Implementation of the Kinetic Plasma Code with Locally Recursive non-Locally Asynchronous Algorithms

To cite this article: A Yu Perepelkina et al 2014 J. Phys.: Conf. Ser. 510 012042

View the article online for updates and enhancements.
Implementation of the Kinetic Plasma Code with
Locally Recursive non-Locally Asynchronous
Algorithms

A Yu Perepelkina¹, V D Levchenko¹,² and I A Goryachev¹
¹ Keldysh Institute of Applied Mathematics, Moscow, Russia
² National Research Nuclear University "MEPhI", Moscow, Russia
E-mail: mogmi@narod.ru

Abstract.
Numerical simulation is presently considered impractical for several relevant plasma kinetics
problems due to limitations of computer hardware even with the use of supercomputers. To
overcome the existing limitations it is suggested to develop algorithms which would effectively
utilize the computer memory subsystem hierarchy to optimize the dependency graph traversal
rules. The ideas for general cases of numerical simulation and implementation of such algorithms
to particle-in-cell code is discussed in the paper. This approach enables the simulation
of previously unaccessible for modeling problems and the execution of series of numerical
experiments in reasonable time. The latter is demonstrated on a multiscale problem of the
development of filamentation instability in laser interaction with overdense plasma. One variant
of the simulation with parameters typical for simulations on supercomputers is performed with
the use of one cluster node. The series of such experiments revealed the dependency of energy
loss on incoming laser pulse amplitude to be nonmonotonic and reach over 4%, an interesting
result for research of fast ignition concept.

1. Introduction
The exponential growth in computational technologies during last decades open up new
possibilities in solving problems relevant for contemporary physics. First of all it includes
multiscale problems, the numerical solution of which was unachievable for a long time.

For this concept one of the most indicative fields is plasma kinetics simulation. Practical
simulation problems require large amount of mesh cells and number of particles to satisfy the
given accuracy criteria.

While the computational systems and simulation algorithms have evolved considerably since
the invention of numerical schemes for plasma kinetics simulation, there remain problems which
are still considered impossible to solve even with the use of massively parallel computations.

In this paper we present the CFHall code which is aimed at significantly multiscale problems
in plasma kinetics simulations [1, 2].

For example, numerical modeling is seen as a most reliable tool to aid in the research of
the causes of electron transport to anode in Hall thruster plasma. For this problem state-of-art
codes perform 2D3V (two dimensional in space, three dimensional in velocity) simulation with
the use of various approximations [3, 4]. These approximations, as well as a 2D3V geometry,
may alter the physics of the studied processes, but the simulation without them is still thought
to be impossible on modern equipment.

The other problem which was considered in the development of the presented code is the
simulation of filamentation instability in laser interaction with plasma. In this case the simulation
domain should be large enough to cover the laser pulse envelope and the plasma layer, at the same
time each filament appearing in the instability should contain several mesh steps. Numerical
simulation has aided the research of the problem ever since it was carried out in 2D3V geometry
[5]. But since the nonlinear plasma processes may be misrepresented when confined to two
dimensions, fully 3D simulations are needed for more accurate results.

Such relevant problems of plasma physics require processing of large amounts of data. In the
Hall thruster problem the fields are usually modeled by Poisson equation with implicit schemes.
The presence of memory subsystem hierarchy leads to degradation of the absolute efficiency of
calculations with the increase of the amounts of processed data due to the nonlocality of data
access. It also results in inefficient parallelization.

Recent 3D3V numerical studies utilize the mesh with no more than $1.5 \cdot 10^6$ cells [6, 7]. The use
of supercomputers may raise this number to $10^8$ [8]. Nevertheless with such grids either the mesh
step approaches the size of the filament, or the simulation domain is not enough to simulate laser
pulses with Gaussian envelope. The acceleration of calculations achieved by massively parallel
computations is impeded by the necessity of a large number of synchronizations.

One of the methods to improve the situation is to use more efficient computational algorithms
which could both take advantage of memory subsystem hierarchy for computations on systems
with shared memory, and minimize the synchronization events in parallel codes. In this work
we show how this can be achieved in a plasma kinetics code with the use of Locally Recursive
non-locally Asynchronous (LRnLA) [9] algorithms.

The algorithms are universal in the sense that one may use them for any numerical scheme
with local stencil. The implementations with the use of algorithm family similar to the one
described in this paper already exist for solution of Maxwell equations in optics simulation
[10], solution of elasticity equations in seismic media [11], gas dynamics simulation with RKDG
(Runge-Kutta Discontinuous Galerkin) method [12].

The principal ideas for the effective algorithms are locality and asynchrony. The greater the
locality of the algorithm is, the closer the subsequently processed data is placed in computer
memory. It also means that if some data is processed at some time, it will sooner be processed
again. The better is the asynchrony of the algorithm, the larger portions of data may be processed
independently before the message passing between their processors becomes necessary.

In this paper we present the basics of the LRnLA algorithms that are used in CFHall code,
show the specifics of their application to plasma kinetics calculations, and demonstrate the
efficiency achieved by such approach on the example of a problem of filamentation instability
simulation.

2. Kinetic model
As a mathematical model of plasma the Vlasov-Maxwell system of equations is used. We use the
dimensionless units. Reciprocal electron Langmuir frequency $\omega_p$ is used for time units, electron
masses $m_e$ for mass, electron charges $e$ for charge, light speed $c$ for velocities. From these the
units for distances $c/\omega_p$ and for fields $m_e \omega_p c/e$ follow. This unit are scalable by electron density,
so the physical quantities the simulation value correspond to are also scalable in a definite range.

Electromagnetic field evolution is calculated in accordance with the FDTD (Finite Difference
Time Domain) numerical scheme [13]. PIC (Particle-in-Cell) method is used for plasma
simulation [14, 15] with second-order form-factor. Superparticle relativistic acceleration is
determined by the Boris scheme.
The field evolution time step is limited by the Courant condition:

\[ dt_{FLD} \leq \frac{dx}{\sqrt{3c}}. \]  

(1)

Time step for particle motion should be small enough to resolve the Langmuir plasma frequency:

\[ dt_{PIC} \leq \frac{1}{\omega_e}. \]  

(2)

These two conditions may be of different scale. That is why it is possible to use 2 different time steps in a simulation.

We shall first demonstrate how the algorithms are constructed for the 3 dimensional FDTD scheme for Maxwell equations with the second order of approximation, which involves only the electromagnetic field evolution, without calculations for plasma particles. The two equations used explicitly in calculation are:

\[ \frac{\partial \vec{B}}{\partial t} = -\nabla \times \vec{E}, \quad \frac{\partial \vec{E}}{\partial t} = \nabla \times \vec{B} - \vec{j}. \]  

(3)

Electric and magnetic field components are shifted against each other both in time and space, according to a Yee cell [13]. This scheme is simple and can be considered independently, and the algorithms built on it will comprise the basis for the more complicated codes.

When proceeding to the algorithms for PIC method it is essential to note the width of its stencil. According to the method, the distribution function of plasma is approximated by a model function, and the integration of Vlasov equation then takes the form of Newtonian equations.

For the construction of this model distribution we use the family of functions \( \Lambda^\ell \), parametrized by \( \ell \).

\[ \Lambda^\ell(x - x') = \int \Lambda^{\ell-1}(x - x'') \Lambda^0(x'' - x') dx'', \quad \Lambda^0(x) = \begin{cases} 1 & |x| < 0.5, \\ 0 & |x| > 0.5 \end{cases} \]  

(4)

\[ \Lambda^\ell(\vec{r}) = \Lambda^\ell(x) \Lambda^\ell(y) \Lambda^\ell(z). \]  

(5)

With \( \ell = 0 \) \( \Lambda^\ell \) is a rectangle function, with \( \ell = 1 \) it is a tent function, with \( \ell = 2 \) it is comprised of three stitched parabola pieces. To calculate the charge density in the point \( \vec{r}_i \) knowing the positions of particles \( \vec{r}_j(t) \) we use

\[ \rho(\vec{r}_i) = \sum_{\alpha} e_\alpha \sum_j \Lambda^{\ell+1}(\vec{r}_j(t) - \vec{r}_i), \]  

(6)

and the current densities are found from the continuity equation. With higher \( \ell \) the order of approximation increases, and the stencil becomes wider. Since the FDTD scheme is of second order, it is reasonable to use the second order of approximation for PIC method. It also gives the computational advantage over the majority of PIC codes, since the required accuracy can be achieved with lower number of particles per cell. With the second order of approximation the stencil covers 3 cells in each direction. Since the particles can travel between cells during their motion, the actual size becomes equal to 4 if we restrict its movements by one cell radius.

3. Methods of calculation optimization

3.1. Data storage

Firstly we shall show how we can take advantage of the idea of locality when choosing the data structures for the simulation.
The finite difference schemes common property is the fact that the cross-shaped stencil is usually involved. Let us note that when we use the 3D array in the calculation the data in the array is stored in a single line in computer memory. If we have an array of the size \( N_x \times N_y \times N_z \), and attempt the calculation for the element with \((i_x, i_y, i_z)\) indices in it (which means that it is the element with index \( i = i_x + N_x \cdot i_y + N_x \cdot N_y \cdot i_z \)) if we use a scheme with a cross-shaped stencil we need its closest neighbors along the three axes. These neighbor cells on the \( x \) axis are located 1 cell away from the point in question, but neighbor cells along \( y \) and \( z \) directions are located \( N_x \) and \( N_x \cdot N_y \) cells away correspondingly. It means that for large meshes each calculation is not local in itself, and such data structures lowers the effectiveness of calculations.

One way to avoid this is to store the data in Morton order Z-curve [16] (see fig. 1). While there may be other more efficient solutions for such problems, Z-curve is also easy to implement with the use of recursive templates in programming languages. In the current work it is utilized in a \textsc{cubeLR} structure. Since this works only when the dimensions of the array are equal to each other and may be represented as \( 2^n \), where \( n \) is some integer number, when array dimensions differ, the simulation domain is subdivided into cubical blocks. The relations between blocks are determined by a structure of ordered lists \textsc{nLAnet}. The further discussion is related to cubical grid without the loss of generality.

![Figure 1](image)

**Figure 1.** Mapping of a 2D data on Z-order curve and common array

### 3.2. LRnLA algorithms

Z-curve may be seen as a recursive traversal rule in the 3D data space. Let us now consider a 4D operations-data space. In this space one can imagine a dependency graph of the calculations for the problem. In fig. 2 the dependency graph for 1D projection of the problem is illustrated. The arrows show the data exchanges, the operations are represented by points. It is convenient to place it on a 2D space. Vertical axis corresponds to operations, and can be thought of as a time axis, horizontal axis corresponds to data, and can be associated with coordinates. The first layers of points along the horizontal axis show data initialization operations, the following ones show the calculations on some time step.

LRnLA algorithms deal with optimizing dependency graph traversal rules. To understand how the algorithms of the LRnLA family are constructed one needs to be able to visualize the algorithms as shapes in the operation-data space.

Algorithm is a rule which determines the particular order in which the given calculations in the dependency graph are processed. Let us enclose several (or all) points of the dependency graph in a 4D polytope (Fig. 2). This will comprise some “calculation block”. This polytope
Figure 2. An example of a dependency graph for FDTD scheme, projected on two dimensional plane. The stencil of a numerical scheme (left). A polytope covers several operations (center). A polytope is subdivided into smaller ones so that the dependencies between calculation blocks is unilateral (right).

may be divided into smaller ones, each of which may (or may not) cover some points of the graph, and the data dependencies (represented by arrows) between blocks are unilateral. These dependencies determine to some extent the order in which these calculation blocks are carried out. The block which are not connected by arrows (asynchronous blocks) may be calculated either subsequently or in parallel. The resulting polytope may also be divided further, until they cover only 1 calculation. This way the shape of the polytope, along with the rule it is divided into smaller shapes with unilateral dependencies, visually illustrates and determines a calculation algorithm.

For further illustration let us explain how this concept works for conventional traversal rules. The conventional algorithm deals with the problem layer by layer, calculating all operation for one given time step before advancing to the next one. In terms of a polytope division illustration one can cover the whole dependency graph by a rectangular polytope, then subdivide it into layers. Each layer is represented by a rectangle with height equal to one time step in the fourth dimension and cover all simulation domain in other dimensions. This layers comprise the calculation blocks that are processed subsequently. This algorithm is not effective in terms of both locality and asynchrony. With the increase of mesh size the locality of such algorithm decreases significantly, and with the use of parallel calculations the data exchanges between processors is required on each time step.

One of the solutions provided by the LRnLA algorithms family is illustrated on fig. 2. The triangle (cone in greater dimensions) is subdivided recursively into triangle and diamond shapes. After the calculation in triangles is processed, the data is transfered to calculation in the diamond. The algorithm is easiest to implement if the shapes of the polytopes involved are similar, so the division becomes recursive.

3.3. ConeFold

Since the polytope subdivision may produce new shapes where the rules of subdivision also differ, the algorithm types for multidimensional problems may become large and difficult to program. To cover the whole operation space while using only one shape and rule of its division the algorithm by the name of “ConeFold” (CF) is used in the presented plasma kinetics simulation code. The shape is illustrated in figs. 3. Its division is \((d + 1)\)-binary, where \(d\) is the dimension of the simulation domain. We can associate a number, let it be called “rank”, with the CF size. One CF with rank \(R\) divides into \(2^{d+1}\) CFs with rank \(R - 1\).

In the implementation the rank \(R\) CF algorithm corresponds to some function call, which conducts \(2^{d+1}\) function calls which correspond to CFs of rank \(R - 1\) in the order determined by the data dependencies. CF of rank zero performs calculation according to the numerical scheme.
Figure 3. ConeFold algorithm shape and decomposition. Projection for 2D and 1D simulation in one data cell. For FDTD scheme the cell is chosen as a Yee cell, so that it contains one complete set of electromagnetic field components: 3 electric field components and 3 magnetic field components. The minimal cell for the algorithm may also contain several Yee cells in case the stencil size of the scheme is larger.

The CF which has all the data of the given problem in its projection on the data space has rank \( R = \text{MaxRank} \). Its height corresponds to \( 2^\text{MaxRank} \) time steps. As illustrated by the slant sides of the ConeFold shape, after one such algorithm is processed in the domain, the data cells involved are situated on the different time steps. To bring all the data to the single time evolution moment by covering the 4D rectangular domain \( 2^d \) CFs should be processed (fig. 4).

3.4. Algorithms for boundaries

The first complication arises when we note that calculations may differ depending on the cell. The boundary cells are processed differently since boundary conditions applies. One should take note that different decomposition rules apply to shapes corresponding to the algorithms which intersect the boundary. Figure 5 shows how the decomposition works in 1D simulation (2D operation-data space). The first ConeFold of the maximal rank intersects the right boundary of the domain. Let us denote this algorithm with \( X \) symbol. It divides into 2 CFs of the same type \( (X) \), one that is situated completely inside the domain (denoted by \( D \) symbol), and one that is outside and thus should not be processed. This way the right boundary CF \( (X) \) calls 3 CFs. The \( D \) CF consists of 4 CFs with \( D \) symbol. On zero rank \( X \) CF performs the calculation with the account of boundary conditions. The left boundary is considered likewise, and the boundary CF is denoted by \( I \).

Figure 4. Coverage of the rectangular simulation domain by ConeFolds of the maximal rank, 1D simulation case

In a 2D case 9 cases should be considered corresponding to the internal calculation, calculation on each edge and each corner. The intersection of algorithm shapes with boundaries define the rules of their decomposition.
3.5. Vectorization
It is not necessary to use the 4D LRnLA decomposition of the dependency graph for 3D simulation domains. One can use the conventional data storage and algorithms along one axis, and locally recursive decomposition along the others. In this case the calculation can be accelerated further with the use of SSE/AVX vectorization along this isolated axis in the following way.

The component values along z axis are stored in a vectorized array. Array elements are stored in vectors with \( N_v \) elements, where \( N_v=4 \) if SSE with float data type or AVX with double data type is used or \( N_v=8 \) for AVX with float. Operations on such vectors take the same amount of time as operations on scalars (double or float). Therefore it is possible make the calculation up to 8 times faster. The actual acceleration is slightly less due to costs of vector formation.

3.6. Other algorithms based on ConeFold
Among the advantages of ConeFold algorithm is its versatility in a sense that it can be adapted to various numerical schemes, local stencil being the only limitation, and to various computer architectures.

The distinctive feature of shared memory systems is the bottleneck in memory subsystem hierarchy. To overcome the difficulty the algorithms locality may be further improved by using a tower-like polytope “ConeTorre” (see fig. 6), which consists of several CFs stacked onto each other in operations (time) dimension. For multicore processors it is convenient to use a polytope “TorreFold” that has a CF shape and subdivides into several ConeTorres, each of which can be processed by separate core.

For NUMA architectures it is more profitable to stack ConeFolds not in a single tower, but with an additional shift, so that all shifted CFs are asynchronous, and it constitutes the “ChessFold” algorithm.

The “ConeTorre” and “ChessFold” algorithms can be combined for the implementations on systems with several levels of parallelization, such as clusters, up to supercomputers.
3.7. Specifics of algorithms for PIC simulation

As it was mentioned earlier, the stencil of PIC method is larger than the one in FDTD calculations, and this leads to the necessity of several modifications of the basic ConeFold algorithm. Also, as previously stated, for CFHall code we choose different time steps for field and plasma evolution. Here is how we modify the algorithm by taking these two points into consideration.

The subdivision of the ConeFold of maximal rank continues recursively to rank \( R = \text{PicRank} + 1 \) (instead of continuing to zero rank). \( \text{PicRank} \) here denotes a chosen number which controls the \( dt_{\text{PIC}} / dt_{\text{FLD}} \) ratio. After the subsequent subdivision we get \( 2^{d+1} \) CFs. The \( 2^d \) CFs of \( R = \text{PicRank} \) from the top layer continue the subdivision until zero rank, and on the zero rank the calculation for field evolution according to FDTD scheme is performed on one data cell. The lower layer of CFs with \( R = \text{PicRank} \) are subdivided according to the same rule as described above for the CF with \( R = \text{PicRank} + 1 \), until the rank reaches \( R = \text{FFrank} \). This number is determined by the form-factor of PIC method and equals 2 in the current code. This CF performs the calculation for the PIC method.

The concept is illustrated on fig. 7 for \( \text{PicRank} = 3 \). ConeFold with rank \( R = \text{FFrank} \) calculates the forces applied to particles from the field values in cells, performs the acceleration and movement of particles and calculates the contribution to charge current densities through cell faces from this movement. The current densities established after this movement are used in \( 2^{\text{PicRank} + 1} - 2^{\text{FFrank}} \) calculations of field components. For \( \text{PicRank} = 3 \) the time step ratio becomes \( dt_{\text{PIC}} / dt_{\text{FLD}} = 12 \).

![Figure 7. Subdivision of CF with rank \( R = \text{PicRank} + 1 \). Any algorithm with larger rank consists of some number of such shapes](image)

The optimization methods applied in CFHall can be summarized as follows. The data is stored in a 2D locally-recursive (Z-order) array associated with \( x \)-\( y \) plane. In each cell all values along \( z \)-axis are stored in a vectorized array. ConeFold LRnLA algorithm is applied for the two directions of locally recursive array. The subdivision includes special ConeFold type (fig. 7) to account for PIC calculations and field and particle time step difference. On higher ranks CFs combine into TorreFold algorithms.

4. Simulation examples

As an example where the described optimization is essential, in the scope of this paper the performance of CFHall code is demonstrated on the problem of laser interaction with overdense plasma.

The problem statement is illustrated on fig. 8. Plasma region occupies approximately half of the simulation domain. Laser pulse propagates in the positive direction of \( x \)-axis. Its transverse envelope is Gaussian and depends only on \( y \) coordinate. Periodic boundary is applies on \( z \) axis, other boundaries are reflective. The initial problem statement is 2D, but the simulation domain
is 3D3V. The 3D structures may only arise due to nonlinear plasma effects, which are intended to be observed in the study.

![Image](image_url)

**Figure 8.** Illustration of the problem statement for the simulation

Such problem is relevant for the advance of fast ignition concept. The interaction of ultraintense laser pulse with plasma layer leads to magnetic field generation due to the nonlinear process of Weibel filamentation instability. The effect of the resulting stable magnetic field structures is crucial for the understanding of electron acceleration efficiency in the system.

To study the nonlinear process of filamentation instability with calculations on a grid, it is essential that one field oscillation covers several mesh steps. The filament size is usually considered to be of the order of skin depth length $c/\omega_p$ [17].

In the current modeling, mesh size is $2048\times1024\times128$ cells, spatial and time steps are $dx = dy = dz = 0.1c/\omega_p$, $dt = 0.025\omega_p^{-1}$. In each cell there are 4 electrons and one ion, about $0.6\cdot10^9$ particles in total. Ions displacement during the simulation time is negligible, however their movement is not restricted. Pulse frequency is three times lower than electron Lengmuir frequency, its longitudinal envelope covers approximately ten wavelengths. The amplitude was varied in $1/3 \div 9$ range in the chosen units. The simulation continues to $250 \div 400\omega_p^{-1}$.

The data for computation takes $\sim 50G\delta$. The simulation has been performed on dual processor CPU Intel Xeon X5670 node. Asynchronous calculations are managed by POSIX threads. One time step in simulation takes $\sim 3$ seconds, so the results for 1 variant are obtained after $\sim 16$ hours.

During the simulation process the electrons are accelerated from the plasma layer boundary. 3D filamentary magnetic structures appear in areas through which accelerated electrons pass. The stable magnetic fields are formed on the plasma boundary.

![Image](image_url)

**Figure 9.** Simulation results. Evolution of filamentation in magnetic field components. $x$-$y$ projection on the left, $x$-$z$ projection on the right
As a separate study, trial simulations were carried out with 256 cells along $z$ axis and with 16 electrons in cell. The difference in the studied effects was negligible. Since the calculation time increases for these parameter values reaching up to two days, they weren’t used for the final series.

The series of experiments was carried out for different amplitude of incoming laser pulse. The ratio of the energy transferred to magnetic field formation was studied in each variant. The nonmonotonic dependency of this ratio on pulse amplitude was discovered (fig. 10). Its maximum reaches a considerable value of $\sim 4.5\%$.

Figure 10. Simulation results. Dependency of the energy ratio transferred to magnetic field — energy in magnetic fields divided by particle kinetic energy

5. Conclusion
The LRnLA algorithm family “ConeFold” had been constructed and implemented in the plasma kinetics simulation code CFHall. The algorithms ensure the locality of subsequently processed data and minimize data synchronization, enabling more efficient use of computer hardware for numerical simulation. The calculation time in comparison with standard approaches is $\sim 10 \div 1000$ times lower. The precise value of the acceleration may vary depending on the specific implementation with which CFHall is compared, on the dimensions and characteristic sizes in problem statement and thus the amount of data being processed. The precise comparison requires separate work and is not included in the scope of the current paper.

The code performance had been demonstrated to achieve physically relevant results in the modeling of laser pulse interaction. From the simulation the non-monotonic dependency of the ratio of energy transferred to magnetic field growth was obtained. The ratio may become up to $4.5\%$ for some values of laser intensity.

Omitting the further consideration of physical results obtained, in the scope of this paper we focus the attention on the benefit of the use of LRnLA algorithms for simulation. We have chosen enough mesh resolution for simulation to separate individual magnetic field filaments, used 3D3V geometry and thus we ensure that the increment in field growth would not be misrepresented.

On the other hand the calculation takes reasonable time, so the series of simulations with varying parameters, being at the same time relevant for front-line plasma physics questions may be carried out on the hardware available for individual researchers.

Aknowledgements
The work is supported by RFBR grants 12-01-00708-a, 12-01-00490-a.
References

[1] Perepelkina A Yu, Levchenko V D and Goryachev I A 2013 J. Phys.: Conf. Series 441 012014
[2] Perepelkina A Yu, Levchenko V D and Goryachev I A 2013 Matematicheskoe Modelirovanie 25(11) 98–110 (in Russian)
[3] Justin M 2007 Advances in Fully-Kinetic PIC Simulations of a Near Vacuum Hall Thruster and Other Plasma Systems, PhD thesis, MIT
[4] Cho S, Komurasaki K and Arakawa Y 2013 Physics of Plasmas, 20, 063501
[5] Elkina N V and Levchenko V D 2003 Appl. Phys. 5 16-23
[6] Okada T and Ogawa K 2007 Phys. Plasmas 14, 072702
[7] Borthakur A and Das N 2013 IJ BSS 2(1) 107-111
[8] Ruhl H 2002 Plasma Sources Sci. Technol. 11 A154
[9] Levchenko V D 2005 J. of Inf. Tech. and Comp. Systems (in Russian) 1 68
[10] Zakirov A V and Levchenko V D 2009 PIERS proceedings, Moscow, Russia 580–584
[11] Ivanov A V, Kaplan S A, Karakin A V, Levchenko V D, Levchenko T V and Rok V E 2006 Geoinformatika 3 59-61
[12] Korneev B A and Levchenko V D 2013 A Locally Recursive non- Locally Asynchronous Algorithm to RKDG Methods for Gas Dynamics Equations (preprint KIAM RAS, Moscow, Russia) 28
[13] Tallove A and Hagness SC 2000 Computational Electrodynamics: The Finite-Difference Time-Domain Method (2nd ed., Boston: Artech House)
[14] Sigov Yu S Numerical Experiment - the Link Between the Past and the Future of Plasma Physics (in Russian), (FIZMATLIT, Moscow, 2001)
[15] Birdsall C and Langdon A 1985 Plasma Physics via Computer Simulation (New York:Mc Graw Hill.)
[16] Morton G M 1966 A Computer Oriented Geodetic Data Base; and a New Technique in File Sequencing (Technical Report, Ottawa, Canada: IBM Ltd.)
[17] Treumann R A and Baumjohann W 2012 Annales Geophysicae 30 427-431,