ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Chikitkin A.V., Rogov B.V.

Two variants of parallel implementation of high-order accurate bicompact schemes for multi-dimensional inhomogeneous transport equation

Recommended form of bibliographic references: Chikitkin A.V., Rogov B.V. Two variants of parallel implementation of high-order accurate bicompact schemes for multi-dimensional inhomogeneous transport equation // Keldysh Institute Preprints. 2018. No. 177. 24 p. doi:**10.20948/prepr-2018-177-e**
URL: **http://library.keldysh.ru/preprint.asp?id=2018-177&lg=e**

KELDYSH INSTITUTE OF APPLIED MATHEMATICS

Russian Academy of Sciences

A.V.Chikitkin, B.V.Rogov

Two variants of parallel implementation of high-order accurate bicompact schemes for multi-dimensional inhomogeneous transport equation

Moscow — 2018

Aleksandr Viktorovich Chikitkin, Boris Vadimovich Rogov

Two variants of parallel implementation of high-order accurate bicompact schemes for multi-dimensional inhomogeneous transport equation

In this paper, we compare the efficiency of two parallel algorithms for solution of the equations of multidimensional high-order accurate bicompact schemes for a multidimensional inhomogeneous transport equation. The first one is a space-marching algorithm for computing non-factorized schemes, and the second one is based on the approximate factorization of multidimensional schemes. The latter algorithm uses iterations to preserve the high (higher than second) order of accuracy of bicompact schemes in time. The convergence of these iterations is proved for a nonstationary two-dimensional and three-dimensional linear inhomogeneous transport equation with constant positive coefficients. Model computations show that the factorization scheme is preferable from the point of view of parallel implementation.

Key words: multidimensional inhomogeneous transport equation, bicompact schemes, parallel algorithms, iterative approximate factorization method

Чикиткин А.В., Рогов Б.В.

Два варианта параллельной реализации высокоточных бикомпактных схем для многомерного неоднородного уравнения переноса

В работе проведено сравнение эффективности двух параллельных алгоритмов решения уравнений высокоточных бикомпактных схем для многомерного неоднородного уравнения переноса. Первый из них есть пространственный маршевый алгоритм счета нефакторизованных схем, а второй основан на приближенной факторизации многомерных схем. В последнем алгоритме используются итерации для сохранения высокого (выше второго) порядка точности бикомпактных схем по времени. Доказана сходимость этих итераций для нестационарного двухмерного и трехмерного линейного неоднородного уравнения переноса с постоянными положительными коэффициентами. Модельные расчёты показывают, что алгоритм на основе факторизованных схем является предпочтительным с точки зрения параллельной реализации.

Ключевые слова: многомерное неоднородное линейное уравнение переноса, бикомпактные схемы, параллельные алгоритмы, метод итерируемой приближенной факторизации

This work was supported by the Russian Foundation for Basic Research, project no. 18-01-00857-a.

1. Introduction

The linear inhomogeneous transport equation needs to be solved in problems related to radiative transfer and transport of uncharged particles (neutrons). Such problems arise in many areas of science and engineering raging from computing atmospheric radiation [1] and thermonuclear targets [2] to computing nuclear reactor cores [3].

The solution of the transport equation is a distribution function in a phase space and, according to its physical interpretation, it must be nonnegative. In many cases, this equation has to be solved in a strongly heterogeneous medium with numerous contact discontinuities (atmospheric clouds or various materials in a nuclear reactor core). Accordingly, a scheme for computing the transport equation must satisfy the following requirements. First, it must be conservative and its resolution must be sufficient for adequately taking into account the discontinuities in the coefficients and sources of the equation. This goal is best achieved when the governing equation is approximated on a stencil lying within a single mesh cell. Second, the scheme must ensure that the numerical solution is positive.

A scheme satisfying these requirements was proposed in [4] for the one-dimensional (1D) homogeneous linear transport equation. It is constructed using the method of lines on a minimum spatial stencil consisting of two nodes and is known as a bicompact scheme. In semidiscrete form, this scheme has fourth-order accuracy in space, which is achieved by supplementing the sought functions with an auxiliary dependent variable, namely, the primitive of the main sought function. For this dependent variable, an additional discrete equation is derived that approximates an integral consequence of the basic differential equation. Since only a finite difference of the primitive was involved in the scheme in [4], starting from [5], the bicompact scheme was written in a computationally more convenient form: the finite differences of the primitive were replaced by integral averages of the sought function over grid cells. Later, bicompact schemes were constructed for the two-dimensional (2D) and three-dimensional (3D) nonstationary inhomogeneous linear transport equations [6, 7] and for systems of nonstationary multidimensional quasilinear hyperbolic equations [8]. Note that, in bicompact schemes [8, 9] for quasilinear hyperbolic equations, an increase in the order of accuracy is ensured not with the help of auxiliary integral averages of the sought function over grid cells, but rather with the help of auxiliary values of the sought function at half-integer spatial grid nodes. To find these quantities, additional difference equations are derived that approximate consequences of the basic differential equations. The equations of semidiscrete bicompact schemes are usually integrated with respect to time using A- and L-stable diagonally implicit Runge–Kutta (DIRK) methods [4–8]. In the case of smooth solutions, the important advantages of bicompact schemes include the preservation of their order of accuracy on an arbitrary nonuniform spatial grid and the low cost of the marching method used to solve the equations of the scheme in each spatial variable. In [6, 8] high-order accurate bicompact schemes were used to construct hybrid schemes for computing discontinuous solutions of hyperbolic equations.

For efficient computations of the nonstationary multidimensional inhomogeneous transport equation, a parallel space marching algorithm for computing difference equations of high-order accurate bicompact schemes was proposed in [6]. In this paper, we consider another parallel algorithm for solving the difference equations of high-order accurate bicompact schemes. It is based on an approximate factorization of multidimensional difference operators. This algorithm makes use of iterations in order to preserve the high (higher than the second) order of accuracy of the schemes in time [9, 10]. We compare the efficiency of two parallel algorithms for computing the equations of multidimensional bicompact schemes, namely, the space marching algorithm for unfactorized schemes and an algorithm for computing factorized schemes.

2. Bicompact schemes

Without loss of generality, the basic points of the method used to construct multidimensional bicompact schemes [6, 7] can be described as applied to the three-dimensional nonstationary transport equation

$$L_3(u) = 0,\ L_3(u) \equiv \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} + c\frac{\partial u}{\partial z} + \sigma u - q(x, y, z, t) \qquad (1)$$

and its two-dimensional analogue

$$L_2(u) = 0,\ L_2(u) \equiv \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} + \sigma u - q(x, y, t), \qquad (2)$$

assuming that a, b, c, and σ are constant positive parameters.

Semidiscrete schemes for the transport equation are derived by discretizing spatial derivatives on a minimum stencil by applying the method of lines. The spatial stencil consists of two nodes x_{j+1}, x_j in the one-dimensional (1D) case, four nodes (x_j, y_k), (x_{j+1}, y_k), (x_j, y_{k+1}), (x_{j+1}, y_{k+1}) in the two-dimensional (2D) case, and eight nodes $(x_\alpha, y_\beta, z_\gamma)$, $\alpha = j, j+1$, $\beta = k, k+1$, $\gamma = l, l+1$, in the three-dimensional (3D) case. The schemes are called *bicompact* because the stencil consists of two points in each spatial direction. A bicompact scheme will be derived for the 3D case. Since we use the minimum spatial stencil, the derivation procedure is valid in the general case of nonuniform Cartesian grids with steps $h_x \equiv h_{x,j+1/2} = x_{j+1} - x_j$, $h_y \equiv h_{y,k+1/2} = y_{k+1} - y_k$, $h_z \equiv h_{z,l+1/2} = z_{l+1} - z_l$.

To obtain a semidiscrete scheme of high order of accuracy in space, along with the values of the sought function u at the stencil nodes,

$$u_{\alpha,\beta,\gamma},\ \alpha = j, j+1,\ \beta = k, k+1,\ \gamma = l, l+1, \qquad (3)$$

we introduce the following auxiliary values:

(1) the integral averages over twelve edges of the 3D cell $G_{3D} = [x_j, x_{j+1}] \times [y_k, y_{k+1}] \times [z_l, z_{l+1}]$:

$$\bar{u}^x_{j+1/2,\beta,\gamma} = \frac{1}{h_x} \int_{x_j}^{x_{j+1}} u(x, y_\beta, z_\gamma, t)dx, \quad \bar{u}^y_{\alpha,k+1/2,\gamma} = \frac{1}{h_y} \int_{y_k}^{y_{k+1}} u(x_\alpha, y, z_\gamma, t)dy,$$
$$\bar{u}^z_{\alpha,\beta,l+1/2} = \frac{1}{h_z} \int_{z_l}^{z_{l+1}} u(x_\alpha, y_\beta, z, t)dz, \quad \alpha = j, j+1, \ \beta = k, k+1, \ \gamma = l, l+1, \tag{4}$$

(2) the integral averages over six faces of the cell:

$$\bar{u}^{xy}_{j+1/2,k+1/2,\gamma} = \frac{1}{h_x h_y} \int_{x_j}^{x_{j+1}} \int_{y_k}^{y_{k+1}} u(x, y, z_\gamma, t)dydx, \quad \bar{u}^{xz}_{j+1/2,\beta,l+1/2} = \frac{1}{h_x h_z} \int_{x_j}^{x_{j+1}} \int_{z_l}^{z_{l+1}} u(x, y_\beta, z, t)dzdx,$$
$$\bar{u}^{yz}_{\alpha,k+1/2,l+1/2} = \frac{1}{h_y h_z} \int_{y_k}^{y_{k+1}} \int_{z_l}^{z_{l+1}} u(x_\alpha, y, z, t)dzdy, \quad \alpha = j, j+1, \ \beta = k, k+1, \ \gamma = l, l+1, \tag{5}$$

(3) the integral average over this cell:

$$\bar{u}^{xyz}_{j+1/2,k+1/2,l+1/2} = \frac{1}{h_x h_y h_z} \int_{x_j}^{x_{j+1}} \int_{y_k}^{y_{k+1}} \int_{z_l}^{z_{l+1}} u(x, y, z, t)dzdydx . \tag{6}$$

In formulas (3)–(6), the quantities $u_{j+1,k+1,l+1}$, $\bar{u}^x_{j+1/2,k+1,l+1}$, $\bar{u}^y_{j+1,k+1/2,l+1}$, $\bar{u}^z_{j+1,k+1,l+1/2}$, $\bar{u}^{xy}_{j+1/2,k+1/2,l+1}$, $\bar{u}^{xz}_{j+1/2,k+1,l+1/2}$, $\bar{u}^{yz}_{j+1,k+1/2,l+1/2}$, and $\bar{u}^{xyz}_{j+1/2,k+1/2,l+1/2}$ are unknown functions of time for $a > 0$, $b > 0$, and $c > 0$ (the other signs are treated in a similar manner), while the other quantities are known either from the computations in the neighboring cells or from the initial and boundary conditions. Equations for these variables can be obtained by integrating the original equation (1) and its seven independent differential consequences

$$\frac{\partial L_3(u)}{\partial x} = 0, \quad \frac{\partial L_3(u)}{\partial y} = 0, \quad \frac{\partial L_3(u)}{\partial z} = 0,$$
$$\frac{\partial^2 L_3(u)}{\partial x \partial y} = 0, \quad \frac{\partial^2 L_3(u)}{\partial x \partial z} = 0, \quad \frac{\partial^2 L_3(u)}{\partial y \partial z} = 0, \quad \frac{\partial^3 L_3(u)}{\partial x \partial y \partial z} = 0 \tag{7}$$

over the spatial cell G_{3D}.

The results of integrating Eqs. (1) and (7) over G_{3D} can be represented in the compact form

$$
\begin{aligned}
&A^zA^yA^x(u_t)_C+(aA^zA^y\Lambda_1^x+bA^z\Lambda_1^yA^x+c\Lambda_1^zA^yA^x+\sigma A^zA^yA^x)u_C=A^zA^yA^xq_C,\\
&A^zA^y\Lambda_1^x(u_t)_C+(aA^zA^y\Lambda_2^x+bA^z\Lambda_1^y\Lambda_1^x+c\Lambda_1^zA^y\Lambda_1^x+\sigma A^zA^y\Lambda_1^x)u_C=A^zA^y\Lambda_1^xq_C,\\
&A^z\Lambda_1^yA^x(u_t)_C+(aA^z\Lambda_1^y\Lambda_1^x+bA^z\Lambda_2^yA^x+c\Lambda_1^z\Lambda_1^yA^x+\sigma A^z\Lambda_1^yA^x)u_C=A^z\Lambda_1^yA^xq_C,\\
&A^z\Lambda_1^y\Lambda_1^x(u_t)_C+(aA^z\Lambda_1^y\Lambda_2^x+bA^z\Lambda_2^y\Lambda_1^x+c\Lambda_1^z\Lambda_1^y\Lambda_1^x+\sigma A^z\Lambda_1^y\Lambda_1^x)u_C=A^z\Lambda_1^y\Lambda_1^xq_C,\\
&\Lambda_1^zA^yA^x(u_t)_C+(a\Lambda_1^zA^y\Lambda_1^x+b\Lambda_1^z\Lambda_1^yA^x+c\Lambda_2^zA^yA^x+\sigma\Lambda_1^zA^yA^x)u_C=\Lambda_1^zA^yA^xq_C,\\
&\Lambda_1^zA^y\Lambda_1^x(u_t)_C+(a\Lambda_1^zA^y\Lambda_2^x+b\Lambda_1^z\Lambda_1^y\Lambda_1^x+c\Lambda_2^zA^y\Lambda_1^x+\sigma\Lambda_1^zA^y\Lambda_1^x)u_C=\Lambda_1^zA^y\Lambda_1^xq_C,\\
&\Lambda_1^z\Lambda_1^yA^x(u_t)_C+(a\Lambda_1^z\Lambda_1^y\Lambda_1^x+b\Lambda_1^z\Lambda_2^yA^x+c\Lambda_2^z\Lambda_1^yA^x+\sigma\Lambda_1^z\Lambda_1^yA^x)u_C=\Lambda_1^z\Lambda_1^yA^xq_C,\\
&\Lambda_1^z\Lambda_1^y\Lambda_1^x(u_t)_C+(a\Lambda_1^z\Lambda_1^y\Lambda_2^x+b\Lambda_1^z\Lambda_2^y\Lambda_1^x+c\Lambda_2^z\Lambda_1^y\Lambda_1^x+\sigma\Lambda_1^z\Lambda_1^y\Lambda_1^x)u_C=\Lambda_1^z\Lambda_1^y\Lambda_1^xq_C,
\end{aligned}
\tag{8}
$$

if we introduce the operators

$$
\begin{aligned}
&A^xu_{j+1/2,\beta,\gamma}=\bar{u}^x_{j+1/2,\beta,\gamma},\ A^yu_{\alpha,k+1/2,\gamma}=\bar{u}^y_{\alpha,k+1/2,\gamma},\ A^zu_{\alpha,\beta,l+1/2}=\bar{u}^z_{\alpha,\beta,l+1/2},\\
&\Lambda_1^xu_{j+1/2,\beta,\gamma}=(u_{j+1,\beta,\gamma}-u_{j,\beta,\gamma})/h_x,\\
&\Lambda_1^yu_{\alpha,k+1/2,\gamma}=(u_{\alpha,k+1,\gamma}-u_{\alpha,k,\gamma})/h_y,\\
&\Lambda_1^zu_{\alpha,\beta,l+1/2}=(u_{\alpha,\beta,l+1}-u_{\alpha,\beta,l})/h_z,\\
&\Lambda_2^xu_{j+1/2,\beta,\gamma}=6(u_{j+1,\beta,\gamma}-2\bar{u}^x_{j+1/2,\beta,\gamma}+u_{j,\beta,\gamma})/h_x^2,\\
&\Lambda_2^yu_{\alpha,k+1/2,\gamma}=6(u_{\alpha,k+1,\gamma}-2\bar{u}^y_{\alpha,k+1/2,\gamma}+u_{\alpha,k,\gamma})/h_y^2,\\
&\Lambda_2^zu_{\alpha,\beta,l+1/2}=6(u_{\alpha,\beta,l+1}-2\bar{u}^z_{\alpha,\beta,l+1/2}+u_{\alpha,\beta,l})/h_z^2.
\end{aligned}
\tag{9}
$$

In Eqs. (8), $u_t\equiv du/dt$ and $C=(j+1/2,k+1/2,l+1/2)$ is a multi-index. The operators $A^r,\Lambda_1^r,\Lambda_2^r$ ($r=x,y,z$) are interpreted as the operators of averaging and of the first and second difference derivatives, respectively.

The first equation in (8) is the result of the exact integration of the basic equation (1), while the other equations in (8) are derived from (7) with the use of the Euler–Maclaurin quadrature rules

$$
\begin{aligned}
&\int_{x_j}^{x_{j+1}}udx=\frac{h_x}{2}(u_{j+1}+u_j)-\frac{h_x^2}{12}(\partial_xu_{j+1}-\partial_xu_j)+O(h_x^5),\quad \partial_xu\equiv\frac{\partial u}{\partial x},\\
&\int_{y_k}^{y_{k+1}}udy=\frac{h_y}{2}(u_{k+1}+u_k)-\frac{h_y^2}{12}(\partial_yu_{k+1}-\partial_yu_k)+O(h_y^5),\quad \partial_yu\equiv\frac{\partial u}{\partial y},\\
&\int_{z_l}^{z_{l+1}}udz=\frac{h_z}{2}(u_{l+1}+u_l)-\frac{h_z^2}{12}(\partial_zu_{l+1}-\partial_zu_l)+O(h_z^5),\quad \partial_zu\equiv\frac{\partial u}{\partial z}
\end{aligned}
\tag{10}
$$

and have the fourth order of accuracy in space.

The equations of a semidiscrete 2D bicompact scheme can be obtained by integrating the original equation (2) and its three independent differential consequences

$$\frac{\partial L_2(u)}{\partial x}=0,\quad \frac{\partial L_2(u)}{\partial y}=0,\quad \frac{\partial^2 L_2(u)}{\partial x\partial y}=0 \tag{11}$$

over the spatial cell $G_{2D}=[x_j,x_{j+1}]\times[y_k,y_{k+1}]$. A semidiscrete bicompact scheme for Eq. (2) consists of four ordinary differential equations (ODEs)

$$\begin{aligned}
&A^yA^x(u_t)_C+(aA^y\Lambda_1^x+b\Lambda_1^yA^x+\sigma A^yA^x)u_C=A^yA^xq_C,\\
&A^y\Lambda_1^x(u_t)_C+(aA^y\Lambda_2^x+b\Lambda_1^y\Lambda_1^x+\sigma A^y\Lambda_1^x)u_C=A^y\Lambda_1^xq_C,\\
&\Lambda_1^yA^x(u_t)_C+(a\Lambda_1^y\Lambda_1^x+b\Lambda_2^yA^x+\sigma\Lambda_1^yA^x)u_C=\Lambda_1^yA^xq_C,\\
&\Lambda_1^y\Lambda_1^x(u_t)_C+(a\Lambda_1^y\Lambda_2^x+b\Lambda_2^y\Lambda_1^x+\sigma\Lambda_1^y\Lambda_1^x)u_C=\Lambda_1^y\Lambda_1^xq_C
\end{aligned} \tag{12}$$

for determining four unknowns $u_{j+1,k+1}$, $\bar{u}^x_{j+1/2,k+1}$, $\bar{u}^y_{j+1,k+1/2}$, $\bar{u}^{xy}_{j+1/2,k+1/2}$ for $a > 0$ and $b > 0$ (the other signs are treated in a similar fashion). The multi-index C in (12) is equal to $(j+1/2,k+1/2)$.

As a time-stepping technique for ODEs (8) and (12), we propose A- and L-stable diagonally implicit Runge–Kutta (DIRK) methods, which ensure the absolute stability of fully discrete bicompact schemes. Moreover, DIRK methods are low-cost techniques as compared with fully implicit Runge–Kutta (RK) methods. At each stage of a DIRK method, the computations are reduced to solving the equations of a fully discrete *baseline* bicompact scheme with its own stage initial condition and time step. The baseline bicompact scheme is obtained by integrating the semidiscrete scheme by the implicit Euler method. For the 2D transport equation (2), this scheme consists of four difference equations and has the form

$$\begin{aligned}
&\left((1+\sigma\tau)A^yA^x+\tau aA^y\Lambda_1^x+\tau b\Lambda_1^yA^x\right)\hat{u}_C=A^yA^x\left(u_C+\tau\hat{q}_C\right),\\
&\left((1+\sigma\tau)A^y\Lambda_1^x+\tau aA^y\Lambda_2^x+\tau b\Lambda_1^y\Lambda_1^x\right)\hat{u}_C=A^y\Lambda_1^x\left(u_C+\tau\hat{q}_C\right),\\
&\left((1+\sigma\tau)\Lambda_1^yA^x+\tau a\Lambda_1^y\Lambda_1^x+\tau b\Lambda_2^yA^x\right)\hat{u}_C=\Lambda_1^yA^x\left(u_C+\tau\hat{q}_C\right),\\
&\left((1+\sigma\tau)\Lambda_1^y\Lambda_1^x+\tau a\Lambda_1^y\Lambda_2^x+\tau b\Lambda_2^y\Lambda_1^x\right)\hat{u}_C=\Lambda_1^y\Lambda_1^x\left(u_C+\tau\hat{q}_C\right),
\end{aligned} \tag{13}$$

where τ is the time step, $\hat{u}=u^{n+1}$, $u=u^n$, and n is the time level number. For the 3D transport equation (1), the baseline bicompact scheme consists of eight difference equations

$$
\begin{aligned}
&\left((1+\sigma\tau)A^zA^yA^x + a\tau A^zA^y\Lambda_1^x + b\tau A^z\Lambda_1^yA^x + c\tau\Lambda_1^zA^yA^x\right)\hat{u}_C = A^zA^yA^x(u_C+\tau\hat{q}_C),\\
&\left((1+\sigma\tau)A^zA^y\Lambda_1^x + a\tau A^zA^y\Lambda_2^x + b\tau A^z\Lambda_1^y\Lambda_1^x + c\tau\Lambda_1^zA^y\Lambda_1^x\right)\hat{u}_C = A^zA^y\Lambda_1^x(u_C+\tau\hat{q}_C),\\
&\left((1+\sigma\tau)A^z\Lambda_1^yA^x + a\tau A^z\Lambda_1^y\Lambda_1^x + b\tau A^z\Lambda_2^yA^x + c\tau\Lambda_1^z\Lambda_1^yA^x\right)\hat{u}_C = A^z\Lambda_1^yA^x(u_C+\tau\hat{q}_C),\\
&\left((1+\sigma\tau)A^z\Lambda_1^y\Lambda_1^x + a\tau A^z\Lambda_1^y\Lambda_2^x + b\tau A^z\Lambda_2^y\Lambda_1^x + c\tau\Lambda_1^z\Lambda_1^y\Lambda_1^x\right)\hat{u}_C = A^z\Lambda_1^y\Lambda_1^x(u_C+\tau\hat{q}_C),\\
&\left((1+\sigma\tau)\Lambda_1^zA^yA^x + a\tau\Lambda_1^zA^y\Lambda_1^x + b\tau\Lambda_1^z\Lambda_1^yA^x + c\tau\Lambda_2^zA^yA^x\right)\hat{u}_C = \Lambda_1^zA^yA^x(u_C+\tau\hat{q}_C),\\
&\left((1+\sigma\tau)\Lambda_1^zA^y\Lambda_1^x + a\tau\Lambda_1^zA^y\Lambda_2^x + b\tau\Lambda_1^z\Lambda_1^y\Lambda_1^x + c\tau\Lambda_2^zA^y\Lambda_1^x\right)\hat{u}_C = \Lambda_1^zA^y\Lambda_1^x(u_C+\tau\hat{q}_C),\\
&\left((1+\sigma\tau)\Lambda_1^z\Lambda_1^yA^x + a\tau\Lambda_1^z\Lambda_1^y\Lambda_1^x + b\tau\Lambda_1^z\Lambda_2^yA^x + c\tau\Lambda_2^z\Lambda_1^yA^x\right)\hat{u}_C = \Lambda_1^z\Lambda_1^yA^x(u_C+\tau\hat{q}_C),\\
&\left((1+\sigma\tau)\Lambda_1^z\Lambda_1^y\Lambda_1^x + a\tau\Lambda_1^z\Lambda_1^y\Lambda_2^x + b\tau\Lambda_1^z\Lambda_2^y\Lambda_1^x + c\tau\Lambda_2^z\Lambda_1^y\Lambda_1^x\right)\hat{u}_C = \Lambda_1^z\Lambda_1^y\Lambda_1^x(u_C+\tau\hat{q}_C).
\end{aligned}
\tag{14}
$$

3. The iterative approximate factorization method

As the dimension d of the problem increases, the number of equations in each cell grows as 2^d. According to the space marching method, a system of dimension 2^d is solved in each cell with the help of direct Gaussian elimination, which has the complexity $C\times(2^d)^3 = C\times 2^{3d}$. For $d=3$, this leads to considerable growth of the computational costs as compared with the one-dimensional case. Among the methods used to avoid such growth, we can mention dimensional splitting (local one-dimensional scheme—LOD scheme), the alternating direction method, and approximate factorization of linear systems of difference equations. In most of these techniques, the accuracy of the solution degrades to the second order.

The difference equations of multidimensional bicompact schemes can also be approximately factorized. To preserve the high order of time-stepping schemes, an iterative procedure is constructed on the basis of approximate factorization. Below, this procedure is described for 2D and 3D bicompact schemes intended for the numerical solution of the inhomogeneous transport equation.

3.1. Two-dimensional scheme

Consider baseline scheme (13) in the case of ODE system (12) integrated by applying the implicit Euler method. Define the operators

$$
\begin{aligned}
&B_1^r(s) = A^r + \tau C_1^r(s),\ B_2^r(s) = \Lambda_1^r + \tau C_2^r(s),\ r = x, y,\\
&C_1^r(s) = \frac{1}{2}\sigma A^r + s\Lambda_1^r,\ C_2^r(s) = \frac{1}{2}\sigma\Lambda_1^r + s\Lambda_2^r.
\end{aligned}
\tag{15}
$$

The operators on the left-hand sides of Eqs. (13) can be represented in the form

$$
\begin{aligned}
&A^y A^x(1+\sigma\tau)+\tau a A^y\Lambda_1^x+\tau b\Lambda_1^y A^x = B_1^y(b)B_1^x(a)-\tau^2 C_1^y(b)C_1^x(a),\\
&A^y\Lambda_1^x(1+\sigma\tau)+\tau a A^y\Lambda_2^x+\tau b\Lambda_1^y\Lambda_1^x = B_1^y(b)B_2^x(a)-\tau^2 C_1^y(b)C_2^x(a),\\
&\Lambda_1^y A^x(1+\sigma\tau)+\tau a\Lambda_1^y\Lambda_1^x+\tau b\Lambda_2^y A^x = B_2^y(b)B_1^x(a)-\tau^2 C_2^y(b)C_1^x(a),\\
&\Lambda_1^y\Lambda_1^x(1+\sigma\tau)+\tau a\Lambda_1^y\Lambda_2^x+\tau b\Lambda_2^y\Lambda_1^x = B_2^y(b)B_2^x(a)-\tau^2 C_2^y(b)C_2^x(a).
\end{aligned}
\tag{16}
$$

In view of (16), Eqs. (13) can be written as

$$
\begin{aligned}
B_1^y(b)B_1^x(a)\hat{u}_C &= A^y A^x(u_C+\tau\hat{q}_C)+\tau^2 C_1^y(b)C_1^x(a)\hat{u}_C,\\
B_1^y(b)B_2^x(a)\hat{u}_C &= A^y\Lambda_1^x(u_C+\tau\hat{q}_C)+\tau^2 C_1^y(b)C_2^x(a)\hat{u}_C,\\
B_2^y(b)B_1^x(a)\hat{u}_C &= \Lambda_1^y A^x(u_C+\tau\hat{q}_C)+\tau^2 C_2^y(b)C_1^x(a)\hat{u}_C,\\
B_2^y(b)B_2^x(a)\hat{u}_C &= \Lambda_1^y\Lambda_1^x(u_C+\tau\hat{q}_C)+\tau^2 C_2^y(b)C_2^x(a)\hat{u}_C.
\end{aligned}
\tag{17}
$$

The structure of system (17) suggests that it can be approximately factorized for small τ with the last terms dropped from the right-hand sides. As a result, system (17) can be solved using the iterative method

$$
\begin{aligned}
B_1^y(b)B_1^x(a)\hat{u}_C^{(i+1)} &= A^y A^x(u_C+\tau\hat{q}_C)+\tau^2 C_1^y(b)C_1^x(a)\hat{u}_C^{(i)},\\
B_1^y(b)B_2^x(a)\hat{u}_C^{(i+1)} &= A^y\Lambda_1^x(u_C+\tau\hat{q}_C)+\tau^2 C_1^y(b)C_2^x(a)\hat{u}_C^{(i)},\\
B_2^y(b)B_1^x(a)\hat{u}_C^{(i+1)} &= \Lambda_1^y A^x(u_C+\tau\hat{q}_C)+\tau^2 C_2^y(b)C_1^x(a)\hat{u}_C^{(i)},\\
B_2^y(b)B_2^x(a)\hat{u}_C^{(i+1)} &= \Lambda_1^y\Lambda_1^x(u_C+\tau\hat{q}_C)+\tau^2 C_2^y(b)C_2^x(a)\hat{u}_C^{(i)},
\end{aligned}
\tag{18}
$$

where i is the iteration number. The initial approximation $\hat{u}^{(0)}$ can be set to zero. The solution $\hat{u}^{(i+1)}$ of Eqs. (18) can be found by solving two systems for the grid functions v_1, v_2, namely,

$$
\begin{cases}
B_1^y(b)v_1 = A^y A^x(u_C+\tau\hat{q}_C)+\tau^2 C_1^y(b)C_1^x(a)\hat{u}_C^{(i)},\\
B_2^y(b)v_1 = \Lambda_1^y A^x(u_C+\tau\hat{q}_C)+\tau^2 C_2^y(b)C_1^x(a)\hat{u}_C^{(i)},
\end{cases}
\qquad
\begin{cases}
B_1^y(b)v_2 = A^y\Lambda_1^x(u_C+\tau\hat{q}_C)+\tau^2 C_1^y(b)C_2^x(a)\hat{u}_C^{(i)},\\
B_2^y(b)v_2 = \Lambda_1^y\Lambda_1^x(u_C+\tau\hat{q}_C)+\tau^2 C_2^y(b)C_2^x(a)\hat{u}_C^{(i)},
\end{cases}
\tag{19}
$$

and then solving the system of equations

$$\begin{cases} B_1^x(a)\hat{u}^{(i+1)} = v_1, \\ B_2^x(a)\hat{u}^{(i+1)} = v_2. \end{cases} \tag{20}$$

Boundary conditions for v_1, v_2 in problems (19) are calculated using (20).

The one-dimensional equations (19) have to be solved on the coordinate lines $x = x_{j+1/2}$, $j = \overline{1, N_x}$ (N_x is the number of grid cells in the *x* direction):

$$\begin{cases} B_1^y(b)v_{1,j+1/2,k+1/2} = A^y A^x(u_C + \tau\hat{q}_C) + \tau^2 C_1^y(b)C_1^x(a)\hat{u}_C^{(i)}, \\ B_2^y(b)v_{1,j+1/2,k+1/2} = \Lambda_1^y A^x(u_C + \tau\hat{q}_C) + \tau^2 C_2^y(b)C_1^x(a)\hat{u}_C^{(i)}, \end{cases}$$
$$\begin{cases} B_1^y(b)v_{2,j+1/2,k+1/2} = A^y \Lambda_1^x(u_C + \tau\hat{q}_C) + \tau^2 C_1^y(b)C_2^x(a)\hat{u}_C^{(i)}, \\ B_2^y(b)v_{2,j+1/2,k+1/2} = \Lambda_1^y \Lambda_1^x(u_C + \tau\hat{q}_C) + \tau^2 C_2^y(b)C_2^x(a)\hat{u}_C^{(i)}. \end{cases} \tag{21}$$

The one-dimensional equations (20) have to be solved on the coordinate lines $y = y_{k+1/2}$,

$$\begin{cases} B_1^x(a)(\hat{\bar{u}}^y)_{j+1/2,k+1/2}^{(i+1)} = \bar{v}_{1,j+1/2,k+1/2}^y, \\ B_2^x(a)(\hat{\bar{u}}^y)_{j+1/2,k+1/2}^{(i+1)} = \bar{v}_{2,j+1/2,k+1/2}^y \end{cases} \tag{22}$$

and on the lines $y = y_{k+1}$, $k = \overline{1, N_y}$ (N_y is the number of grid cells in the y direction)

$$\begin{cases} B_1^x(a)\hat{u}_{j+1/2,k+1}^{(i+1)} = v_{1,j+1/2,k+1}, \\ B_2^x(a)\hat{u}_{j+1/2,k+1}^{(i+1)} = v_{2,j+1/2,k+1}. \end{cases} \tag{23}$$

Equations (22) are derived from Eqs. (20) by applying the averaging operator A^y.

Finally, at every time level, we need to solve $2(N_x + N_y)$ one-dimensional problems. Each one-dimensional problem is solved by the space marching method. The factorized scheme has two advantages over the original (unfactorized) scheme. First, the solution of one-dimensional problems involves the inversion of 2×2 rather than $2^d \times 2^d$ matrices. Second, one-dimensional problems along different grid lines at every iteration step can be solved independently. Therefore, the algorithm can be efficiently parallelized, in contrast to the original scheme with a more complicated dependence of data.

Let us analyze the convergence of iterative method (18). This method can be written in the matrix-vector form

$$\mathbf{M}_y\mathbf{M}_x\hat{\mathbf{v}}^{(i+1)} = \mathbf{N}\hat{\mathbf{v}}^{(i)} + \boldsymbol{\psi}, \quad \mathbf{v} = \begin{pmatrix} \bar{u}^{xy}_{j+1/2,k+1/2} \\ \bar{u}^{y}_{j+1,k+1/2} \\ \bar{u}^{x}_{j+1/2,k+1} \\ u_{j+1,k+1} \end{pmatrix}, \quad \boldsymbol{\psi} = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{pmatrix},$$

$$\mathbf{M}_y = \begin{bmatrix} \theta & 0 & \kappa_y & 0 \\ 0 & \theta & 0 & \kappa_y \\ -12\kappa_y & 0 & \theta+6\kappa_y & 0 \\ 0 & -12\kappa_y & 0 & \theta+6\kappa_y \end{bmatrix}, \mathbf{M}_x = \begin{bmatrix} \theta & \kappa_x & 0 & 0 \\ -12\kappa_x & \theta+6\kappa_x & 0 & 0 \\ 0 & 0 & \theta & \kappa_x \\ 0 & 0 & -12\kappa_x & \theta+6\kappa_x \end{bmatrix} \quad (24)$$

,

$$\mathbf{N} = \begin{bmatrix} (\theta-1)^2 & (\theta-1)\kappa_x & (\theta-1)\kappa_y & \kappa_x\kappa_y \\ -12(\theta-1)\kappa_x & (\theta-1)(\theta-1+6\kappa_x) & -12\kappa_x\kappa_y & \kappa_y(\theta-1+6\kappa_x) \\ -12(\theta-1)\kappa_y & -12\kappa_x\kappa_y & (\theta-1)(\theta-1+6\kappa_y) & \kappa_x(\theta-1+6\kappa_y) \\ 144\kappa_x\kappa_y & -12\kappa_y(\theta-1+6\kappa_x) & -12\kappa_x(\theta-1+6\kappa_y) & (\theta-1+6\kappa_x)(\theta-1+6\kappa_y) \end{bmatrix}$$

,

where $\theta = 1+(\sigma\tau)/2$; $\kappa_x = a\tau/h_x$, $\kappa_y = b\tau/h_y$ are the Courant numbers; $\boldsymbol{\psi}$ is a column vector containing known values from the preceding time level, values known from the boundary conditions, or computed in preceding cells and the values of the given source function q.

The matrices $\mathbf{M}_x, \mathbf{M}_y, \mathbf{N}$ can be represented in the form of Kronecker matrix products

$$\mathbf{M}_x = \mathbf{E}\otimes\mathbf{P}_x, \ \mathbf{M}_y = \mathbf{P}_y\otimes\mathbf{E}, \ \mathbf{N} = (\mathbf{P}_y - \mathbf{E})\otimes(\mathbf{P}_x - \mathbf{E}), \quad (25)$$

where

$$\mathbf{P}_r = \theta\mathbf{E} + \kappa_r\mathbf{R}, \ \mathbf{E} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \mathbf{R} = \begin{bmatrix} 0 & 1 \\ -12 & 6 \end{bmatrix}, \ r = x, y. \quad (26)$$

The product of the matrices $\mathbf{M}_y, \mathbf{M}_x$ is transformed as follows:

$$\mathbf{M}_y\mathbf{M}_x = (\mathbf{P}_y\otimes\mathbf{E})(\mathbf{E}\otimes\mathbf{P}_x) = \mathbf{P}_y\otimes\mathbf{P}_x. \quad (27)$$

The iterative method (24) is written in the recurrence form

$$\mathbf{v}^{(i+1)} = \mathbf{S}\mathbf{v}^{(i)} + \xi, \quad (28)$$

where the step matrix of the method is given by

$$\mathbf{S} = \left(\mathbf{M}_y \mathbf{M}_x\right)^{-1} \mathbf{N}. \tag{29}$$

In view of formulas (25) and (27), the step matrix (29) of the iterative method can be written as a Kronecker product of two matrices, each associated only with one spatial direction:

$$\begin{aligned} \mathbf{S} &= \left(\mathbf{P}_y \otimes \mathbf{P}_x\right)^{-1} \left[(\mathbf{P}_y - \mathbf{E}) \otimes (\mathbf{P}_x - \mathbf{E})\right] = \left(\mathbf{P}_y^{-1} \otimes \mathbf{P}_x^{-1}\right) \left[(\mathbf{P}_y - \mathbf{E}) \otimes (\mathbf{P}_x - \mathbf{E})\right] = \\ &= \left(\mathbf{E} - \mathbf{P}_y^{-1}\right) \otimes \left(\mathbf{E} - \mathbf{P}_x^{-1}\right) = \left(\mathbf{E} - \mathbf{T}_y\right) \otimes \left(\mathbf{E} - \mathbf{T}_x\right), \end{aligned} \tag{30}$$

where $\mathbf{T}_r = \mathbf{P}_r^{-1},\ r = x, y$.

Due to representation (30) for the matrix **S**, its eigenvalues $\lambda(\mathbf{S})$ are equal to the product of the eigenvalues λ_y, λ_x of the matrixes $\mathbf{E} - \mathbf{T}_y$, $\mathbf{E} - \mathbf{T}_x$:

$$\lambda(\mathbf{S}) = \lambda_y \lambda_x, \tag{31}$$

where

$$\lambda_r = \frac{12\kappa_r^2 + (6\theta - 3 \pm i\sqrt{3})\kappa_r + \theta(\theta - 1)}{12\kappa_r^2 + 6\theta\kappa_r + \theta^2},\ r = x, y. \tag{32}$$

The moduli of the complex conjugate numbers λ_r (32) are identical and given by

$$\begin{aligned} |\lambda_r| &= \frac{\sqrt{\left(12\kappa_r^2 + 3(2\theta - 1)\kappa_r + \theta(\theta - 1)\right)^2 + 3\kappa_r^2}}{12\kappa_r^2 + 6\theta\kappa_r + \theta^2} = \\ &= \sqrt{1 - \frac{6\kappa_r + 2\theta - 1}{12\kappa_r^2 + 6\theta\kappa_r + \theta^2}} < 1 \ \ \forall \kappa_r > 0,\ \theta \geq 1, \end{aligned} \tag{33}$$

therefore, in view of (31), the spectral radius $\rho(\mathbf{S})$ of the step matrix **S** is

$$\rho(\mathbf{S}) = |\lambda_y||\lambda_x| < 1 \quad \forall \kappa_x > 0,\ \kappa_y > 0,\ \theta \geq 1. \tag{34}$$

Since the spectral radius of the stationary iterative process (27) is less than unity for any $\kappa_y > 0, \kappa_x > 0, \theta \geq 1$, the iterations always converge.

Note that the convergence of the iterative approximate factorization method for bicompact schemes as applied to the nonstationary 2D linear *homogeneous* transport equation was proved in [9, 10].

3.2. Three-dimensional scheme

Consider the baseline bicompact scheme (14) in the case of ODE system (8) integrated by the implicit Euler method. Define the operators

$$B_1^r(s) = A^r + \tau C_1^r(s),\ B_2^r(s) = \Lambda_1^r + \tau C_2^r(s),\ r = x, y, z,$$
$$C_1^r(s) = \frac{1}{3}\sigma A^r + s\Lambda_1^r,\ C_2^r(s) = \frac{1}{3}\sigma\Lambda_1^r + s\Lambda_2^r. \tag{35}$$

The operators on the left-hand sides of (14) can be approximately factorized as

$$\begin{aligned}
&(1+\sigma\tau)A^zA^yA^x + a\tau A^zA^y\Lambda_1^x + b\tau A^z\Lambda_1^yA^x + c\tau\Lambda_1^zA^yA^x = B_1^z(c)B_1^y(b)B_1^x(a) - \tau^2 I_1,\\
&(1+\sigma\tau)A^zA^y\Lambda_1^x + a\tau A^zA^y\Lambda_2^x + b\tau A^z\Lambda_1^y\Lambda_1^x + c\tau\Lambda_1^zA^y\Lambda_1^x = B_1^z(c)B_1^y(b)B_2^x(a) - \tau^2 I_2,\\
&(1+\sigma\tau)A^z\Lambda_1^yA^x + a\tau A^z\Lambda_1^y\Lambda_1^x + b\tau A^z\Lambda_2^yA^x + c\tau\Lambda_1^z\Lambda_1^yA^x = B_1^z(c)B_2^y(b)B_1^x(a) - \tau^2 I_3,\\
&(1+\sigma\tau)A^z\Lambda_1^y\Lambda_1^x + a\tau A^z\Lambda_1^y\Lambda_2^x + b\tau A^z\Lambda_2^y\Lambda_1^x + c\tau\Lambda_1^z\Lambda_1^y\Lambda_1^x = B_1^z(c)B_2^y(b)B_2^x(a) - \tau^2 I_4,\\
&(1+\sigma\tau)\Lambda_1^zA^yA^x + a\tau\Lambda_1^zA^y\Lambda_1^x + b\tau\Lambda_1^z\Lambda_1^yA^x + c\tau\Lambda_2^zA^yA^x = B_2^z(c)B_1^y(b)B_1^x(a) - \tau^2 I_5,\\
&(1+\sigma\tau)\Lambda_1^zA^y\Lambda_1^x + a\tau\Lambda_1^zA^y\Lambda_2^x + b\tau\Lambda_1^z\Lambda_1^y\Lambda_1^x + c\tau\Lambda_2^zA^y\Lambda_1^x = B_2^z(c)B_1^y(b)B_2^x(a) - \tau^2 I_6,\\
&(1+\sigma\tau)\Lambda_1^z\Lambda_1^yA^x + a\tau\Lambda_1^z\Lambda_1^y\Lambda_1^x + b\tau\Lambda_1^z\Lambda_2^yA^x + c\tau\Lambda_2^z\Lambda_1^yA^x = B_2^z(c)B_2^y(b)B_1^x(a) - \tau^2 I_7,\\
&(1+\sigma\tau)\Lambda_1^z\Lambda_1^y\Lambda_1^x + a\tau\Lambda_1^z\Lambda_1^y\Lambda_2^x + b\tau\Lambda_1^z\Lambda_2^y\Lambda_1^x + c\tau\Lambda_2^z\Lambda_1^y\Lambda_1^x = B_2^z(c)B_2^y(b)B_2^x(a) - \tau^2 I_8,
\end{aligned} \tag{36}$$

where

$$\begin{aligned}
I_1 &= A^zC_1^yC_1^x + C_1^zA^yC_1^x + C_1^zC_1^yA^x + \tau C_1^zC_1^yC_1^x,\\
I_2 &= A^zC_1^yC_2^x + C_1^zA^yC_2^x + C_1^zC_1^yC_1^x + \tau C_1^zC_1^yC_2^x,\\
I_3 &= A^zC_2^yC_1^x + C_1^zC_1^yC_1^x + C_1^zC_2^yA^x + \tau C_1^zC_2^yC_1^x,\\
I_4 &= A^zC_2^yC_2^x + C_1^zC_1^yC_2^x + C_1^zC_2^yC_1^x + \tau C_1^zC_2^yC_2^x,\\
I_5 &= C_1^zC_1^yC_1^x + C_2^zA^yC_1^x + C_2^zC_1^yA^x + \tau C_2^zC_1^yC_1^x,\\
I_6 &= C_1^zC_1^yC_2^x + C_2^zA^yC_2^x + C_2^zC_1^yC_1^x + \tau C_2^zC_1^yC_2^x,\\
I_7 &= C_1^zC_2^yC_1^x + C_2^zC_1^yC_1^x + C_2^zC_2^yA^x + \tau C_2^zC_2^yC_1^x,\\
I_8 &= C_1^zC_2^yC_2^x + C_2^zC_1^yC_2^x + C_2^zC_2^yC_1^x + \tau C_2^zC_2^yC_2^x.
\end{aligned}$$

An iterative algorithm for solving the equations of the baseline scheme relies on the approximate factorization (36) of the difference operators and is given by the formulas

$$\begin{aligned}
B_1^z(c)B_1^y(b)B_1^x(a)\hat{u}_C^{(i+1)} &= A^zA^yA^x(u_C + \tau\hat{q}_C) + \tau^2 I_1\hat{u}_C^{(i)},\\
B_1^z(c)B_1^y(b)B_2^x(a)\hat{u}_C^{(i+1)} &= A^zA^y\Lambda_1^x(u_C + \tau\hat{q}_C) + \tau^2 I_2\hat{u}_C^{(i)},\\
B_1^z(c)B_2^y(b)B_1^x(a)\hat{u}_C^{(i+1)} &= A^z\Lambda_1^yA^x(u_C + \tau\hat{q}_C) + \tau^2 I_3\hat{u}_C^{(i)},\\
B_1^z(c)B_2^y(b)B_2^x(a)\hat{u}_C^{(i+1)} &= A^z\Lambda_1^y\Lambda_1^x(u_C + \tau\hat{q}_C) + \tau^2 I_4\hat{u}_C^{(i)},\\
B_2^z(c)B_1^y(b)B_1^x(a)\hat{u}_C^{(i+1)} &= \Lambda_1^zA^yA^x(u_C + \tau\hat{q}_C) + \tau^2 I_5\hat{u}_C^{(i)},\\
B_2^z(c)B_1^y(b)B_2^x(a)\hat{u}_C^{(i+1)} &= \Lambda_1^zA^y\Lambda_1^x(u_C + \tau\hat{q}_C) + \tau^2 I_6\hat{u}_C^{(i)},\\
B_2^z(c)B_2^y(b)B_1^x(a)\hat{u}_C^{(i+1)} &= \Lambda_1^z\Lambda_1^yA^x(u_C + \tau\hat{q}_C) + \tau^2 I_7\hat{u}_C^{(i)},\\
B_2^z(c)B_2^y(b)B_2^x(a)\hat{u}_C^{(i+1)} &= \Lambda_1^z\Lambda_1^y\Lambda_1^x(u_C + \tau\hat{q}_C) + \tau^2 I_8\hat{u}_C^{(i)},
\end{aligned} \tag{37}$$

where i is the iteration number.

Algorithm (37) can be written in the matrix-vector form

$$\mathbf{M}_z\mathbf{M}_y\mathbf{M}_x\hat{\mathbf{v}}^{(i+1)} = \mathbf{N}\hat{\mathbf{v}}^{(i)} + \boldsymbol{\psi},\ \ \mathbf{v} = \begin{pmatrix} \bar{u}^{xyz}_{j+1/2,k+1/2,l+1/2} \\ \bar{u}^{yz}_{j+1,k+1/2,l+1/2} \\ \bar{u}^{xz}_{j+1/2,k+1,l+1/2} \\ \bar{u}^{z}_{j+1,k+1,l+1/2} \\ \bar{u}^{xy}_{j+1/2,k+1/2,l+1} \\ \bar{u}^{y}_{j+1,k+1/2,l+1} \\ \bar{u}^{x}_{j+1/2,k+1,l+1} \\ u_{j+1,k+1,l+1} \end{pmatrix},\ \ \boldsymbol{\psi} = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \\ \psi_6 \\ \psi_7 \\ \psi_8 \end{pmatrix}, \tag{38}$$

$$\mathbf{M}_z = \mathbf{P}_z \otimes \mathbf{E} \otimes \mathbf{E},\ \ \mathbf{M}_y = \mathbf{E} \otimes \mathbf{P}_y \otimes \mathbf{E},\ \ \mathbf{M}_x = \mathbf{E} \otimes \mathbf{E} \otimes \mathbf{P}_x, \tag{39}$$

$$\begin{aligned} \mathbf{N} = {} & \mathbf{E} \otimes (\mathbf{P}_y - \mathbf{E}) \otimes (\mathbf{P}_x - \mathbf{E}) + (\mathbf{P}_z - \mathbf{E}) \otimes \mathbf{E} \otimes (\mathbf{P}_x - \mathbf{E}) + \\ & + (\mathbf{P}_z - \mathbf{E}) \otimes (\mathbf{P}_y - \mathbf{E}) \otimes \mathbf{E} + (\mathbf{P}_z - \mathbf{E}) \otimes (\mathbf{P}_y - \mathbf{E}) \otimes (\mathbf{P}_x - \mathbf{E}), \end{aligned} \tag{40}$$

where $\boldsymbol{\psi}$ is a known column vector and the matrices $\mathbf{P}_r$ $(r = x, y, z)$ and $\mathbf{E}$ are given by the formulas

$$\mathbf{P}_r = \theta\mathbf{E} + \kappa_r\mathbf{R},\ \ \theta = 1 + \frac{1}{3}\sigma\tau,\ \ \mathbf{E} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \ \mathbf{R} = \begin{bmatrix} 0 & 1 \\ -12 & 6 \end{bmatrix}. \tag{41}$$

Here, $\kappa_x = a\tau/h_x$, $\kappa_y = b\tau/h_y$, $\kappa_z = c\tau/h_z$ are Courant numbers.

The matrix product on the left-hand side of (38) can be brought to the form

$$\mathbf{M}_z\mathbf{M}_y\mathbf{M}_x = \mathbf{P}_z \otimes \mathbf{P}_y \otimes \mathbf{P}_x.$$

Iterative method (38) is written in the recurrence form

$$\hat{\mathbf{v}}^{(i+1)} = \mathbf{S}\hat{\mathbf{v}}^{(i)} + \xi, \tag{42}$$

where the step matrix of the method is given by

$$\mathbf{S} = \left(\mathbf{M}_z\mathbf{M}_y\mathbf{M}_x\right)^{-1}\mathbf{N} = \left(\mathbf{P}_z \otimes \mathbf{P}_y \otimes \mathbf{P}_x\right)^{-1}\mathbf{N} = \left(\mathbf{P}_z^{-1} \otimes \mathbf{P}_y^{-1} \otimes \mathbf{P}_x^{-1}\right)\mathbf{N}. \tag{43}$$

In view of formulas (40) and (43), the step matrix $\mathbf{S}$ of the iterative method can be brought into the form

$$\begin{aligned}\mathbf{S} = {} & \mathbf{T}_z \otimes \left(\mathbf{E}-\mathbf{T}_y\right)\otimes\left(\mathbf{E}-\mathbf{T}_x\right)+\left(\mathbf{E}-\mathbf{T}_z\right)\otimes\mathbf{T}_y\otimes\left(\mathbf{E}-\mathbf{T}_x\right)+ \\ & +\left(\mathbf{E}-\mathbf{T}_z\right)\otimes\left(\mathbf{E}-\mathbf{T}_y\right)\otimes\mathbf{T}_x+\left(\mathbf{E}-\mathbf{T}_z\right)\otimes\left(\mathbf{E}-\mathbf{T}_y\right)\otimes\left(\mathbf{E}-\mathbf{T}_x\right),\end{aligned} \tag{44}$$

where $\mathbf{T}_r = \mathbf{P}_r^{-1}$, $r = x, y, z$.

Formula (44) shows that $\mathbf{S}$ is the sum of four 8×8 matrices, each being a Kronecker product of three 2×2 matrices. In turn, the column eigenvectors of the 8×8 matrices on the right-hand side of (44) are Kronecker products of column eigenvectors of 2×2 matrices [13, p.596]. Since the column eigenvectors of the 2×2 matrices $\mathbf{T}_r$ and $\mathbf{E}-\mathbf{T}_r$ coincide, the column eigenvectors of all four 8×8 matrices on the right-hand side of (44) coincide as well. Therefore, the eigenvalues $\lambda(\mathbf{S})$ of the step matrix $\mathbf{S}$ are equal to the sum of the eigenvalues of four 8×8 matrices on the right-hand side of (44):

$$\begin{aligned}\lambda(\mathbf{S}) = {} & \lambda\left(\mathbf{T}_z\otimes\left(\mathbf{E}-\mathbf{T}_y\right)\otimes\left(\mathbf{E}-\mathbf{T}_x\right)\right)+\lambda\left(\left(\mathbf{E}-\mathbf{T}_z\right)\otimes\mathbf{T}_y\otimes\left(\mathbf{E}-\mathbf{T}_x\right)\right)+ \\ & +\lambda\left(\left(\mathbf{E}-\mathbf{T}_z\right)\otimes\left(\mathbf{E}-\mathbf{T}_y\right)\otimes\mathbf{T}_x\right)+\lambda\left(\left(\mathbf{E}-\mathbf{T}_z\right)\otimes\left(\mathbf{E}-\mathbf{T}_y\right)\otimes\left(\mathbf{E}-\mathbf{T}_x\right)\right)= \\ & =(1-\lambda_z)\lambda_y\lambda_x+\lambda_z(1-\lambda_y)\lambda_x+\lambda_z\lambda_y(1-\lambda_x)+\lambda_z\lambda_y\lambda_x,\end{aligned} \tag{45}$$

where λ_r are the eigenvalues of the matrix $\mathbf{E}-\mathbf{T}_r$. In writing the last equality in (45), we used the properties of eigenvalues of a matrix that is a Kronecker matrix product. Since the matrix $\mathbf{E}-\mathbf{T}_r$ has two complex conjugate eigenvalues

$$\lambda_r = \frac{12\kappa_r^2+(6\theta-3\pm i\sqrt{3})\kappa_r+\theta(\theta-1)}{12\kappa_r^2+6\theta\kappa_r+\theta^2}, \tag{46}$$

formula (45) determines eight different eigenvalues of the matrix $\mathbf{S}$. It follows from (46) that

$$\begin{aligned}&\mathrm{Re}(\lambda_r) = \frac{12\kappa_r^2+(6\theta-3)\kappa_r+\theta(\theta-1)}{12\kappa_r^2+6\theta\kappa_r+\theta^2} = 1-\frac{3\kappa_r+\theta}{12\kappa_r^2+6\theta\kappa_r+\theta^2}, \\ &\mathrm{Im}(\lambda_r) = \pm\frac{\sqrt{3}\kappa_r}{12\kappa_r^2+6\theta\kappa_r+\theta^2}, \\ &|\lambda_r|^2 = \frac{\left(12\kappa_r^2+(6\theta-3)\kappa_r+\theta(\theta-1)\right)^2+3\kappa_r^2}{(12\kappa_r^2+6\theta\kappa_r+\theta^2)^2} = 1-\frac{6\kappa_r+2\theta-1}{12\kappa_r^2+6\theta\kappa_r+\theta^2}.\end{aligned} \tag{47}$$

The squared modulus of the eigenvalue $\lambda(\mathbf{S})$ is calculated as

$$
\begin{aligned}
&|\lambda(\mathbf{S})|^2 = \lambda(\mathbf{S})\bar{\lambda}(\mathbf{S}) = (1-4\,\mathrm{Re}\,\lambda_z)|\lambda_y|^2|\lambda_x|^2 + (1-4\,\mathrm{Re}\,\lambda_y)|\lambda_z|^2|\lambda_x|^2 + \\
&+(1-4\,\mathrm{Re}\,\lambda_x)|\lambda_z|^2|\lambda_y|^2 + 4|\lambda_z|^2|\lambda_y|^2|\lambda_x|^2 + 2(\mathrm{Re}\,\lambda_z\,\mathrm{Re}\,\lambda_y + \mathrm{Im}\,\lambda_z\,\mathrm{Im}\,\lambda_y)|\lambda_x|^2 + \\
&+2(\mathrm{Re}\,\lambda_z\,\mathrm{Re}\,\lambda_x + \mathrm{Im}\,\lambda_z\,\mathrm{Im}\,\lambda_x)|\lambda_y|^2 + 2(\mathrm{Re}\,\lambda_y\,\mathrm{Re}\,\lambda_x + \mathrm{Im}\,\lambda_y\,\mathrm{Im}\,\lambda_x)|\lambda_z|^2.
\end{aligned}
\tag{48}
$$

Using (47) and (48), we find the squared spectral radius of $\mathbf{S}$:

$$
\begin{aligned}
&\rho^2(\mathbf{S}) = \max|\lambda(\mathbf{S})|^2 = 1 - \frac{f(\kappa_z,\kappa_y,\kappa_x)}{(12\kappa_z^2+6\theta\kappa_z+\theta^2)(12\kappa_y^2+6\theta\kappa_y+\theta^2)(12\kappa_x^2+6\theta\kappa_x+\theta^2)}, \\
&f(\kappa_z,\kappa_y,\kappa_x) = -144\kappa_z\kappa_y\kappa_x(\kappa_z+\kappa_y+\kappa_x) - \\
&-72\theta\left[3\kappa_z\kappa_y\kappa_x + (\kappa_y+\kappa_x)\kappa_z^2 + (\kappa_z+\kappa_x)\kappa_y^2 + (\kappa_z+\kappa_y)\kappa_x^2\right] - \\
&-12\left[(2\theta^2-1)(\kappa_z^2+\kappa_y^2+\kappa_x^2) + (7\theta^2-2\theta-2)(\kappa_z\kappa_y+\kappa_z\kappa_x+\kappa_y\kappa_x)\right] - \\
&-6(4\theta^3-2\theta^2-3\theta+2)(\kappa_z+\kappa_y+\kappa_x) - (3\theta-2)(2\theta^3-3\theta+2).
\end{aligned}
\tag{49}
$$

It follows from (49) that the spectral radius of the step matrix of the stationary iterative process (42) is less that unity for any $\kappa_z > 0,\ \kappa_y > 0,\ \kappa_x > 0,\ \theta \geq 1$. Therefore, the iterations always converge.

Note that the convergence of the iterative approximate factorization method for bicompact schemes as applied to the nonstationary 3D linear *homogeneous* transport equation was proved in [10].

4. Parallel implementation of numerical algorithms

Although bicompact schemes are implicit, due to their logical simplicity, they can be efficiently solved by applying the space marching method, which is well suited for parallelization. A parallel space marching algorithm is a method for wave data processing with pipelining. The algorithm can be implemented on both shared and distributed memory systems. It should be emphasized that the compact stencil of a bicompact scheme leads to a minimum amount of data to be exchanged between the processors in the case of distributed memory.

Moreover, the factorized version of the scheme can also be parallelized with higher efficiency, since the one-dimensional problems along each grid line can be solved independently.

In this section, we describe parallel implementations of the original and factorized schemes. The efficiency of parallel processing and the scalability of both algorithms are compared as applied to a test initial–boundary value problem for the two-dimensional nonstationary transport equation with a stiff source term.

4.1. Parallel space marching algorithm

Consider a parallel implementation of the algorithm for the scalar inhomogeneous transport equation on shared memory systems with OpenMP

technology. The sequence of computations in the parallel algorithm is schematically shown in Fig. 1.

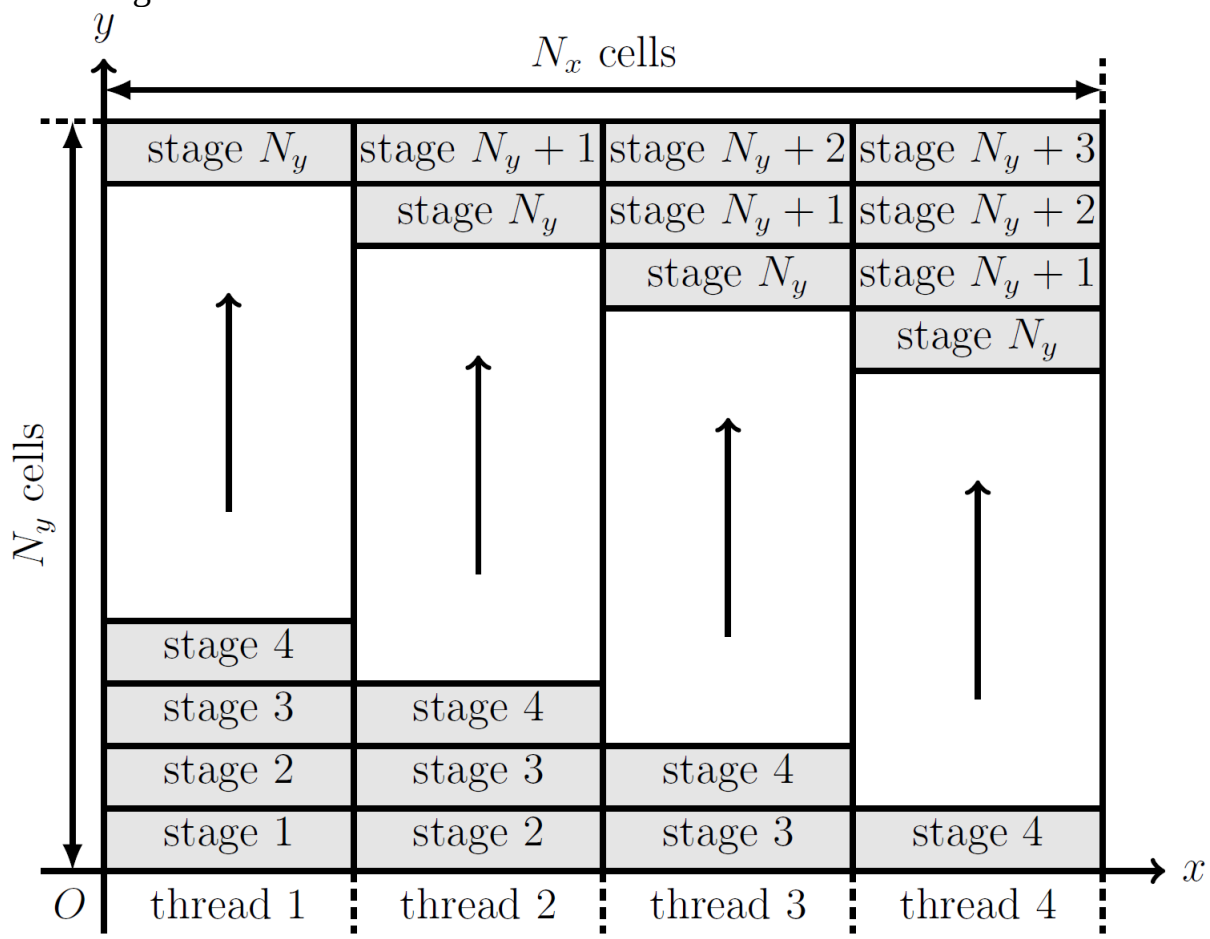

Fig. 1. Schematic view of parallel space marching computations for four threads.

Suppose that the computational domain consists of $N_x \times N_y$ cells and the number of threads is equal to p. The grid is divided into p parts in the x direction. Assume that N_x / p is an integer. The computation at a single time step within each stage of the Runge–Kutta method is represented as a sequence of stages. At every stage, each of the threads involved executes computations within a cell block of size $(N_x \times p)\times 1$. All threads perform computations in parallel and independently. The next stage begins when all threads have completed all their procedures at the preceding stage; i.e., a barrier for threads is necessary at the end of every stage. The sequence of stages shown in Fig. 1 guarantees that, at every stage, in each block all boundary values required for the space marching method are either known from the boundary conditions or have been computed earlier.

The efficiency of parallelization can be roughly estimated neglecting the loss due to the synchronization of the threads. Let T_1 denote the computation time within a single cell. Then the running time of the sequential algorithm at a single time step is $N_x N_y T_1$. The number of stages in the parallel method is $N_y + p - 1$, and each thread

computes N_x / p cells, so the computation time is $T_1(N_y + p - 1)N_x$ and the speedup S is given by

$$S = \frac{N_x N_y T_1}{T_1(N_y + p - 1)N_x / p} = \frac{p}{1 + (p-1)/N_y}. \tag{50}$$

This formula shows that, even if the ratio p / N_y is not very small, the theoretical speedup is close to the ideal one.

It should be noted that the block size can be chosen depending on the cache size and other parameters of the computing device in order to improve the scalability of the computations.

The scalability of the parallel algorithm was checked by computing the test initial–boundary value problem

$$\begin{aligned} &u_t + au_x + bu_y + \sigma u - q = 0, (x, y) \in [0,1] \times [0,1], t \in [0, T], \\ &u(x, y, 0) = u^0(x, y), u(0, y, t) = 0, u(x, 0, t) = 0 \end{aligned} \tag{51}$$

with an initial condition in the form of the compactly supported smooth pulse

$$u^0(x, y) = \begin{cases} \left(1 - (4x-1)^2\right)^7 \left(1 - (4y-1)^2\right)^6, & |4x-1| \le 1, |4y-1| \le 1, \\ 0, & |4x-1| > 1, |4y-1| > 1. \end{cases} \tag{52}$$

The parameters were specified as $a = b = 1, T = 0.5, \sigma = 1$, the source was defined as $q = \sigma u^0(x - at, y - bt)$, and the exacxt solution was given by $u^{ex}(x, y, t) = u^0(x - at, y - bt)$. The computations were executed on a grid consisting of 2048×2048 cells. The fifth-order accurate ESDIRK75 method from [11] was used for time stepping.

The test computations were performed on a computational node of the MVS-10P system consisting of two 8-core Xeon E5-2690 processors at the Joint Supercomputer Center of the Russian Academy of Sciences and on an Intel Xeon Phi 5110p coprocessor at the Laboratory of Mathematical Simulation of Nonlinear Processes in Gas Media of the Moscow Institute of Physics and Technology. The Intel Xeon Phi 5110p coprocessor contains 61 physical cores, each supporting four threads.

Figure 2 shows the computation time of a single time step as a function of the number of threads on a logarithmic scale. In both cases, the speedup of the algorithm is close to the ideal one. On the Xeon E5-2690 processor with 16 threads, the speedup is equal to 11 and the efficiency of parallelization is approximately 70%. On the Intel Xeon Phi 5110p coprocessor with 60 threads, the speedup is 47 and the efficiency of parallelization is about 80%. These results suggest that the parallel space marching algorithm is well scalable on shared memory systems. Moreover, the simple logical structure of the algorithm makes it possible to easily use vectorization, which can

potentially speed up the computations by four times on Xeon processors and by eight times on Xeon Phi processors.

The scheme of the algorithm described can be extended nearly without modifications to the three-dimensional case. Specifically, the grid is divided, for example, in the *z* direction into two-dimensional layers, and the computations in each layer are executed in the order shown in Fig. 1. The only difference is that a linear system of eight equations is solved in each cell.

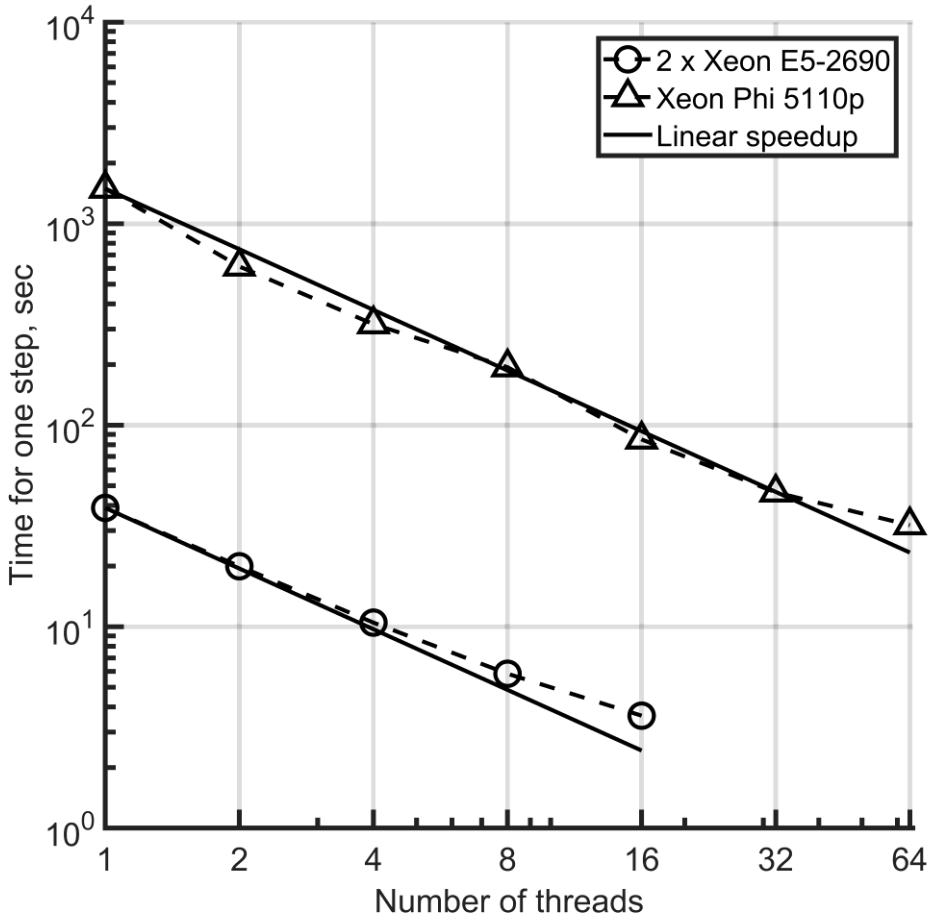

Fig. 2. Computation time per step against the number of threads.

4.2. Parallel factorized algorithm

At every iteration step of the factorized 2D scheme, the space marching method can be used to independently solve the one-dimensional problems (21) along each cell layer $j = const, j = \overline{1, N_x}$, and the one-dimensional problems (22), (23) along the cell layer $k = const, k = \overline{1, N_y}$. Algorithmically, these operations represent one-dimensional loops. On shared memory systems, they can be parallelized with the help of OpenMP by adding the simple instruction !$omp parallel do. Good load balancing and good scalability of the algorithm can be expected if the number of cells in each direction is a multiple of the number of threads or if it is not a multiple, but much greater than the number of threads.

The scalability of the parallel factorized algorithm was examined as applied to test problem (51), (52). In practice, such problems are solved on relatively small grids with a number of cells being $\leq 10^2$ in each direction. In the three-dimensional case, this yields a grid with 10^6 cells. Therefore, it is important to compare the scalability of the parallel space marching algorithm and the parallel factorized algorithm on such grids: in the sequential case, the factorized scheme takes more time than space marching, because of the large number of iterations. However, the parallel space marching algorithm is poorly scalable when the ratio p / N_x is large.

In the factorized scheme, the stopping criterion was as follows: at every step s of the Runge–Kutta method, in solving an ODE system of semidiscrete bicompact scheme with a right-hand side f, the difference between the values $k_s = f(t + c_s\tau, u^n + \tau\sum_{q=1}^{s} a_{sq}k_q)$ at the current ith and preceding $(i-1)$th iteration steps must be less in absolute value than the prescribe parameter tol:

$$\max_{j,k} |(k_s^{(i)})_{j,k} - (k_s^{(i-1)})_{j,k}| < tol, \qquad (53)$$

where c_s, a_{sq} are the coefficients of the Runge–Kutta method (see Appendix). As an initial approximation, we used zero values of $(k_s)_{j,k}$.

These features of algorithms are especially important in using modern computational architectures with a large number of threads and a relatively low efficiency of an individual core (computing device). For example, Intel processors of the last Xeon Phi generation have 61 to 72 cores per processor, and up to four threads can be executed on each core. Moreover, such processors possess low efficiency and storage as compared with processors having fewer cores (8–12). Therefore, to use the capabilities of such processors to a full extent, we need algorithms that are well scalable on small-sized problems.

In Fig. 3 the computation time required for the problem is plotted as a function of the number of threads used in the space marching and factorized algorithms run on a 12-core Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70 GHz processor. Problem (51), (52) was solved by applying the SDIRK54 method [12] on a grid consisting of 48x48 cells. The number of time steps was 48. The parameter tol was specified as $tol = 10^{-2}$, and the average number of iterations required for achieving the prescribed accuracy was equal to 3.65. The factorized scheme gave the same error as the usual scheme: $\| Err \|_\infty = 4.0 \times 10^{-4}$.

As was expected, the space marching algorithm is poorly scalable on such a fine grid, since the ratio of the number of threads to the number of cells in each direction is rather large (0.25 for 12 threads). The efficiency of parallel execution is 40% for 12 threads.

The factorized algorithm demonstrates good scalability up to 12 threads, and its efficiency is equal to 74%. The sequential factorized algorithm is slower by a

factor of 1.6 than sequential space marching. However, due to its good scalability, the factorized scheme on 12 threads is faster than space marching.

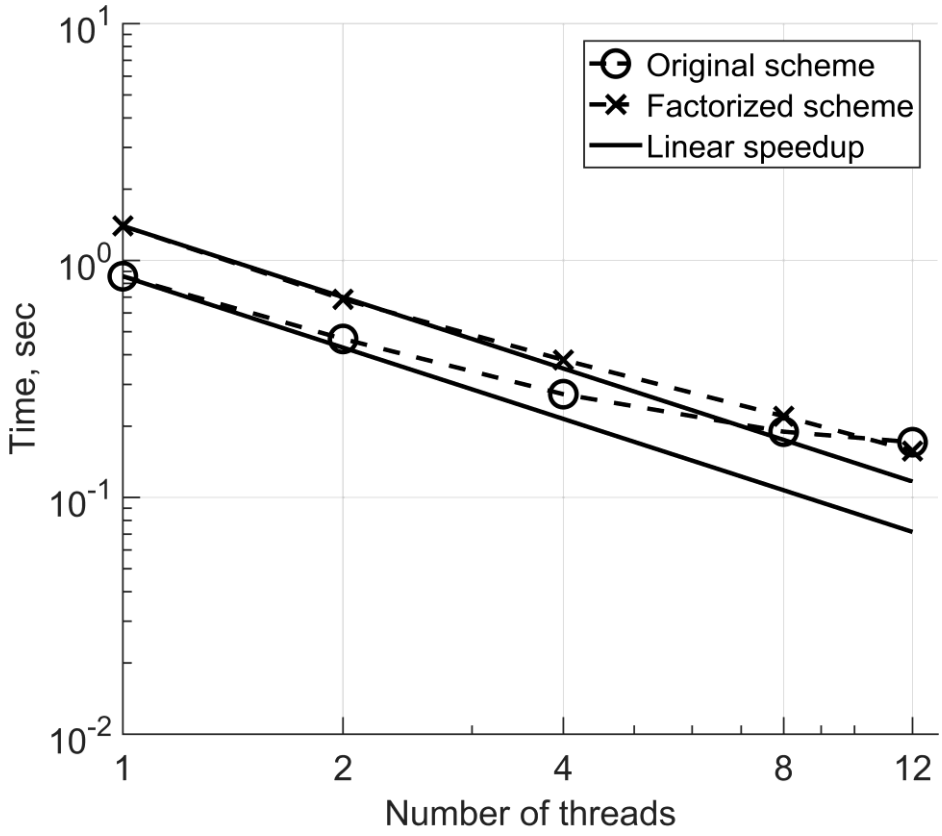

Fig. 3. Computation time against the number of threads for the original and factorized schemes.

In Fig. 4, the computation time required for solving problem (51), (52) on a grid of 60x60 cells with the number of time steps equal to 60 is plotted against the number of threads for the space marching and factorized algorithms run on an Intel Xeon Phi 5110p processor with 61 physical cores, each executing up to four threads. Here, $tol = 10^{-2}$ and the average number of iterations used for achieving the prescribed accuracy is equal to 2.77. The factorized scheme gives the same error as the usual scheme: $\| Err \|_{\infty} = 1.6 \times 10^{-4}$.

In this example, the ratio of the maximum number of threads to the number of cells in each direction is 1, so the space marching algorithm is scalable only up to 12 threads. The efficiency is 50% for 12 threads and 4% for 60 threads. This means that the parallel space marching algorithm fails to use the entire efficiency of the Xeon Phi processor as applied to such problems.

The factorized algorithm exhibits a satisfactory scalability. Its efficiency is equal to 50% for 60 threads. On a single core, the factorized algorithm is slower by a factor of 1.6 than the sequential space marching algorithm. On 60 cores, the computation time required for the factorized scheme is 1/3 times as much as the least computation time for space marching (on 12 cores).

These test computations show that the factorized scheme is preferable for parallel implementation: first, the factorized sequential algorithm is easy to parallelize and, second, on grids of middle size (which are most frequently used in practice) the factorized scheme requires less time. It should be noted that the advantage of the factorized scheme is even greater when three-dimensional problems are solved.

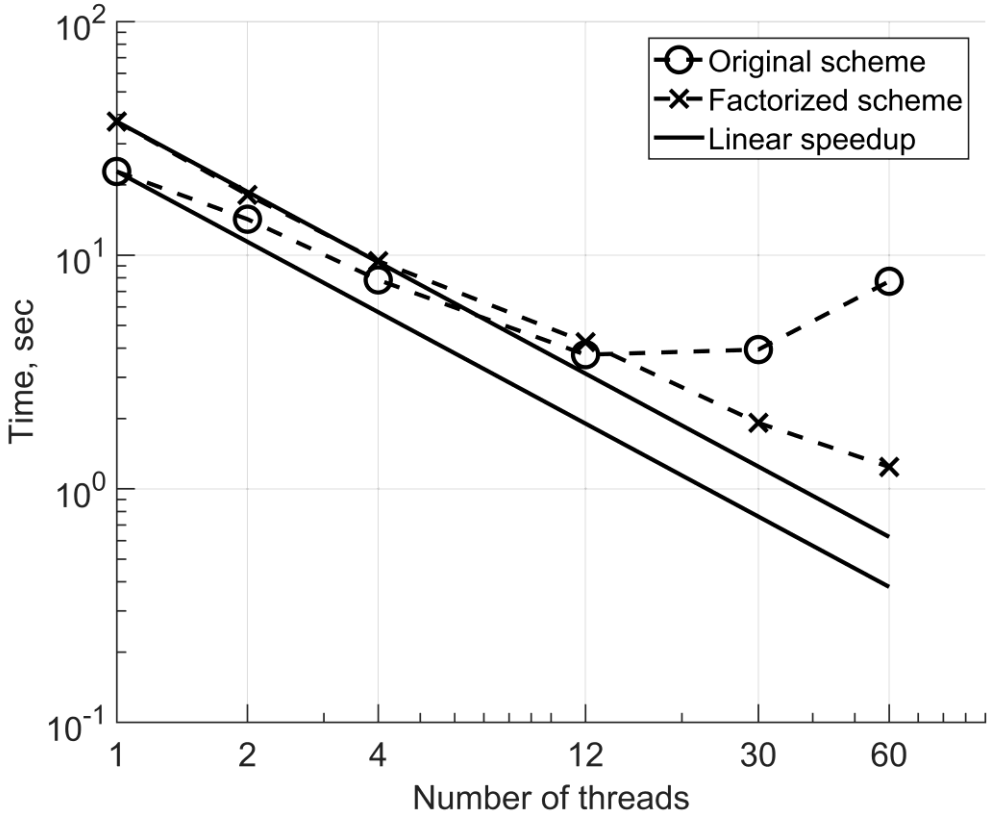

Fig. 4. Computation time against the number of threads for the original and factorized schemes.

5. Conclusions

The efficiency of two parallel algorithms for solving equations of high-order accurate bicompact schemes as applied to the multidimensional inhomogeneous transport equation was compared. The first of them is the space marching algorithm for computing unfactorized schemes, and the second one is based on the approximate factorization of multidimensional schemes. The latter algorithm involves iterations used to preserve the high (higher than the second) order of accuracy of bicompact schemes in time. The convergence of these iterations for the nonstationary two- and three-dimensional linear inhomogeneous transport equations with constant positive coefficients was proved. The test computations showed that the algorithm based on factorized bicompact schemes is preferable in terms of parallel implementation.

Appendix: Coefficients of diagonally implicit Runge–Kutta methods

The Butcher tableau for SDIRK54 [12], which is a fourth-order DIRK method with five implicit stages, is given by

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|ccccc} 1/4 & 1/4 & & & & \\ 0 & -1/4 & 1/4 & & & \\ 1/2 & 1/8 & 1/8 & 1/4 & & \\ 1 & -3/2 & 3/4 & 3/2 & 1/4 & \\ 1 & 0 & 1/6 & 2/3 & -1/12 & 1/4 \\ \hline & 0 & 1/6 & 2/3 & -1/12 & 1/4 \end{array}$$

Below is the Butcher tableau for ESDIRK75, which is a fifth-order DIRK method with six implicit stages and one explicit stage [11]:

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|cccccc} 0 & 0 & & & & & \\ 0.4 & 0.2 & 0.2 & & & & \\ 0.6 & 0.25 & 0.15 & 0.2 & & & \\ 1 & 0.6214285714 & -0.9642857142 & 1.1428571428 & 0.2 & & \\ 0.8 & 0.2657743915 & 0.4927153439 & -0.3035089947 & 0.1450192592 & 0.2 & \\ 0.2 & 0.0448367885 & -0.2368271451 & 0.2540183114 & -0.1402888244 & 0.0782608695 & 0.2 \\ 1 & 0.0659722222 & 0.1736111111 & 0.1736111111 & -0.1340277777 & 0.2604166666 & 0.2604166666 & 0.2 \\ \hline & 0.0659722222 & 0.1736111111 & 0.1736111111 & -0.1340277777 & 0.2604166666 & 0.2604166666 & 0.2 \end{array}$$

References

1. Shilkov A.V., Gertsev M.N., Aristova E.N., Shilkova S.V. Benchmark «line-by-line» calculations of atmospheric radiation // Computer Research and Modeling. 2012. V. 4, No. 3, P. 553–562 (Russian).

2. Rozanov V.B., Vergunova G.A., Aristova E.N. et al. Interaction of laser radiation with a low-density structured absorber // JETP. 2016. V. 122. No. 2, P. 256–276.

3. Baydin D.F., Aristova E.N. 3D hexagonal parallel code QuDiff for calculating a fast reactor's critical parameters // Math. Models Comput. Simul. 2016. V. 8, No. 4. P. 446–452.

4. Rogov B.V., Mikhailovskaya M.N. Fourth-order accurate bicompact schemes for hyperbolic equations // Dokl. Math. 2010. V. 81, No. 1. P. 146–150.

5. Aristova E.N., Rogov B.V. Boundary conditions implementation in bicompact schemes for the linear transport equation // Math. Models Comput. Simul. 2013. V. 5, No. 3. P. 199–207.

6. Aristova E.N., Rogov B.V., Chikitkin A.V. Optimal monotonization of a high-order accurate bicompact scheme for the nonstationary multidimensional transport equation // Comput. Math. Math. Phys. 2016. V. 56, No. 6. P. 962–976.

7. Chikitkin A.V., Rogov B.V., Aristova E.N. High-order accurate bicompact schemes for solving the multidimensional inhomogeneous transport equation and their efficient parallel implementation // Dokl. Math. 2016. V. 94, No. 2. P. 517–522.

8. Bragin M.D., Rogov B.V. Minimal dissipation hybrid bicompact schemes for hyperbolic equations // Comput. Math. Math. Phys. 2016. V. 56, No. 6. P. 947–961.

9. Bragin M.D., Rogov B.V. Iterative approximate factorization of difference operators of high-order accurate bicompact schemes for multidimensional nonhomogeneous quasilinear hyperbolic systems // Comput. Math. Math. Phys. 2018. V. 58, No. 3. P. 295–306.

10. Rogov B.V., Bragin M.D. On the convergence of the method of iterative approximate factorization of difference operators of high-order accurate bicompact scheme for nonstationary three-dimensional hyperbolic equations // Keldysh Institute Preprints. 2018. No. 132. 16 p. doi:10.20948/prepr-2018-132-e. URL: http://library.keldysh.ru/preprint.asp?id=2018-132&lg=e.

11. Skvortsov L.M. Diagonally implicit Runge–Kutta FSAL methods for stiff and differential-algebraic systems // Matem. Mod. 2002. V. 14, No. 2. P. 3–17 (Russian).

12. Skvortsov L.M. Diagonally implicit Runge-Kutta methods for stiff problems // Comput. Math. Math. Phys. 2006. V. 46, No. 12. P. 2110–2123.

13. Watkins D.S. Fundamentals of matrix computations. New York: Wiley Interscience, 2002.

Contents