УДК 004.92

V. Ianushkevych[1], **M. Dosta**[2] (M.Sc.), **S. Antonyuk**[2] (Dr.-Ing.)
S. Heinrich[2] (Prof.), **V.A. Svyatnyy**[1] (Prof.)
[1]Donetsk National Technical University, Ukraine
[2]Hamburg University of Technology, Germany
dosta@tu-harburg.de

ADVANCED DATA STORAGE OF DEM SIMULATIONS RESULTS

Usage of the discrete element method (DEM) is corresponded with the high computational demands. One of the main efforts is caused by a huge volume of the generated data, which should be stored for further analysis and post-processing. In this contribution an advanced methodology to store such data is proposed and implemented into a novel simulation system. Performed tests have shown high efficiency of the developed approach.

Keywords: DEM, data storage, granular material, data compression

Introduction

In order to perform simulation of the granular materials on the microscale the discrete element method has been introduced by Cundall and Strack [1]. The usage of this approach implies that the investigated system is represented as a set of individual particles. Each particle is described as a separate entity and in the every time step the Newtonian equations of motion are solved for each granule.

Due to the enormous high computational effort of the DEM, the number of particles of the modelled material is drastically decreased in compare to the original system. However, to obtain meaningful results and to perform realistic simulation, the usage of the DEM requires a modelling of large particle sets. The number of simulated particles can be varied between 103 and 107.

As results from the DEM simulations a set of time-dependent particle characteristics such as coordinates, velocities, accelerations, etc. are obtained. These characteristics should be saved onto the hard-drive and loaded from it for the further data post-processing. The large amount of particles (107), the small magnitudes of the simulation time step 1 μs and the huge set of time-dependent characteristics lead to the enormous volume of generated data. The typical simulation of the fluidized bed apparatus with 150000 particles for 1 second of real-time results in the data set about 60 GB.

Therefore, to solve an above described problem, the novel data storage approach has been developed. The new concept has been implemented into the multiscale simulation system of particulate materials [2] and its performance has been compared with already developed textual format.

Basic concept

During the DEM modelling a large volume of data is generated. For each particle in the system the following entries should be stored:

- *time-independent:* weight, density, modulus of elasticity, etc.
- *time-dependent:* coordinates, velocities, time progression of particle temperature, etc.

To store the characteristics which are not depends on the time the relatively small volume is needed. Contrary to this, the time-dependent data is continuously generated during simulation and the total amount of this data can reach enormous size. This can leads to the situations when the simulation results cannot be stored in the Random Access Memory (RAM) of modern computer and the data buffering onto a hard drive should be done.

Textual format

To store the DEM data two different file formats have been implemented into the multiscale simulation environment [2]. First file format has a relatively simplified structure that allows for a user to modify such files from any text editors or to generate them from own program. The new particles can be easily added or the already presented particles can be modified or removed without data corruption.

In the text format every particle is stored as a single file line. In the first part of line the time-independent data is placed. Afterwards, in the second part, the time-dependent data are consistently stored accordingly to the increased time. In Figure 1 the general organization of the data in the file is illustrated. The whole file represented as a continuous stream of bytes.

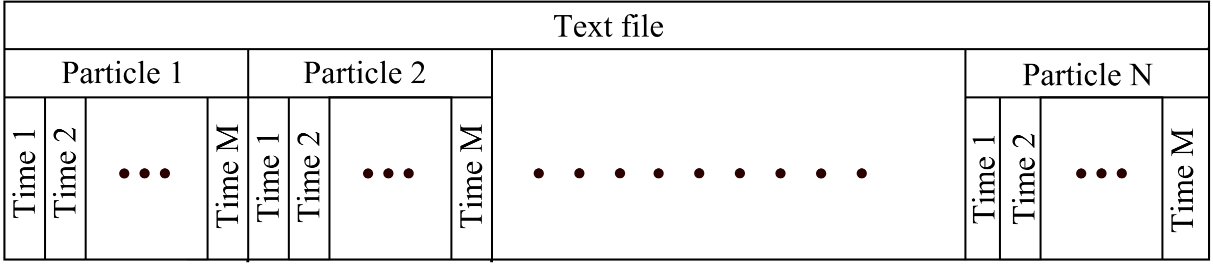

Figure 1 – Data layout in text file.

Despite the set of previously described benefits, this storage approach has a set of drawbacks which significantly limits its applicability. The first disadvantage is a large size of generated files. The textual format implies that any digit is stored as one-byte character. For example, instead of 2 bytes which are needed to store the value ‘12345’ the 5 bytes will be used.

Another disadvantage is caused due to the relatively slow data access. To store the entries for a different time points the various numbers of bytes can be used. That is why, it is impossible to determine the exact offset of specified time point in the file. For the visualization or for the post-processing of the modelling results, it is necessary to get particle properties for a specific time point.

However, required information is placed all over the file, and since it is impossible to determine the exact offset of this data, file must be read entirely, even if the information is needed only for the short time interval.

During the simulation it is necessary to store all data completely in the RAM. This limits the maximal size of the generated data and makes it equal to the size of the virtual address space. For instance, in the 32-bit architecture IA32 it is equal to 2 GB. Taking into account all mentioned above disadvantages, the new file format was developed to provide correct handling of large data sets.

Binary format

The novel storage approach is based on the binary data format, therefore the size of generated files are significantly smaller compare to the textual. The binary data is located in the file in different order in compare to the textual format. The time-dependent and time-independent data are stored in different files. All data relating to a single time point are stored as a one block. Due to this feature, to render a single frame it is sufficient to load only one single data block from the file and there is no need to load all data into the RAM.

In the file with time-independent particle properties supplementary information about data placement in time-dependent files are stored. This allows to find the data for specific time point quickly.

In the both file formats the linear interpolation is used. It means, when the value for specified time point can be predicted from already saved data, then the new data point will not be saved. Such concept reduces the size of the file, but in the case with binary files it increase the overhead. With the textual format there is no such problem, because it is loaded in the RAM entirely. But in the case of binary format the data for some particle can be absent in the specific time point. Hence, it is impossible to obtain required information. To avoid this problem the binary file is spitted into a set of data blocks.

The general storage concept which is used the case of binary format is shown in Figure 2. At the first time point of any data block all information about every particle is stored. To get required information for the specific time point, it is sufficient to load just one data block.

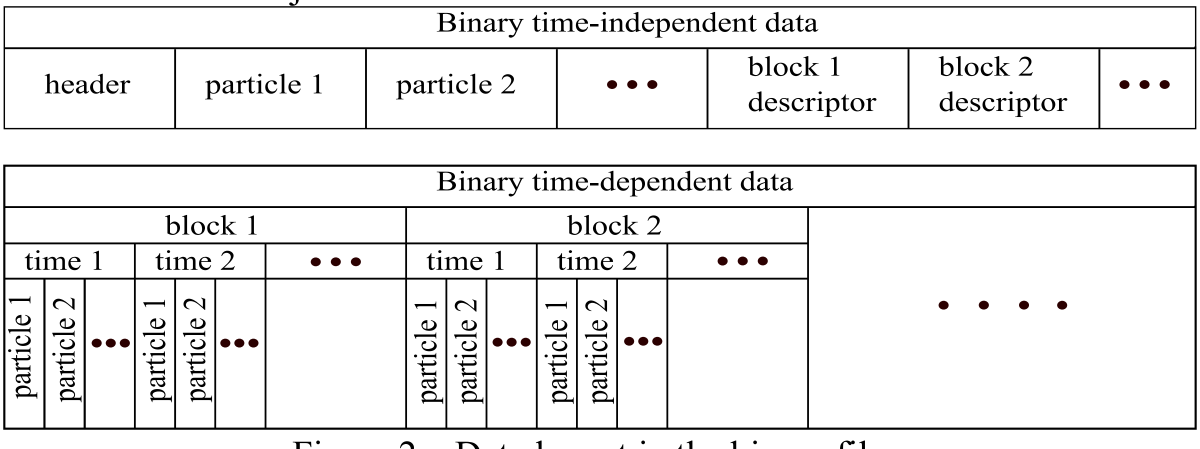

Figure 2 – Data layout in the binary files.

Implementation of binary format

In the current version of the simulation environment the operations related to the data storage are encapsulated into the kernel of the system. In order to create new particles, to delete existing and to perform other modification operations an interface to the kernel has been implemented. This interface does not depend on the storage format. If the binary file format is used, then the swapping of data block is done automatically.

In the RAM memory is always loaded at least 2 data blocks. This is necessary, to start an algorithm of linear interpolation. Every time, when the kernel tries to get the access to some time-dependent data, the low-level Input/Output (I/O) manager is called. This manager checks whether the required block is present in the RAM. If the required block was not loaded, then the current blocks are removed from the RAM and new data is loaded.

In the case of the writing operations, the I/O manager analyse the total volume of the data which has been already saved to the RAM. When the amount of data in the memory exceeds the maximum allowable size of the block then the data is saved to the hard-drive as a separate block.

The general representations of the algorithms to save and to load data are illustrated in Figure 3.

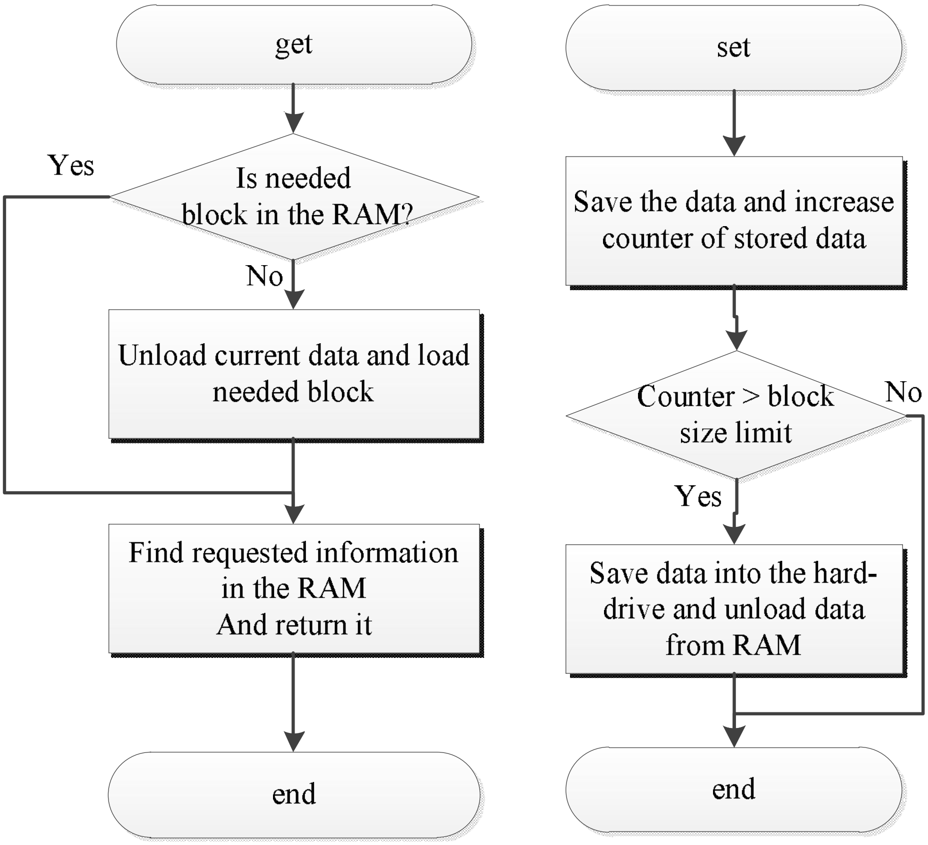
get
Yes
Is needed block in the RAM?
No
Unload current data and load needed block
Find requested information in the RAM And return it
end
set
Save the data and increase counter of stored data
Counter > block size limit
No
Yes
Save data into the hard-drive and unload data from RAM
end

Figure 3 – Algorithms for getting and setting data.

Performance tests

In order to compare the efficiencies of the text and the binary formats the performance tests were carried out. The results of these tests are depicted in Figure 4.

In Figure 4.a (left side) the time which is required to load the data file is shown. From the received results the conclusion can be drawn, that the binary files in most cases can be loaded more than 10 times faster as the textual. In Figure 4.b (right side) the dependency between the random access time and the file size is shown. With the binary format there is no need to store all the data in the RAM, hence the memory usage is significantly smaller. Such approach allows to simulate processes with the large number of particles on a long time interval.

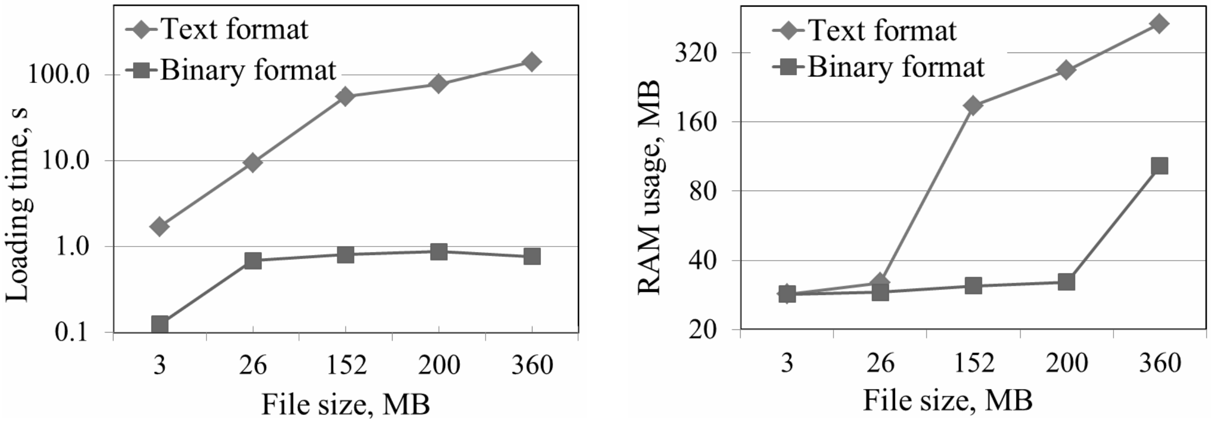

Figure 4 – Performance tests results. Part I.

However, in the case of binary format reading and writing data goes much slower, than in the textual format. In the case of binary data it takes about 100 ms to obtain the data which is placed in another block. This can be observed from the curves depicted in Figure 5.a.

The ratio between the size of binary and textual format is shown in Figure 5.b. From the illustrated results the conclusion can be drawn that the binary files significantly smaller than the textual files.

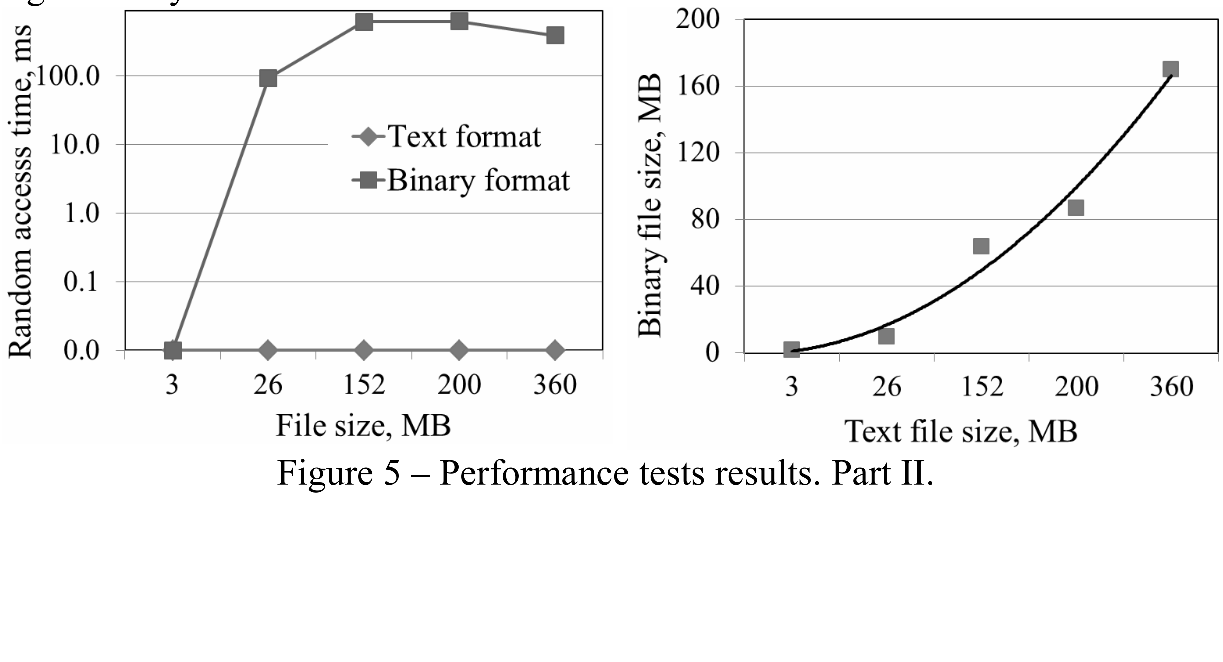

Figure 5 – Performance tests results. Part II.

Conclusions

The simulation results which are obtained from the discrete element simulations consist from a huge number of data entries. In order to perform the post-processing of results the effective data storage concept should be developed. In this contribution the novel approach to store the DEM data has been proposed and implemented into the multiscale simulation environment.

The novel concept is based on the usage of the binary data format, whereby the DEM results are divided between separate data blocks. These blocks are stored in separate files and continuously loaded into the RAM. Such approach significantly optimizes memory usage and increases program efficiency. As another advantage of the novel concept the minimization of the size of the data files can be underlined.

References

1. Cundall, P. A. & Strack, O. D. L., 1979. A discrete numerical model for granular assemblies. Geotechnique, pp. 47-65.

2. M. Dosta, S. Antonyuk, S. Heinrich. Multiscale simulation of fluidized bed granulation process, Chemical Engineering & Technology, 2012, Vol. 35, 1373-1380.

Надійшла до редакції 07.09.2012 р.Рецензент:д-р.тех.наук,проф.Святний В.А.

В. Янушкевич[1], М. Доста[2], С. Антонюк[2], Ш. Хайнрих[2], В.А. Святный[1]
[1]Донецкий национальный технический университет, Украина
[2]Гамбургский технический университет, Германия

Улучшенный формат для хранения результатов моделирования, полученных при использовании метода дискретных элементов. Использование метода дискретных элементов связано с большими требованиями, накладываемыми на вычислительную систему. Одно из требований вызвано высокими объемами генерируемых данных, которые должны сохраняться для последующего их анализа и обработки. В данной статье представлен новый формат хранения данных который был внедрен в моделирующую среду. Проведенные исследования показали высокую эффективность предложенного формата.

Ключевые слова: метод дискретных элементов, хранение данных, сыпучие материалы, сжатие данных

В. Янушкевіч[1], М. Доста[2], С. Антонюк[2], Ш. Хайнріх[2], В.А. Святний[1]
[1]Донецький національній технічний університет, Україна
[2]Гамбургський технічний університет, Німеччина

Покращений формат зберігання даних, які отримані при моделюванні за допомогою метода дискретних елементів. Використання методу дискретних елементів пов’язано зі значними вимогами до обчислювальної системи. Одна із значних вимог пов’язана з великим обсягом даних, які повинні зберігатись для їх подальшого аналізу та обробки. У представленій роботі був розроблений новий формат зберігання даних, який був реалізований у моделююче середовище. Проведені дослідження посвідчили високу ефективність запропонованого формату.

Ключові слова: метод дискретних елементів, зберігання даних, сипки матеріали, компримування даних