A Parallel Algorithm for Solving the Advection Equation with a Retarded Argument

Svyatoslav I. Solodushkin[1,2], Arsen A. Sagoyan[1], and Irina F. Iumanova[1]

[1] Ural Federal University, Yekaterinburg, Russia,
[2] IMM UB RAS, Yekaterinburg, Russia
solodushkin_s@mail.ru

Abstract. We describe a parallel implementation of a difference scheme for the advection equation with time delay on a hybrid architecture computation system. The difference scheme has the second order in space and the first order in time and is unconditionally stable. Performance of a sequential algorithm and several parallel implementations with the MPI technology in the C++ language has been studied.

Keywords: parallel numerical methods · difference scheme · advection equation · time delay

1 Introduction and Formal Statement

First order partial differential equations with time delay, also known as advection equations, with distributed parameters arise in the modeling of birds migration, expansion of influence virus and transfer of nuclear particles [1, 2]. Papers which deal with an advection equation with time delay which is also combined with a retardation of a state variable have numerous applications in cell dynamics [3].

The qualitative theory of partial functional differential equations (PFDE) is developed quite well (see, for example, [4] and references therein). Questions of existence, uniqueness, stability and asymptotic behaviour are usually considered by authors. At the same time only a few PFDE could be solved in the the explicit form (analytically), so the elaboration, substantiation and a program realization of numerical methods for these equations are of hight interest.

Especially effective difference schemes for PFDE of parabolic, hyperbolic and advection type were elaborated in [6–10]. The main idea in these works is a separation principle the essence of which is the separation of finite and infinite dimensional components in the structure of PFDE. To take into account the time delay effect, interpolation and extrapolation of discrete prehistory is used. This extrapolation is needed for the realization of implicit methods and allows the authors to avoid the necessity of solving nonlinear systems. Our approach is close to [6, 10] and is based on a combination of the stability verification methods for two-layer difference schemes [11] and the separation principle mentioned above.

A dramatic improvement of supercomputers architecture and their performance led to the increased interest in the elaboration of parallel numerical methods. Below we survey some of approaches. The first class of methods of concern

are a domain decomposition (DD) methods which are based on a physical decomposition of a global solution domain. The very first of them is a classical alternating Schwarz method for the elliptic equation which is sequential in its nature. In order to obtain parallel analogs of Schwarz alternating method many new techniques have been introduced, such as additive Schwarz methods, parallel multilevel precondition algorithms, parallel weighted Schwarz algorithms, parallel subspace correction methods and etc., see e.g. [15]. The parallel DD methods have established themselves as very efficient PDE solution methods [12] and were extended to the advection–diffusion type equation [16, 17].

A parallel implementation of a method of the semi-Lagrangian type for the advection equation was considered in [18]. The difference scheme with variable template is constructed on the base of an integral equality between the neighboring time levels.The proposed approach allows the authors to avoid the Courant-Friedrichs-Lewy restriction on the relation between time step and mesh size.

Time domain decomposition algorithm for the parallel-in-time approximation of solution of advection equation was developed in [19]. It could be interpreted as a multiple shooting method for evolution problem with a particular choice of the approximate Jacobian on a coarse grid. Because this method computes the numerical solution for multiple time steps in parallel, it is categorized as a parallel across the steps method [20]. This is in contrast to approaches using parallelism across the method like parallel Runge-Kutta methods, where independent stages can be computed in parallel or parallel across the system methods like waveform relaxation.

In this article we present a parallel version of one algorithm which is based on a difference scheme, first published in [13].

Let $\tau > 0$ and consider the following advection equation with aftereffect

$$\frac{\partial u(x,t)}{\partial t} + a\frac{\partial u(x,t)}{\partial x} = f(x,t,u(x,t),u_t(x,\cdot)), \tag{1}$$

where $x \in [x_0, X]$ is a state and $t \in [t_0, \theta]$ is time; $u(x,t)$ is an unknown function; $u_t(x,\cdot) = \{u(x,t+\xi),\ -\tau \le \xi < 0\}$ is a prehistory-function of the unknown function to the moment t and $a > 0$ is a constant. Together with the advection equation we have the following initial and the boundary conditions

$$u(x,t) = \varphi(x,t), \quad x \in [x_0, X],\ t \in [t_0 - \tau, t_0], \tag{2}$$

$$u(x_0,t) = g(t), \quad t \in [t_0, \theta]. \tag{3}$$

We adopt the compatibility condition $g(t_0) = \varphi(x_0, t_0)$. Questions of the existence and uniqueness of a solution to the stated boundary value problem (1)–(3) were considered in [4] and we assume that the functional f and functions φ and g are such that problem has a unique solution.

We denote by $\mathcal{Q} = \mathcal{Q}[-\tau, 0)$ the set of functions $u(\xi)$ that are piecewise continuous on $[-\tau, 0)$ with a finite number of points of discontinuity of the first kind and right continuous at the points of discontinuity. We define a norm on $\mathcal{Q}$ by $\|u\|_{\mathcal{Q}} = \sup_{\xi\in[-\tau,0)} |u(\xi)|$. We additionally assume that the functional

$f(x,t,u,v(\cdot))$ is given on $[0,X]\times[t_0,\theta]\times R\times\mathcal{Q}$ and is Lipschitz in the last two arguments:

$$\exists\, L_f\in R\ \ \forall\, x\in[x_0,X],\ t\in[t_0;\theta],\ u^1\in R,\ u^2\in R,\ v^1\in Q,\ v^2\in Q:$$

$$\big|f(x,t,u^1,v^1(\cdot))-f(x,t,u^2,v^2(\cdot))\big|\le L_f\Big(|u^1-u^2|+\|v^1(\cdot)-v^2(\cdot)\|_Q\Big).$$

2 The difference scheme

We consider an equidistant partition of $[x_0,X]$ into parts with step $h=(X-x_0)/N$ and split the segment of variation of the time variable $[t_0,\theta]$ into parts with the step Δ. The uniform grid $\{x_i,t_j\}_{j=0}^{M}{}_{i=0}^{N}$ can be constructed, here $x_i=x_0+ih,\ i=0,\dots,N$, and $t_j=t_0+j\Delta,\ j=0,\dots,M$. Denote by u_j^i approximations of the functions $u(x_i,t_j),\ i=0,\dots,N,\ j=0,\dots,M$, at the nodes. Without loss of generality and to simplify the narration we assume that the value $\tau/\Delta=m$ is a natural number.

Since functional $f(x_i,t_j,u(x_i,t_j),u_{t_j}(x_i,\cdot))$ may depend on values of the function u between grid nodes, interpolation may be needed. For every fixed node (x_i,t_j) and time delay $\xi\in[-\tau,0)$ there are only two possibilities: if $t_j+\xi\le t_0$, interpolation is not needed, we use the initial function, $u(x_{(i)},t_j+\xi)=\varphi(x_{(i)},t_j+\xi)$, otherwise we use the interpolation as described below.

For every fixed time moment $t_j,\ j=1,\dots,M$, we introduce its discrete prehitory

$$\left\{u_k^i\right\}_j=\left\{u_k^i\mid\max\{0,j-m\}\le k\le j\right\}.$$

Definition 1. *A mapping I defined on the set of all admissible discrete prehistories and acting by the rule $I:\left\{u_k^i\right\}_j\to v^{i,j}(\cdot)\in\mathcal{Q}[-\min\{\tau,t_j\},0]$ is called an* interpolation operator *for the discrete history.*

Let us give an example of a concrete interpolation operator, which has the properties required for the numerical method that we are going to construct. For the discrete prehistory $\left\{u_k^i\right\}_j$ we define

$$v^{i,j}(t_j+\xi)=\frac{1}{\Delta}\Big((t_{l+1}-t_j-\xi)\,u_l^i+(t_j+\xi-t_l)\,u_{l+1}^i\Big),\ t_l\le t_j+\xi\le t_{l+1}.\ (4)$$

We say an interpolation operator has order of error p on the exact solution, if there exist constants C_1 and C_2 such that, for all $i=1,\dots,N,\ j=1,\dots,M$, and $t\in[\max\{0,t_j-\tau\},t_j]$ the following inequality holds:

$$\exists\, C_1\in R, C_2\in R\quad\forall\, i=1,\dots,N,\ j=1,\dots,M,\ t\in[\max\{0,t_j-\tau\},t_j]:$$

$$\big|v^{i,j}(t)-u(x_i,t)\big|\le C_1\max_{\max\{0,j-m\}\le l\le j}\big|u_l^i-u(x_i,t_l)\big|+C_2\Delta^p.$$

For example, the operator of interpolation (4) is of the second order.

We consider the following family of difference schemes (parametrized by $0\le s\le 1$), with $j=0,\dots,M-1$:

$$\begin{aligned}
&\frac{u^1_{j+1}-u^1_j}{\Delta}+a\left(s\,\frac{-4u^0_{j+1}-2h/a(f^0_{j+1}-\dot g_{j+1})+4u^1_{j+1}}{2h}+\right.\\
&\qquad\left.+\;(1-s)\,\frac{-4u^0_j-2h/a(f^0_j-\dot g_j)+4u^1_j}{2h}\right)=f^1_j,\\
&\frac{u^i_{j+1}-u^i_j}{\Delta}+a\left(s\,\frac{u^{i-2}_{j+1}-4u^{i-1}_{j+1}+3u^i_{j+1}}{2h}+\right.\\
&\qquad\left.+\;(1-s)\,\frac{u^{i-2}_j-4u^{i-1}_j+3u^i_j}{2h}\right)=f^i_j,\quad i=2,\dots,N,
\end{aligned}\tag{5}$$

with the initial and boundary conditions

$$u^i_0=\varphi(x_i,t_0),\quad i=0,\dots,N;\qquad v^{i,0}(t)=\varphi(x_i,t),\quad t<t_0,\quad i=0,\dots,N,\tag{6}$$

$$u^0_j=g_0(t_j),\quad j=0,\dots,M.\tag{7}$$

Here $f^i_j=f\left(x_i,t_j,u^i_j,v^{i,j}(\cdot)\right)$ is the value of the functional f, calculated on an approximate solution, $v^{i,j}(\cdot)$ is the result of an interpolation, $\dot g_j=g'(t_0+j\Delta)$. For constructing a numerical method, we additionally assume that $g(t)$ is a differentiable function.

Let us explain the way in which we have obtained the scheme. The derivative $\partial u/\partial t$ in (1) is approximated by a finite difference over two nodes. For nodes $(i,j), i=2,\dots,N, j=0,\dots,M-1$, the derivative $\partial u/\partial t$ is approximated by a finite difference over three nodes on the right edge. For $i=1$ such an approximation requires to calculate u^{-1}_j. For $i=1$ we apply the approximation over three nodes with the double node $(0,j)$:

$$\frac{\partial u^1_j}{\partial x}\approx\frac{1}{2h}\left(-4u^0_j-2h\frac{\partial u^0_j}{\partial x}+4u^1_j\right).$$

Because of (1) we have $\dfrac{\partial u^0_j}{\partial x}=\dfrac{1}{a}\left(f^0_j-\dfrac{\partial u^0_j}{\partial t}\right)$, and due to (3) we obtain $\dfrac{1}{a}\left(f^0_j-\dot g_j\right)$.

Theorem 1. *Let the exact solution $u(x,t)$ of problem* (1) *be thrice continuously differentiable with respect to state variable x, twice continuously differentiable with respect to time t and the first derivative of the solution with respect to x is continuously differentiable in t. Then if $2s>1$ method* (5) *converges with order $h^2+\Delta$, i. e. there exists a constant C such that $\|u(x_i,t_j)-u^i_j\|\le C(h^2+\Delta)$ for all $i=0,\dots,N$ and $j=0,\dots,M$.*

3 Formulation of parallel algorithm

The method (5) allows one to find the items u_j^i recurrently and could be programmed in the form of two nested loops with informational dependance on each other. To find u_i^{j+1}, $i \geq 2$, three values from the previous time layer $u_{i-2}^j, u_{i-1}^j, u_i^j$ and two from the current layer $u_{i-2}^{j+1}, u_{i-1}^{j+1}$ are required.

For efficient parallel implementation it is necessary to organize the calculation so that the influence of the informational dependencies were eliminated. We will use the approach described in [21]. The essence of proposed method is to find a *separable line*, that passes through the grid nodes so, that the nodes on a given line are informationally dependent on the nodes that lie on one side of the line and only on them. At each step, one calculates in parallel the values at the nodes which are lying on the separable line.

We share the task of finding the approximate solution in the grid nodes between processes. Namely, we cut the grid into the K spatial layers. We call the k-th, $k = 1, .., K$, spatial layer the set of nodes of the grid as follow

$$\{(i,j)_k \,|i = (k-1)N/K + 1, .., kN/K, j = 1, .., M\}.$$

Let the k-th process calculates a solution in the k-th spatial layer.

The first process has no informational dependencies on other processes. To find the solution on the $(j+1)$-th time layer the k-th process should receive form the $(k-1)$-th process two values: $u_{(k-1)N/K-1}^{j+1}$ and $u_{(k-1)N/K}^{j+1}$; this constitutes the informational dependencies. Notice, that values $u_{(k-1)N/K-1}^{j}$ and $u_{(k-1)N/K}^{j}$ have already been transferred to k-th process and the value $u_{(k-1)N/K+1}^{j}$ have been found.

We call the block $B_{k,j}$, $k \geq 2$, the set of grid nodes

$$\{(i,j) \,|i = (k-1)N/K + 1, .., kN/K, j = k\}.$$

To find the solution in the block $B_{k,j}$, $k \geq 2$, it is necessary to complete the calculation in the block $B_{k-1,j}$. The blocks, where we search the solution in parallel, are on the same separable line. In other words we can iterate through the grid nodes in a direction perpendicular to this line and perform all the operations on the line at each iteration in parallel. Since the task of finding the solution in a single block is complex and data transfer between adjacent processes requires few time, the efficiency of parallelization is high.

Let the K is the number of processes in the pool. In cases of practical importance $K \ll m$, therefore the situation when the first process has finished its work and the last process has not started yet is excluded. Let us describe the implementation. In the line 4 the blocking function "receive" is used and non-blocking function "send" in the line 9.

```
for (j from 1 to M){
 for (k from 1 to min(j,K)){//To perform in the k-th process
  if(k>1)
    Receive last two dots of Block [k-1,j-k+1];
```

```
  else
    Use boundary function;
  Perform the calculations in the Block [k,j-k+1];
  if (k<K)
    Send last two dotes of the Block [k,j-k+1]
    to (k+1)-th process;
 }
}
```

The idea of the algorithm is depicted in Fig. 1. Here arrows symbolise the direction in which the approximate solution is sought; little empty rounds represent data, i.e. approximate solution in two nodes, which $(k-1)$-th process send to k-th one.

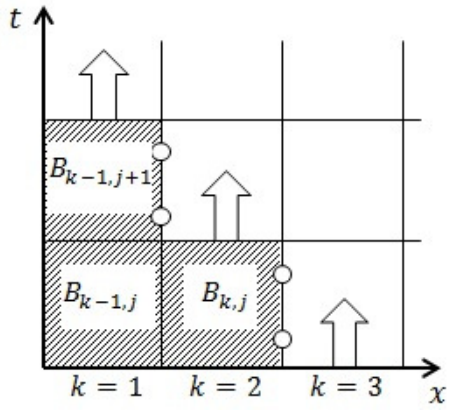

Fig. 1. Decomposition of the domain into the blocks

4 Numerical experiments: results and analysis

To verify the parallel implementation of the algorithm in question we used the test equation

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial u(x,t)}{\partial x} = \cos x \cos t - u(x, t-\pi/2) + \psi(x,t),$$

with the following initial and the boundary conditions

$$u(x,t) = \sin x \cos t, \quad 0 \le x \le \pi,\ \pi/2 \le t \le 0,$$
$$u(0,t) = 0, \quad 0 \le t \le 10\pi.$$

Here the function ψ is defined as follow

$$\psi(x,t) = 10^{-10} \ln\left(\sin^2\left(\arctan\sqrt{\frac{x^2+t^2}{t^2+10}}\right) + \cos^2\left(\arctan\sqrt{\frac{x^2+t^2}{t^2+10}}\right)\right)$$

and is identically equal to zero; it was added in the right-hand side of the equation to increase the computational complexity in each iteration.

This initial-boundary problem has the exact solution $u(x,t) = \sin x \cos t$.

The calculations were performed on a cluster of Ural Branch of RAS "Uran". The following hardware and software configuration were used: CPU INTEL XEON E5450, 2×4 cores, 3GHz; Cache memory 2×6 MB Level 2 cache (5400 Sequence); RAM 16 Gb DDR2; OS Linux 2.6.32; language: C++ with Intel C++ compiler (ICC) v14.0.0; MPI library MVAPICH2 Intel 13.0. We call our computation system architecture as hybrid, since it has distributed memory with shared memory on each computing node.

In the previous paragraph we used the term "process" to refer an abstract unit that performs a calculation; now let us concretize this notion. According to the principles of the number of processes is equal to the number of computational nodes.

Let us present the results of computational experiments for the test equation (see Tab. 1). Recall, that we consider an equidistant partition of $[x_0, X]$ and $[t_0, \theta]$, the grid consists of N and M dotes with respect to space and time. The parameter M was constant through the experiments, $M = 4000$, parameter N ranges from 500 to 50000 dotes. The number of processes is reported in the first line and ranges from 1 to 64. Time in seconds is reported in the table cells. Speedup is calculated for multi processes variants (compared to the single process variant).

Table 1. Numerical results: time and speedup

Num. of processes	1	2	4	8	16	32	64
N=500	0.16	0.09	0.07	0.03	0.03	0.02	0.05
N=5000	1.68	1.29	0.74	0.47	0.31	0.21	0.21
N=50000	19.79	12.92	7.25	4.22	2.29	1.40	1.38
Speedup		1.5	2.7	4.7	6.8	14.1	14.3

To characterize the quality of the parallel algorithm the two types of scalability were estimated. Weak scaling is defined as how the solving time varies with the number of processors for a fixed problem size per processor. Weak scaling could be numerically characterized by the function $eff_{weak}(N) = T(N)/T(N_0)$, where N_0 is a fixed initial number of processors, in our case $N_0 = 2$, and N is a current number of processors, $T(N)$ is a time of the solution of the task using N processors. An ideal $eff_{weak}(N)$ is identically equal to one and is unreachable according to Amdahl's Law.

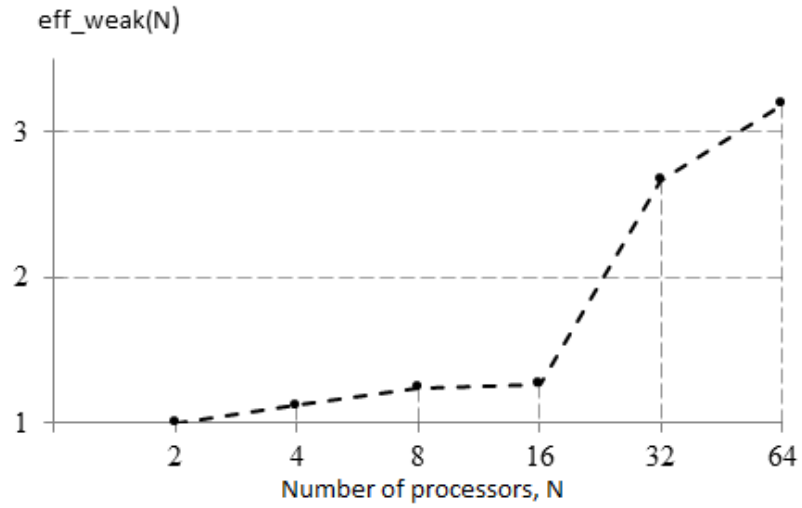

Fig. 2. Weak scaling

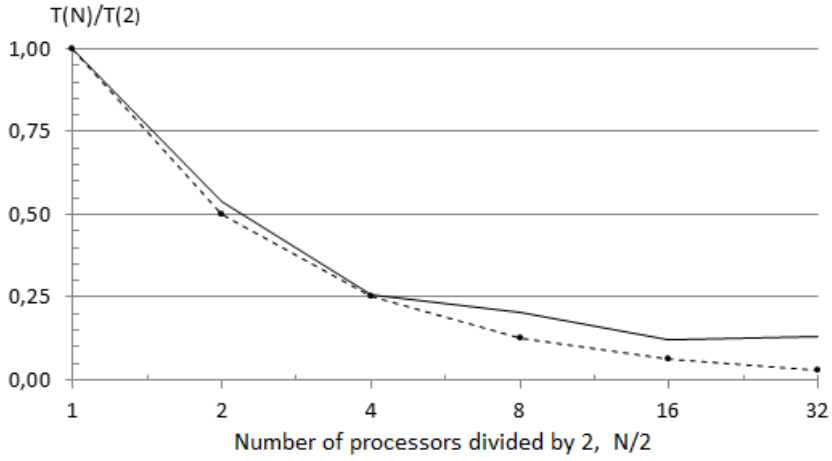

Fig. 3. Strong scaling

Strong scaling is defined as how the solving time varies with the number of processors for a fixed total problem size. Strong scaling could be numerically characterized by the function $eff_{strong}(N) = T(N)/(N_0\,T(N_0))$. In the case of perfect scalability the computation time is inversely proportional to the number of computing nodes. To estimate the strong scalability the grid of size 50000×5000 was used.

Experimental assessment of function eff_{strong} is shown in Fig. 3 (solid line); it is extremely close to perfect scaling (dashed line), which is shown for comparison.

5 Conclusion

For the advection equation with time delay the difference scheme originally proposed in [13] was parallelized. The proposed method of parallelization is based on the geometric partitioning of the computational grid into blocks that have weak informational dependencies.

The parallel version was implemented for the "Uran" supercomputer and could be executed on any number of processors; MPI technology was used. Performed tests proved showed high scaling of the method. We observed the fourteen-fold decrease in time compared with the sequential version. Numerical results are consistent with the theory.

Acknowledgements

This research is supported by RFBR 14-01-00065 and 13-01-00089. We acknowledge the support by the program 02.A03.21.0006 on 27.08.2013.

References

1. Bart A. van Tiggelen, Skipetrov S. (Eds.) Wave Scattering in Complex Media: From Theory to Applications Proceedings of the NATO Advanced Study Institute, Corsica, France 10-22 June 2002.
2. S.A. Gourley, Liu R., Wu J. *Spatiotemporal Distributions of Migratory Birds Patchy Models with Delay* SIAM J. Appl. Dyn. Syst., 9(2), 589-610.
3. T. Luzyanina, J. Cupovic, B. Ludewig and G. Bocharov, *Mathematical models for CFSE labelled lymphocyte dynamics: asymmetry and time-lag in division*, J. Math. Biol. (2013), **69(6-7)** 1547-83. doi: 10.1007/s00285-013-0741-z.
4. J. Wu, *Theory and applications of partial functional differential equations,* Springer-Verlag, New York, 1996.
5. V. G. Pimenov, *General Linear Methods for the Numerical Solution of Functional-Differential Equations*, Differential Equations **37**(1) (2001) 116-127
6. V. G. Pimenov and A. B. Lozhnikov, *Difference schemes for the numerical solution of the heat conduction equation with aftereffect*, Proceedings of the Steklov Institute of Mathematics **275** (2011) 137-148.

7. V. G. Pimenov and A. V. Lekomtsev, *Convergence of the Alternating Direction Methods for the Numerical solution of a Heat Conduction Equation with Delay*, Proceedings of the Steklov Institute of Mathematics **272** (2011) 101-118.
8. V. G. Pimenov and E. E. Tashirova *Numerical methods for solving a hereditary equation of hyperbolic type*, Proceedings of the Steklov Institute of Mathematics **281** (2013) 126-136.
9. V. G. Pimenov and A. Lozhnikov *Numerical methods for evolutionary equations with delay and software package PDDE*, Lecture Notes in Computer Science **8236** (2013) 437-444.
10. V. G. Pimenov and S.Sviridov *Numerical methods for advection equations with delay* American Institute of Physics. Conference Proceeding. Proceedings of 40th International Conference Applications of Mathematics in Engineering and Economics. **1631** 114 (2014) P. 114-121.
11. A. A. Samarskii, *Theory of Difference Schemes*, Nauka, Moscow, 1989 [in Russian].
12. B. F. Smith, P. E. Bjorstad, and W. Gropp. Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge University Press, 1996.
13. S. I. Solodushkin, *A difference scheme for the numerical solution of an advection equation with aftereffect*, Russian Mathematics, Allerton Press **57** (2013) 65-70.
14. S. I. Solodushkin, I. F. Yumanova and R. H. De Staelen, *First order partial differential equations with time delay and retardation of a state variable*, Journal of Computational and Applied mathematics **289** (2015) 322-330.
15. Bramble, J.H., Pasciak, J.E., Xu, J.: Parallel multilevel preconditioners. Math.Comput. 55, 1-22 (1990)
16. P.N. Vabishchevich, Parallel domain decomposition algorithms for parabolic problems, Mat. Model. 9 (5) (1997) 77-86.
17. Yang, D.P.: Parallel domain decomposition procedures of improved D-D type for parabolic problems. J. Comput. Appl. Math. 233, 2779-2794 (2010)
18. Efremov A. et al. A Computational Realization of a Semi-Lagrangian Method for Solving the Advection Equation. Journal of Applied Mathematics Volume 2014, Article ID 610398.
19. Gander M. J. Analysis of the parareal algorithm applied to hyperbolic problems using characteristics. Bol. Soc. Esp. Math. Apl. No 42, 5–19 (2008).
20. Burrage K. Parallel methods for ODEs. Advances in Computational Mathematics. V. 7, Issue (1-2), 1–31 (1997).
21. McCool M., Robison A. D., Reinders J. Structured Parallel Programming. – Waltham, USA: Elsevier, 2012. 406 p.