High-Performance Computation of Initial Boundary Value Problems

Valery Il'in(✉)

Institute of Computational Mathematics and Mathematical Geophysics,
Novosibirsk State University, Novosibirsk, Russia
ilin@sscc.ru
https://icmmg.nsc.ru/ru/content/employees/ilin-valeriy-pavlovich

Abstract. This paper considers the efficient methods and high- performance parallel technologies for the numerical solution of the multi-dimensional initial boundary value problems, with a complicated geometry of a computational domain and contrast properties of a material on the heterogeneous multi-processor systems with distributed and hierarchical shared memory. The approximations with respect to time and space are carried out by implicit schemes on the quasi-structured grids. At each time step, the iterative algorithms are used for solving the systems of linear or nonlinear equations that, in general, are non-symmetric with a special choice of the initial guess. The scalable parallelism is provided by two-level iterative domain decomposition methods, with parameterized intersection of subdomains in the Krylov subspaces, which are accelerated by means of a coarse grid correction and polynomial or other types of preconditioning. A comparative analysis of the performance and speed up of the computational processes is presented, based on a simple model of parallel computing and data structures.

Keywords: Nonstationary boundary value problems
High-order approximations · Stability · Initial guess
Iterative processes · Domain decomposition · Scalable parallelism

1 Introduction

In this paper, we consider various numerical approaches to solving multi-dimensional initial boundary value problems (IBVPs) for nonstationary partial differential equations (PDEs) in complicated computational domains with real data, and offer a comparative analysis of their performance on modern heterogeneous multi-processor systems (MPS) with distributed and hierarchical shared memory. In general, we will assess the efficiency of numerical solutions of a class of mathematical problems by the volume of computational resources required to provide the accuracy needed on a particular type of MPS. Of course, such a

The work is supported by the Russian Science Foundation (grant 14-11-00485 P) and the Russian Foundation for Basic Research (grant 16-29-15122 ofi-m).

L. Sokolinsky and M. Zymbler (Eds.): PCT 2018, CCIS 910, pp. 186–199, 2018.
https://doi.org/10.1007/978-3-319-99673-8_14

statement is not quite clear, and we should refine many details in this important concept. A simple way to do this consists in measuring run time and using these measurements as a performance criterion. Other tools could be the estimation of computing time and communication time based on a certain model for the implementation of the problem on MPS. In what follows, we will use the second approach and consider simple representations for both the arithmetical execution time T_a and the communication time T_c. In total, the performance of the numerical solution of the problem will be defined by the run time $T_t = T_a + T_c$. We suppose here that the arithmetical units do not work during data transfer, although the whole picture can be more complicated.

The performance is characterized by two main aspects: mathematical efficiency of numerical methods and computational technologies for software implementation on a particular hardware architecture. The algorithmic issues depend on two main mathematical stages: discrete approximation of the original continuous problem and numerical solution of the resulting algebraic task. It is important that we do not examine model problems but problems with real data: multi-dimensional boundary value problems in computational domains with a complicated geometry, multi-connected and multi-scaled (in general) piece-wise smooth boundaries and contrast properties of a material, which provide singularities of the solution to be sought. It means that in order to ensure a high numerical resolution and accuracy of the computational model, we must use fine grids with a very small time step τ and a spatial step h. So we have, in principle, a "super task" with a very large number of degrees of freedom (d. o. f.) or a high dimension of the corresponding discrete problem. In general, the original problem can be nonlinear and multi-disciplinary or multi-physical, i.e. it is described either by a system of PDEs or by the corresponding variational relations for unknown vector functions. Also, the mathematical statement may not be a direct one with all the coefficients of the equations given and with initial and boundary conditions, it may be instead an inverse problem that includes variable parameters to be found from the condition of minimization of some given objective functional of the unknown solution. For simplicity, however, we will mainly consider direct IBVPs for a single linear scalar equation. A review of the corresponding models can be found in [11] (see also the literature cited therein). We also do not consider in detail other computational steps of the mathematical modeling (grid generation, post-processing, visualization of the results, etc.) since they are of a more general type and are almost defined by the problem specifications.

The approximation approaches are divided into temporal and spatial discretizations. If we carry out the spatial approximation at first by the finite volume method, the finite element method or any other method [2], then we will obtain a system of ordinary differential equations (ODEs). There are various explicit and implicit, multi-stage and/or multi-step algorithms of different orders [3] that may be applied to solve such a system. It is important to remark that modern computational trends give preference to methods of high order of accuracy since

they make it possible to decrease the amount of data communication, which is not only a slow operation but an energy consuming process.

If we use schemes that are implicit with respect to time, thereby providing a stable procedure for numerical integration, it will be necessary to solve at each step a system of linear algebraic equations (SLAEs) of special type, with large sparse matrices. This is the most expensive computational stage as it requires a large number of arithmetical operations and a big amount of memory, and both grow nonlinearly when the number of d. o. f. increases [4]. In this case, the main tool to ensure a high performance is the scalable parallelization of domain decomposition methods (DDM), which belongs to a special field of computational algebra (see, for example, [5–7]). A detailed review of parallelization approaches for nonstationary problems is presented in [8,9]. In what follows, we will consider direct IBVPs only, whereas the ideal of engineering problems consists in solving inverse problems, which involves computing optimized parameters of the mathematical model under the condition of constrained optimization of a given objective functional. However, this is a topic that requires a special research.

The paper is structured as follows. In Sect. 2, the example of the heat transfer equation is considered regarding various aspects of temporal and spatial approximations. Section 3 deals with geometrical and algebraic issues of DDM as applied to nonstationary problems. In the last Section, we discuss an application of the given analysis for the parallel solution of practical problems.

2 Discretization Issues of Nonstationary Problems

Let us consider the initial boundary value problem (IBVP)

$$\begin{gathered} \frac{\partial u}{\partial t} + L(u) = f(\mathbf{x}, t), \quad \mathbf{x} \in \Omega \subset \mathcal{R}^d, \quad d \geq 2, \\ \bar{\Omega} = \Omega \cup \Gamma, \quad 0 < t \leq T_e < \infty, \quad u\big|_{t=0} = u^0(\mathbf{x}), \\ l(u)\big|_{\Gamma} = g(\mathbf{x}, t), \quad \mathbf{x} = (x_1, \ldots, x_d), \end{gathered} \tag{1}$$

where t and $\mathbf{x}$ are, respectively, temporal and spatial variables; $u^0(\mathbf{x})$ is a given initial guess; L is some differential operator, possibly, a nonlinear one and, in general, a matrix operator. In this case, the unknown $u = (u_1, \ldots, u_{N_u})^T$ is a vector function. We call task (1) a multi-disciplinary or multi-physics problem. Here $\bar{\Omega}$ denotes a bounded d-dimensional computational domain with boundary $\Gamma = \bigcup_{k=1}^{N_\Gamma} \Gamma_k$; l is a boundary-condition operator, which can be of various types l_i (Dirichlet, Neumann or Robin) at the corresponding boundary segments Γ_i; f and g are functions that may depend on the unknown solution. We suppose that IBVP (1) describes a practical problem with real data. This means, for example, that the computational domain $\bar{\Omega}$ may have a complicated geometry, possibly, with multi-connected piecewise smooth curvilinear boundary surfaces Γ_k. As an illustration, the following linear scalar differential operator of the second order is considered in (1):

$$Lu = -\sum_{i,j=1}^{n} \frac{\partial}{\partial x_i}\left(a_{i,j}(\mathbf{x})\frac{\partial u}{\partial x_j}\right) + \sum_{i=1}^{n} b_i \frac{\partial u}{\partial x_i} + cu = f(\mathbf{x}). \tag{2}$$

The corresponding boundary conditions can be written down as

$$\alpha_k u + \beta_k \sum_{i,j=1}^{d} a_{i,j}\frac{\partial u}{\partial x_j}\cos(\mathbf{n}, x_j) = g_k, \quad |\alpha_k| + |\beta_k| \neq 0, \quad \mathbf{x} \in \Gamma_k, \tag{3}$$

where $\mathbf{n}$ denotes the outward unit normal to Γ_k.

Note that if the original system of PDEs is complex and has temporal derivatives of high order, it can always be transformed into a first order real system by including additional unknown functions. Also, formulas (1) can describe an inverse problem if it contains variable parameters $p = (p_1, \ldots, p_{N_p})^T$, which should be optimized by means of the minimization of a prescribed objective functional. For simplicity, the original IBVP is written in the classical differential form, and it can be re-described in a variational style. It is supposed that the input data ensures the smoothness of the numerical methods in all cases. One more remark: in general, some boundary segments Γ_k can move, but we will primarily consider the boundary Γ fixed.

The approximation of the original problem (1) can be made in two steps. In the first step, we generate a spatial grid Ω^h, which, in the three-dimensional case ($d = 3$), for example, a set of nodes (vertices), edges, faces (possibly, curvilinear), and finite elements or volumes. After applying the spatial approximation using the finite volume method, the finite element method, the discontinuous Galerkin method or other approaches, we obtain a system of N ordinary differential equations:

$$\begin{gathered} B\dot{u}^h + Au^h = f^h, \\ \dot{u}^h, u^h, f^h \in \mathcal{R}^N; \quad B, A \in \mathcal{R}^{N,N}, \end{gathered} \tag{4}$$

where $\dot{u}$ denotes the time derivative of u, and the components of the vector $f^h = \{f_l\}$ and of the matrices $B = \{b_{l,l}\}$ and $A = \{a_{l,l}\}$ may, in general, depend on the unknown solution.

In a simple case, the unknown vector $u^h = \{u_l^h\}$ consists of approximate nodal values of the original solution $(u)^h = \{u(\mathbf{x}_l)\}$ but, basically, it can include, for instance, other functionals, and some derivatives of u at different points. The vector $(u)^h$ of the discretized unknown solution satisfies the equation

$$B(\dot{u})^h + A(u)^h = (f)^h + \psi^h, \quad \psi = O(h^\gamma), \tag{5}$$

where ψ^h is the spatial approximation (truncation) error of Eq. (4), h is the maximal distance between neighboring grid nodes, and $\gamma > 0$ is the order of the approximation. The matrix A in (4) can be defined as

$$(Au^h)_l \equiv a_{l,l}u_l + \sum_{l' \in \omega_l} a_{l,l'}u_{l'} = f_l, \quad l \in \Omega^h, \tag{6}$$

where Ω^h can be considered to be a set of indices that determine the number $N = O(h^{-1})$ of all unknowns, and ω_l denotes the stencil of the lth node, i.e. the

set of neighboring nodes. In other words, ω_l is the union of the column numbers of the nonzero elements in the lth row of the matrix A (the number of such values will be denoted as N_l). The total set made up by all ω_l, $l = 1, \ldots, N$, determines the portrait of the sparse matrix A ($N_l \ll N$). Note that N_l does not depend on the matrix dimension N, which can be estimated as $N \approx 10^7 \div 10^{10}$ for a large-size real problem. Moreover, for $d = 3$, we have $N_l \approx 10 \div 30$ for the first or the second order schemes, whereas $N_l > 100$ for the fourth to sixth orders of accuracy.

To solve ODEs (4), it is possible to apply various multi-stage and/or multi-step numerical integrators of different orders of accuracy with respect to the time step τ_n. For simplicity, we consider the two-step weighted scheme

$$B\frac{u^{n+1}-u^n}{\tau_n} + \theta(Au^{n+1} - f^{n+1}) = (1-\theta)(f^n - Au^n), \qquad \theta \in [0,1], \quad n = 0, 1, \ldots, \tag{7}$$

where n is a time-step number; $\theta = 0$ corresponds to the explicit Euler method, otherwise, we have an implicit algorithm. If $\theta = 1/2$, formula (7) corresponds to the Crank–Nicolson scheme, which has the second order approximation error $\psi^\tau = O(\tau^2)$, $\tau = \max_n\{\tau_n\}$; besides, $\psi^\tau = O(\tau)$ for $\theta \neq 1/2$. Here and in what follows, we omit the index "h" for the sake of brevity. If we denote by $(u)^n$ a vector whose components are the values of the exact solution $u(t_n, \mathbf{x}_l)$, and substitute it for u^n in (7), then we have

$$B\frac{(u)^{n+1}-(u)^n}{\tau_n} + \theta[A(u)^{n+1} - (f)^{n+1}] = (1-\theta)[(f)^n - A(u)^n] + \psi^n, \tag{8}$$

where $\psi^n = \psi^\tau + \psi^h$ is the total, i.e. temporal and spatial, approximation error of the numerical scheme.

If relations (7) are nonlinear, we should use quasi-linearization for each n, i.e. apply the iterative process and solve SLAEs at each "nonlinear" step.

In the implicit scheme with $\theta \neq 0$, we have to solve a large algebraic system by some iterative approach, even for the original linear IBVP, since direct (noniterative) algorithms are too expensive in our case (matrices $B + \tau_n\theta A$ are supposed to be nonsingular). Finally, from (7), we do not calculate u^{n+1} but some approximate value $\tilde{u}^{n+1}$, which produces the residual vector

$$r^n = (1-\theta)(\tilde{f}^n - A\tilde{u}^n) - B\frac{\tilde{u}^{n+1}-\tilde{u}^n}{\tau_n} + \theta(A\tilde{u}^{n+1} - \tilde{f}^{n+1}). \tag{9}$$

Now let us determine the total vector of the original solution, $z^{n+1} = (u)^{n+1} - \tilde{u}^{n+1}$. It follows from (8) and (9) that the vectors z^{n+1}, r^n and ψ^n are connected by a relation that, for the reduced original problem (the elements of the matrices A and B, as well as those of the vectors f^n are supposed to be independent of u and t), can be written down as

$$C_1 z^{n+1} = C_2 z^n + \tau_n(\psi^n - r^n), \qquad C_1 = B + \theta A, \quad C_2 = B - (1-\theta)A. \tag{10}$$

If

$$\|\tau_n(\psi^n - r^n)\| \le \tau\|\psi\| \tag{11}$$

for some vector norm, then we obtain from (10) the following estimate:

$$\begin{gathered}\|z^{n+1}\| \le \rho\|z^n\| + \tau\rho_1\|\psi\|, \\ \rho = \|C_1^{-1}C_2\|, \quad \rho = \|C_1^{-1}\|.\end{gathered} \tag{12}$$

It follows from the considerations above that if the iterative residual r^n at each time step has the same order of accuracy as the approximation error ψ^n, then the total solution error does not change the order of accuracy. One important issue in solving a nonstationary problem consists in choosing the initial guess for the iterative solution of SLAEs at each time step. It is natural that the u^n values would be a good approximation to u^{n+1} to reduce the number of iterations, provided that the time step τ_n is sufficiently small. Another simple approach is based on the linear extrapolation with respect to time:

$$u^{n+1} = u^n + (u^n - u^{n-1})\tau_n/\tau_{n-1} + O(\tau^2). \tag{13}$$

In this case, we need to save the numerical solution for one additional time step. One of the popular methods for solving ODEs is based on the application of predictor-corrector schemes. For example, if we use in (7) the Crank–Nicolson scheme, for which $\theta = 1/2$ and $\psi^\tau = O(\tau^2)$, or any other implicit method, this involves including a preliminary predictor stage for computing an approximate value of u^{n+1} by the simple explicit formula

$$B(\hat{u}^{n+1} - u^n) = \tau_n(f^n - Au^n) \equiv \tau_n r^n, \tag{14}$$

where B is a diagonal or another easily invertible matrix, and $\hat{u}^{n+1}$ is considered to be a predicted value of u^{n+1}. It can be interpreted as a zero iteration, $u^{n+1,0} = \hat{u}^{n+1}$, and corrected by m iterations of the form

$$\begin{gathered}B(u^{n+1,s} - u^n) = \tau_n[\theta(f^{n+1} - Au^{n+1,s-1}) + (1-\theta)(f^n - Au^n)], \\ s = 1, \ldots, m.\end{gathered} \tag{15}$$

This approach is called PC^m and in practice provides an acceptable small residual

$$r^{n+1,s} = \tau_n[\theta(f^{n+1} - Au^{n+1,s-1}) + (1-\theta)r^n] - B(u^{n+1,s} - u^n)$$

in a few iterations.

An improved idea to choose the initial guess can be proposed based on the least-squares method (LSM; see [10]). Let us save several previous time-step solutions $u^{n-1}, \ldots, u^{n-q}$, and compute the value $u^{n+1,0}$ by means of the linear combination

$$\begin{gathered}u^{n+1,0} = u^n + c_1v_1 + \ldots + c_qv_q = u^n + Vc, \\ v_l = u^n - u^{n-l}, \quad l = 1, \ldots, q, \\ c = (c_1, \ldots, c_q)^T \in \mathcal{R}^q, \quad V = (v_1, \ldots, v_q) \in \mathcal{R}^{N,q}.\end{gathered} \tag{16}$$

The system of Eq. (7) can be rewritten as

$$\begin{aligned} Cu^{n+1} &\equiv (\tau_n^{-1}B + \theta A)u^{n+1} = g^{n+1}, \\ g^{n+1} &= [\tau_n^{-1}B + (1-\theta)A]u^n + \theta f^{n+1} + (1-\theta)f^n. \end{aligned} \tag{17}$$

So it follows from relation (16) that the initial residual $r^{n+1,0} = g^{n+1} - Cu^{n+1,0}$ of system (17) satisfies the equality

$$r^{n+1,0} = r^n - CVc. \tag{18}$$

Formally, here we can set $r^{n+1,0} = 0$ and obtain overdetermined SLAE for the vector c:

$$Wc \equiv CVc = \tau^n, \quad W \in \mathcal{R}^{N,q}. \tag{19}$$

The generalized normal (with a minimal residual) solution of this system can be computed by the SVD (Singular Value Decomposition) algorithm or by the least-squares method (LSM), which gives the same result in exact arithmetics. The LSM gives the "small" symmetric system

$$Gc \equiv W^T Wc = W^T \tau^n, \quad G = V^T C^T CV \in \mathcal{R}^{q,q}, \tag{20}$$

which is nonsingular if W is a full-rank matrix. It is easy to verify that Eq. (20) implies the orthogonality property of the residual:

$$W^T r^{n+1,0} = 0. \tag{21}$$

Note that, instead of the LSM approach (20), (21), it is possible to apply the so-called deflation principle [11], which uses the following orthogonality property instead of (21):

$$V^T r^{n+1,0} = 0. \tag{22}$$

In this case, we have to solve SLAE

$$Hc \equiv V^T CVc = V^T c, \quad H \in \mathcal{R}^{q,q}, \tag{23}$$

to determine the vector c. If this vector is computed from system (20) or (23), then the initial guess $u^{n+1,0}$ for SLAEs (17) is determined from (16). For solving system (17) at each time step, it is natural to apply some preconditioned iterative method in Krylov subspaces. The stopping criterion of such iterations is

$$\|r^{n+1,m}\| = \|g^{n+1} - Cu^{n+1,m}\| \le \varepsilon \|g^{n+1}\| \tag{24}$$

for some given tolerance $\varepsilon \ll 1$. If condition (24) is satisfied, we set $u^{n+1} = u^{n+1,m}$ and go to the next time step.

3 Geometrical and Algebraic Issues of Algorithms

The general scheme of solution of nonsteady IBVPs can be described as having two main parts. The first one consists in generating an algebraic system at each time step. Usually, this stage is parallelized easily enough, with a linear speedup when the number of computer units grows. The more complicated stage includes solving the algebraic system of equations, linear or nonlinear (SLAEs or SNLAEs); such tasks require a large amount of computational resources (memory and number of arithmetic operations) as the number of d. o. f. grows.

If we have SNLAEs at each time step, the solution methods involve a two-level iterative process. At first, some type of quasi-linearization is applied, and at each "nonlinear" iteration (Newton or Jacobi type, for example), we need to solve SLAEs, usually with a large sparse ill-conditioned matrix. This second stage will be the main issue in our considerations in what follows.

The main tool to achieve scalable parallelism on modern MPS is based on a domain decomposition method that can be interpreted in an algebraic or geometrical framework. Also, domain decomposition methods can be considered at both the continuous and the discrete levels. We use the second approach and suppose that the original computational domain Ω has already been discretized into a grid computational domain Ω^h. So, in what follows, the DDM is implemented only in grid computational domains, and the upper index "h" will be omitted for brevity.

Let us decompose Ω into P subdomains (with or without overlap):

$$\Omega = \bigcup_{q=1}^{P} \Omega_q, \quad \bar{\Omega}_q = \Omega_q \cup \Gamma_q, \quad \Gamma_q = \bigcup_{q' \in \omega_q} \Gamma_{q,q'}, \quad \Gamma_{q,q'} = \Gamma_q \cap \bar{\Omega}_{q'}, \quad q' \neq q. \tag{25}$$

Here Γ_q is the boundary of Ω_q, which is composed of the segments $\Gamma_{q,q'}$, $q' \in \omega_q$, and $\omega_q = \{q_1, \ldots, q_{M_q}\}$ is a set of M_q contacting or conjugate subdomains. Formally, we can also denote by $\Omega_0 = R^d \setminus \Omega$ the external subdomain:

$$\bar{\Omega}_0 = \Omega_0 \cup \Gamma, \quad \Gamma_{q,0} = \Gamma_q \cap \bar{\Omega}_0 = \Gamma_q \cap \Gamma, \quad \Gamma_q = \Gamma_q^i \cup \Gamma_{q,0}, \tag{26}$$

where $\Gamma_q^i = \bigcup_{q' \neq 0} \Gamma_{q,q'}$ and $\Gamma_{q,0} = \Gamma_q^e$ stand for the internal and external parts of the boundary of Ω_q. We also define the overlap $\Delta_{q,q'} = \Omega_q \bigcap \Omega_{q'}$ of neighboring subdomains. If $\Gamma_{q,q'} = \Gamma_{q',q}$ and $\Delta_{q,q'} = \emptyset$, then the overlap of Ω_q and $\Omega_{q'}$ is empty. In particular, we suppose in (25) that each of the P subdomains has no intersection with Ω_0 ($\Omega_q \bigcap \Omega_0 = \emptyset$).

The idea of the DDM involves the definition of the sets of IBVPs that should be equivalent to the original problem (1) in all subdomains:

$$\begin{gathered} \frac{\partial u_q}{\partial t} + L u_q(\mathbf{x}) = f_q, \quad \mathbf{x} \in \Omega_q, \quad l_{q,q'}(u_q)\big|_{\Gamma_{q,q'}} = g_{q,q'} \equiv l_{q',q}(u_{q'})\big|_{\Gamma_{q',q}}, \\ q' \in \omega_q, \quad l_{q,0} u_q\big|_{\Gamma_{q,0}} = g_{q,0}, \quad q = 1, \ldots, P. \end{gathered} \tag{27}$$

Interface conditions in the form of Robin boundary conditions (instead of (3), for simplicity) are imposed in each segment of the internal boundaries of the subdomains, with the operators $l_{q,q'}$ from (27):

$$\alpha_q u_q + \beta_q \frac{\partial u_q}{\partial \mathbf{n}_q}\bigg|_{\Gamma_{q,q'}} = \alpha_{q'} u_{q'} + \beta_{q'} \frac{\partial u_{q'}}{\partial \mathbf{n}_{q'}}\bigg|_{\Gamma_{q',q}}, \qquad |\alpha_q| + |\beta_q| > 0, \quad \alpha_q \cdot \beta_q \geq 0. \tag{28}$$

Here $\alpha_{q'} = \alpha_q$ and $\beta_{q'} = \beta_q$; $\mathbf{n}_q$ is the outer normal to the boundary segment $\Gamma_{q,q'}$ of the subdomain Ω_q. Strictly speaking, two pairs of different coefficients, $\alpha_q^{(1)}, \beta_q^{(1)}$ and $\alpha_q^{(2)}, \beta_q^{(2)}$, should be given for conditions of type (28) on each piece $\Gamma_{q,q'}, q' \neq 0$, of the internal boundary. For example, $\alpha_q^{(1)} = 1, \beta_q^{(1)} = 0$ and $\alpha_q^{(2)} = 0, \beta_q^{(2)} = 1$ formally correspond, respectively, to the continuity of the solution sought and its normal derivative. The additive Schwarz algorithm in DDM is based on an iterative process in which the BVPs in each subdomain Ω_q are solved simultaneously, and the right-hand sides of the boundary conditions in (27) and (28) are taken from the previous iteration.

We implement the domain decomposition in two steps. At the first one, we define subdomains Ω_q without overlap, i.e. contacting grid subdomains have no common nodes, and each node belongs to only one subdomain. Then we define the grid boundary $\Gamma_q = \Gamma_q^0$ of Ω_q, as well as the extensions of $\bar{\Omega}_q^t = \Omega_q^t \cup \Gamma_q^t$, $\Omega_q^0 = \Omega_q$, $t = 0, \ldots, \Delta$, layer by layer:

$$\begin{aligned} &\Gamma_q \equiv \Gamma_q^0 = \left\{ l' \in \hat{\omega}_l, \;\; l \in \Omega_q, \;\; l' \notin \Omega_q, \;\; \Omega_q^1 = \bar{\Omega}_q^0 = \Omega_q \cup \Gamma_q^0 \right\}, \\ &\Gamma_q^t = \left\{ l' \in \hat{\omega}_l, \;\; l \in \Omega_q^{t-1}, \;\; l' \in \Omega_q^{t-1}, \;\; \Omega_q^t = \bar{\Omega}_q^{t-1} = \Omega_q^{t-1} \cup \Gamma_q^{t-1} \right\}. \end{aligned} \tag{29}$$

Here Δ stands for the parameter of extension or overlap.

At each time step, the algebraic interpretation of the DDM, after the approximations of BVPs (27) and (28), is described by the block version of SLAEs (17),

$$C_{q,q} u_q + \sum_{r \in \hat{\omega}_q} C_{q,r} u_r = g_q, \quad q = 1, \ldots, P, \tag{30}$$

where indices "$n+1$" have been omitted for brevity; $C_{q,q}$ and $u_q, f_q \in \mathcal{R}^{N_q^\Delta}$ are a block diagonal matrix and subvectors with components belonging to the corresponding subdomain Ω_q^Δ; N_q^Δ is the number of nodes in Ω_q^Δ.

The implementation of the interface conditions between adjacent subdomains can be described as follows. Let the lth node be a near-boundary one in the subdomain Ω_q. Then we write down the corresponding equation in the form

$$\begin{aligned} (D_{q,q} u)_l \equiv \left(c_{l,m} + \theta_l \sum_{m \notin \omega_q} c_{l,m} \right) u_l + \sum_{m \in \omega_q} c_{l,m} u_m = \\ = g_l + \sum_{m \notin \omega_q} c_{l,m} (\theta_l u_l - u_m). \end{aligned} \tag{31}$$

Here θ_l is some parameter that corresponds to different types of boundary conditions at the boundary Γ_q, namely $\theta_l = 0$ corresponds to the Dirichlet condition,

$\theta_l = 1$ corresponds to the Neumann condition, and $\theta_l \in (0,1)$ corresponds to the Robin boundary condition.

If we denote $D = \text{block-diag}\{D_{l,l}\}$, then a simple variant of DDM is described as the Schwarz (or block Schwarz–Jacobi) iterative method

$$Du^{s+1} = (D - C)u^s + g, \quad s = 0, 1, \ldots. \tag{32}$$

Improved versions of this approach are given by preconditioned algorithms in Krylov subspaces. Firstly, let us consider the advanced choice of the preconditioning matrices.

In the case of an overlapping domain decomposition, the additive Schwarz iterative algorithm is defined by the corresponding preconditioning matrix B_{AS}, which can be described as follows (see [7]). For the subdomain Ω_q^{Δ} with overlap parameter Δ, we define a prolongation matrix $R_{q,\Delta}^T \in \mathcal{R}^{N,N_q^{\Delta}}$ that extends the vectors $u_q = \{u_l,\ l \in \Omega_q^{\Delta}\} \in \mathcal{R}^{N_q^{\Delta}}$ to $\mathcal{R}^N$ according to the relations

$$(R_{q,\Delta}^T u_q)_l = \begin{cases} (u_q)_l & \text{if } \ l \in \Omega_q^{\Delta}, \\ 0 & \text{otherwise.} \end{cases}$$

The transpose of this matrix defines a restriction operator that restricts vectors in $\mathcal{R}^N$ to the subdomain Ω_q^{Δ}. The diagonal block of the preconditioning matrix B_{AS}, which represents the restriction of the discretized BVP to the qth subdomain, is expressed by $\hat{C}_q = R_{q,\Delta} C R_{q,\Delta}^T$. In these terms, the additive Schwarz preconditioner is defined as

$$B_{AS} = \sum_{q=1}^{P} B_{AS,q}, \ \ B_{AS,q} = R_{q,\Delta}^T \hat{C}_q^{-1} R_{q,\Delta}.$$

Also, it is possible to define the so-called restricted additive Schwarz (RAS) preconditioner by considering the prolongation $R_{q,0}^T$ instead of $R_{q,\Delta}^T$, i.e.

$$B_{RAS} = \sum_{q=1}^{P} B_{RAS,q}, \ \ B_{RAS,q} = R_{q,0}^T \hat{C}_q^{-1} R_{q,\Delta}.$$

Note that B_{RAS} is a nonsymmetric matrix, even if C is a symmetric one.

The third way to define the preconditioner consists in the weighted determination of the iterative values in the intersections of the subdomains. For example, if the set of node indexes $S_q^h = \bigcap_{q'} \Omega_{q'}^h$ belongs to n_q^{s+1} grid subdomains $\Omega_{q'}^h$, and we have n_q^{s+1} different values of u_l^{s+1} for $l \in S_q^h$, then it is natural to compute the real next iterative value of the subvector u_q^{n+1} by means of the least-squares condition for the corresponding residual subvector.

Another type of preconditioning matrix which is used for DDM iterations in Krylov subspaces is responsible for the coarse grid correction or aggregation approach, which is based on a low-rank approximation of the original matrix C. We define a coarse grid, or macrogrid, Ω_c and the corresponding coarse space

with $N_c \ll N$ degrees of freedom, as well as some basic functions $w^k \in \mathcal{R}^N$, $k = 1, \ldots, N_c$. We suppose that the rectangular matrix $W = (w_1, \ldots, w_{N_c}) \in \mathcal{R}^{N,N_c}$ has full rank. Then we define the coarse grid preconditioner B_c as

$$B_c^{-1} = W\hat{C}^{-1}W^T, \;\; \hat{C} = W^T C W \in \mathcal{R}^{N_c,N_c},$$

where the small matrix $\hat{C}$ is a low-rank approximation of C; W is called the restriction matrix, and the transposed matrix W^T is the prolongation matrix.

Let us consider now the construction of the preconditioned iterative processes in Krylov subspaces. We offer a general description of the multi-preconditioned semi-conjugate residual (MPSCR) iterative method [12]. Let $r^0 = f^0 - Cu^0$ be the initial residual of algebraic system (17), and let $B_0^{(1)}, \ldots, B_0^{(m_0)}$ be a set of some nonsingular easily invertible preconditioning matrices. Using them, we define a rectangular matrix composed of the initial direction vectors p_k^0, $k = 1, \ldots, m_0$:

$$P_0 = [p_1^0 \cdots p_{m_0}^0] \in \mathcal{R}^{N,m_0}, \quad p_l^0 = (B_0^{(l)})^{-1} r^0, \tag{33}$$

which are assumed to be linearly independent.

Successive approximations u^n and the corresponding residuals r^n will be determined with the help of the recursions

$$\begin{aligned} u^{n+1} &= u^n + P_n\bar{\alpha}_n = u^0 + P_0\bar{\alpha}_0 + \cdots + P_n\bar{\alpha}_n, \\ r^{n+1} &= r^n - CP_n\bar{\alpha}_n = r^0 - CP_0\bar{\alpha}_0 - \cdots - CP_n\bar{\alpha}_n. \end{aligned} \tag{34}$$

Here $\bar{\alpha}_n = (\alpha_n^1, \ldots, \alpha_n^{m_n})^T$ are m_n-dimensional vectors. The direction vectors p_l^n, $l = 1, \ldots, m_n$, which form the columns of the rectangular matrices $P_n = [P_1^n \cdots P_{m_n}^n] \in \mathcal{R}^{N,m_n}$, are defined as orthogonal vectors in the sense of satisfying the relations

$$P_n^T C^T C P_k = D_{n,k} = 0 \quad \text{for } k \neq n, \tag{35}$$

where $D_{n,n} = \text{diag}\{\rho_{n,l}\}$ is a symmetric positive definite matrix since the matrices P_k have full rank, as is supposed.

Orthogonality properties (35) provide the minimization of the residual norm $\|r^{n+1}\|_2$ in the Krylov block subspace of dimension M_n:

$$K_{M_n} = \text{Span}\{P_0, \ldots, C^{n-1}P_{n-1}\}, \quad M_n = \sum_{k=0}^{n-1} m_k \tag{36}$$

provided that we define the coefficient vectors $\bar{\alpha}_n$ and the matrices P_n by the formulas

$$\bar{\alpha}_n = \{\alpha_{n,l}\} = (D_{n,n}^{-1})^{-1} P_n^T C^T r^0, \tag{37}$$

$$P_{n+1} = Q_{n+1} - \sum_{k=0}^{n} P_k \bar{\beta}_{k,n}, \tag{38}$$

where the auxiliary matrices

$$Q_{n+1} = [q_1^{n+1} \cdots q_{m_n}^{n+1}], \quad q_l^{n+1} = (B_{n+1}^{(l)})^{-1} r^{n+1}, \quad l = 1, \ldots, m_n, \tag{39}$$

have been introduced; $B_{n+1}^{(l)}$ are some nonsingular easily invertible preconditioning matrices, and $\bar{\beta}_{k,n}$ are coefficient vectors that are determined, after substitution of (38) into orthogonality conditions (35), by the formula

$$\bar{\beta}_{k,n} = D_{k,k}^{-1} P_k^T C^T C Q_{n+1}. \tag{40}$$

Let us remark that a successful acceleration of various Krylov algorithms can be attained by least-squares approaches [13].

4 Parallel Implementation of the Method

The parallel implementation of the numerical approaches we have considered consists, in general, of the following main stages:

(a) at each time step the grid constructing and or reconstructing the mesh at each time step if it is necessary, i.e. if the solution changes dresfiarlly in time;
(b) computing the coefficients of a discrete algebraic system, and recomputing these coefficients if the input data of the original problem depend on time;
(c) at each time step, implementing nonlinear iterations if the coefficients of the original IBVP depend on the unknown solution;
(d) solving SLAEs by means of domain decomposition methods in Krylov subspaces;
(e) postprocessing and visualization of the numerical results obtained;
(f) solving the inverse or the optimal IBVP which includes constraint minimization of the objective parameterized functional based on the optimization methods and on solution of a set of direct problems, presented by the above stages;
(g) control of the general computational process and decision-making in the results of mathematical modeling.

The "d" stage is the most expensive in terms of the required computational resources, and it is also the most investigated in the sense of achieving scalable parallelism. The main numerical and technological tools here are based on both domain decomposition methods and hybrid programming: MPI (Message Passing Interface system), open-MP type multi-thread computing, vectorization of operations and use of special computational units, for instance, GPGPU (see [14] and references therein). The DDMs represent two-level iterative processes in the Krylov subspaces. The upper level includes the distributed version of the MPSCR method (33)–(40), for example. In the case of a symmetric matrix C, this algorithm becomes simpler and transforms into a multi-preconditioned conjugate residual (MPCR) method with short recursions. Here matrix-vector operations are parallelized easily by means of efficient functions from the SPARSE

BLAS library. To minimize inter-processor communication time, a special array buffering is implemented. The main speedup is attained by synchronously solving the auxiliary algebraic subsystems for subdomains on the corresponding processors. It is important that SLAEs can have diverse matrix structures and be solved by various direct or iterative algorithms. In a sense, we have here a heterogeneous block iterative process, and minimizing the general run-time is not simply in such cases. In this situation, the balancing domain decomposition problem is a nonstandard task that should be solved in terms of general computer resource consuming minimization.

The scalable parallelization of the other computational stages (a–c) should also be based, naturally, on the domain decomposition principle. Within the conception of the basic system of modeling (BSM; see [15]), each stage would be implemented by the corresponding BSM kernel subsystem which is interacted by means of distributed data structures.

References

1. Il'in, V.P.: Mathematical Modeling, Part I: Continuous and Discrete Models. SBRAS Publ., Novosibirsk (2017). (in Russian)
2. Il'in, V.P.: Finite Element Methods and Technologies. ICM&MG SBRAS, Novosibirsk (2007). (in Russian)
3. Il'in, V.P.: Methods of Solving the Ordinary Differential Equations. NSU Publ., Novosibirsk (2017). (in Russian)
4. Il'in, V.P.: Problems of parallel solution of large systems of linear algebraic equations. J. of Math. Sci. **216**, 795–804 (2016). https://doi.org/10.1007/s10958-016-2945-4
5. Saad, Y.: Iterative Methods for Sparse Linear Systems. PWS Publ., New York (2002). https://doi.org/10.1137/1.9780898718003
6. Il'in, V.P.: Finite Difference and Finite Volume Methods for Elliptic Equations. ICM&MG SBRAS Publisher, Novosibirsk (2001). (in Russian)
7. Dolean, V., Jolivet, P., Nataf, F.: An Introduction to Domain Decomposition Methods: Algorithms, Theory and Parallel Implementaion. SIAM, Philadelphia (2015). https://doi.org/10.1137/1.9781611974065
8. Gander, M.J., Guttel, S.: ParaExp: a parallel integrator for linear initial value problems. SIAM J. Sci. Comput. **35**, 123–142 (2013). https://doi.org/10.1137/110856137
9. Karra, S.: A hybrid Pade ADI scheme of high-order for convection-diffusion problem. Int. J. Numer. Methods Fluids **64**, 532–548 (2010). https://doi.org/10.1002/fld2160
10. Lawson, G.L., Hanson, R.J.: Solving Least Squares Problems. Prentice-Hall, Inc., Upper Saddle River (1974). https://doi.org/10.1137/1.9781611971217
11. Saad, Y., Yeung, M., Erhel, J., Guyomarc'h, F.: A deflated version of the Conjugate Gradient Algorithm. SIAM J. Sci. Comput. **21**, 1909–1926 (2000). https://doi.org/10.1137/s1064829598339761
12. Il'in, V.P.: Multi-preconditioned domain decomposition methods in the Krylov subspaces. In: Dimov, I., Faragó, I., Vulkov, L. (eds.) NAA 2016. LNCS, vol. 10187, pp. 95–106. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57099-0_9

13. Il'in, V.P.: Least squares methods in Krylov subspaces. J. Math. Sci. **224**, 900–910 (2017). https://doi.org/10.1007/s10958-017-3460-y
14. Il'in, V.: On the parallel strategies in mathematical modeling. In: Sokolinsky, L., Zymbler, M. (eds.) PCT 2017. CCIS, vol. 753, pp. 73–85. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67035-5_6
15. Gladkikh, V.S., Il'in, V.P.: Basic System of Modeling (BSM): conception, architecture and methodology. In: Conference Proceedings of "Modern Problems of Mathematical Modeling, Image Processing and Parallel Computing", pp. 151–158. RTU Publ., Rostov (2017). https://doi.org/10.23947/2587-8999-2017-2-194-200. (in Russian)