomena.

In order to overcome this dilemma, we proposed a real time compression algorithm available for numerical simulations with mesh code in the previous paper (Ref.1). The employment of data compression, i.e., reduction of data size, in the output routine in simulation codes can save both disk space and CPU time spent in the routine and thus it is possible to store more frames of the results of dynamical simulation and to attain higher resolution in time.

There exist many data compression methods (Ref.2 and references therein). They can be put into two categories: lossless methods and not lossless ones called 'lossy' methods. For the lossless methods the original data can be restored perfectly from the compressed data. These methods are suitable for sequential text data. One of algorithms of this type compression tools, e.g., LZ77, uses the redundancy in the data to minimize information(Ref.2, Ref.3). The higher redundancy that data have, the larger compression efficiency is expected. An example of compression tools based on this algorithm is gzip which is very popular for unix users.

On the other hand, we can allow some degree of information loss when drawing graphical image to look over data. It can be applied to data compression: we can find desired information in the compressed data by the appropriate lossy compression tool. Moreover image data have huge volume generally, so those data are apposite to the lossy compression methods. Drastic compression efficiency can be achieved by the lossy compression tools in general because they lose some information of the original data literally.

However, those lossy methods have a risk that the desired information is lost or the unnecessary information verbosely occupies the stored data volume instead after compression, which are called "failure" of data compression generally. On the other hand, if the information obtained from compressed data essentially covers the desired one expected from the original data, the compression is regarded as "success". JPEG(Ref.4, Ref.2) is a typical example for such lossy compression tools of the image data, which includes the parameter, called quality coefficient. That parameter is set in order to regulate the quality of the image after compression. Indeed, the quality of the image and the compression ratio have a close connection. The 'desired' information, or enough high quality parameter in JPEG is ambiguous and we will define it in next section. It is noted that the lossless methods also have a risk that the size of 'compressed' data exceeds that of the original data when the data is less verbose.

In Ref.1, we have proposed an algorithm of data compression which is appropriate for visualizing outputs of simulations with mesh code in multi-dimensional space. Our proposed algorithm consists

of the two steps. At the first step, we compress the simulation data by converting the data format into character type format, i.e., 1byte for each data. It is lossy type since the data for visualization need drastic compression due to their huge volume. Moreover some information are allowed to be lost when those data are converted into the image. The compression efficiency(CE) defined in the following is guaranteed about 75% at this stage if the original data are recorded in single precision. Since the compressed data are sequential stream of characters (text) after the first compression step, we further compress it by lossless method in the second compression stage. Any lossless compression methods are available and we adopt LZW15V.C(Ref.2) in Ref.1 and in our extended version of the first step compression algorithm described below since the source program of LZW15V.C is attained to Ref.1. In Ref.I, we have obtained CE $\simeq$ $75\% \sim 98\%$ after the two compression processes in our model calculation and have shown that CE depends on bias of data distribution of the original physical quantities.

The compression algorithm based on conversion of the data format is used for other purpose in FITS (Flexible Image Transport System)(Ref.5) for the image data and in technical application field for the simulation data(Ref.6). FITS, a standard image data format of the image data in the astronomical field, was developed in order to make astronomers compare and interchange the images made at different observatories easily. The researchers in technical application field use the similar algorithm in order to absorb the difference of OS and mainframe of computers (Ref.6) since the binary data format of the floating point depends on computer systems. We notice that this data format conversion algorithm results in data compression, which is thought to be regarded as an additional merit in those field. Our original points are following : we rather take advantage of this merit and combine the conversion method with the lossless compression method to obtain higher compression efficiency as shown in Ref.1. Moreover, we include the compression process in a main program of numerical simulations, so that more frames of the results of dynamical simulation can be stored and higher resolution in time can be attained.

Our proposed lossy compression algorithm in the first step, however, was limited to the variables such as density which have unique sign, + or − in Ref.1. In this paper, we extend first step part to variables which can have both signs, i.e., we can compress velocities, magnetic fields and the gravitational potential and show that the compression efficiency for those variables is almost same as that of the variables with the unique sign.

Furthermore we develop another version of our proposed compression algorithm in which zlib(Ref.7), one of popular compression libraries, is adopted for the second step compression method instead of LZW15V.C as the next step (hereafter call 'zlib version'). The compression method currently used in zlib never essentially expands the data, while LZW(Ref.8, Ref.2) can double the file size in extreme case. We also show the test calculations of CE by using this zlib version compression method for three models used in Ref.1.

The organization of this paper is as follows. In next section we describe our proposed algorithm of compression, show CE calculated by using simulation results of phenomena including magnetic field and compare the original image with that generated by using the compressed data. We also compare the CE calculated by using zlib version with that calculated by our previous version adopting LZW15V.C for the same models as those used in Ref.1. Section 3 is devoted for summary and discussion.

2 Data Compression Method and Compression Efficiency

Since we have already described compression algorithm in first step in Ref.1, we mainly present the extended part in this paper. 'Success' or 'failure' of a lossy compression method such as JPEG mainly used

for presentation depends on users' impression of the image data generally. Since our target data are used not only for presentation but for scientific analysis, we need more definite criterion of 'success' of compression, which is as follows. If we can get essentially the same information needed for scientific analysis in the image drawn with the compressed data with that in the image drawn with original data, we say that that compression is successful.

We define the compression efficiency, CE. It is presented by

$$\mathsf{CE} = 1 - \frac{m}{r}$$

where m is the size of the compressed data, called map data and r is the size of the original data, called raw data. This means that as CE tends to 1, compression is more efficient. If CE is negative, the size of the 'compressed' data exceeds that of the original data.

The first phase of compression algorithm is very simple.

1. Set the contour levels which cover the dynamic range of the original data.
2. Classify the original data according to the specified contour levels.
3. Make the conversion table which relates the original data and contour levels.

The data belonging to the same contour level are regarded having the same value. Information is partly lost during this compression phase.

If the data distribute smoothly, the following simplest procedure can be applied:

1. Find the minimum and maximum value of the original data.
2. Divide the range between the minimum and the maximum equally into a specified number of contour levels($n_{\rm cl}$), say, $2^8 = 256$ levels.
3. Convert the value of the original data at each grid point into the number of corresponding contour level (Classification).
4. Make the conversion table which records the original value corresponding to each number of contour level.

If the dynamic range of the original data is very large, the logarithm of those data is calculated and then classified into the contour level.

In this paper we have extended the part 1 such that we can compress any variables data. In Ref.1, we tested our method only for density variable which is positive definite. Our extended method is applicable to any variables, i.e., can treat the data whose minimum and maximum value have the opposite sign. We will show test results of compression of magnetic field components in the following.

As for the number of the contour levels we can use more than 256 levels, if necessary. The file size of the conversion table is (4byte) $\times n_{\rm cl}+$ (header of file) whose size is $\sim$ 1kbyte. If the original data are recorded in single precision floating point variable and $n_{\rm cl} = 256$, 4byte data at each cell of the original data can be compressed to 1byte, namely CE$\sim$ 75% including file size of the conversion table at the first step.

Since the compressed data obtained by the first step of this method are sequential text (i.e., stream of characters), we can further compress them by lossless compression tool(second step). Thus, in this algorithm, the data go through compression two times. For real time compression during the simulation, we adopt LZW15V.C for the extended version of the first step compression method mentioned above as well as Ref.I. Its algorithm is based on LZW which is also adopted by the 'compress' command in UNIX operating system and uses the redundancy of the data such as LZ77. As described in detail later, we develop another version of our proposed compression algorithm in which zlib is adopted for the second step compression method instead of LZW15V.c as the next step.

We have both the first and the second step compression programs at source level, so we link those subroutine programs and call them in main program when we output compressed data during the simulation. The load of CPU time for execution of first and second compression and I/O is very light as shown in Ref.1,

which is 249.80 seconds for 50 frames of compressed snapshot data whose file sizes are 718 kbytes $\sim$ 982 kbytes. It corresponds to 1.7% of the total CPU time.

The important different point between our compression method and conventional compression ones is that we can store 3D compressed data in case of 3D simulation. We can generate any projected surfaces after we finish the main calculation. On the other hand, when using 2D image compression methods such as JPEG, one have to save 2D data whose surface levels should be determined before calculation. So if one cannot find any expected results on the selected surfaces, one may have to run the main program again, which will consume CPU and real time.

In Ref.1, we tested our compression routine for snapshot data of density variables in three models and obtained 93% $\sim$ 98% CE. We concluded that the compression for density variables are successful. Here we test our proposed method for magnetic fields. We adopt test data from the results of 3D numerical simulation of emergence of twisted magnetic flux tubes calculated by R. Matsumoto. Figures 1a and 1b show the isosurface of magnetic fields and the density distribution for the original data and for the compressed data respectively. The surface of the thick tube is the magnetic isosurface, the thin dark gray curves are the magnetic field lines and the shaded walls in the simulation box show the density. We find that the magnetic field lines of the compressed data almost coincide with those of the original data. We can say that it is even difficult to find difference among them. However, there are some differences in the density. The gradation of the wall in Figure 1b (compressed data) is not so smooth compared with that in Figure 1a (original data). We think that $n_{\rm cl}$ (256) is not enough for the density because it has large dynamic range. If this difference is not important for analysis in this model, our compression method is very effective. Actually we get about 75% compression efficiency at the first step and achieve more than 91% compression at the final step for all three magnetic fields as seen in Table 1. This means that we can save ten times more frames of dynamical data under a fixed disk space if we adopt our compression method.

The compression algorithm used in zlib is an LZ77 variant, essentially same as the that in gzip and Zip. One of the merits of zlib is that the compression method currently used in zlib never expands the data, while LZW can double the file size in extreme case. Thus we develop another version of our proposed compression method with zlib library as the next step. In Table 2, we show the CE calculated by using zlib version for snapshot data of density variables in three models used in Ref.1. We also show the CE calculated by using the previous method (hereafter call 'LZW15V.C version') for comparison. The compression level can be set in zlib and changes from 0 (no compression) to 9 (the highest compression with the longest calculation time). The CE does not depend on the compression level largely and the CEs are almost same with those obtained by LZW15V.C version. The file sizes of the data compressed by zlib version are less than 7% of those of the original data and this method is also useful as well as the previous method.

3 Summary and Discussion

We have extended our proposed compression algorithm for multi-dimensional numerical simulation results with mesh code presented in Ref.1 such that we can also compress variables which can have both signs. Our method consists of two steps. The first one is lossy compression method based on quantization or classification of original data. The second one is real time lossless compression method.

We have shown that the compression efficiency is always more than 75% at the first step and is achieved to be more than 91% in our model calculation when the original data are recorded in single precision. We also have presented that the variables with any signs can be compressed successfully as well as the density variable by obtaining CE of the magnetic fields. Namely, e.g., in magnetohydrodynamical simulations, we can

compress and can output 3D volume data of the density, the velocity, the magnetic fields, the energy and the gravitational potential simultaneously during main run. It is also shown that our compression method is useful for visualization or quick-looks of numerical results.

Since any lossless compression methods are available, we have adopted LZW15V.C for the second step compression method in our extended version of the compression method which can treat any variables. As the next step, we have developed another version of our proposed compression method in which zlib is adopted for the second one since the compression algorithm used in zlib can avoid expanding the original data. We have obtained the CE for the snap shot of density variance used in Ref.1 and have found in our test calculations that the CE does not depend on the compression level largely and the CEs obtained by zlib version compression method are $93\% \sim 98\%$, almost same with those obtained by LZW15V.C version. Thus it has been shown that this zlib version is also useful as well as the previous LZW15V.C version.

For the other schemes concerned with setting of the contour levels, the equal cell number scheme may be considered, which determines the contour levels such that the number of cells in every contour level are equal. This scheme is appropriate for the case that there are large statistic bias in data distribution, i.e., the case that there are sharp peaks in the histogram $n_{\rm cell}(\rho)$ where ρ denotes the variable such as density. A lot of mesh cells represent the density around at such peaks. Let us consider an example: the minimum density is $\rho = 0.1$, the maximum is $\rho = 1.1$, the most cells are in $0.1 < \rho < 0.2$, i.e., the peak in $n_{\rm cell}(\rho)$, and there are no cells of $0.2 < \rho < 1.1$. According to the equal cell number scheme, the fine contour levels are set at $0.1 < \rho < 0.2$ corresponding to the peak and little numbers of contour levels are set at $0.2 < \rho < 1.1$. This scheme may lead to successful compression. Actually this scheme can reproduce the original data in the global simulation space more correctly compared with our regular interval scheme in the case of extreme statistical biased data distribution. However, the physically interesting phenomena may be often seen in the behavior at high gradient region in the usual space. Especially when using mesh code, the number of cells for such region is small compared with the total number of cells generally. The equal cell number scheme, which expresses many cells as correctly as possible, is not appropriate for extraction of the interesting phenomena since this scheme does not evaluate the cells outside the peak correctly, in other words, it loses more information in the cells outside the peak than those in the cells inside the peak. On the other hand, although our regular interval scheme is simple, it can reproduce the local and fine structures expressed with smaller numbers of mesh cells which may be rare events compared with the global simulation volume because it reproduces the global structure fair. When applying our regular interval scheme to the above example without any contrivance, the number of the contour levels assigned to the peak is $(1.1(\max) - 0.1(\min))/n_{\rm cl} \times 0.1$(effective width of the peak) ~ 26 with $n_{\rm cl} = 256$. If complicated structures exist in the peak, specified ~ 26 contour levels may not be enough for resolution of such structures. However, if we notice before compression that the interesting phenomena can be found in the peak, our method can lead to successful compression by introducing the proper cut-off in the histogram for this example and other general cases including sharp peaks in the histogram. So it can be said that our regular interval scheme is better choice. Thus we conclude that our compression methods, both LZW15V.C version and zlib version for the second compression step, are applicable and are useful for almost various simulation data.

As noted in Ref.1, this method also can be used for I/O module for AVS and IDL, one of popular graphical softwares. This is future work.

AVS is a trademark of Advanced Visual Systems Inc.
IDL is a trademark of Research Systems Inc.

References

1. Den M., Yamashita, K., Matsumoto R. 1997, J. for Academic Computing and Networking vol.1, p25.
2. Nelson M., Gailly J.-L. 1996, The Data Compression Book, Second Edition, (Prentice Hall, Toppan, Tokyo).
3. Ziv, J. and Lempel, A., 1977, IEEE Transactions on Information Theory, vol.23, No.3, p337.
4. Wallace, G.K., 1991, Communications of the ACM, vol.34, No.4, p31.
5. Wells, D.C., Greisen, E.W. and Harten, R.H., 1981, Astron. Astrophys. Suppl. Ser. vol.44, p363.
6. Private communication.
7. The zlib home page, http://www.cdrom.com/pub/infozip/zlib/
8. Nelson, M., 1989, Dr. Dobb's journal, vol.14, No.10, p.29.