Bull. Nov. Comp. Center, Comp. Science, 9 (1998), 31–43
© 1998 NCC Publisher

3D mixed boundary value problems: numerical algorithms, data structures and technologies of implementation*

Yana L. Gurieva and Valery P. Il'in

Technological aspects of the numerical solution to multi-dimensional boundary value problems (BVP) for partial differential equations (PDEs) using the finite difference, the finite element or the finite volume methods are described. Among them, the mathematical formulation of the problem, definitions of objects under consideration, algorithms of approximation with different accuracy orders, description of the code FVSDE3 and estimates of parallelization speedup.

Introduction

Numerical solution to multi-dimensional boundary value problems (BVP) for partial differential equations (PDEs) using the finite difference, the finite element or the finite volume methods (FDM, FEM or FVM) includes the following computational stages:

- input and preliminary processing of the data describing the original mathematical formulation of the BVP which includes an information about the geometry of the computational domain, the boundary conditions, the types of PDE's and values of their coefficients in different subdomains and possible instructions for the algorithm applied and for visualization of the numerical results;
- discretisation of the BVP that means an automatic generation and constructing the grid subdomains with their cells, edges and nodes providing connection between the grid objects or/and the functional data;
- approximation of the original BVP which can be efficiently implemented on the basis of element-by-element data processing, computing the local matrices and assembling the global matrix of an algebraic system of the grid equations;
- numerical solution to the algebraic system, for example, by efficient iterative incomplete factorization methods with the conjugate gradient acceleration providing a fast convergence rate and usage of reasonable CPU resources;
- analysis of the computational results with visualization of numerical data and possible solution to the optimal control problem by the run of various variants of BVPs and the search for the optimal parameters in the formulation in the sense of minimization of the given functionals under some restrictions.

*Supported by the Russian Foundation for Basic Research under Grant 96-01-01770 and INTAS-RFBR under Grant 95-98.

Undoubtedly, a subject of considerable interest is the efficiency of parallel computing on the multi-processor systems with distributed or shared memory.

In the present paper, the questions of mathematical and software technologies are considered on the example of the 3D mixed BVP for the diffusion equation

$$-\operatorname{div}\lambda\operatorname{grad}u+\mu u=f, \tag{1}$$

in the bounded computational domain $\Omega=\bigcup_k\Omega_k$ with the piecewise smooth functions $\lambda>0,\ \mu\geq 0,\ f$ in different subdomains Ω_k. On different parts of the domain *outer boundary* $\Gamma^\circ=\Gamma^D\bigcup\Gamma^N,\ \Gamma^D=\bigcup_p\Gamma_p^D,\ \Gamma^N=\bigcup_q\Gamma_q^N$ the Dirichlet or the Neumann (Newton) conditions are satisfied

$$u\big|_{\Gamma_p^D}=g_p,\qquad æ_q u+\frac{\partial u}{\partial n}\Big|_{\Gamma_q^N}=\gamma_q, \tag{2}$$

here $\frac{\partial}{\partial n}$ means the external normal derivative and $g_p,\ æ_q,\ \gamma_q$ are given constants or functions.

The second equation in (2) is called the *Newton condition* in the general case and the *Neumann condition* if $æ_q=0$. On the surfaces Γ^i of possible discontinuity of λ (on common sides of different Ω_k, or the "inner boundaries") the conjugate conditions hold:

$$u\big|_{\Gamma_+^i}=u\big|_{\Gamma_-^i},\qquad \lambda_+\frac{\partial u}{\partial\bar n}\Big|_{\Gamma_+^i}=\lambda_-\frac{\partial u}{\partial\bar n}\Big|_{\Gamma_-^i}, \tag{3}$$

where the signs "+", "−" mean one-sided values of the function and its normal derivative on different sides of Γ^i.

We suppose that the subdomains Ω_k are parallelepipeds whose faces (which form the external and the internal boundaries) are parallel to the coordinate planes.

The approximations used is a set of various finite volume approximations of order $O(h)$, $O(h^2)$, $O(h^3)$ at 7-, 19- or 27-point stencils on the regular non-uniform grids, see [1].

The efficient explicit iterative incomplete factorization procedure EXIFA is used to obtain a numerical solution of large algebraic systems of equations [2].

The above mentioned algorithms are implemented in the program FVSDE3 which is a generalization of the similar 2D code FVSDE2 [3].

The above-mentioned stages of the solution process and the problems arising during their implementation were considered by many authors mostly for the FEM technologies. The construction of the FEM approximations and their theoretical and technological grounds can be found in the papers and books by many authors (see [4–6], for example). The problems of grid generation and the corresponding algorithms are presented in the recent review [7]. Some algorithmic problems of the FEM and the FVM combination were considered in [8].

In Section 1, we describe the basic definitions of the mathematical formulation and the approaches used in the following considerations. Section 2 is devoted to the description of algorithms and data structures. In Section 3, special software technologies in the code FVSDE3 are considered. Some aspects of parallel implementation of the above-mentioned algorithms are reviewed in Section 4.

1. Mathematical formulation and approaches

In this section, the basic functional, geometric and computational objects will be defined. They form the basis for specifications of the algorithms and the data representation used in designing the code to solve the problem. Similar considerations of some objects and data structures were presented in [9] without peculiarities of their implementation.

Mathematical formulation of any BVP is defined by a set of differential equations, computational domain, boundary conditions and specifications of the algorithms applied.

Computational domain Ω (CD) is an open bounded set in R^n ($n = 3$ in our case). Its closure is $\overline{\Omega} = \Omega \cup \Gamma^o$, where $\Gamma^o \in R^{n-1}$ is the outer boundary of Ω. In the general case, Γ^o is multi-connected.

Subdomain Ω_k is an open bounded subset of Ω. Subdomains do not intersect, i.e. $\Omega_k \cap \Omega_l = \emptyset$. From the practical point of view, a subdomain differs from other subdomains in its specific geometry and in values of the coefficients of the PDE. Any domain can be presented as a union of its subdomains: $\Omega = \bigcup_{k=1}^{N_s} \Omega_k$ where N_s is the number of subdomains. In our consideration, each subdomain is a parallelepiped $\Omega_k = \left\{x_1^{(k)} \le x \le x_2^{(k)},\ y_1^{(k)} \le y \le y_2^{(k)},\ z_1^{(k)} \le z \le z_2^{(k)}\right\}$ which has its own coefficients for the differential equation and whose faces are parallel to the coordinate planes.

For each subdomain Ω_k, there is defined its boundary Γ_k and its closure $\overline{\Omega}_k = \Omega_k \cup \Gamma_k$. The subdomain boundary Γ_k can be considered consisting of two parts: *outer boundary part* $\Gamma_k^o = \Gamma_k \cap \Gamma$ and *inner boundary part* $\Gamma_k^i = \Gamma_k \setminus \Gamma_k^o$. As well as the boundary Γ_k of any subdomain, the domain boundary Γ consists of two parts – *inner* and *outer* boundaries: $\Gamma = \Gamma^i \cup \Gamma^o$. *Inner boundary* is the part of Γ formed by the mutual boundary parts of pairs of subdomains: $\Gamma^i = \cup\Gamma_{kl}$, $\Gamma_{kl} = \Gamma_k \cap \Gamma_l$, $k \ne l$.

From the practical point of view, it is important to distinguish the inner and the outer parts of the boundary because they concern different mathematical conditions: boundary conditions (2) and conjugate conditions (3) respectively. Strictly speaking, the latter is a direct consequence of the jump of coefficient λ in (1), and, formally, the approximations are constructed without using the conjugate conditions. According to both types of boundary conditions in the problem formulation, the Dirichlet and the Neumann (the Newton) conditions, we can assume Γ^o (and, also, every Γ_k^o) consisting of two parts $\Gamma^o = \Gamma^D \cup \Gamma^N$ which correspond to Dirichlet and Neumann (Newton) conditions respectively. As a data processing for these types is different, it is important to have a possibility to separately process the corresponding types of conditions, i.e., to traverse the corresponding boundary parts.

Differential equation (DE) of the form of (1) as a computational object is a triple of coefficients (λ, μ, f). As the domain consists of its subdomains, this triple is determined for each subdomain Ω_k: $DE_k = (\lambda_k, \mu_k, f)$, where in our case of the problem formulation λ_k is a constant, μ_k is either zero or non-negative constant or a given function and f_k is, also, either a constant or a given function. Let us note that the main value for the subdomain to distinguish it from the other subdomains is its number k, i.e., it is assumed that having this number, one can easily get the necessary values for DE_k.

Discretization of BVP in the domain Ω is based on construction of some *grid* in the domain and considered as calculation of the coordinates of the grid nodes, defining the grid edges, faces and elements, and some other grid objects. These grid objects depend on the method of approximation used and the type of the grid. Usually, two types of the grids are distinguished: irregular and regular. Their main difference is in the topology of the adjacent grid nodes. For a given grid node, a set of other nodes connected with it by the grid edges is called *grid stencil*. For the regular grid, the grid stencils of all the grid nodes $P \in \Omega$ are the same. We will consider regular rectangular grids. *Rectangular grid* G is a tensor product of the three non-uniform one-dimensional grids G^x, G^y, G^z (x-, y-, and z-grid):

$$G^x : x_{i+1} = x_i + h_i^x, \qquad G^y : y_{j+1} = y_j + h_j^y, \qquad G^z : z_{k+1} = z_k + h_k^z,$$
$$i = 0, \dots, L, \qquad j = 0, \dots, M, \qquad k = 0, \dots, K.$$

The constructed rectangular grid is *consistent with the geometry* of the domain in the sense that, firstly, each side of any subdomain is placed on the coordinate planes defined by some of the coordinates of G^x, G^y or G^z. We suppose that the computational domain is embedded in the parallelepiped $\tilde{\Omega} = \{x_0 \le x \le x_L, y_0 \le y \le y_M, z_0 \le z \le z_N\}$ and that the domain boundaries (outer and inner) cross the grid lines in the nodes only. For example, if a subdomain has a side $x = x_i$, $y_{j_1} \le y \le y_{j_2}$, $z_{k_1} \le z \le z_{k_2}$, the grid should have coordinate x_i in its x-grid, and the coordinates y_{j_1}, y_{j_2} and z_{k_1}, z_{k_2} in its y- and z-grids, respectively. The boundary vertices and edges should be grid nodes and grid edges. For a convenient description of the computational domain, we also need to define *external subdomains* $\Omega^e = \tilde{\Omega} \setminus \Omega = \cup_k \Omega_k^e$ which can be represented formally as subdomains with zero diffusion coefficient λ of equation (1) (in fact, no equation will be solved in Ω_k^e).

To approximate equation (1), the finite volume method is applied (for details see [1]). It is based on an approximation of the conservative law relation

$$-w \int\limits_{S_{i,j,k}} \lambda \frac{\partial u}{\partial n} ds - (1-w) \int\limits_{\bar{S}_{i,j,k}} \lambda \frac{\partial u}{\partial n} ds = w \int\limits_{V_{i,j,k}} (f - \mu u) dv + (1-w) \int\limits_{\bar{V}_{i,j,k}} (f - \mu u) dv, \quad (4)$$

which is equivalent to the original problem (1)–(3) and is combined over small and big elementary volumes $V_{i,j,k}$, $\bar{V}_{i,j,k}$ with surfaces $S_{i,j,k}$, $\bar{S}_{i,j,k}$:

$$\begin{aligned} V_{i,j,k} &= \Big\{ \frac{x_i + x_{i-1}}{2} \le x \le \frac{x_i + x_{i+1}}{2}, \ \frac{y_j + y_{j-1}}{2} \le y \le \frac{y_j + y_{j+1}}{2}, \\ &\qquad \frac{z_k + z_{k-1}}{2} \le z \le \frac{z_k + z_{k+1}}{2} \Big\}, \\ \bar{V}_{i,j,k} &= \{ x_{i-1} \le x \le x_{i+1}, \ y_{j-1} \le y \le y_{j+1}, \ z_{k-1} \le z \le z_{k+1} \}, \end{aligned}$$

with a weight parameter w. The resulting grid equations are, in general, of the 27-point type:

$$p^0_{i,j,k} u_{i,j,k} - p^3_{i,j,k} u_{i+1,j,k} - p^4_{i,j,k} u_{i,j+1,k} - p^6_{i,j,k} u_{i,j,k+1} -$$
$$p^3_{i-1,j,k} u_{i-1,j,k} - p^4_{i,j-1,k} u_{i,j-1,k} - p^6_{i,j,k-1} u_{i,j,k-1} -$$

$$
\begin{aligned}
&p^7_{i,j,k}u_{i-1,j+1,k} - p^8_{i,j,k}u_{i+1,j+1,k} - p^9_{i-1,j,k-1}u_{i-1,j,k-1} - \\
&p^{10}_{i,j,k}u_{i,j+1,k+1} - p^{11}_{i,j,k}u_{i-1,j,k+1} - p^{12}_{i,j,k}u_{i,j-1,k+1} - \\
&p^7_{i+1,j-1,k}u_{i+1,j-1,k} - p^8_{i-1,j-1,k}u_{i-1,j-1,k} - p^9_{i-1,j-1,k-1}u_{i-1,j,k-1} - \\
&p^{10}_{i,j-1,k-1}u_{i,j-1,k-1} - p^{11}_{i+1,j,k-1}u_{i+1,j,k-1} - p^{13}_{i,j,k}u_{i-1,j-1,k+1} - \\
&p^{14}_{i,j,k}u_{i+1,j+1,k+1} - p^{15}_{i,j,k}u_{i-1,j+1,k+1} - p^{16}_{i,j,k}u_{i+1,j-1,k+1} - \\
&p^{13}_{i+1,j+1,k-1}u_{i+1,j+1,k-1} - p^{14}_{i-1,j-1,k-1}u_{i-1,j-1,k-1} - \\
&p^{15}_{i+1,j-1,k-1}u_{i+1,j-1,k-1} - p^{16}_{i-1,j+1,k-1}u_{i-1,j+1,k-1} = f_{ijk}, \qquad (5)
\end{aligned}
$$

where the symmetry of the coefficients is taken into account, and coefficients $p^n_{i,j,k}$ are computed via the values of mesh steps and the parameters λ, μ, æ of the original BVP, see [1] for details.

For the weight parameter $w = 32/31$, this equation turns into the 19-point equation with zero coefficients $p^{13}, \dots, p^{16}$ and for $w = 16/15$ it turns into the 7-point scheme with zero coefficients $p^7, \dots, p^{16}$.

The final system of linear algebraic equations has the following form:

$$
Au = f, \qquad (6)
$$

in which the equation for the grid node with the grid indexes (i, j, k) is given by (5). The approximations of relation (4) used to obtain system (6) bring about the symmetric matrix A.

The order of approximation accuracy is different for different values w and grid types:

- $O(h)$ for $w = 16/15$ and a general non-uniform grid,
- $O(h^2)$ for $w = 16/17$ and a general non-uniform grid or for any $0 \le w \le 1$ and a piecewise uniform grid (the finite number of grid zones with constant grid steps in each one),
- $O(h^4)$ for $w = 32/31$, the uniform grid, the Dirichlet boundary conditions and the constant coefficients λ, μ.

The matrix A of the final system (6) is calculated via the element-by-element approach which is well-known in the FEM. It means that the main elementary calculation object is not a grid node but a grid cell. The calculations made in each cell use the "local" information, i. e., the information about the current cell only: its grid steps, grid indexes of the cell vertexes and the value λ in the cell. The calculations in one grid cell result in the entries of the local matrix. Matrix A of the final system (or the global matrix) is a sum of extended local matrices of all the grid cells.

To apply the element-by element technique for the computation of *local balance* and assembling *global balance* matrix of the final system of grid equations, we introduce some grid objects. *Computational cell* (or a grid cell) is a volume restricted by six grid planes (or six coordinates): $\{x_i \le x \le x_{i+1},\ y_j \le y \le y_{j+1},\ z_k \le z \le z_{k+1}\}$ and we will refer to it as (i, j, k)-cell later. The grid nodes are the vertices of the grid cells. Each grid cell belongs only to one subdomain and so has its own value of parameters of the PDE, i. e. the cells differ in the numbers of the subdomains to which they belong. There is no cell containing the inner boundary – the plane dividing the subdomains with different λ. The computational domain is a union of all the grid cells. And, finally, the faces of a grid cell (*grid faces*) are eihter parts of the domain boundary Γ or the inner faces formed by the grid. In the first case, the faces have the same boundary condition as the part of Γ to which they belong.

2. Algorithms and data structures

The first step of solution of the original problem, which requires some calculations, is the construction of a grid. The coordinates of a non-uniform parallelepipedoidal grid are calculated as follows. Along each coordinate, the grid consists of the finite number of grid zones in which the grid steps are defined as constants (u-zone) or by a geometric progression (g-zone) or by an arithmetic progression (a-zone, or a-type zone).

Let the values n_x, n_y, z_z mean the numbers of zones in x-, y- and z-directions and

$$X_p,\ p=0,1,\dots,n_x, \qquad Y_q,\ q=0,1,\dots,n_y, \qquad Z_r,\ r=0,1,\dots,n_z,$$

are the boundaries of the zones with the properties

$$\begin{aligned} &X_0 = x_0, && X_p > X_{p-1}, && X_{n_x} = x_{L+1},\\ &Y_0 = y_0, && Y_q > Y_{q-1}, && Y_{n_y} = y_{M+1},\\ &Z_0 = z_0, && Z_r > Z_{r-1}, && Z_{n_z} = z_{K+1}. \end{aligned}$$

The numbers of grid steps l_p, m_q, and n_r in each x-, y-, and z-zones are defined so that

$$L+1=\sum_{p=1}^{n_x} l_p, \qquad M+1=\sum_{q=1}^{n_y} m_q, \qquad K+1=\sum_{r=1}^{n_z} n_r.$$

The constant grid steps in the u-zones are defined as

$$\begin{aligned} h_i^x &= h_x^{(p)} = \frac{X_p - X_{p-1}}{l_p}, && i = L_{p-1},\dots,L_p-1, && L_p=\sum_{s=1}^{p} l_s,\\ h_j^y &= h_y^{(q)} = \frac{Y_q - Y_{q-1}}{m_q}, && j = M_{q-1},\dots,M_q-1, && M_q=\sum_{s=1}^{q} m_s,\\ h_k^z &= h_z^{(r)} = \frac{Z_r - Z_{r-1}}{n_r}, && k = N_{r-1},\dots,N_r-1, && N_r=\sum_{s=1}^{r} n_s. \end{aligned}$$

The grid steps in the g-type zones are sought for by the formulas

$$\begin{aligned} h_i^x &= h_x^{(p)}\alpha_p^{i-L_{p-1}}, && i = L_{p-1},\dots,L_p-1,\\ h_j^y &= h_y^{(q)}\beta_q^{j-M_{q-1}}, && j = M_{q-1},\dots,M_q-1,\\ h_k^z &= h_z^{(r)}\gamma_r^{k-N_{r-1}}, && k = N_{r-1},\dots,N_r-1. \end{aligned}$$

Here α_p, β_q, γ_r are denominators of the geometrical progressions, and $h_x^{(p)}$, $h_y^{(q)}$, $h_z^{(r)}$ are the initial grid steps in respective zones defined by the relations

$$h_x^{(p)} = (X_p - X_{p-1})\frac{\alpha_p - 1}{\alpha_p^{l_p} - 1}, \qquad \alpha_p \neq 1, \quad 1 \leq p \leq n_x,$$

$$h_y^{(q)} = (Y_p - Y_{p-1})\frac{\beta_q - 1}{\beta_q^{m_q} - 1}, \qquad \beta_q \neq 1, \quad 1 \leq q \leq n_y,$$

$$h_z^{(r)} = (Z_r - Z_{r-1})\frac{\gamma_r - 1}{\gamma_r^{n_r} - 1}, \qquad \gamma_r \neq 1, \quad 1 \leq r \leq n_z.$$

The grid steps in *a*-zones are defined as

$$h_i^{(x)} = h_x^{(p)} + (i - L_{p-1})d_x^{(p)}, \qquad i = L_{p-1}, \dots, L_p - 1,$$

$$h_j^{(y)} = h_y^{(q)} + (j - M_{q-1})d_y^{(q)}, \qquad j = M_{q-1}, \dots, M_q - 1,$$

$$h_k^{(z)} = h_z^{(r)} + (k - N_{r-1})d_z^{(r)}, \qquad k = N_{r-1}, \dots, N_r - 1,$$

where $d_x^{(p)}$, $d_y^{(q)}$, $d_z^{(r)}$ are differences of the arithmetical progressions,

$$d_x^{(p)} = \left(\frac{X_p - X_{p-1}}{l_p} - h_x^{(p)}\right)\frac{2}{l_p - 1}, \qquad 1 \leq p \leq n_x,$$

$$d_y^{(q)} = \left(\frac{Y_q - Y_{q-1}}{m_q} - h_y^{(q)}\right)\frac{2}{m_q - 1}, \qquad 1 \leq q \leq n_y,$$

$$d_z^{(r)} = \left(\frac{Z_r - Z_{r-1}}{n_r} - h_z^{(r)}\right)\frac{2}{n_r - 1}, \qquad 1 \leq r \leq n_z,$$

satisfying the positiveness conditions of the initial grid steps

$$X_p - X_{p-1} > h_x^{(p)} = \frac{X_p - X_{p-1}}{l_p} - d_x^{(p)}\frac{l_p - 1}{2} > 0,$$

$$Y_q - Y_{q-1} > h_y^{(q)} = \frac{Y_q - Y_{q-1}}{m_q} - d_y^{(q)}\frac{m_q - 1}{2} > 0,$$

$$Z_r - Z_{r-1} > h_z^{(r)} = \frac{Z_r - Z_{r-1}}{n_r} - d_z^{(r)}\frac{n_r - 1}{2} > 0.$$

In papers [9–10], the some definitions of grid data structures and algorithms of their building were described for the considered type of geometry and grid. We will describe them here in details because these structures as well as the algorithms of grid building, local and global matrices calculation and solution of the linear system form the basis for the program code FVSDE3 which will be performed in the next section.

The grid data structure for the problem under consideration is presented in the form of several arrays containing a cell-oriented information, i. e. the value of λ in the cell (or some identifier, e.g. number of the subdomain which contains this grid cell) and some information on grid faces (e. g., boundary conditions if any).

To compute a local balance matrix under the element-by-element approach, one should have the values of the functions λ, μ, f in every computational cell, therefore a three-dimensional $(L+1) \times (M+1) \times (K+1)$ array *med* is introduced, such that its element *med*(i,j,k) equals the subdomain number in the grid (i,j,k)-cell (from the algorithmic point of view, we identify subdomains by their numbers). The array of the subdomain numbers in the cells is built by the subdomain traversing, taking into account their mutual topology, see [11] for details.

To calculate the contributions into the local matrices from the Neumann or the Newton boundary conditions, it is required to determine on which face of a grid cell this condition is given. In the initial data, the boundary condition number (we identify the boundary conditions by their numbers) should be given on every face of each subdomain. The three-dimensional arrays *jface* of $(L+2)\times(M+1)\times(K+1)$ dimension, *jface* of $(L+1) \times (M+2) \times (K+1)$ dimension and *kface* of $(L+1) \times (M+1)\times(K+2)$ dimension are introduced. These arrays contain through numbers of faces of the subdomains on the perpendicular to the axes X, Y and Z faces of the grid cells, respectively (dimensions of the arrays coincide with the number of grid faces in the corresponding direction).

The solution of algebraic system (6) is computed on the basis of the iterative incomplete factorization methods accelerated by means of the conjugate gradient algorithm. Namely, Eisenstadt modification of the generalized symmetric successive over relaxation (SSOR) algorithm with the compensation principle (the row sum criteria, see [2]) is used. This method includes the iterative compensation parameter $0 \le \theta \le 1$ and the relaxation parameter $0 \le \omega \le 1$ to minimize the number of iterations. For Stiltjes matrices, the values $\theta = \omega = 1$ are suitable. The iterative process stops when one of the following conditions holds:

$$||r^0|| \le \frac{\varepsilon}{LMK}, \qquad \frac{||r^n||}{||r^0||} \le \varepsilon, \qquad n \le n_{\max}. \tag{7}$$

Here

$$||r^n|| = \Big(\sum_i \sum_j \sum_k (r^n_{i,j,k})^2\Big)^{1/2}$$

is the Euclidian norm of the residual vector $r^n = f - Au^n$, n is the number of iterations, u^0 and r^0 are the initial solution and residual, ε and $n_{\max}$ are prescribed small and large values. To solve system (6), this approach is implemented in the above-mentioned subroutine EXIFA.

The matrix A after the approximation stage is presented in the form of arrays of the coefficients $p^0, \dots, p^{16}$ from equation (5). They are then transformed into the form of the sparse row-wise format [12] because the solver needs this matrix representation as it is universal and independent of the portrait of the matrix A.

3. FVSDE3 code

The program FVSDE3 is made as one subroutine which solves the mixed boundary value problem (1)–(3) for the diffusion equation with piece-wise smooth coefficients in a composed of parallelepipeds computational domain using fast incomplete factorization solvers on the basis of 7-point, 19-point or 27-point finite volume schemes, on a non-uniform or a uniform grid. The programming language is Fortran-77. The call of the subroutine is the following:

```
call FVSDE3(fname,u,x,y,z,nmax,l,m,n,rh,dfun,userk,userg,cfun,ido)
```

The arguments here mean the following:

`fname`	is a name of the data input file (input);
`u`	is an array of size lmn containing the solution values at the grid points (output);
`x`	is an array of size l containing the x-coordinates of the grid nodes (output);
`y`	is an array of size m containing the y-coordinates of the grid nodes (output);
`z`	is an array of size n containing the z-coordinates of the grid nodes (output);
`nmax`	is a maximum allowable number of iterations (input) and the real number of iterations (output);
`l,m,n`	are numbers of grid points in x-, y- and z-directions (output);
`ido`	is a flag indicating the current state of computations (input/output); `ido` $\neq$ 0 means the abnormal return from the subroutine;
`rh`	is a user-supplied function for the right-hand side f of (1). Its form is `rh(isr,x,y,z)`, where `isr` is a number of the subdomain which contains the point (x, y, z) (input), `x,y,z` are a point coordinate values (input), `rh` is the value of the right-hand side at the point (x, y, z) (output);
`dfun`	is an user-supplied function for the right side function g in the Dirichlet boundary conditions (2). Its form is `dfun(nbcond,x,y,z)`, where `nbcond` is a number of boundary condition (input), `x,y,z` are a point coordinate values (input), `dfun` is the value of the function at the point (x, y, z) (output);
`cfun`	is a user-supplied function for the Helmholtz coefficient μ in (1). Its form is `cfun(isr,x,y,z)`, where the input parameters are the same as in `rh` function. It returns the value of the function μ at the point (x, y, z);
`userk`	is a user-supplied function for the coefficient æ in the Newton boundary condition (2). Its form is `userk(nbcond,x,y,z)`, where the input parameters are the same as in `dfun` function. It returns the value of coefficient æ at the point with the coordinates (x, y, z);
`userg`	is a user-supplied function for the coefficient γ in the Newton boundary condition (2). Its form is `userg(nbcond,x,y,z)` with the same parameters as in `userk` function.

The subroutine FVSDE3 consists of several successive standard stages, i. e., input of the initial data describing the problem, grid construction, building the grid data structures, approximation and solution of the linear system.

The program from which the FVSDE3 is called should have the description of four arrays: x of length $L + 2$, y of length $M + 2$, z of length $K + 2$, and u of the length $N = (L+2) \times (M+2) \times (K+2)$. Also, five user functions should be defined.

Memory requirements for the subroutine are determined by the approximation stencil, which is 27-point in the general case (it means that for each grid point there exist 27 values of coefficients of equation (5)), by the memory requirements of the solving procedure EXIFA, and by the double precision representation of real values. Taking into account the symmetry of the matrix (only 14 values of coefficients in the left side of equation (5) are enough), the total number of arrays of length N for the program is 16 (14 for the coefficients and 2 for the right-hand side f and for the vector of unknowns u). Because of the memory requirements of the solver EXIFA, 22N double precision real values are needed in total.

4. The parallelization speedup estimates

There are many publications on parallelization, mainly, on numerical solution of the algebraic systems of equations, for example, see [2] and cited references. We briefly consider the parallel general data processing for numerical solution to a BVP.

Formally, the numerical solution to a BVP is a set of data computed using the following data structures (DS):

Statement (input) DS which consists of *geometric, functional* and *algorithmic data structures* and includes full information about the computational domain, the original PDE, the boundary conditions and some numerical methods; this DS contains integer and real values about geometric objects, functional representation of coefficients and algorithmic parameters; the total volume of such an information is not large as compared to the rest DS and depends on complexity of the BVP, but not on the number of mesh points; for the practical problems it can be estimated by $N_s \leq 1000$.

Grid data structure consists of information about all grid objects – nodes, edges, faces and cells as well as their relations with geometric, functional and algorithmic objects from the statement DS; the purpose of the grid DS is to provide an efficient implementation of constructing the grid system of equations which approximate the BVP; this DS consists of integer values which are either numbers or references to other objects and data; the total volume of this DS is estimated for large L, M, K by $N_g \approx 4LMK$.

Algebraic DS means real values of the matrix entries $pk_{i,j,k}$ and right hand side $f_{i,j,k}$ from the grid system of equations and, also, the integer data to indicate to numbers of the neighboring stencil points for each grid point (in other words, it provides numbers of nonzero matrix entries in the sparse row-wise format). The total volume of the algebraic DS for L, M, $K \gg 1$ equals $N_a = 7LMK + N_z$ where $N_z \leq m_{st}LMK/2$ is the number of the off-diagonal matrix entries and $m_{st} = 6, 18$ or 26 is the number of the neighboring points in the grid stencil for each node depending on the type of approximation. This DS consists of real values for the coefficients of the final system and some integer data to use the sparse row-wise format.

Resulting data structure contains one real array of numerical solution at the grid nodes and various postprocessing information including the data for tables, graphics, isolines, etc.

The numerical process includes three main stages (we do not consider preprocessing and postprocessing stages which provide a user interface and are mainly connected not with calculations but with graphics):

(a) formation of the grid DS from the statement DS (discretisation);

(b) transformation of the grid DS to the algebraic DS (approximation);

(c) numerical solution to the algebraic system (the iterative incomplete factorization method implemented in the subroutine EXIFA).

The main approach to solution of the multidimensional BVP is a domain decomposition (DD). In the case of a parallelization on p processors, it can be formulated as follows.

We define the grid subdomains Ω_s^h, $s = 1, \ldots, p$, with a possible overlapping, which are not connected with the computational subdomains Ω_k and divide the grid and the algebraic data structures into substructures which are distributed on the processors. The ideal domain decomposition, in the sense of the best utilization of SIMD processors, is the definition of the grid subdomains Ω_s^h with the equal number of nodes (or cells).

The implementation of the stage (a) on p processors can be readily arranged as follows. The copies of the statement DS are put on all the processors because of their relatively small volume. The purpose of data processing is to prescribe the number of its computational domain Ω_k to each grid cell (formally this number is zero if the cell is outside of the computation domain Ω), and to prescribe the number of boundary segment to each grid face. The latter number allows us to find the type of the corresponding boundary condition, and it equals to zero if the grid face is internal, i. e., it is not on the outer boundary Γ^o. On each s-th processor, computations are done in loops over the cells and the grid faces which are in the s-th grid subdomain Ω_s^h. Really, this stage is implemented without any communications and redundant arithmetic, i. e., we have an ideal parallelization.

At the stage (b) we have the following data processing:

$$\text{statement DS} + \text{grid DS} \to \text{algebraic DS}.$$

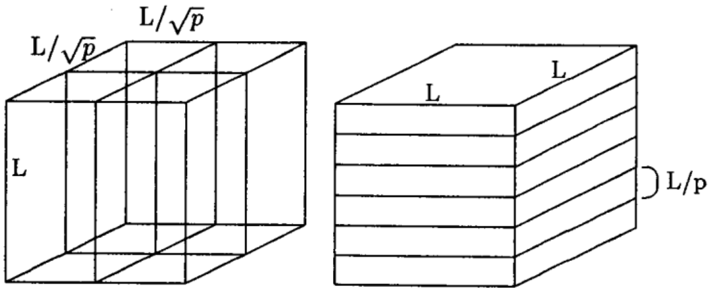
The computations for s-th grid subdomain are done in the cell-by-cell ordering on corresponding s-th processor. The resulting algebraic DS is node-oriented (the matrix entries and right-hand side for q-th equation correspond to q-th grid point) and is distributed on p processors in accordance with their belonging to the grid subdomains. The parallelization at this stage is not ideal because some communications between processors are needed when assembling the global matrix from the local ones: for the grid nodes on the common boundary of two subdomains there are local matrices computed on different processors according to their correspondence to the subdomains. The last operation to get the final global matrix is to traverse the nodes on the mutual boundaries and to sum up their additives from the local matrices computed in different subdomains. In so doing, the volume of arithmetical operations is proportional to $O(h^2)$ (as compared to $O(h^3)$ – the total number of grid nodes). If N is the total number of grid points and N/p is the number of nodes in one grid subdomain for an ideal distribution, then the CPU-times T_p equal to the values $C_a N/p$ and $C_b N/p$ for the stages (a) and (b), respectively. Here $C_a \approx 10$ and $C_b \approx 100$ are the numbers of arithmetic operations per node for the stages.

The third stage has the main computational complexity both for direct and iterative discrete methods for solution of the BVP systems of equations because the number of arithmetic operations is a nonlinear function of N. The memory requirements for the 3D BVP make the iterative algorithms the most efficient tool on multi-processor systems. Various versions of the algebraic domain decomposition methods can be implemented in our case, see [2, 13], for example, and we do not intend to consider this topic in detail.

One of the main approaches here is the double iterative process with communications between processors on the outer iteration only (the inner iterations deal with the linear subsystems in the subdomains corresponding to the processors and the outer iterations make possible to get the final solution to the whole algebraic system). In this case, the total computational complexity of numerical solution depends on the way of performing the double iterative procedure, but the speedup does not depend on the numbers of the inner and outer iterations.

It is important to note that in this DD approach, the communications and the total efficiency of parallelization are defined by the way of domain decomposition in the sense of the shape of subdomains. The corresponding estimates can be demonstrated on the computer system in the form of the square processor grid of the size of $p^{1/2} \times p^{1/2}$ with distributed memory ($p^{1/2}$ is supposed to be an integer).

Let a computational cube grid domain with the total number of nodes $LMK = L^3$ ($L = M = K$) be decomposed into p subdomains (we assume that $p \ll L^3$) without overlapping so that each subdomain is processed on its own processor, and the number of nodes in one subdomain is L^3/p. Than the number of arithmetic operations on one processor will be proportional to L^3/p. The number of communications between processors depends on the way of division. We will consider two ways of decomposition (the figure): in layers and in columns.



Two ways of decomposition

First, let the domain be divided into p layers. This way can be called the 1D-decomposition as it is made in the direction of only one coordinate. The number of nodes on the adjacent side of two layers is L^2. Than the number of communications between the neighbouring processors is proportional to the number of nodes on the two adjacent sides of one layer and in our case is $C_1 = 2L^2$.

The second way of distributing the nodes is to divide the domain into p columns. By analogy with the first case, this way of division can be called 2D-decomposition as the dividing is made along two coordinates. The number of nodes of the adjacent side of one subdomain is $L^2/\sqrt{p}$. Than the number of communications between processors is $C_2 = 4L^2/\sqrt{p}$.

So, if the relation $C_2/C_1 = 2/\sqrt{p} < 1$, i.e., $p > 4$, then it is better to divide a region into columns in the sense of decreasing the communications.

References

[1] Y.L. Gurieva and V.P. Il'in, *On second order finite-volume approximations for 3D mixed boundary value problems*, Bull. of the Novosibirsk Computing Center, Series: Numerical Analysis, Issue 7, 1996, 51–70.

[2] V.P. Il'in, *Iterative Incomplete Factorization Methods*, World Scientific Publishing Co., Singapore, 1993.

[3] Y. Gurieva and V. Il'in, *Finite volume approached for 2-D BVPs: algorithms, data structures, software and experiments*, Univ. of Nijmegen, 1997, Univ. of Nijmegen, Dep. of Math. Rep., No. 9715.

[4] G. Fix and G. Strang, *Analysis of the Finite Element Method*, Prentice-Hall, Inc., New Jersey, 1973.

[5] O. Axelsson and V.A. Barker, *Finite Element Solution of Boundary Value Problems. Theory and Computations*, Academic Press, New York, 1984.

[6] B. Szabo and I. Babuška, *Finite Element Analysis*, John Wiley & Sons, Inc., New York, 1991.

[7] L.V. Krugljakova, A.V. Neledova, B.F. Tishkin, and A.Yu. Filatov, *Unstructured adaptive grids for mathematical physics problems*, Math. Modeling, **10**, No. 3, 1998, 93–116 (in Russian).

[8] E.P. Shurina and T.V. Vojtovich, *Analysis of the finite element and finite volume methods based upon unstructured grids for sloution of the Navier-Stokes equations*, Computational Technologies, Novosibirsk, ICT SD RAS, **2**, No. 4, 1997, 84–104 (in Russian).

[9] V.P. Il'in, *On data structures and algorithms of mathematical physics problems*, Preprint 938, Novosibirsk, Computing Center, 1991 (in Russian).

[10] Y.L. Gurieva, V.P. Il'in, and M.R. Larin, *Inner data structures of 2D boundary value problems*, Preprint 1090, Novosibirsk, Computing Center, 1997 (in Russian).

[11] Y.L. Gurieva, *Technological aspects of solving the mixed boundary value problems by the finite volume method*, Avtometriya, No. 5, 1997, 100–109 (in Russian).

[12] A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems* Prentice-Hall, Inc., New Jersey, 1981.

[13] V.P. Il'in, *Parallel Implicit Methods of Alternating Directions*, Comput. Math. & Math. Phys., **37**, No. 8, 1997, 871–878.