UDC 519.6 10.23947/2587-8999-2022-1-1-18-28

APPLICATION OF THE GRID-CHARACTERISTIC METHOD FOR SOLVING THE PROBLEMS OF THE PROPAGATION OF DYNAMIC WAVES USING HPC SYSTEMS

N. I. Khokhlov

Moscow Institute of Physics and Technology, Dolgoprudny, Russia

✉ khokhlov.ni@mipt.ru

The paper considers the application of various modern technologies of high-performance computing to accelerate the numerical solution of the problems of propagation of dynamic wave disturbances using the grid-characteristic method. Technologies are considered both for central processing units (CPUs) and for graphic processors (GPUs). Comparative results of applying MPI, OpenMP, CUDA technologies are presented. As examples of the work of the developed software package, a number of examples of calculating the problems of seismic and geophysics are given. Separately, the issue of parallelizing problems with the presence of contacts of many grids and the topography of the day surface using curvilinear grids is considered.

Keywords: grid-characteristic method, seismic, geophysics, MPI, OpenMP, CUDA.

Introduction. Numerical simulation of the propagation of dynamic wave disturbances in solids is used to solve a wide range of problems. These tasks include, among other things, the tasks of seismic exploration and computational geophysics. The role of numerical simulation in each of these areas is very important. The search for minerals, such as hydrocarbons, is an expensive task, for which various methods of petrophysics, seismic and geology are used. As a result of applying these methods, an approximate model of the geological environment is built. Further, this model is used to test various hypotheses about the position in space of geological layers and the distribution of minerals. One of the methods of such an analysis is the exact solution of the direct problem of the propagation of elastic waves using numerical simulation. In addition, the solution of the direct problem is an integral part of the inversion and migration methods, the purpose of which is to determine the parameters of the environment and the positions in the space of sections between the media, which ultimately makes it possible to detect minerals in the earth's thickness. Numerical modeling of the propagation of seismic waves is an essential part of the work in geological exploration in the oil industry. Mathematical modeling is carried out in various geological environments, including stratified media and media with various inhomogeneities (for example, cracks or caverns). Tasks of this kind seem to be very resource intensive in terms of computational resources.

The characteristic dimensions of computational domains in such problems for each measurement are about 1000–10000 nodes, and reasonable computation times are about a day. The performance of modern desktop computers makes it possible to solve the direct problem by explicit numerical methods in the two-dimensional case in a reasonable time. And the performance of clusters is already enough to solve three-dimensional problems. The increased interest of industry in solving the direct problem is due to the increase in the performance of modern central processors, as this has

allowed fast simulation results for large geological models. The intensive development of technologies for parallel programming of general-purpose applied problems on graphic processors also contributes to an increase in interest. As a rule, explicit numerical methods can be rewritten for efficient execution on the GPU, which, depending on the type of problem and the numerical method, can improve performance by several orders of magnitude compared to the implementation for the CPU. Thus, the increased interest in numerical simulation in the industry explains the relevance of this work. The paper considers a grid-characteristic method for solving the equation of wave propagation in an elastic medium. This method has a number of features, due to which its use in certain formulations is more appropriate than the use of classical methods such as finite difference. In [1], the discontinuous Galerkin method was compared with the grid-characteristic method on an unstructured grid and on a regular grid [24–25]. The authors demonstrated a higher calculation speed using the CX method, and the accuracy of the calculation showed the applicability of the method to practical problems.

In this paper, we consider a software package designed to simulate the problems of propagation of dynamic wave disturbances in solids. The software package uses two-dimensional and three-dimensional structural block grids with inhomogeneities. For numerical integration, grid-characteristic [1,2] and finite-volume [3] methods of 2-4 orders of accuracy are used. The algorithm is parallelized using various technologies for writing parallel applications. At present, parallelization efficiency of up to 70% has been achieved using MPI technology while scaling up to 16 thousand computing cores. In systems with shared memory, the algorithm is parallelized using OpenMP technology. Also, the code is parallelized using CUDA technology, which gives an acceleration of up to 50 times compared to a single CPU core. The program can use several cards within one host. In this paper, the results of the same algorithm using different technologies are considered. Parallelization tests for up to 16 thousand CPU cores and 8 CUDA devices are given.

Mathematical model and method. The basic equations of motion of the linear-elastic medium can be written as follows [24]:

$$\rho \mathbf{v}_t = (\nabla \cdot \mathbf{T})^{\mathrm{T}}, \tag{1}$$

$$\mathbf{T} = \lambda(\nabla \cdot \mathbf{v})\mathbf{I} + \mu(\nabla \otimes \mathbf{v} + (\nabla \otimes \mathbf{v})^{\mathrm{T}}), \tag{2}$$

where ρ is density, $\mathbf{v}$ is velocity, $\mathbf{T}$ is the stress tensor, and λ, μ are the Lamé parameters, characterizing the elastic properties of the medium.

We use the following mathematical notations throughout this paper:

$\mathbf{a}_t \equiv \frac{\partial a}{\partial t}$ is the partial derivative of field $\mathbf{a}$ with respect to time t;

$\mathbf{a}\otimes\mathbf{b}$ is a tensor product of two vectors, $\mathbf{a}$ and $\mathbf{b}$, $(\mathbf{a}\otimes\mathbf{b})_{ij} = a_i b_j$;

$\mathbf{I}$ is identity tensor of rank 2.

The grid-characteristic method (GCM) uses the characteristic properties of the systems of hyperbolic equations, describing the elastic wave propagation [24]. The mathematical principles of the GCM approach are summarized in Appendix A. It is based on representing the equations of motion of the linear-elastic medium in the following form:

$$\mathbf{q}_t + \mathbf{A}_1\mathbf{q}_x + \mathbf{A}_2\mathbf{q}_y + \mathbf{A}_3\mathbf{q}_z = 0. \tag{5}$$

In the last equation, $\mathbf{q}$ is a vector of unknown fields, having 9 components and equal to

$$\mathbf{q} = \begin{bmatrix} \mathbf{v} \\ \mathbf{T} \end{bmatrix} = [v_1 \quad v_2 \quad v_3 \quad T_{11} \quad T_{22} \quad T_{33} \quad T_{23} \quad T_{13} \quad T_{12}]^{\mathrm{T}}. \tag{6}$$

where $\mathbf{q}(t, x, y, z)$ is a vector of the unknown fields; $\mathbf{q}_t$ denotes the partial derivative of $\mathbf{q}$ with respect to t; $\mathbf{q}_x$, $\mathbf{q}_y$ and $\mathbf{q}_z$ denote the partial derivatives of vector $\mathbf{q}$ with respect to x , y, and z, respectively.

Matrices $\mathbf{A}_k$, $k = 1,2,3$, are the 9×9 matrices given by the following expression:

$$\mathbf{A}_1 = -\begin{bmatrix} 0 & 0 & 0 & \rho^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho^{-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho^{-1} & 0 \\ \lambda + 2\mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{7}$$

matrix $\mathbf{A}_2$ is given by the following expression:

$$\mathbf{A}_2 = -\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho^{-1} \\ 0 & 0 & 0 & 0 & \rho^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho^{-1} & 0 & 0 \\ 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda + 2\mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{8}$$

and matrix $\mathbf{A}_3$ can be written as follows:

$$\mathbf{A}_3 = -\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho^{-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho^{-1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho^{-1} & 0 & 0 & 0 \\ 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda + 2\mu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{9}$$

The product of matrix $\mathbf{A}_k$ and vector $\mathbf{q}$ can be calculated as follows:

$$\mathbf{A}_k \begin{bmatrix} \mathbf{v} \\ \mathbf{T} \end{bmatrix} = -\begin{bmatrix} \rho^{-1}(\mathbf{T} \cdot \mathbf{n}) \\ \lambda(\mathbf{v} \cdot \mathbf{n})\mathbf{I} + \mu(\mathbf{n} \otimes \mathbf{v} + \mathbf{v} \otimes \mathbf{n}) \end{bmatrix}, \tag{10}$$

where $\otimes$ denotes the tensor product of two vectors.

In the last equation $\mathbf{n}$ is a unit vector directed along the $\mathbf{x}$, $\mathbf{y}$, or $\mathbf{z}$ directions for matrices $\mathbf{A}_1$, $\mathbf{A}_2$, or $\mathbf{A}_3$, respectively.

As we discussed above, the GCM approach is based on representing the solutions of the acoustic and/or elastic wave equations at later time as a linear combination of the displaced at a certain spatial step solutions at some previous time moment. This representation can be used to construct a direct time-stepping iterative algorithm of computing the wave fields at any time moment from the

initial and boundary conditions. In order to develop this time-stepping formula, we represent matrices $\mathbf{A}_k$ using their spectral decomposition. For example, for matrix $\mathbf{A}_1$ we have:

$$\mathbf{A}_1 = (\mathbf{\Omega}_1)^{-1}\mathbf{\Lambda}_1\mathbf{\Omega}_1, \tag{11}$$

where $\mathbf{\Lambda}_1$ is a 9×9 diagonal matrix, formed by the eigenvalues of matrix $\mathbf{A}_1$; and $(\mathbf{\Omega}_1)^{-1}$ is a 9×9 matrix formed by the corresponding eigenvectors. Note that, matrices $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_3$ have the same set of eigenvalues:

$$\{c_\mathrm{p}, -c_\mathrm{p}, c_\mathrm{s}, -c_\mathrm{s}, c_\mathrm{s}, -c_\mathrm{s}, 0,0,0\}. \tag{12}$$

In the last formula, c_p is a P-wave velocity being equal to $\left(\rho^{-1}(\lambda + 2\mu)\right)^{1/2}$ and c_s is an S-wave velocity being equal to $(\rho^{-1}\mu)^{1/2}$.

Let us consider some direction $\mathbf{x}$. We assume that the unit vector $\mathbf{n}$ is directed along this direction, while the unit vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ form a Cartesian basis together with $\mathbf{n}$. We also introduce the following symmetric tensors of rank 2:

$$\mathbf{N}_{ij} = \frac{1}{2}\left(\mathbf{n}_i \otimes \mathbf{n}_j + \mathbf{n}_j \otimes \mathbf{n}_i\right), \tag{13}$$

where indices i and j vary from 0 to 2 in order to simplify the final formulas, and $\mathbf{n}_0 = \mathbf{n}$.

It is shown in Appendix A that, the solution of equation (5), vector $\mathbf{q}$, along the x, y, and z directions can be written as follows:

$$\begin{aligned}
\mathbf{q}(t+\tau, x, y, z) &= \textstyle\sum_{j=1}^{J} \mathbf{X}_{1,j}\mathbf{q}\left(t, x - \Lambda_{1,j}\tau, y, z\right), \\
\mathbf{q}(t+\tau, x, y, z) &= \textstyle\sum_{j=1}^{J} \mathbf{X}_{2,j}\mathbf{q}\left(t, x, y - \Lambda_{2,j}\tau, z\right), \\
\mathbf{q}(t+\tau, x, y, z) &= \textstyle\sum_{j=1}^{J} \mathbf{X}_{3,j}\mathbf{q}\left(t, x, y, z - \Lambda_{3,j}\tau\right).
\end{aligned} \tag{14}$$

Here τ is the time step of the solution, and $\mathbf{X}_{1,j}$, $\mathbf{X}_{2,j}$ and $\mathbf{X}_{3,j}$ are the characteristic matrices expressed through the components of matrices $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_3$ and their eigenvalues as follows:

$$\mathbf{X}_{\mathrm{i},j} = \boldsymbol{\varpi}_{*i,j}\boldsymbol{\varpi}_{i,j}, \ \mathrm{i}=1,2,3; \tag{15}$$

where $\boldsymbol{\varpi}_{*i,j}$ is the j's column of matrix $(\mathbf{\Omega}_\mathrm{i})^{-1}$, and $\boldsymbol{\varpi}_{i,\mathrm{j}}$ is the j's row of matrix $\mathbf{\Omega}_\mathrm{i}$. The scalar components of the column matrices $\boldsymbol{\varpi}_{*\mathrm{I,j}}$ are defined by the following expressions:

$$\omega_{1,2} = \left(\mathbf{\Omega}_1 \begin{bmatrix}\mathbf{v}\\ \mathbf{T}\end{bmatrix}\right)_{1,2} = \mathbf{n}\cdot\mathbf{v} \mp \left(c_\mathrm{p}\rho\right)^{-1}(\mathbf{N}_{00} * \mathbf{T}), \tag{16}$$

$$\omega_{3,4} = \mathbf{n}_1\cdot\mathbf{v} \mp (c_\mathrm{s}\rho)^{-1}(\mathbf{N}_{01} * \mathbf{T}), \tag{17}$$

$$\omega_{5,6} = \mathbf{n}_2\cdot\mathbf{v} \mp (c_\mathrm{s}\rho)^{-1}(\mathbf{N}_{02} * \mathbf{T}), \tag{18}$$

$$\omega_7 = \mathbf{N}_{12} * \mathbf{T}, \tag{19}$$

$$\omega_8 = (\mathbf{N}_{11} - \mathbf{N}_{22}) * \mathbf{T}, \tag{20}$$

$$\omega_9 = \left(\mathbf{N}_{11} + \mathbf{N}_{22} - \frac{2\lambda}{\lambda+2\mu}\mathbf{N}_{00}\right) * \mathbf{T}. \tag{21}$$

In equations (16) - (21) the asterisk "*"denotes the convolution of two tensors of rank 2.

Expressions (14) can be used to find the solution, vector $\mathbf{q}$, at any time moment, $t + \tau$, from the given initial conditions, thus representing a direct time-stepping algorithm of numerical modeling the elastic wave propagation in inhomogeneous media.

MPI parallelization. To work in systems with distributed memory, the software package is parallelized using MPI technology [16]. In parallelization, standard algorithms for decomposition of the computational domain and exchange of boundary cells were used, which are widely used for explicit grid methods [17–20]. Since this implementation of the grid-characteristic solver uses regular

grids, the nodes can be stored in 2D/3D arrays (they are stored in memory as continuous one-dimensional arrays). The input to the program is the number of processes for each grid, which will recalculate the nodes inside these grids. The algorithm determines the best partitioning of arrays into blocks of equal size for distributing work between processes by calling MPI_Dims_create. In addition, it is possible to manually set the desired distribution of processes. The number of blocks into which the grid is divided is the same as the number of processes associated with the grid. This means that each process recalculates nodes for the next time step for at most one grid block. While processes can be responsible for no more than one grid block, they can process multiple blocks from different grids at the same time.

To simplify the synchronization of nodes on block boundaries within one grid, a special communicator (MPI_Comm) and a special group (MPI_Group) are created for this grid. The number of processes that are given as input determines the list of processes from the MPI_COMM_WORLD communicator, on the basis of which a new communicator is created. Synchronization within the grid is performed between time steps, and each process can obtain information about the processes processing neighboring blocks. Knowing the exact numbers of processes that are ready to exchange recalculated nodes on block boundaries, processes can perform asynchronous receive and send operations (MPI_Isend, MPI_Irecv).

In contrast to synchronization at the block boundary, when synchronizing at the boundaries of the contact surfaces between two meshes, the processes do not know the numbers of the processes with which to exchange data, since the geometry of the contacts can be arbitrary. The original implementation used the MPI_Alltoall function to ensure that all MPI_COMM_WORLD communicator processes contain the same decomposition information. The main reason for the inefficiency of such a strategy was that processes that did not need synchronization to continue the calculation were still required to participate in this interaction. In other words, a global synchronization point was created at each time step that could be eliminated.

Then, lists accessible to all processes with the properties of all grids were introduced. These properties contain information about the sizes of the grids, the numbers of the processes associated with the grids, the exact positions of the blocks and their sizes for each of these processes. Lists of properties are created at the beginning of the initialization of the computational algorithm, are synchronized between all processes and then remain unchanged (this is possible due to the static structure of grids and connections between them). For each mesh it is possible to get the process numbers from MPI_COMM_WORLD that process each block. Therefore, when a process needs to update the nodes on a contact boundary to synchronize with another process, it knows which process contains data from another mesh with that contact, so it can only communicate with that process. Moreover, this approach allows us to take into account the case in which several processes on the boundary of one grid communicate with independently distributed processes on the boundary of another grid. This means that mesh decompositions can be independent of each other.

Information about contacts at the grid boundaries is specified in a generalized way. The position of a region in one mesh that is to be sent to a given region in another mesh is input to the solver. The list of such hints contains complete information about the contacts in the multi-grid model. This information is used at each time step when synchronization occurs. If the positions of the grid nodes do not match exactly, then re-interpolation is performed.

The performance of our parallel implementation of the grid-characteristic method was measured. Fig. 1. shows the results. In the test, a computational grid with a size of 1000x1000x1000 nodes was used. Testing was carried out on the HECToR cluster. This supercomputer consists of 2816 computing nodes. Each of the nodes is equipped with two 16-core AMD Opteron 2.3GHz Interlagos processors. RAM is 32 GB per node. The countdown is from 128 MPI processes, in order to get correct results, and with a small number of processes, the calculation times can be unacceptably long. It can be seen that the presented implementation scales up to 4096 processes with almost no efficiency loss. As the number of processes increases, there is an expected decrease in efficiency, but the algorithm still performs well even at 16384 processes.

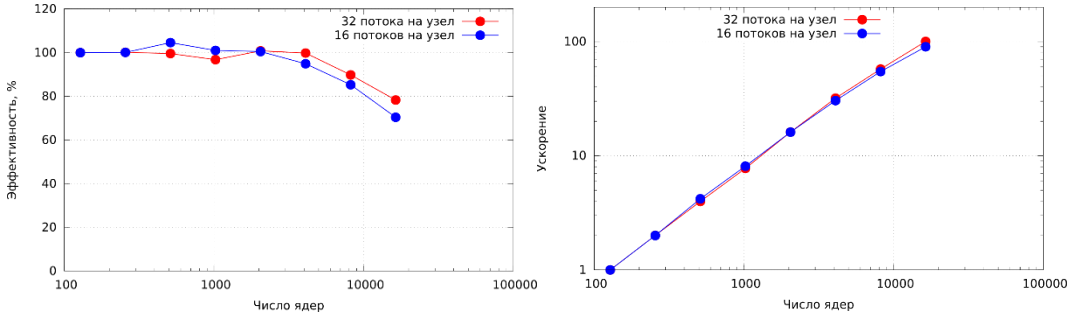

Fig. 1. MPI efficiency (left) and speedup (right)

OpenMP parallelization. Parallelization was carried out for systems with shared memory using OpenMP [10, 21] and POSIX Threads technologies. The resulting acceleration is shown in fig. 2. Due to insignificant differences in acceleration and ease of implementation of parallel code using OpenMP, it was decided to use only OpenMP technology in calculations. The code is parallelized according to the principle of geometric parallelism. The computational grid is divided into rectangular areas, each of which is recalculated by one OpenMP thread. The superimposed meshes used to specify cracks are split in the same way. Global thread synchronization occurs between time steps. In this case, values are exchanged in nodes that are on the boundary of partitions and must be available simultaneously to several threads.

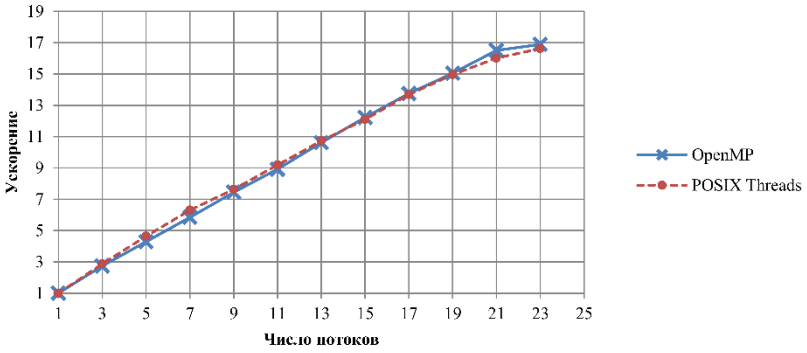

Fig.2. Dependence of acceleration on the number of threads for systems with shared memory

74% acceleration efficiency of the resulting algorithm was achieved on a machine with Intel Xeon E5-2697 processors (on 24 cores).

CUDA parallelization. The algorithm was also parallelized on NVidia GPU GPGPUs using CUDA technology. This technology is widely used for parallelization, including explicit, computational algorithms [11-15]. A complete rewriting of a part of the calculation module for this architecture was required [21]. The version of this program optimized for execution on the central processor was taken as the basis for the implementation of the algorithm on graphic processors. The most computationally expensive part of the algorithm was subjected to optimizations. Since splitting along the spatial coordinate was used, two steps were required to recalculate the entire grid: along the X axis and along the Y axis. In the initial version (cuda1 in Fig. 3), 2 times more memory is allocated on the graphics device than is required to store the computational grid. Two copies of it are kept in memory.

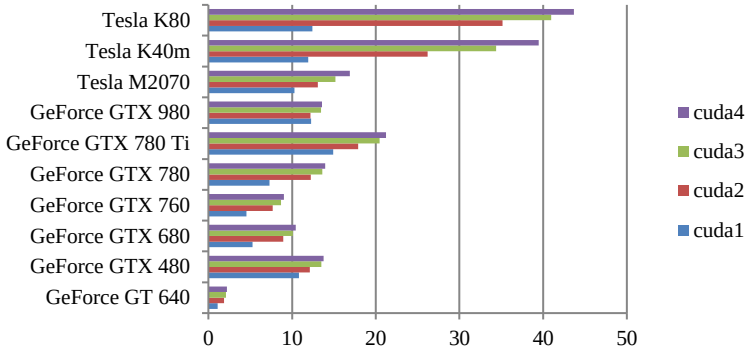

Fig. 3. Speed up compared to CPU

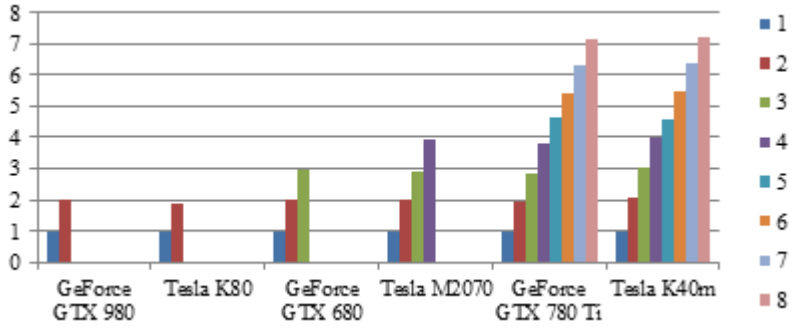

Fig. 4. Acceleration depending on the number of graphics devices

At each step, the values of the grid nodes stored in one of the copies are recalculated. In this case, the calculation result is written to another copy of the computational grid. As a result, synchronization occurs only between calls to CUDA kernels. This was done because the global synchronization of the entire device creates large time delays. Thus, it was possible to reduce the number of global synchronizations to two times per time step. The next optimization (cuda2) was to use the device's shared memory. Reading from global memory occurs only once at each step, and accesses become sequential (coalesced), which also leads to better performance. Further, optimizations are used that give a slight increase in performance. In the cuda3 version, additional data is computed on the CPU and passed through kernel call parameters. Thus, each thread is expected to have a local copy of these values. In the cuda4 version, block sizes are selected in such a way as to minimize the number of nodes that require memory exchanges between blocks.

For execution on multiple GPUs, the most optimized variant was used. GPUDirect technology was also used, which allows data to be transferred via the PCI Express bus, bypassing the central processor. The maximum acceleration obtained compared to the CPU on a single graphics device is 55 times on the GeForce GTX 780 Ti in single precision calculations and 44 times on the Tesla K80 in double precision calculations (Fig. 3). The maximum achieved acceleration from the number of graphics devices is 7.1 times on 8 graphics devices (Fig. 4) for double precision. GPUDirect technology has increased the acceleration by 10% from what was achieved without it in single precision calculations. When calculating with double precision, the acceleration was 2.4%.

Computational examples. We carried out the computations of the wave field for the models with and without the fractures [26]. The height of the model was 4.65 km, the width and length were 20 km and 20 km, accordingly. The point source of Reiker impulse was set in the center of the computational grid on the surface of the medium. The central frequency of the seismic source was 10 Hz as the full waveform inversion is usually carried out using the low frequencies. The receivers were situated on the surface of the medium with the 50 x 50 m steps in the x and y directions. First, we computer simulated the seismic waves in heterogeneous medium without fractures. Fig. 5 presents a schematic view of the model. The color bar denotes the values of the longitudinal velocity.

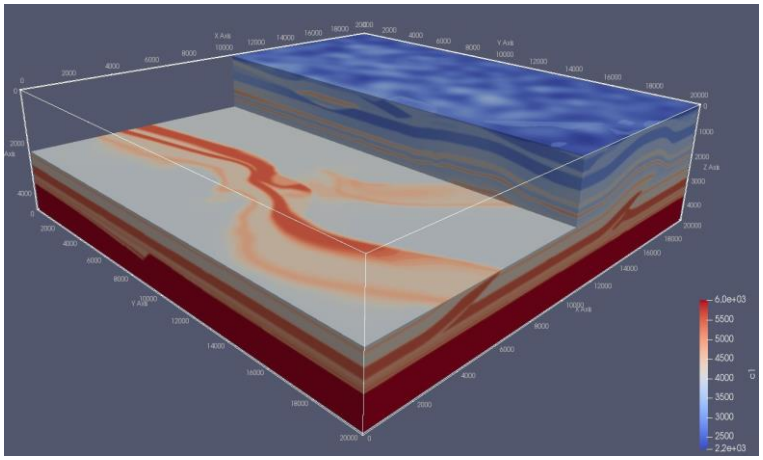

Fig. 5 Schematic representation of the model without fractures

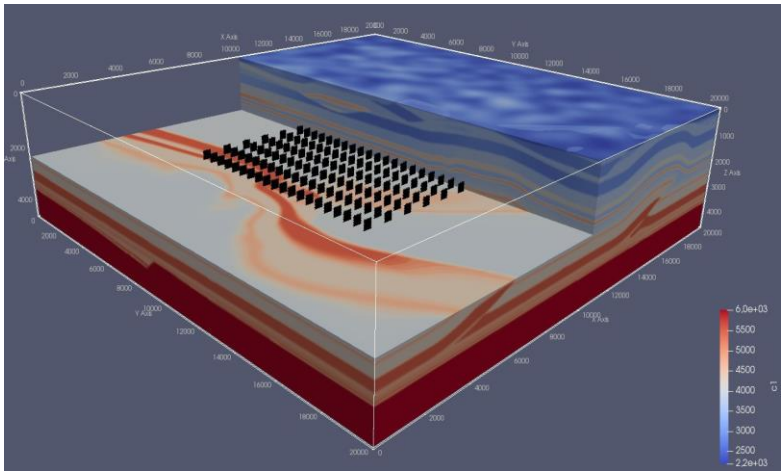

Fig. 6. Schematic representation of the model with the fractures

Then, we calculated the wave field in the heterogeneous medium with 200 vertical extremely thin fractures. The size of a single fracture was 200 x 200 m^2. The fractures were situated at a depth of 2 km in the center of the computational grid. Fig. 6 depicts a schematic representation of the model with fractured structures.

The seismograms for the models with and without fractures are shown in Fig. 7. The pictures in Fig. 4 reflect the values for the vertical field components V_z in the receivers for the XY plane on the surface (Z = 0) at the same moment of time equal to 2.4 s. The leftmost picture in Fig. 7 shows the anomalous wave field from the cluster of fractures. The middle picture shows the wave field for the model without fractures. The rightmost picture presents the overall wave field for the model with the cluster of vertical fractures. The color scale is the same for all the pictures in Fig.7.

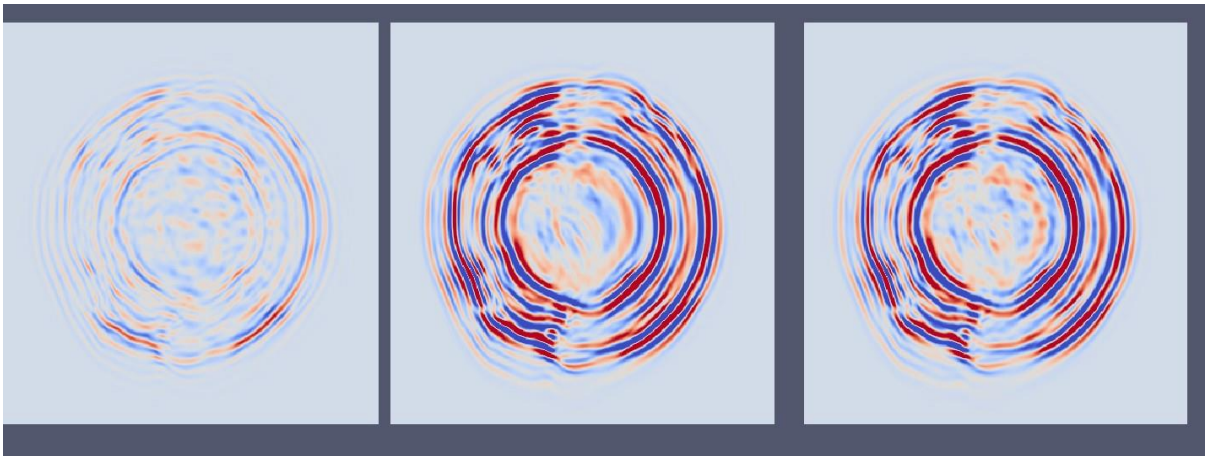

Fig. 7. The vertical field component (V_z) in the receivers at time moment of 2.4 s for different models

a) the anomalous field from the model

b) the total field in the model without fractures

c) the total field in the with fractures

The last figure shows that the response from the fractured structures is smaller than the value of the total response, but it is still quite noticeable. This result illustrates a possibility of reconstructing a response from the fractured structures with the help of full-wave seismic modeling.

Conclusions. The paper considers a software package designed to simulate the propagation of dynamic wave disturbances in solids. The calculation algorithm is based on the grid-characteristic method for solving systems of hyperbolic equations in partial derivatives. This method makes it possible to take into account the wave structure of the equation and to correctly set the boundary and contact conditions. An algorithm has been implemented that makes it possible to perform calculations using block-structural grids. The algorithm is parallelized using MPI, OpenMP, CUDA technologies. At present, parallelization efficiency of up to 70% has been achieved using MPI technology while scaling up to 16 thousand computing cores. In systems with shared memory, the algorithm is parallelized using OpenMP technology. Also, the code is parallelized using CUDA technology, which gives an acceleration of up to 50 times compared to a single CPU core. The program can use several cards within one host. In this paper, the results of the same algorithm using different technologies are considered. Parallelization tests for up to 16 thousand CPU cores and 8 CUDA devices are given. The performance of the algorithm on test examples is shown.

References

1. V. A. Biryukov, V. A. Miryakha, I. B. Petrov, and N. I. Khokhlov, Simulation of elastic wave propagation in geological media: Intercomparison of three numerical methods // Comput. Math. Math. Phys., Vol. 56, No. 6, 2016.
2. Голубев В.И., Петров И.Б., Хохлов Н.И. Численное моделирование сейсмической активности сеточно-характеристическим методом // Журнал вычислительной математики и математической физики, 2013. Т. 53, № 10. С. 1709 – 1720.
3. P.L.Roe. Characteristic-Based Schemes for the Euler Equations // Annual Review of Fluid Mechanics. 1986. No.18. pp. 337-365.
4. LeVeque R. J. Finite volume methods for hyperbolic problems. Vol. 31. Cambridge university press, 2002.
5. LeVeque R. J. Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems. Vol. 98. Siam, 2007.
6. Strang G. On the construction and comparison of difference schemes // SIAM Journalon Numerical Analysis. 1968. Vol. 5, No. 3. pp. 506–517.
7. Courant R., Isaacson E., Rees M. On the solution of nonlinear hyperbolic differential equations by finite differences // Communications on Pure and Applied Mathematics. 1952. Vol. 5, No. 3. pp. 243–255.
8. Rusanov V. V. Difference schemes of the third order of accuracy for the forward calculation of discontinuous solutions // Doklady Akademii Nauk. Vol. 180. Russian Academy of Sciences. 1968. pp. 1303-1305.
9. Favorskaya A. V., Petrov I. B. Grid-characteristic method // Innovations in Wave Processes Modelling and Decision Making / ed. by A. V. Favorskaya, I. B. Petrov. Springer, 2018. pp. 117–160.
10. Dagum L., Menon R. OpenMP: an industry standard API for shared-memory programming // IEEE computational science and engineering. 1998. Vol. 5. No. 1. pp. 46–55.
11. Nakata N., Tsuji T., Matsuoka T. Acceleration of computation speed for elastic wave simulation using a Graphic Processing Unit // Exploration Geophysics. 2011. Vol. 42, No. 1. pp. 98–104.
12. Weiss R. M., Shragge J. Solving 3D anisotropic elastic wave equations on parallel GPU devices // Geophysics. 2013. Vol. 78, No. 2. F7–F15.
13. Finite-difference staggered grids in GPUs for anisotropic elastic wave propagation simulation / F. Rubio [et al.] // Computers & geosciences. 2014. Vol. 70. pp. 181–189.
14. Komatitsch D., Michéa D., Erlebacher G. Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using CUDA // Journal of Parallel and Distributed Computing. 2009. Vol. 69, No. 5. pp. 451-460.
15. Modeling the propagation of elastic waves using spectral elements on a cluster of 192 GPUs / D. Komatitsch [et al.] // Computer Science-Research and Development. 2010. Vol. 25, No. 1/2. pp. 75–82.
16. Message P Forum. 1994. MPI: a Message-Passing Interface Standard. Technical Report. University of Tennessee, Knoxville, TN, USA.
17. Якобовский М.В. Введение в параллельные методы решения задач // Предисл.: В. А. Садовничий. М.: Издательство Московского университета, 2013. 328 с.
18. Ivanov A.M., Khokhlov N.I. Efficient Inter-process Communication in Parallel Implementation of Grid-Characteristic Method, 2019. pp. 91–102.
19. Ivanov A.M., Khokhlov N.I. Parallel implementation of the grid-characteristic method in the case of explicit contact boundaries // Comput. Res. Model. 2018. V. 10. № 5. pp. 667–678.
20. Khokhlov N. и др. Solution of Large-scale Seismic Modeling Problems // Procedia Comput. Sci. 2015. V. 66. pp. 191–199.
21. Khokhlov N. и др. Applying OpenCL Technology for Modelling Seismic Processes Using Grid-Characteristic Methods, 2016. pp. 577–588.

Author:

Nikolay I. Khokhlov, PhD, Moscow Institute of Physics and Technology (9 Institutskiy per., Dolgoprudny, Moscow Region, Russian Federation), khokhlov.ni@mipt.ru

УДК 519.6 10.23947/2587-8999-2022-1-1-18-28

ПРИМЕНЕНИЕ СЕТОЧНО-ХАРАКТЕРИСТИЧЕСКОГО МЕТОДА ДЛЯ РЕШЕНИЯ ЗАДАЧ РАСПРОСТРАНЕНИЯ ДИНАМИЧЕСКИХ ВОЛНОВЫХ ВОЗМУЩЕНИЙ С ИСПОЛЬЗОВАНИЕМ СИСТЕМ HPC

Н. И. Хохлов

Московский физико-технический институт, Долгопрудный, Российская Федерация

✉ khokhlov.ni@mipt.ru

В работе рассматривается применение различных современных технологий высокопроизводительных вычислений для ускорения численного решения задач распространения динамических волновых возмущений с использованием сеточно-характеристического метода. Рассматриваются технологии как для центральных процессоров (CPU), так и для графических процессоров (GPU). Приведены сравнительные результаты применения технологий MPI, OpenMP, CUDA. В качестве примеров работы разработанного программного комплекса приводится ряд примеров расчета задач сейсмики и геофизики. Отдельно рассмотрен вопрос распараллеливания задач с наличием контактов многих сеток и топографии дневной поверхности, используя криволинейные сетки.

Ключевые слова: сеточно-характеристический метод, сейсмика, геофизика, MPI, OpenMP, CUDA.

Автор:

Хохлов Николай Игоревич, кандидат физико-технических наук, Московский физико-технический институт (141701, Московская область, г. Долгопрудный, Институтский пер., 9.), khokhlov.ni@mipt.ru