Numerical simulation of the ion focusing process in a dust impact time of flight mass spectrometer

I V Piyakov[1], D V Rodin[1], M A Rodina[1] and A M Telegin[1]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

Abstract. The article presents the description of a design of dust-impact time of flight mass-spectrometer. A method for calculating the trajectory of a charged particle on a triangular grid is considered. The implementation of the calculation algorithm for the personal computer, the GPU accelerator and the supercomputer is described. The main differences in implementations, comparison of performance and accuracy of calculation are given. The advantages and disadvantages of each implementation are considered.

1. Introduction

Property analysis of the micrometeoroid and debris particles *in-situ* is impossible without the use of appropriate recording equipment. Thus, for the analysis of the chemical composition of particles, dust-impact time-of-flight mass spectrometers are used. The advantage of this type of analyzers is high speed, the ability to register single events, small size and power consumption. A unique feature of dust-impact time-of-flight mass spectrometers is ionization by a high-velocity collision of a particle-striker and target, which occurs randomly at an arbitrary point of the target. An increase in the probability of ionization in this way is achieved by increasing the active area of the target, which leads to the need for additional spatial focusing of the ions obtained using parabolic reflectors or electrostatic reflectors with a special form of electric field. The plasma produced as a result of a high-velocity collision has a high energy spread, which necessitates the use of electrostatic ion mirrors that provide a temporary focusing of ion packets.

The principle of operation of time-of-flight devices is based on measuring the time of flight of particles with the known energy value from the ion source to the receiver. In an ideal case, the time of flight depends only on the mass of the ion. The development of real devices is based on the determination of design parameters that ensure maximum independence of the time of flight from the place of collision to the ion receiver from the ion initial energy. For this purpose, a one-dimensional field structure [1] is calculated that provides the required temporal focusing and the requirements for design elements are developed by numerical simulation. It allows to form field structure close to the desired in the maximum volume of the device structure. Verification of each of the design options is carried out by simulating the trajectories of model ion packets, with the subsequent calculation of the resolving power. In this paper we consider the software that implements a massively parallel algorithm for calculating particle trajectories on a triangular grid, which provides the possibility of modeling curvilinear structural elements.

2. Formulation of the problem

The structure diagram of the device under consideration is shown in Figure 1. A detailed description of the device is given in [2].

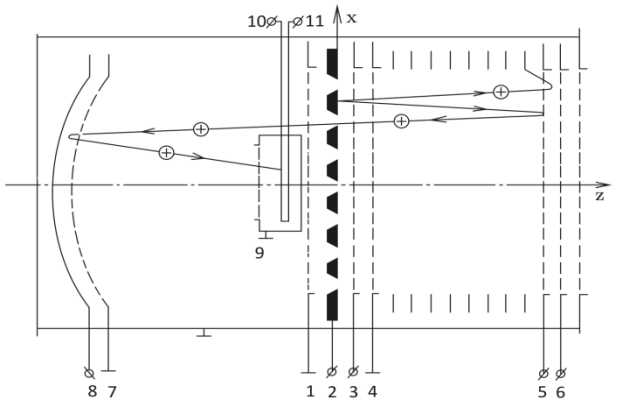

Figure 1. Dust-impact mass spectrometer with a parabolic reflector (2 – target; 3 – accelerating grid; 4 – retarding grid, grounded; 5, 6 – mesh of the ion mirror, 1 – grounded grid; 7, 8 – parabolic reflector, 10, 11 – ion receiver; 9 – receiver casing, grounded).

The device works as follows. When the particle strikes the target, the particle substance is ionized, accelerated by an electric field and enters the ion mirror, bounded by grids 4, 5, 6, where it is reflected and, passing back through the target (by this moment the accelerating potential is turned off), flies through the fieldless space between the meshes 1 and 8, when ion packet is reflected in a parabolic reflector and enters the receiver input 10, an ion pulse is formed at its output 11.

A special feature of the reflector is the shape of a paraboloid with a focus in the ion receiver. This facilitates additional spatial focusing of ion beams. Thus, the loss of ions decreases, leading to increase of the device sensitivity. The energy focusing of ions is performed by the ion mirror.

The main parameters of the time-of-flight mass spectrometer are its resolving power and sensitivity. At the same time, their values depend on many factors:

– design parameters of a time-of-flight mass spectrometer;
– the energy spread of ions formed as a result of high-velocity collision;
– direction of the initial ion velocity vectors;
– coordinate of the particle impact on the target;
– chemical composition of particles

To determine the influence of these factors on the characteristics we are interested in, we need to calculate the trajectories of ions in the device path. For this purpose, for the given mass spectrometer structure, the following parameters are specified: the distances between the grids and the accelerating potential, the target and ring dimensions, the curvature of the parabolic reflector, the diameter of the central target hole, the parameters of the field-setting rings. The grid with the distribution of electrostatic potential inside the device path that is used on the stage of modelling the operation of the device is preliminary calculated.

3. The calculation algorithm

Numerical simulation of the motion of charged particles can be implemented both with the use of a grid of field values obtained by the method of finite differences, and with the use of nodal values obtained by the finite element method. The advantage of the first method is the clarity, simplicity of implementation of the cell search functions and cubic interpolation, ease of mesh joining when calculating in several areas. However, an essential disadvantage of this type of grids is the impossibility of locating nodes on curvilinear surfaces. The finite element method allows you to divide arbitrary regions into separate elements (in certain cases, triangles), which allows to create an exact reconstruction of the shape of the regions of interest. At the same time, the use of finite element grids requires the implementation of additional functions, for example, the search for the triangle to which the point belongs, as well as the more complex than in the finite difference method data structures for storing grid information, which is a graph with a description of nodes and triangles.

Based on the data on the interaction coordinates and velocity, an ion packet with a known value of the energy spread is formed [3]. For each ion in the ion packet with a given time step, the increment of coordinates is calculated. After each iteration, a check is made for impact with the structural element, escape and detection. Times of flights of the detected ions are conserved. To estimate the design characteristics for each mass, it is required to calculate the number of ions, to estimate the collection

coefficient, the mean time of flight, the standard deviation and the resolution of the instrument on two sigma level.

Step by step algorithm of the program on a triangular grid can be described as follows.

1. Download the data about the calculated grid with the field from the file.
2. File parsing - linking nodes, triangles, regions and field values.
3. Creating a Mesh object with a description of the relationships between nodes and triangles.
4. Formation of a model ion packet, starting from the given coordinates and the speed of interaction.
5. Finding the current triangle for each ion.
6. Interpolation of the field at the location of each ion.
7. Calculation of the displacement of each ion in time d*t*.
8. Checking the location of the ion in the current triangle.
9. Recursive search for a new triangle.
10. Check for escape / detection - return to point 6.

In general, the implementation structure is similar to the one described in [4]. So in step 4 the Box-Muller algorithm is similarly used for the formation of a packet with a Maxwellian velocity distribution. The ion displacement can also be calculated by the trapezoidal rule or by the Runge-Kutta method. The difference between this implementation is the use of the bilinear interpolation method for a triangular grid and the need to check whether the coordinates of the particle belong to the current triangle. Both problems are solved using the known approaches with small modifications. To accelerate the interpolation operation, each instance of the triangle class stores the previously calculated interpolation coefficients *s*1, *t*1, *s*2, *t*2, *s*3, *t*3, *A*, coeff and to speed up the search for a new triangle to which the particle coordinates belong. The same instance stores references to neighbouring triangles, which allows to replace the search for all grid triangles by a recursive search from the current triangle.

4. Features of the algorithm implementation for calculating the trajectories for CPU, GPU, HPC

The implementation for the CPU is written in C# and contains four main classes for implementing trajectory calculation. The Point class is used to store graph nodes, contains the node index, coordinate information, field values, and the number of triangles to which this node belongs. The Triangle class contains the index of the triangle, three references to the nodes-vertices of this triangle, information about the domain to which the triangle belongs, three references to adjacent triangles, pre-calculated coefficients for interpolation, and methods for bilinear interpolation of the field and the search for a new triangle when the point doesn't belong to the current triangle. The Mesh class contains arrays of nodes and triangles, as well as the methods for finding the nearest point, triangle, reading and writing files. The Particle class contains information about the mass and charge of a particle, its speed, coordinates, the current triangle, and also a method that implements an iterative calculation of the trajectory on a triangular grid.

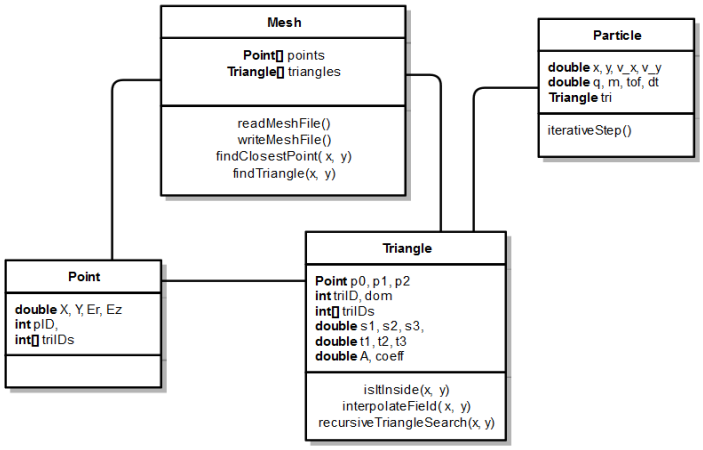

Figure 2. Class structure for storing information about the calculated grid, PC implementation in C#.

The use of a graphic accelerator imposes severe restrictions on the implementation of software, leading to the need, if possible, to represent all information in the form of arrays. Thus, the implementation for the GPU contains the following arrays: triangleNeighbours, trianglePoints - arrays of vector format uint4 for storing information about neighbouring triangles and triangle vertices. Arrays s1_t1, s2_t2, s3_t3, A_coeff of vector format double2 are used for storing information about the coefficients for interpolating the field. Arrays pointsCoordinates and pointsFields of vector format double2 contain information about grid nodes. Arrays particlesCoordinates and particlesSpeeds vector format double2 are used for storing information about the coordinates and velocities of particles. An array of particlesTimes of double1 format is used for storing the current time of the particle. An array of particlesTriangles of the uint1 format contains the index of the current triangle of the particle. The index in the array is used to refer to nodes, triangles, and particles instead of explicit indexes. These arrays are created in the host memory and populated with data from the CPU implementation, after those arrays are created in the GPU memory and data from the host arrays is copied to the GPU. Then, the texture and surface links are bound to the GPU arrays, which allows using the operations of texture samples that implement hardware caching. It should be noted that linking texture references to arrays of vector double types is impossible, so when copying data from the CPU to the GPU, a bit conversion of one double number to a pair of int numbers takes place. After the texture sample, a reverse bit conversion occurs.

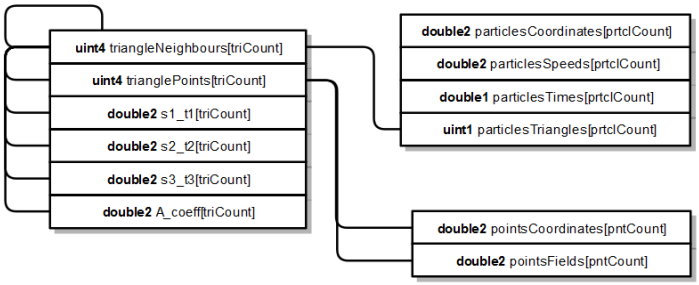

Figure 3. The structure of arrays for storing information about the calculated grid, GPU implementation in the CUDA language.

Implementation for the supercomputer Sergey Korolev differs both from the version for the PC, and from the version for the GPU. Since this implementation is written for the MPICC compiler, and the version of the C language that does not support OOP is used. The data about triangles and their interrelationships and the relations with nodes are stored in triangleNeighbours, trianglePoints of int format, data on coefficients for interpolation in triangleCoeff array, double format , the coordinates of the nodes and field values in them - in the pointsCoordinatesAndFields array of the double format, the particle data in the particles array of the double format. This implementation does not require additional type conversions when accessing memory, all arrays also lack the index of nodes, triangles and particles, instead of them an index in the array is used.

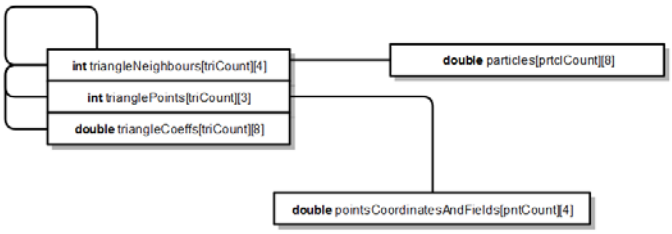

Figure 4. Structure of arrays for storing information about the calculated grid, HPC implementation in C language.

5. Results of calculation and performance comparison

To visualize the modelling process and evaluate the performance of the design, a visualization module was written to depict the movement of ion packets in the volume of the structure. Figure 5 shows several combined frames, the focusing of ion packets over time is clearly visible - two masses are well separated at the receiver plane.

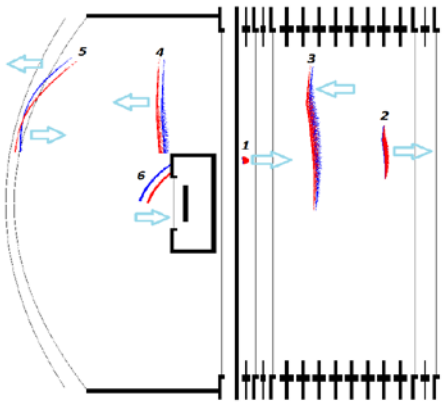

Figure 5. Visualization of the trace of ion packets. A packet of two masses is formed at point 1, accelerates and flies into the ion mirror 2, reflects and flies back to the target 3, flies through the target, part of the ions settles on the receiver shell 4, is reflected in the parabolic reflector 5 and flies into the receiver 6.

The results of modelling the construction of a time-of-flight mass spectrometer with a linear ion mirror and a parabolic reflector are presented in Figures 6 and 7. Calculations were made for 16382 particles without taking into account the transparency coefficients of the grids and the target.

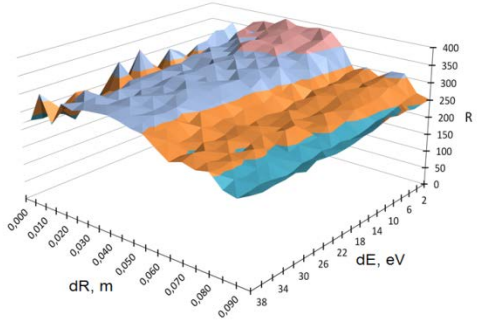

Figure 6. Dependence of the resolving power of the instrument on the collision coordinate and the energy spread.

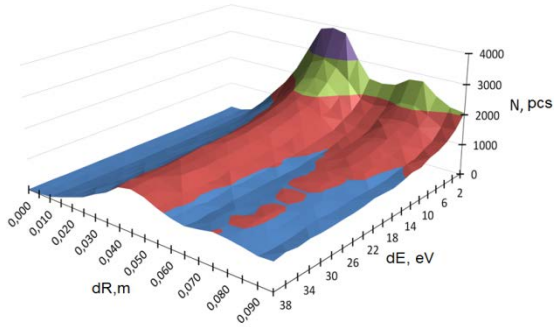

Figure 7. Dependence of the number of received ions on the collision coordinate and energy spread.

The resolving power has a local maximum at zero radial offset. The ion collection coefficient has a maximum of 35 mm and 75 mm radial displacement with minimal energy spread.

Comparison of implementations for a personal computer, accelerator GPU and supercomputer was performed for a different number of particles. The comparison did not reveal any differences in the accuracy of calculations, which is generally explained by the fact that all calculation modules are implemented using variables of the double type. HPC implementation was launched on four processors.

The results of measurements of the execution time of 10,000 iterations are given in Table 1.

6. Conclusion

Using the above mentioned software, the authors simulated various designs of the dust-mass spectrometer. Simulation of a design with a linear mirror and a parabolic reflector showed a relatively low resolving power of about 250-300 units regardless of the energy spread and the collision coordinates. However, for a combination of factors, such as the achieved resolution, ion collection coefficient, ease of adjustment, this design is more preferable than others. Thus, the ion collection coefficient drops sharply in designs with a flat reflector. In constructions with a nonlinear ion mirror, it is not possible to achieve independence of the resolving power from the collision coordinate.

The presented software provides the possibility of modelling structures with axial symmetry, containing curvilinear electrodes and non-uniform electric fields. In addition to modelling point single-shot sources, such as laser or high-speed impact ion sources, it is also possible to simulate the trajectories of charged particles in various instruments forming ion fluxes, such as gas mass spectrometers with continuous ionization or gas-discharge sources of off-electrode plasma [4, 5].

In contrast to using rectangular meshes [6, 7], the implementation of the accelerator for the GPU (2,880 cores per 1020 MHz) did not show a significant increase in performance, which is explained by a higher number of branches, during the execution of the algorithm that implements the calculation on a triangular grid. However, even such a modest increase in the speed of execution can be useful in solving design optimization problems that require multiple tracing of ion packets, with a total number of estimated passes reaching several tens of thousands, and a total execution time of several days or weeks.

The implementation for HPC, proved to be more efficient with the same number of processors, which can be explained by greater memory performance of the binary code produced by the MPICC compiler. Due to the fact that this task is parallel to the initial data, performance can be increased by increasing the number of compute nodes.

Table 1. Results of measuring the execution time of 10,000 trajectory iterations for implementation on PC, GPU, HPC.

	PC parallel. (*4CPU*, 8 threads)	*PC Single thread*	GPU Host calls	GPU Device calls	HPC Sergey Korolev
16384	7с	24с	6с	6с	6с
65536	29с	1м 34с	23с	23с	18с
262144	1м 53с	6м 19с	1м29с	1м29с	1м19с

7. References

[1] Semkin N D, Piyakov I V, Rodin D V and Pomel'nikov R A 2012 Analytical method for computing the electrostatic field distribution in the deflector of the time-of-flight mass spectrometer *Technical Physics. The Russian Journal of Applied Physics* **57(10**) 1400-1405 DOI: 10.1134/S106378421200192

[2] Semkin N D, Piyakov I V, Rodin D V and Pomelnikov R A 2012 An onboard dust-and-dust mass spectrometer for studying the elemental composition of micrometeoroids *Nauchnoe priborostroenie* **22(3**) 13-20

[3] Hornung K, Malama Y G and Kestenboim K S 2000 Impact vaporization and ionization of cosmic dust particles *Astrophysics and Space Science* **274**(1-2) 355-363

[4] Podlipnov V V, Kolpakov V A and Kazanskiy N L 2016 Etching of silicon dioxide in off electrode plasma using a chrome mask *Computer Optics* **40(6)** 830-836 DOI: 10.18287/2412-6179-2016-40-6-830-836

[5] Kazanskiy N L, Kolpakov V A and Podlipnov V V 2014 Gas discharge devices generating the directed fluxes of off-electrode plasma *Vacuum* **101** 291-297

[6] Piyakov A V, Rodin D V, Rodina M A andTelegin A M 2017 Numerical simulation of motion of dust particles in an accelerator path *CEUR Workshop Proceedings* **1902** 55-61

[7] Semkin N D, Rodin D V and Piyakov I V 2013 A method of calculating the potentials of the field-setting elements of a nonlinear ion mirror *Nauchnoe priborostroenie* **23(3**) 69-75