УДК 519.6
https://doi.org/10.47533/2020.1606-146X.04

A. *ALTYBAY, N. TOKMAGAMBETOV, Z. SPABEKOVA

Al-Farabi Kazakh National University, Almaty, Kazakhstan

GPU COMPUTING FOR 2D WAVE EQUATION BASED ON IMPLICIT FINITE DIFFERENCE SCHEMES

In this paper we will consider the numerical implementation of the 2d wave equation which is a fundamental equation in many engineering problems. An approximate solution of a function is calculated from discrete points in spatial grid based on discrete time steps. The initial values are given by the initial value condition. First we will interpret how to transform a differential equation into an implicit finite-difference equation, respectively, a set of finite-difference equations that can be used to calculate an approximate solution. Then we will change this algorithm to parallelize this task on GPU. Special focus is on improving the performance of

the parallel algorithm. In addition, we will run the implemented parallel code on the GPU and serial code the central processor, calculate the acceleration based on the execution time. We present that the parallel code that runs on a GPU gives the expected results by comparing our results to those obtained by running serial code of the same simulation on the CPU. In fact, in some cases, simulations on the GPU are found to run 22 times faster than on a CPU.

Key words*: *Numerical simulation, GPU, CUDA technology, wave equation, finite difference.

INTRODUCTION. The application of high-performance parallel computing in mathematical modeling opens up new possibilities for studying physical processes in longer time and more extensive spatial domains. Currently, various high-performance parallel computing is used in many areas. One of such applications is acoustics. One of the most important tasks of acoustics is the problem of wave field modeling. Already for several years, GPUs have been used to accelerate well parallelizable computing, only with the advent of a new generation of GPUs with multicore architecture; this direction began to give tangible results. The goal of this work is to develop a parallel implementation of the finite difference method for solving two-dimensional wave equation on a graphics processor using CUDA technology and to study the efficiency of parallelization by comparing the time of solving two-dimensional wave equation on a GPU and a central processor. There is a large amount of work devoted to numerical methods developed for the study of wave processes in recent decades. It includes a finite-difference method [1], a finite-volume method [2], the finite-element method [3], a spectral-element method [4] a two-level compact ADI method [5] , the

*Адрес для переписки. E-mail: arshyn.altybay@gmail.com implicit Finite Difference Time Domain Methods [6], a boundary [integral] element method [7], and spectral methods [8]. A completely non-linear model must be applied to many problems. Most models have been developed for technical applications. These numerical methods provide some of the most natural methods for modeling the propagation and scattering of underlying waves in electromagnetic, acoustic and elastic studies. However, as indicated in [9], the aforementioned methods have several disadvantages if the second-order equations are converted to first-order systems before discretization, especially in the presence of several spatial dimensions. Therefore, recently, much attention has been paid to the development of efficient finite-difference methods that directly discretize second-order differential equations [[10][11]]. The two- dimensional approach considers a highly idealized wave field, since even monochromatic waves in the presence of side perturbations quickly acquire a two-dimensional structure. The difficulties encountered are not a direct result of the increase in size.

The main complication is that the problem cannot be reduced to a two-dimensional problem, and even for the case of a two-periodic wave field, the problem of solving the Laplace-type equation for the velocity potential arises. Most models designed to study the three-dimensional dynamics of waves are based on simplified equations, such as second-order perturbation methods, in which higher-order terms are ignored.

In general, it is unclear what effects are missing in such simplified models. Our current work is motivated by recent interest in the development and application of high-order compact difference methods for solving partial differential equations. Obviously, higher-order compact difference schemes have better resolution on stencils with a compact grid compared to non-compact or low-level methods [12, 13]. For multidimensional problems, the efficiency of an implicit compact difference scheme depends on the computational efficiency of the corresponding matrix solvers. From this point of view, the ADI method [14] is promising because they can decompose a multidimensional problem into a series of one-dimensional problems. It has been shown that schemes acquired are unconditionally stable. For the proper assignment of large domains of modeling, two- or three-dimensional computational grids with a sufficient number of points are used. Calculations on such grids require more CPU time and computer memory resources. To accelerate the computation process, GPU technology was used in this paper, which allows the program to operate on larger grids. The graphics processing unit (GPU) is a highly parallel, multi-threaded, and multi-core processor with enormous processing power. Its low cost and high bandwidth floating point operations and memory access bandwidth are attracting more and more high performance computing researchers [15]. In addition, compared to cluster systems, which consist of several processors, computing on a GPU is inexpensive and requires low power consumption with equivalent performance. In many disciplines of science and technology, users were able to increase productivity by several orders of magnitude using graphics processors [16, 17]. GPU programming on NVIDIA graphics cards has become significantly easier with the introduction at the end of 2006 of the CUDA programming language (NVIDIA Corporation 2009a), which is relatively easy to learn because its syntax is similar to C. With GPU becoming available alternative to CPU for parallel computing, aforementioned parallel tridiagonal solvers and other hybrid methods have been implemented on GPUs [18-25].

Zhang et al. [18] first implemented parallel cyclic reduction (PCR) and then proposed a CR-PCR hybrid algorithm. A hybrid of PCR-Thomas method was proposed by Sakharnykh

[24], and it was also studied by Zhang et al. [18]. There are many examples in the literature of successfully using GPUs for wave propagation simulation [16- 31].

Here we consider some issues in the numerical simulation of some problems in the propagation of the wave in acoustic on GPU.

NUMERICAL EXPERIMENTS. We consider two-dimensional wave equation

$$\frac{\partial^2 u}{\partial t^2} - c^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = f(x,y,t),(t,(x,y)) \in [0;T]\times[0;l]; \tag{2.1}$$

subject to the initial conditions

$$u(0, x, y) = \varphi_1(x, y);\ x;\ y \in [0;\ l]; \tag{2.2}$$

$$\frac{\partial u(0,x,y)}{\partial t} = \varphi_2(x, y);\ x;\ y \in [0;\ l]; \tag{2.3}$$

and boundary conditions

$$u(t, x, 0) = 0,\ u(t, x, l) = 0,\ t \in [0;\ T],\ x \in [0;\ l]; \tag{2.4}$$

$$u(t, 0, y) = 0,\ u(t, l, y) = 0,\ t \in [0;\ T],\ y \in [0;\ l]. \tag{2.4}$$

We introduce a space-time grid with steps h_1, h_2, τ respectively, in the variables x; y; t:

$$w^{\tau}_{h_1 h_2} = \{x_i = ih_1, i = \overline{0,N};\ y_j = jh_2,\ j = \overline{0,N};\ t_k = k\tau;\ k = 0,1,...,2T\} \tag{2.8}$$

2.1. Alternating direction implicit (ADI) method. The Alternating Direction Implicit (ADI) method is a finite difference scheme and has long been used to solve partial differential equations (PDEs) in higher dimension. Originally it was introduced by Peaceman and Rachford [14], but many variants have been invented throughout the years [32-34]. In ADI method, each numerical step is split into several sub-steps based on the spatial dimension of the problem, and the linear equation system is solved implicitly in one direction while treating information in the other direction(s) explicitly. With this alternating calculations, ADI method is unconditionally stable and second order in time and space. Another favorable property of the ADI method is that in each sub-step the equations to be solved have a tridiagonal structure and can be solved efficiently with Tridiagonal Matrix Algorithm (TDMA).

For problem (2.1) the ADI method has the form

$$\frac{u^{k+1/2}_{i,j} - 2u^{k}_{i,j} + u^{k-1/2}_{i,j}}{\tau^2} - \frac{c^2}{2h^2}\left(u^{k+\frac{1}{2}}_{i+1,j} - 2u^{k+\frac{1}{2}}_{i,j} + u^{k+\frac{1}{2}}_{i-1,j} + u^{k-\frac{1}{2}}_{i+1,j} - 2u^{k-\frac{1}{2}}_{i,j} + u^{k-\frac{1}{2}}_{i-1,j}\right) = f^{k}_{i,j} \tag{2.9}$$

$$\frac{u^{k+1}_{i,j} - 2u^{k+1/2}_{i,j} + u^{k}_{i,j}}{\tau^2} - \frac{c^2}{2h^2}\left(u^{k+1}_{i,j+1} - 2u^{k+1}_{i,j} + u^{k+1}_{i,j-1} + u^{k}_{i,j+1} - 2u^{k}_{i,j} + u^{k}_{i,j-1}\right) = f^{k+\frac{1}{2}}_{i,j} \tag{2.10}$$

Equation (2.1) then can be efficiently solved by ADI method [14] in two sub-steps.

At the first sub-step, Equation (2.9) is solved in i direction:

$$a_i u^{k+1/2}_{i+1,j} + b_i u^{k+1/2}_{i,j} + c_i u^{k+1/2}_{i-1,j} = f^{k}_{i,j} \tag{2.11}$$

where $a_i = \frac{\tau^2}{2}, b_i = \tau^2 + h^2, c_i = \frac{\tau^2}{2}$ and $f_i = -\frac{\tau^2}{2} u_{i+1,j}^{k-\frac{1}{2}} + (\tau^2 + h^2) u_{i,j}^{k-\frac{1}{2}} - \frac{\tau^2}{2} u_{i-1,j}^{k-\frac{1}{2}} + 2h^2 u_{i,j}^k +$ $+ f_{i,j}^k$. For the next sub-step, Equation (2.10) is solved in j direction:

$$a_j u_{i,j+1}^{k+1} + b_j u_{i,j}^{k+1} + c_j u_{i,j-1}^{k+1} = f_{i,j}^{k+\frac{1}{2}} \tag{2.12}$$

where $a_j = \frac{\tau^2}{2}, b_j = \tau^2 + h^2, c_j = \frac{\tau^2}{2}$ and $f_j = -\frac{\tau^2}{2} u_{i,j+1}^k + (\tau^2 + h^2) u_{i,j}^k - \frac{\tau^2}{2} u_{i,j-1}^k + 2h^2 u_{i,j}^k +$ $+ f_{i,j}^{k+\frac{1}{2}}$.

2.2. Cyclic reduction algorithm (CR). Cyclic reduction algorithm was invented by W. Hockney in the 1965 [35] and the CR algorithm consists of two steps: forward reduction and backward substitution. The forward reduction step sequentially eliminate the odd-indexed unknowns and then unknowns are re-ordered and the process is continued until one equation with one unknown is left. The backward substitution step solves the remaining one equation and finds the unknown y, consequently finds all unknowns from the previous steps, this algorithm fully described in work zhang[18].

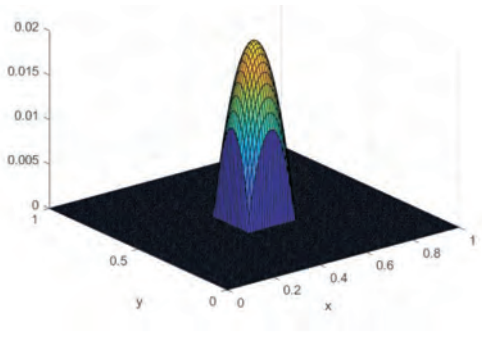
Using the implicit subscheme (2.9), the cyclic reduction method is performed in the x direction, with the result that we get the grid function $u_{i,j}^{k+1/2}$. In the second fractional time step, using the subscheme (2.10), the Cyclic reduction method is performed in the direction of the y axis, with the result that we get the grid function $u_{i,j}^{k+1}$. The Cyclic reduction algorithm has the order $0(\tau + h^2)$, i.e. the first order in time and the second in *x* and *y* variables. In the following, we demonstrate numerical simulations.

All calculations are made in C++ by using the cyclic reduction algorithm. For all simulations $\Delta t = 0{,}01$, $\Delta x = \Delta y = 0{,}01$. In all visualization of result, we use Matlab R2018b.

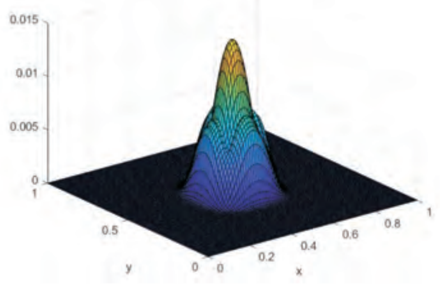
For simulation we use initial condition:

$$U(x,y,0) = \begin{cases} (x-0{,}4)(x-0{,}6)+(y-0{,}4)(y-0{,}6), & if\ 0{,}4 < x < 0{,}6\ and\ 0{,}4 < y < 0{,}6; \\ 0, else. \end{cases}$$

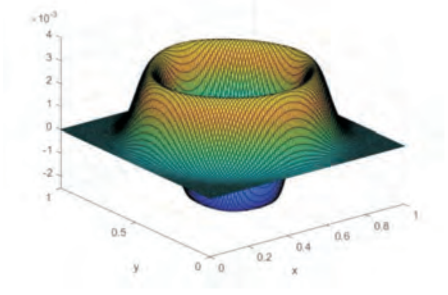
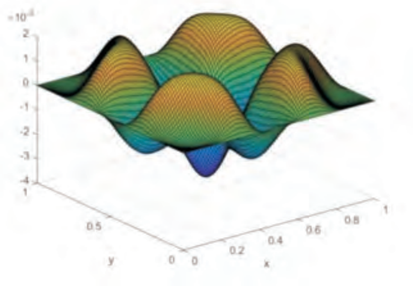
$U_t(x, y, 0) = 0$ and dirichlet boundary conditions. Some results are illustrated in Fig.1.



t = 0



t = 0,05

$t = 0{,}3$ $t = 1{,}0$

Figure 1 – Displacement of wave at different times.

We adopt the unit square $(x, y) \in [0; 1] \times [0;1]$ as the spatial solution domain with 100 elements per each side and 100 interior points, c = 1, with initial condition

$$u(x,y,0) = \sin(2\pi x)\sin(2\pi y), \frac{\partial u(x,y,0)}{\partial t} = 0$$

and $u(0, y, t) = u(1, y, t) = u(x, 0, t) = u(y, 1, t) = 0$ on the boundaries. The analytical solution of equation 2.1 is as flows: $u(x, y, t) = \cos(2\pi\sqrt{2}t)\sin(2\pi x)\sin(2\pi y)$.

Graphic comparisons of the exact solution with the numerical and errors are shown in figure 2.

Table 1 – Maximum norm and norm errors.

h	max error	L^2 – norm
0,001	10^{-3}	10^{-22}
0,0001	10^{-3}	10^{-22}
0,00001	10^{-3}	10^{-23}
0,00001	10^{-3}	10^{-25}
0,000001	10^{-3}	10^{-27}

Table 1 displays the convergence rate of displacement solution under grid refinement. The convergence rates in maximum norm at the final time shows forth order convergence.

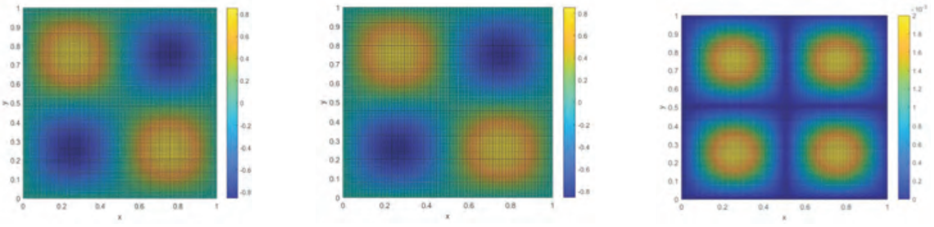


Figure 2 – Solution for time $t = 1$, $\Delta x = \Delta y = 0{,}010101$,
$\Delta t = 0{,}0001001$ from left to right exact, numerical, error

CUDA IMPLEMENTATION. Nowadays Graphics Processing Units(GPUs) or graphics processors have evolved from fixed-function processors specialized for three-dimensional graphics operations to a fully programmable computing platform for a wide variety of computationally demanding applications. Modern GPUs are massively data-parallel throughput-oriented many-core processors capable of providing TFLOPS of computing performance and quite high memory bandwidth compared to a high performance CPU.

In 2007, NVIDIA introduced CUDA, an extension to C programming language, for general purpose computing on graphics processors. It is designed so that its constructions allow a natural expression of concurrency at the data level. A CUDA program consists of two parts: a sequential program running on the CPU, and a parallel part running on the GPU [37, 38]. The parallel part is called the kernel. A C program using CUDA extensions hand out a large number of copies of the kernel into available multiprocessors to be performed contemporaneously. The CUDA code consists of three computational steps: transferring data to the global GPU memory, running the CUDA core, and transferring the results from the GPU to the CPU memory. The algorithm for solving the problem (2.1) is shown in Algorithm 1.

Algorithm 1 – Implementation of 2D wave equation

```
1.  compute initial condition matrix $U_0$
2.  from initial condition (2.2) we can get $u = U_0$
3.  while ($t < t_{end}$) do
4.  for $j = 0 ,..., n$
5.  for $i = 0 ,..., n$
6.  calculate tridiagonal system elements $a_i, b_i, c_i, f_i$
7.  call function CR($a_i, b_i, c_i, f_i, y_i, n$)
8.  calculate matrix $U_x$
9.  for $i = 0, ..., n$
10. for $j = 0, ..., n$
11. calculate tridiagonal system elements $a_j, b_j, c_j, f_j$
12. call function CR($a_j, b_j, c_j, f_j, y_j, n$)
13. calculate matrix $U_y$
14. swap ($u, U_x$)
15. swap ($U_0, U_y$)
16. $t = t + \Delta t$
17. end while
```

Here, u, U_0, U_x, U_y denote $u_{i,j}^{k-1/2}, u_{i,j}^{k}, u_{i,j}^{k+1/2}, u_{i,j}^{k+1}$ respectively

EXPERIMENTAL RESULTS. In this section we show the results obtained on a laptop with configuration, 640 cores GeForce GTX 1050, NVIDIA GPU together with a CPU Intel Core i7 8th gen, 2.20 GHz, RAM 8Gb. Simulation parameters are configured as follows. Mesh size is uniform in both directions with $\Delta x = \Delta y = 1/(N - 1)$, $c = 1$ and numerical time step Δt is 0.01 s, and simulation time is T = 1.0s, therefore the total number of time steps is 100. To present more realistic data, we tested six cases with domain sizes of 64x64; 128x128; 512x512; 1024x1024; 2048x2048 and 4096x4096.

The performance of a parallel algorithm is determined by calculating its speedup.

Speedup is defined as the ratio of the best-case execution time of the sequential algorithm for a particular problem to the worst-case execution time of the parallel algorithm.

$$Speedup = \frac{CPUtime}{GPUtime}$$

In Table 2 we report the execution times in seconds for serial (CPU time) and CUDA (GPU time) implementation of cyclic reduction method to the problem (2.1)-(2.5) together with the values of the speedup computed as the ratio in different devices

Table 2 – Execution timing and speed up with the Intel Core i7 8th gen, 2.20 GHz, NVIDIA GTX 1050.

Domain sizes	CPU time	GPU time	Speedup
64 × 64	2.417	0.863	2.8
128 × 128	7.95	1.558	5.1
512 × 512	155.33	18.71	8,3
1024 × 1024	1198.281	76.813	15.6
2048 × 2048	1885.483	104.343	18.07
4096 × 4096	3590.3	161.33	22.3

CONCLUSIONS AND FUTURE WORK. In this paper, we have introduced a numerical solution of a two-dimensional wave equation based on an implicit finite difference scheme using the cyclic reduction method. We develop an approach parallelization of the cyclic reduction method on the graphic processing unit parallelization of the cyclic reduction method on the graphic processing unit. And we showed how we accelerated the cyclic reduction method on the NVIDIA GPU. From the test results of table 1, it can be seen that the acceleration algorithm proposed by us gives a good result. Our GPU implementation obtained a speedup around 22,3x.

REFERENCES

1 A. P. Engsig-Karup, B. Harry, H. B. Bingham, O.Lindberg. An efficient flexible-order model for 3D nonlinear water waves, J. Comput. Phys., 228, 2100-2118, 2009.

2 D. Greaves. Application Of The Finite Volume Method To The Simulation Of Nonlinear Water Waves, Advances in Numerical Simulation of Nonlinear Water Waves, 11, 357-396, 2010.

3 G.Richter An explicit finite element method for the wave equation, Applied Numerical Mathematics, 16,65-80.1994

4 D. Komatitsch, J. Tromp. Spectral-element simulations of global seismic wave propagation: II Three-dimensional models, oceans, rotation and self-gravitation Geophysical Journal International, 150:1, 303318. 2002.

5 H. L. Liao, Z.Z. Sun. A two-level compact ADI method for solving second-order wave equations, International Journal of Computer Mathematics, 90:7, 1471-1488,2013.

6 H.K. Rouf. Implicit Finite Difference Time Domain Methods. Theory and Applications/Hasan Khaled Rouf. -LAP Lambert Academic, 208, 2011.

7 J.Grue, D.Fructus. Model For Fully Nonlinear Ocean Wave Simulations Derived Using Fourier Inversion Of Integral Equations In 3D, Advances in Numerical Simulation of NonlinearWater Waves, 11, 1-42, 2010.

8 G. Ducrozet, F. Bonnefoy, D. L. Touze, P. Ferrant. Open-source solver for nonlinear waves in open ocean based on High-Order Spectral method, Comp. Phys. Comm., 203, 245-254, 2016.

9 H. O. Kreiss, N.A. Petersson, J. Ystrom. Di_erence approximation for the second-order wave equation, SIAM J. Numer. Anal. 40, 1940-1967, 2002.

10 M. Dehghan, A. Mohebbi. The combination of collocation, finite difference, and multigrid methods for solution of the two-dimensional wave equation, Numer. Methods Partial Differential Equation, 24, 897-910, 2008.

11 H.F. Ding, Y.X. Zhang. A new fourth-order compact finite difference scheme for the two dimensional second-order hyperbolic equation, J. Comput. Appl. Math., 72, 626-632, 2009.

12 G. Cohen. High-Order Numerical Methods for Transient Wave Equations, Springer, NewYork, 2002.

13 J.M. Liu , K.M. Tang. A new unconditionally stable ADI compact scheme for the two spacedimensional linear hyperbolic equation, Int. J. Comput. Math. 87:10, 2259-2267,2010.

14 D. W. Peaceman, H. H. Rachford. The Numerical Solution of Parabolic and Elliptic Differential Equations, Journal of the Society for Industrial and Applied Mathematics, 3.1, 1955, issn: 03684245. url: http://www.jstor.org/stable/2098834

15 A. Klockner, T.Warburton, J. Bridge, and J.S. Hesthaven. Nodal discontinuous Galerkin methods on graphics processors, J. Comput. Phys., 228: 21,78637882,2009.

16 N. Bell, M. Garland. Efficient sparse matrix-vector multiplication on CUDA, NVIDIA Technical Report, 2008.

17 E. Elsen, P. LeGresley, E. Darve. Large calculation of the flow over a hypersonic vehicle using a GPU, J. Comput. Phys, 227,1014810161, 2008.

18 Y. Zhang, J. Cohen, J. Owens, Fast tridiagonal solvers on the GPU, ACM Signplan Notices, 45:5,127136,2010

19 Y. Zhang, J. Cohen, A. Davidson, J. Owens, A hybrid method for solving tridiagonal systems on the GPU, GPU Computing Gems Jade Edition, 117,2011.

20 A. Davidson, J. Owens Register packing for cyclic reduction: a case study, Proceedings of the FourthWorkshop on General Purpose Processing on Graphics Processing Units, ACM, 4,2011.

21 A. Davidson, Y. Zhang, J. Owens An auto-tuned method for solving large tridiagonal systems on the GPU, Parallel and Distributed Processing Symposium (IPDPS), IEEE International, IEEE, 2011,956965,2011.

22 D. Goddeke, R. Strzodka. Cyclic reduction tridiagonal solvers on GPUs applied to mixedprecision multigrid, Parallel and Distributed Systems, IEEE Transactions, 22:1, 2232, 2011.

23 H. Kim, S.Wu, L. Chang, W. Hwu. A scalable tridiagonal solver for GPUs, Parallel Processing (ICPP), 2011 International Conference on, IEEE, 444453, 2011.

24 N. Sakharnykh. Tridiagonal solvers on the GPU and applications to fluid simulation, GPU Technology Conference, 2009.

25 Z. Wei, B. Jang, Y. Zhang, Y.Jia. Parallelizing Alternating Direction Implicit Solver on GPUs, International Conference on Computational Science, ICCS, Procedia Computer Science 18, 389398, 2013.

26 M. A. Diaz, M. Solovchuk, W.H. Tony Sheu. High Performance MultiGPU Solver for Describing Nonlinear Acoustic Waves in Homogeneous Thermoviscous Media, Computers and Fluids, doi: 10.1016/j.compuid.2018.03.008,2018

27 D. Michea, D. Komatitsch. Accelerating a three-dimensional finite-difference wave propagation code using GPU graphics cards, Geophys. J. Int, 182, 389402,2010

28 R. Mehra, N. Raghuvanshi, L. Savioja, M. Lin, D. Manocha. An efficient GPU-based time domain solver for the acoustic wave equation Applied Acoustics, 73, 8394, 2012

29 M. Lastra, M.J.Castro, C. Ureaa, M.Asuncin. Efficient multilayer shallow-water simulation system based on GPUs, Mathematics and Computers in Simulation, 148, 48-65,2018

30 A. Lacasta, M. M. Hernandez, J. Murillo, P. G. Navarro. GPU implementation of the 2D shallow water equations for the simulation of rainfall/runo_ events Environ Earth Sci doi: 10.1007/s12665-015-4215-z

31 R. M. Weiss, J. Shragge. Solving 3D anisotropic elastic wave equations on parallel GPU devices GEOPHYSICS, 78:2, 19,2013.

32 I.J.D. Craig, A.D. Sneyd. An alternating-direction implicit scheme for parabolic equations with mixed derivatives. Computers and Mathematics with Applications, 16:4, 341350, 1988. issn: 0898-1221. doi: http://dx.doi.org / 10.1016/0898-1221(88) 90150-2.

33 J. Douglas, H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables, Transaction of the American Mathematical Society,82, 421489,1956.

34 Jr. Douglas Jim, James E. Gunn. A general formulation of alternating direction methods. Numerische Mathematik, 6.1,428453.1964

35 R. W. Hockney. A fast direct solution of Poissons equation using Fourier analysis. Journal of the ACM, 12:1, 95113, 1965.

36 NVIDIA, Nvidia, http://www.nvidia.com/, Accessed, 2019.

37 P. Song, Z. Zhang, L. Liang, Q. Zhang, Q. Zhao. Implementation and performance analysis of the massively parallel method of characteristics based on GPU, Annals of Nuclear Energy, 131, 257272, 2019

38 J. Nickolls, I. Buck, M. Garland, K.Skadron. Scalable parallel programming with cuda. Queue, 6:2,4053,2008. doi:http://www.doi.acm.org/10.1145/ 1365490.1365500.

39 NVIDIA TURING GPU ARCHITECTURE https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf

А. АЛТЫБАЙ, Н. ТОҚМАҒАМБЕТОВ, З. СПАБЕКОВА

Әл-Фараби атындағы Қазақ ұлттық университеті, Алматы, Қазақстан

ЕКІ ӨЛШЕМДІ ТОЛҚЫН ТЕҢДЕУІН АЙЫҚЫН ЕМЕС АЙЫРЫМДЫЛЫҚ ТЕҢДЕУІ НЕГІЗІНДЕ ГРАФИКАЛЫҚ ПРОЦЕССОРДЕ (GPU) ЕСЕПТЕУ

Бұл жұмыста біз көптеген инженерлік есептердің негізгі теңдеу болып табылатын екі өлшемді толқын теңдеудің сандық шешуді қарастырамыз. Функцияның жуық шешімі кеңістіктік тордағы дискретті нүктелерден уақыттың дискретті қадамдары негізінде есептеледі. Бастапқы мәндер бастапқы мәннің шартымен беріледі. Алдымен біз дифференциалдық теңдеуді айқын емес айырымдық теңдеулерге қалай айналдыруға болатындығын, сәйкесінше, шешімді есептеу үшін қолдануға болатын ақырлы-айырымдық теңдеулер жиынтығын түсіндіреміз. Содан кейін біз бұл тапсырманы GPU-ге параллельдеу үшін осы алгоритмді өзгертеміз. Параллель алгорит-

мде өнімділікті жоғарлатуға ерекше көңіл бөлінеді. Сонымен қатар, біз GPU-де параллель кодты және орталық процессордың сериялық кодын іске қосамыз, орындалу уақытына қарай үдеуді есептейміз. Біз GPU-дегі параллель коды және орталық процессорда жүктелген сериялық кодын қолдану арқылы алынған нәтижелермен салыстыру арқылы күтілетін нәтижелер беретінін ұсынамыз. Шындығында, кейбір жағдайларда GPU-да есептеулер процессорға қарағанда 22 есе тез орындалады.

***Түйін сөздер**: сандық модельдеу, GPU, CUDA технологиясы, толқын теңдеуі, ақырлы айырмашылық.*

А. АЛТЫБАЙ, Н. ТОКМАГАМБЕТОВ, З. СПАБЕКОВА

Казахский национальный университет им. аль-Фараби, Алматы, Казахстан

РЕШЕНИЕ ДВУМЕРНОГО ВОЛНОВОГО УРАВНЕНИЯ С ИСПОЛЬЗОВАНИЕМ НЕЯВНОЙ РАЗНОСТНОЙ СХЕМЫ НА ГРАФИЧЕСКОМ ПРОЦЕССОРЕ(GPU)

Рассмотрим численную реализацию двумерного волнового уравнения, которое является фундаментальным уравнением во многих инженерных задачах. Приближенное решение функции вычисляется из дискретных точек в пространственной сетке на основе дискретных временных шагов. Начальные значения задаются условием начального условия. Сначала мы объясним, как преобразовать дифференциальное уравнение в неявное уравнение конечных разностей, соответственно, систему уравнений в конечных разностях, которые можно использовать для вычисления приближенного решения. Затем мы изменим этот алгоритм, чтобы распараллелить эту задачу на GPU. Особое внимание уделяется повышению производительности параллельный алгоритм. Кроме того, мы запустим реализованный параллельный код на графическом процессоре и серийный код центрального процессора, рассчитав ускорение на основе времени выполнения. Мы представляем, что параллельный код, который выполняется на GPU, дает ожидаемые результаты, сравнивая наши результаты с результатами, полученными при запуске последовательного кода той же симуляции на CPU. Фактически, в некоторых случаях моделирование на GPU выполняется в 22 раза быстрее, чем на CPU.

***Ключевые слова**: численное моделирование, GPU, технология CUDA, волновое уравнение, конечная разность.*