Irreversible data compression concepts with polynomial fitting in time-order of particle trajectory for visualization of huge particle system

To cite this article: H Ohtani et al 2013 J. Phys.: Conf. Ser. 454 012078

View the article online for updates and enhancements.
Irreversible data compression concepts with polynomial fitting in time-order of particle trajectory for visualization of huge particle system

H Ohtani¹,², K Hagita³, A M Ito¹, T Kato⁴, T Saitoh⁵ and T Takeda⁶

¹ Department of Helical Plasma Research, National Institute for Fusion Science, 322-6 Orosi-cho, Toki 509-5292, Japan
² Department of Fusion Science, The Graduate University for Advanced Studies (SOKENDAI), 322-6 Orosi-cho, Toki 509-5292, Japan
³ Department of Applied Physics, National Defense Academy of Japan, 1-10-20 Hashirimizu, Yokosuka 239-8686, Japan
⁴ Department of Physical Science, Graduate School of Science, Hiroshima University, 1-3-1 Kagamiyama, Higashi-Hiroshima 739-8526, Japan
⁵ Interactive Research Center of Science, Graduate School of Science and Engineering, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, Japan
⁶ Center for Computational Astrophysics, National Astronomical Observatory of Japan, 2-21-1 Osawa, Mitaka 181-8588, Japan

E-mail: ohtani.hiroaki @ nifs.ac.jp

Abstract. We propose in this paper a data compression scheme for large-scale particle simulations, which has favorable prospects for scientific visualization of particle systems. Our data compression concepts deal with the data of particle orbits obtained by simulation directly and have the following features: (i) Through control over the compression scheme, the difference between the simulation variables and the reconstructed values for the visualization from the compressed data becomes smaller than a given constant. (ii) The particles in the simulation are regarded as independent particles and the time-series data for each particle is compressed with an independent time-step for the particle. (iii) A particle trajectory is approximated by a polynomial function based on the characteristic motion of the particle. It is reconstructed as a continuous curve through interpolation from the values of the function for intermediate values of the sample data. We name this concept “TOKI (Time-Order Kinetic Irreversible compression)”. In this paper, we present an example of an implementation of a data-compression scheme with the above features. Several application results are shown for plasma and galaxy formation simulation data.

1. Introduction

The peta-scale computers, e.g., Titan, Sequoia, and K computer, have started operation, and ultra-huge-scale simulations can be performed. Such simulations will discover many new features in the various fields of science. Though many scientific discoveries are expected, there are big urgent issues, that are, data input and output (data I/O), transmission, saving, and analysis including visualization for the enormous volumes of data. The calculation speed of computers has become very large, but the speed of data I/O onto a storage system is not as high as the CPU calculation speed. Because the data size of huge-scale simulations is enormous, in some cases not
all data can be saved onto the storage system. Supercomputer systems are, in general, remotely connected, and the generated data is transmitted to a local machine. Data transmission needs very long time. Since the local machine does not have as large memory as the supercomputer, it is impossible to analyze and/or visualize all the data at once. These problems are a major bottleneck in huge-scale simulation in many fields.

In order to resolve the problems of data size, several data compression schemes were proposed for the time-sequential data of process computers [1], electrocardiogram [2, 3], moving objects [4], and GPS trajectories [5], compression of images [6, 7], video and sound [8, 9], and particle trajectory data of molecular dynamics simulation [10, 11, 12]. H.264/MPEG-4 AVC is one of the video compression standards [8, 9]. It adopts the integer transform operation with 16 bit integer accuracy, which enables it to operate with only adding, subtraction and a bit shift. The advantage of this operation is that it does not generate the difference in operation results by the different implementations of a real number operation. It also adopts the motion compensation which describes an image in terms of the transformation of a reference image at an earlier time or even an image from the future, to the current image. Davisson presented a method of theoretically analyzing the data compression techniques which relied on the approximation of the source output by polynomial segments, and considered straight-line interpolation, specifically [7]. Omeltchenko et al. used octree indexing and sorted atoms accordingly on the resulting space-filling curve for scalable I/O of molecular dynamics data [12]. By storing differences of successive atomic coordinates and using an adaptive, variable-length encoding to handle exceptional values, the I/O size was reduced by an order-of-magnitude with a user-controlled error bound. The GROMACS XTC format is a portable format for particle trajectories in the molecular dynamics simulation [10]. The trajectories are written using a reduced precision algorithm which works in the following way: the coordinates are multiplied by a scale factor (typically 1000). These are rounded to integer values. The algorithms by Spangberg et al. for XTC algorithm are based on the observation that differences in atomic coordinates and velocities, in either time or space, are generally smaller than the absolute values of the coordinates and velocities [11]. They also developed an algorithm which used a block sorting. By using an interactive polynomial interpolation, Daehlen and Floater described the decomposition algorithm and presented some numerical examples, such as Franke’s function, which was a well-known standard function used in the investigation of methods for interpolating scattered data, and a discrete height data above sea-level on a uniform grid of coordinates in an area of land [13].

A characteristic particle motion plays an important role in the several phenomena under the simulations of plasma physics, astrophysics, and so on. In the collisionless magnetic reconnection, un-magnetized particles execute a complex thermal motion, and this complex motion is one of the main causes to trigger the reconnection [14]. Several particles are selectively accelerated in the collisionless shock wave [15]. In order to capture these behaviors, it is important to trace the particle trajectory or local averages of physical quantities in detail instead of statistical averages, such as temperature and average velocity. On the other hand, galaxy formation simulation deals with huge-scale spatiotemporal phenomena, such as $10^{10}$ years for age of the universe, $10^8$ years for the galaxy dynamics, $10^6$ years for the giant molecular clouds dynamics, and $10^3$ years for the supernova remnants. Small-scale temporal phenomena take place in the small compact region. In order to see what phenomena occur in the galaxy formation, it is needed to record the simulation data in the small-time scale since the interstellar medium evolves on the smaller scale. If we know how large the scale of the phenomena is and/or, where it takes place in advance, before starting the simulation, it is possible to perform on-the-fly visualization. However, in the case that we know those only after the simulation has proceeded, it is needed to store and visualize all the data as a post process. When we log all data at each uniform time step, the size of stored data becomes enormous, and it needs a long time to input and output the large data. Data compression is indispensable in performing the huge-scale simulation.
We propose in this paper that a compression of simulation data from a large-scale particle simulation holds favorable prospects for scientific visualization of particle systems. Our concepts deal with the data of particle orbits obtained by simulation directly. Switching from a real to an integer representation, as proposed in the field of molecular dynamics simulation, is a kind of approximation, and the data compression of binary digit strings is one of the powerful tools. In this paper, we approximate a particle trajectory with a polynomial function based on the characteristic motion of the particle. The parameters of each polynomial are stored in place of the original data. Moreover, we can use arbitrary mathematical functions based on known behavior of the particle dynamics. The function is changed according to the characteristic of the particle dynamics, or the simulation data of the particle trajectory itself is recorded without the approximation by the polynomial. This is expected to reduce significantly the data size saved onto the storage and improve I/O performance for viewing software. Accordingly, it becomes possible to visualize a larger-scale particle system. We name this concept “TOKI (Time-Order Kinetic Irreversible compression)”. 

In section 2, we propose the concepts of how to compress the data. One implementation of our concept is explained as a flow chart. We apply the compression scheme to plasma and galaxy formation simulations. In section 3, we show the data compression rate, reproducibility, and the error between the simulation results and the reconstructed data from the compressed data. We discuss the results and the future plan in section 4. Finally, we conclude in section 5.

2. Concepts of a compression scheme

Our proposal “TOKI” adopts a data-compression method for particle trajectory data which has following features:

- Through control over the compression scheme, the difference between the simulation variables and the reconstructed values for the visualization from the compressed data becomes smaller than given constant.
- The particles in the simulation are regarded as independent particles and the time-series data for each particle is compressed with independent time-step for the particle.
- A particle trajectory is approximated by a polynomial function based on the characteristic motion of the particle. In other words, the parameters of each polynomial are stored in place of the original data. It is reconstructed as a continuous curve by the interpolation from the values of the function for intermediate values of the sample data.

In short, the continuous curve which is established by a small number of parameters of the interpolation function approximates the original particle trajectory, and the amount of necessary data can be reduced. We consider the following procedures in order to realize these concepts.

(i) In the case that a particle moves rapidly with a high-frequency oscillating motion, the simulation data is stored as raw data without a change, since such motion cannot be approximated by the polynomial. The data size becomes a product of the number of the fast oscillating particles and the number of time steps.

(ii) In the case that a particle moves with a slow-frequency oscillation motion, the trajectory can be traced by the interpolation function of the polynomial. The coefficients of the polynomial are stored in this case. The data size becomes a product of the number of particles, the number of the coefficients, and the number of the approximation calculations. The polynomial adopted for the interpolation function in this compression scheme depends on the physical system we investigate. For example, if a particle executes circular motion, a circular function is suitable for expressing such a motion. After obtaining the coefficients of the interpolation function, we compare the simulation data and the interpolated values. If the difference between those is smaller than a given constant, which is an input parameter
for the error control (we call this an allowed error in the following), the coefficients are adopted and stored. If the difference is larger than the allowed error, we reduce the number of the simulation data which is interpolated in this procedure and calculate the interpolation again. Until the difference becomes smaller than the allowed error, reduction of the data size and calculation of the interpolation are repeated. When the size of the simulation data is equal or smaller than the number of coefficients of the polynomial, the calculation is stopped, and the raw simulation data is stored.

By these procedures, the following data compression is performed. For example, by using a cubic function as an interpolation polynomial, the number of coefficients is four. If the number of the raw data is larger than four, and it is interpolated by a cubic function, four coefficients are stored in place of the data. It means a decrease in the number of the stored data. We call this a data compression. In this paper, we adopt a cubic function for the test of the compression code. If the number of rapid particles which cannot be represented by the interpolation function is much smaller compared to the number in the whole system, an efficient data compression is expected by this scheme. We call the above procedure “encoding”, and reproduction of the particle trajectory from compressed data “decoding”, respectively, in the following.

Figure 1 shows one implementation of our scheme as a flow chart in which the encoder performs the procedures in order to realize the “TOKI” procedure.

Initialization allocates working arrays for storing the time-sequential data of particles, the coefficients and the raw data. The particle position is stored in the working array sequentially (In figure 1, the position data $x$ is stored in the array $work$, and the data number is set as $i$). If we want to record the compressed data (we have a forcing-output flag set), or the working array is full, the encoder calculates the coefficients of the interpolation function by the least-squares method. When those are obtained, we compare the simulation data with the interpolation values. If the difference between them is larger than the allowed error, the data length of the time-sequential particle data for the interpolation in the working array is reduced, and the least-squares method calculates the interpolation again. When the difference is smaller than the allowed error, the data length is increased, and the calculation of the least-squares method is performed again, too. In this procedure, we find the longest data length $n$ for interpolation. Or, when the number of the data for interpolation is equal or smaller than the number of the coefficients of the polynomial, this procedure is stopped.

At this time, if the number of data which is interpolated is larger than the number of the coefficients, the coefficients are stored. Otherwise, the raw simulation data is stored. The particle position data in the work array is shifted by the number $n$ of the raw data, which is interpolated or stored. Because the size of the working array is limited, the data which is not interpolated is shifted to the beginning of the array. After the data shift, new data is added after the last data in order to find the coefficients for the data as large as needed. If we have a forcing-output flag set, then the stored compression data is recorded, and we determine whether the data compression procedure continues or not. If the procedure continues, the position data is stored in the working array again. Finally the working arrays are de-allocated.

In the initialization of the encoder, several initial parameters are input: basename, digit number and extension of the output file, allowed error, order of the polynomial, number of particles and size of working array. The digit number specifies the number of digits when we assign a number to the output file. In the body of the encoder, the simulation time, the enforcing-output flag and the position of particles are inputted. In the output part of the encoder, several data are recorded: particle number, dimension ($x$, $y$ and $z$), number of interpolation calculations, time when the interpolation starts, interpolated data length, and coefficients of the interpolation functions. When the interpolation function cannot represent the particle trajectory, the raw data of the simulation is recorded, and the number of the recorded raw data is recorded instead of the interpolated data length.
3. Applications
We aim to apply this compression method “TOKI” to the simulations in plasma physics and galaxy formation. Since the characteristic time scales in the particle dynamics are different in these systems, it is practically impossible to store visualization data which is logged simply at every smallest time step. We consider that those are suitable systems for application of the scheme.

In this paper, we encode the simulation data as a post process to examine whether our concepts of data compression and the encoder work well or not. Implementing the encoder in the simulation code itself is left as future work. Moreover, we use a cubic function as interpolation polynomial for the encoder test.

As mentioned above, the simulation data is represented by an interpolation function in the encoder. The decoder reconstructs the data from the interpolation function at each time given...
in the time in the simulation. In the following, we compare the reconstructed or decoded data with the raw simulation data and examine the differences. Moreover, we compare the data size of encoded data with that of raw data. The raw data contains the simulation time and the three components of particle positions. Compression rate is defined as the ratio of the encoded data size to the raw data size. We study the dependence of the rate on the size of the working array and the allowed error values. As a first preliminary test, we choose a Fortran binary with double precision as data recording format.

3.1. Plasma particle simulation
We then apply our compression scheme to the plasma particle simulation. The simulation code is a three-dimensional electromagnetic particle-in-cell code [16, 17]. The boundary conditions are periodic. The number of particles per cell is not uniform in space and changes from 10 to 100. The data for compression is a part of the full simulation data, and the number of particles in the encoder test is 800. The data was calculated for 10,000 time steps, and the time step of the simulation satisfies the Courant-Friedrich-Lewy (CFL) condition [18]. Under the CFL condition, in the plasma particle simulation the time step $dt$ needs to be much smaller than the plasma frequency $1/\omega_p$ or the cyclotron frequency $1/\omega_c$. As a result, for calculation stability, the time step $dt$ and space interval $dx$ are defined by the electron plasma frequency $\omega_{pe}$ and the electron Debye length $\lambda_e$, respectively, in the full plasma particle simulation. In fact, in order to express the dynamics of both electrons and ions, $dt$ and $dx$ are smaller than $1/\omega_{pe}$ and $\lambda_e$, respectively. Since $dt$ is small, the size of the output data becomes large.

Figure 2 shows the comparison between the simulation data and the decoded data reconstructed from the encoded data by the decoder for one electron in three-dimension, and the error between the simulation and the decoded data. The size of the working array is 64 and the allowed error is 0.1. The time evolution of $x$ and $z$ coordinates has a discontinuity because of the periodic boundary condition used for analyzing and visualizing the particle trajectory in this research area. In this plasma particle simulation, the particle goes out from the simulation box, and the position of the particle is modified; $x \rightarrow x \pm L_x$, where $L_x$ is the simulation box length. It is found that the decoded data reproduces the high-frequency motion of the particle, and that the error between the simulation data and the decoded data is less than the allowed error value (=0.1). Figure 3 displays the dependence of the error between the simulation and the decoded data on the allowed error. In all cases, the error is smaller than the allowed error. From this result, it is shown that the error introduced by the encoder is under control. Table 1 shows the dependence of CPU time, the data size and the data compression rate on the working array size and on the allowed error. The larger the size of working array becomes, the longer the CPU time that the calculation needs and the smaller the compressed data size becomes. It is because the encoder tries to obtain an interpolation function for time-sequential data as long as possible.

3.2. N-body/gas simulation
We next apply our compression scheme to $N$-body/gas simulation of galaxy formation [19, 20]. Gravitational interaction is calculated through a tree method with GRAPE, and the gas is simulated by smoothed particle hydrodynamics under the open boundary condition. The number of sequential simulation data is 31. The number of particles in the original simulation data is 1,005,600, and we sample 10,560 particles from it for the first test of the encoder.

Table 2 shows the size of compressed data and the data compression rate. When the allowed error value is too small, the compressed data size becomes almost the same as the raw data size. In such a simulation case, it is convenient that the allowed error value is 0.005 to 0.0001. Figure 4 shows the comparison between the simulation data and the decoded data reconstructed from the encoded data by the decoder for two particles with high- and low-frequency motions,
**Figure 2.** Reproducibility and error between the simulation and the decoded data in the case of plasma particle simulation. The time evolution of the particle coordinates ((a) x, (b) y, (c) z) is shown. The red line indicates the decoded data, and the green plus symbols show the simulation data. (d) Errors (difference) between the simulation and decoded data are shown. Plus, cross, and square symbols are the errors for x, y, and z coordinates, respectively.

**Table 1.** Dependence of CPU time, data size and data compression rate on work array size and on allowed error in the case of plasma particle simulation.

| Work array size | Allowed error | CPU time (sec) | Data size (MB) | Compression rate (%) |
|-----------------|----------------|----------------|----------------|---------------------|
| Raw data        | -              | -              | 183.45         | -                   |
| 32              | 0.1            | 2.67           | 50.13          | 27.3                |
| 64              | 0.1            | 2.28           | 25.81          | 14.1                |
| 128             | 0.1            | 3.44           | 14.01          | 7.6                 |
| 256             | 0.5            | 4.11           | 7.59           | 4.1                 |
| 256             | 0.1            | 7.75           | 10.54          | 5.7                 |
| 256             | 0.05           | 8.88           | 12.39          | 6.8                 |

respectively. The original data was sampled at coarse time intervals when the simulation ran. That is, the simulation time step is small enough, but the time step of the output is coarse. For the simulation data obtained with the coarse output intervals, the encoder cannot obtain a
suitable interpolation function of the rapid particle, as shown in figure 4(a). On the other hand, for the low-frequency motion shown in figure 4(b), the encoder can produce an interpolation function to represent the motion. If the encoder used the sufficiently large data set obtained with small time-step output, it is expected that the fitting function can represent the high-frequency wave motion, and that the data compression ratio becomes better. In order to deal with such large data set directly, we will implement the encoder into the simulation code and calculate simultaneously the interpolation and the running simulation. Then, it would not be necessary to sample the simulation data only at coarse time intervals, instead the full data set will be stored.

4. Discussion
In this paper, we study how to encode simulation data as a post process and examine whether our concepts of data compression and our encoder work well or not. The compression ratios are about 5% in the plasma simulation and about 30% in the galaxy formation simulation, respectively. These values are good for practical applications. As next stage, we want to implement this encoder into the simulation code and encode simultaneously the data while performing the simulation. In this case, there are some problems: The encoder needs memory for the working array and calculation time for obtaining the interpolation function. The issue is whether these costs are comparable to the I/O time for the raw data and whether the memory size of the working array can be reduced. Those are important points in trade-off. We need to
Table 2. Dependence of data size and data compression rate on allowed error in the case of $N$-body/gas simulation.

| Number of particles | Allowed error | Data size (MB) | Compression rate (%) |
|---------------------|---------------|---------------|----------------------|
| 10,560              | Raw data      | 11.24         | -                    |
| 10,560              | 0.00005       | 11.21         | 99.4                 |
| 10,560              | 0.0001        | 3.64          | 32.4                 |
| 10,560              | 0.005         | 3.26          | 29.0                 |
| 1,005,600           | Raw data      | 1070.26       | -                    |
| 1,005,600           | 0.001         | 344.49        | 32.2                 |

optimize the encoder for high-performance computing.

We define $dt$ as the time step of data output in the simulation here. In this case, the output data size is proportional to $\sim 1/dt$. On the other hand, in this paper, we trace the particle trajectory through an interpolation function. We define $\Delta t$ as a time interval in which a single interpolation function can represent the trajectory. $\Delta t$ is dependent on the character of particle dynamics, since the determination of the interpolation function depends on the aspect of the dynamics. Accordingly, $\Delta t$ takes different values from one particle to the other, that is, $\Delta t$ time step is an independent time interval. The data size which is recorded every $\Delta t$ is proportional to $\sim 1/\Delta t$. Since $\Delta t$ is larger than the uniform $dt$ defined in the simulation, the data size for $\Delta t$ becomes smaller than that for $dt$.

For the test of the encoder, we used a cubic function for fitting the sequential simulation data. As mentioned in section 2, we consider that the polynomial adopted for the interpolation function in this compression scheme should depend on the actual physical system we investigate. Selecting the preferred function represents a problem. If we change the interpolation fitting function from a cubic function to higher-order functions, complex data can be expressed better, but the function may over-fit an occasional noise, and the compression rate may become worse. If $\Delta t$ becomes $N$ times as large as $dt$ because of the interpolation of an $n$-order polynomial function, the data compression ratio is $N/n$. When $n$ increases, the ratio becomes better and the objective of the data compression is achieved. Moreover, even if it is concluded by the examination that the data should be interpolated by a high-order polynomial function, we also need to investigate the conclusion from the engineering view point, such as calculation cost and memory size.

As shown in section 3, the size of the working array has an influence on the compression rate, the calculation time, and the memory size of the encoder. Optimization of the size of the working array by auto-tuning is also important. As mentioned above, the reductions of the memory size of the working array and of the calculation cost of interpolation remain technical problems. And, it is important to compare the schemes as mentioned in section 1 with same simulation data in order to discuss advantages and disadvantages of our method. In the future we plan to realize this task by applying our concepts to a variety of systems.

The concept “TOKI” can be applied to other vector and scalar simulation data, such as electromagnetic field, temperature, etc, which does not change with a high-frequency oscillation. Needless to say, ultra-huge-scale simulation data are a common issue also for fields other than plasma physics and astrophysics. Further we are going to apply our concepts to other simulation variables and to a variety of systems.

In future, we will implement a decoder for our compression method on viewer applications, for example, Zindaiji [21] and realize visualization of a large-scale particle system.
Figure 4. Reproducibility in the case of $N$-body/gas simulation data. We show only the $x$ coordinate of a particle which moves in the three-dimensional space. Red, green, blue, magenta, and sky blue lines show the simulation raw data, decoded data under the conditions of the allowed errors 0.01, 0.001, 0.005, and 0.0001, respectively. In the high-frequency particle case (a), raw data and data for error=0.0001 overlap, and data for error=0.01, 0.001 and 0.005 overlap.

5. Conclusions
We proposed the data compression method “TOKI” for particle trajectory data produced by simulation. It was possible to realize a good reproducibility from encoded data and to control
the error between the decoded values and simulation data.

By using this compression method, the data size saved onto the storage is expected to be reduced and I/O performance for visualization software significantly improved. As a result, it will be possible to visualize more large-scale simulation data and to realize reactive animation visualization. It will be possible to visualize time-sequential simulation data of a large-scale system and to analyze them by exploring in space. Plasma phenomena, such as, reconnection and shock waves, time evolution of plasma particles and large-scale structure, will be simultaneously studied in such real-time visualization. In research on galaxy formation, it will be possible to describe a fully dynamical evolution from a region of high-density star formation to a whole galaxy. The data compression scheme could open up a new path to understand the physics of these systems through capturing phenomena occurring in local regions with the visualization of the detailed behavior of particles.

Acknowledgments

This work was supported by National Institutes of Natural Sciences (NINS) Program for Cross-Disciplinary Study.

References

[1] Hale J C and Sellars H L 1981 Chem. Eng. Prog. 77 38
[2] Jalaleddine S M S, Hutchens C G, Strattan R D and Coberly W A 1990 IEEE Trans. Biomedical Eng. 37 329
[3] Nygaard R, Melnikov G and Katsaggelos A K 2001 IEEE Trans. Biomedical Eng. 48 28
[4] Meratnia N and de By R A 2004 Lecture Notes in Computer Science 2992 765
[5] Chen M, Xu M and Franti P, 2012 Proc. Data Compression Conference 62
[6] Staadt O G, Gross M H and Weber R 1997 Proc. IEEE Vis. ’97 337
[7] Davison L D 1968 IEEE Trans. Information Theory 14-1 390
[8] http://www.itu.int/
[9] http://www.iso.org/iso/jtc1_home.html
[10] Green D G, Meacham K E, Surridge M, van Hoesel F and Berendsen H J C, 1955 Metecc-95 Proceedings
[11] Spangberg D, Larsson D S D and van der Spoel D 2011 J. Mol. Model 17 2669
[12] Omeltchenko A, Campbell T J , Kalis R K, Liu X and Nakano A 2000 Comp. Phys. Comm. 131 78
[13] Daehlen M and Floater M 1993 Numerical Algorithms 5 165
[14] For example, Horton W and Tajima T 1991 J. Geophys. Res. 96 15811. Ishizawa A and Horiuchi R 2005 Phys. Rev. Lett. 95 045003, and Ohtani H, Horton W, Petrosky T and Horiuchi R 2009 J. Plasma Fusion Res. SERIES 8 203
[15] For example, Sugiyama T 2011 Phys. Plasmas 18 022302, Kato T and Takabe H 2010 Astrophys. J. 721 828, Riquelme M A and Spitkovsky A 2011 Astrophys. J. 733 63, and Matsumoto Y, Amano T and Hoshino M 2012 Astrophys. J. 755 109
[16] Birdsall C K and Langdon A B 1985 Plasma Physics Via Computer Simulation (McGraw-Hill, New York)
[17] Ohtani H and Horiuchi R 2009 Plasma Fusion Res. 4 024
[18] Courant R, Friedrichs K O, and Lewy H 1928 Math. Ann. 100 32
[19] Saitoh T R, Koda J, Okamoto T, Wada K, and Habe A 2006 Astrophysical J. 640 22
[20] Takeda T 2007 Proc. SIGGRAPH ’07 ACM SIGGRAPH 2007 computer animation festival 62
[21] http://www.cfca.nao.ac.jp/~takedatk/COMPUTER/ZINDAJI3/Zindaji3Top_E.html