Bull. Nov. Comp. Center, Num. Anal., 11 (2002), 33–50

Program package for 3D boundary-value elliptic problem*

Y.L. Gurieva, V.P. Il'in

The main topic of the paper is to present a program package to solve 3D BVP for the Helmholtz-type equation. Numerical algorithms concerning data structures, approximations, and solving are described. Structure of the input data file and the usage examples are presented. Some useful recommendations for users are given.

The purpose of the described program is to solve a mixed boundary value problem (BVP) for the Helmholtz type equation with the piece-wise smooth coefficients in composed of parallelepipeds computational domain using fast incomplete factorization solvers on the basis of 7-, 19-, or 27-point finite–volume schemes on a non-uniform or uniform grid.

1. The BVP statement

Let $u(x,y,z)$ be a solution of the partial differential equation (PDE)

$$-\frac{\partial}{\partial x}\left(\lambda(x,y,z)\frac{\partial u}{\partial x}\right)-\frac{\partial}{\partial y}\left(\lambda(x,y,z)\frac{\partial u}{\partial y}\right)-\frac{\partial}{\partial z}\left(\lambda(x,y,z)\frac{\partial u}{\partial z}\right)+$$
$$\mu(x,y,z)u=f(x,y,z),\quad (x,y,z)\in\Omega=\bigcup_{k=1}^{m}\Omega_k, \tag{1}$$

in a bounded domain Ω composed of m rectangular subdomains Ω_k, where the given function λ is the positive constant or the function and functions $\mu\geq 0$, f are smooth in each Ω_k.

On the surfaces Γ_i of possible of discontinuity of λ (on common sides of different Ω_k, or "internal boundaries") the conjugate conditions hold:

$$u|_{\Gamma_+}=u|_{\Gamma_-},\qquad \lambda_+\frac{\partial u}{\partial\bar{n}}\bigg|_{\Gamma_+}=\lambda_-\frac{\partial u}{\partial\bar{n}}\bigg|_{\Gamma_-}, \tag{2}$$

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

On the external boundary $\Gamma=\Gamma_d\cup\Gamma_n$ of Ω the following Dirichlet (first kind) or Neumann (Newton if $æ\neq 0$) conditions are given:

*Supported by the Russian Foundation for Basic Research under Grants 02-01-01176 and 01-07-90367, and by Grant 10 of the 6-th Expertise Competition of 1999 year of Young Scientists.

$$u|_{\Gamma_d} = g, \qquad æu + \frac{\partial u}{\partial \bar{n}}\Big|_{\Gamma_n} = \gamma, \tag{3}$$

where g, $æ$, γ are given functions and $\bar{n}$ is an outerward normal to the boundary Γ.

The external and internal boundaries of Ω are considered to be multi-connected, all their parts being parallel to the coordinate axis.

2. Algorithms

The domain Ω is discretized by non-uniform parallelepipedoidal grid

$$x_{i+1} = x_i + h_i^x, \quad y_{j+1} = y_j + h_j^y, \quad z_{k+1} = z_k + h_k^z,$$
$$i = 0, \ldots, L-1, \quad j = 0, \ldots, M-1, \quad k = 0, \ldots, N-1,$$

and supposed to be embedded in parallelepiped $x_0 \le x \le x_L$, $y_0 \le y \le y_M$, $z_0 \le z \le z_N$, so that the boundary crosses the grid lines in the nodes only.

Along each coordinate, the grid consists of the finite number of mesh zones in which mesh steps are defined as a constant (so-called "u-zone" or "u-type" zone) or by a geometric progression ("g-zone") or by an arithmetic one ("a-zone").

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

$$X_p, \ p = 0, 1, \ldots, n_x, \quad Y_q, \ q = 0, 1, \ldots, n_y, \quad Z_r, \ r = 0, 1, \ldots, n_z,$$

be the boundaries of such zones with the following properties

$$X_0 = x_0, \quad X_p > X_{p-1}, \quad X_{n_x} = x_L,$$
$$Y_0 = y_0, \quad Y_q > Y_{q-1}, \quad Y_{n_y} = y_M,$$
$$Z_0 = z_0, \quad Z_r > Z_{r-1}, \quad Z_{n_z} = z_N.$$

The numbers of mesh steps in the zones are defined as l_p, m_q, and n_r and total numbers of nodes in each direction are given as

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

If the mesh zone is the zone of u-type, then the constant steps are defined as follows:

$$h_i^x = h_x^{(p)} = \frac{X_p - X_{p-1}}{l_p}, \quad i = L_{p-1} + 1, \ldots, L_p, \quad L_p = \sum_{s=1}^{p} l_s,$$

$$h_j^y = h_y^{(q)} = \frac{Y_q - Y_{q-1}}{m_q}, \quad j = M_{q-1}+1, \ldots, M_q, \quad M_q = \sum_{s=1}^{q} m_s,$$

$$h_k^z = h_z^{(r)} = \frac{Z_r - Z_{r-1}}{n_r}, \quad k = N_{r-1}+1, \ldots, N_r, \quad N_r = \sum_{s=1}^{r} n_s.$$

The mesh steps in g-type zones can be found using the formulas

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

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

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

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

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

The mesh steps in a-zones are defined as

$$h_i^{(x)} = h_x^{(p)} + (i - L_{p-1}) d_x^{(p)}, \quad i = L_{p-1}+1, \ldots, L_p,$$
$$h_j^{(y)} = h_y^{(q)} + (j - M_{q-1}) d_y^{(q)}, \quad j = M_{q-1}+1, \ldots, M_q,$$
$$h_k^{(z)} = h_z^{(r)} + (k - N_{r-1}) d_z^{(r)}, \quad k = N_{r-1}+1, \ldots, N_r,$$

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

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

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

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

satisfying the conditions of positiveness of the initial mesh steps

$$X_p - X_{p-1} > h_x^{(p)} = \frac{X_p - X_{p-1}}{l_p} - d_x^{(p)} \frac{l_p - 1}{2} > 0,$$
$$Y_q - Y_{q-1} > h_y^{(q)} = \frac{Y_q - Y_{q-1}}{m_q} - d_y^{(q)} \frac{m_q - 1}{2} > 0,$$
$$Z_r - Z_{r-1} > h_z^{(r)} = \frac{Z_r - Z_{r-1}}{n_r} - d_z^{(r)} \frac{n_r - 1}{2} > 0.$$

The finite-volume schemes for the problem are based on an approximation of the conservative law relation

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

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

$$V_{i,j,k} = \Big\{ (x_i + x_{i-1})/2 \le x \le (x_i + x_{i+1})/2,$$
$$(y_j + y_{j-1})/2 \le y \le (y_j + y_{j+1})/2,$$
$$(z_k + z_{k-1})/2 \le z \le (z_k + z_{k+1})/2 \Big\},$$
$$\bar{V}_{i,j,k} = \{ x_{i-1} \le x \le x_{i+1},\ y_{j-1} \le y \le y_{j+1},\ z_{k-1} \le z \le z_{k+1} \},$$

with a weight parameter w.

The resulting grid equations depend on the approximations of the integrals and the derivatives in equation (4) and are, in general, of 27-point type:

$$p^0_{i,j,k} u_{i,j,k} - p^3_{i,j,k} u_{i+1,j,k} - p^4_{i,j,k} u_{i,j+1,k} - p^6_{i,j,k} u_{i,j,k+1} -$$
$$p^3_{i-1,j,k} u_{i-1,j,k} - p^4_{i,j-1,k} u_{i,j-1,k} - p^6_{i,j,k-1} u_{i,j,k-1} -$$
$$p^7_{i,j,k} u_{i-1,j+1,k} - p^8_{i,j,k} u_{i+1,j+1,k} - p^9_{i-1,j,k-1} u_{i-1,j,k-1} -$$
$$p^{10}_{i,j,k} u_{i,j+1,k+1} - p^{11}_{i,j,k} u_{i-1,j,k+1} - p^{12}_{i,j,k} u_{i,j-1,k+1} -$$
$$p^7_{i+1,j-1,k} u_{i+1,j-1,k} - p^8_{i-1,j-1,k} u_{i-1,j-1,k} - p^9_{i-1,j-1,k-1} u_{i-1,j,k-1} -$$
$$p^{10}_{i,j-1,k-1} u_{i,j-1,k-1} - p^{11}_{i+1,j,k-1} u_{i+1,j,k-1} - p^{13}_{i,j,k} u_{i-1,j-1,k+1} -$$
$$p^{14}_{i,j,k} u_{i+1,j+1,k+1} - p^{15}_{i,j,k} u_{i-1,j+1,k+1} - p^{16}_{i,j,k} u_{i+1,j-1,k+1} -$$
$$p^{13}_{i+1,j+1,k-1} u_{i+1,j+1,k-1} - p^{14}_{i-1,j-1,k-1} u_{i-1,j-1,k-1} -$$
$$p^{15}_{i+1,j-1,k-1} u_{i+1,j-1,k-1} - p^{16}_{i-1,j+1,k-1} u_{i-1,j+1,k-1}$$

$$
\begin{aligned}
= \frac{h_i^x h_j^y h_k^z}{4}\Big[\frac{w}{16}(3f_{i,j,k} + f_{i+1,j,k} + f_{i,j+1,k} + f_{i,j,k+1}) + \\
\Big(1 - \frac{31}{32}w\Big)(f_{i+1,j,k} + f_{i,j+1,k} + f_{i+1,j,k+1} + f_{i,j+1,k+1})\Big] + \\
\frac{h_{i-1}^x h_j^y h_k^z}{4}\Big[\frac{w}{16}(3f_{i,j,k} + f_{i-1,j,k} + f_{i,j+1,k} + f_{i,j,k+1}) + \\
\Big(1 - \frac{31}{32}w\Big)(f_{i-1,j,k} + f_{i-1,j,k+1} + f_{i,j+1,k} + f_{i,j+1,k+1})\Big] + \\
\frac{h_{i-1}^x h_{j-1}^y h_k^z}{4}\Big[\frac{w}{16}(3f_{i,j,k} + f_{i-1,j,k} + f_{i,j-1,k} + f_{i,j,k+1}) + \\
\Big(1 - \frac{31}{32}w\Big)(f_{i,j-1,k} + f_{i,j-1,k+1} + f_{i-1,j,k} + f_{i-1,j,k+1})\Big] + \\
\frac{h_i^x h_{j-1}^y h_k^z}{4}\Big[\frac{w}{16}(3f_{i,j,k} + f_{i,j-1,k} + f_{i+1,j,k} + f_{i,j,k+1}) + \\
\Big(1 - \frac{31}{32}w\Big)(f_{i,j-1,k} + f_{i,j-1,k+1} + f_{i+1,j,k} + f_{i+1,j,k+1})\Big] + \\
\frac{h_i^x h_j^y h_{k-1}^z}{4}\Big[\frac{w}{16}(3f_{i,j,k} + f_{i,j,k-1} + f_{i+1,j,k} + f_{i,j+1,k}) + \\
\Big(1 - \frac{31}{32}w\Big)(f_{i+1,j,k} + f_{i,j+1,k} + f_{i+1,j,k-1} + f_{i,j+1,k-1})\Big] + \\
\frac{h_{i-1}^x h_j^y h_{k-1}^z}{4}\Big[\frac{w}{16}(3f_{i,j,k} + f_{i-1,j,k} + f_{i,j,k-1} + f_{i,j+1,k}) + \\
\Big(1 - \frac{31}{32}w\Big)(f_{i-1,j,k} + f_{i-1,j,k-1} + f_{i,j+1,k} + f_{i,j+1,k-1})\Big] + \\
\frac{h_{i-1}^x h_{j-1}^y h_{k-1}^z}{4}\Big[\frac{w}{16}(3f_{i,j,k} + f_{i-1,j,k} + f_{i,j-1,k} + f_{i,j,k-1}) + \\
\Big(1 - \frac{31}{32}w\Big)(f_{i,j-1,k} + f_{i,j-1,k-1} + f_{i-1,j,k} + f_{i-1,j,k-1})\Big] + \\
\frac{h_i^x h_{j-1}^y h_{k-1}^z}{4}\Big[\frac{w}{16}(3f_{i,j,k} + f_{i,j-1,k} + f_{i,j,k-1} + f_{i+1,j,k}) + \\
\Big(1 - \frac{31}{32}w\Big)(f_{i,j-1,k} + f_{i,j-1,k-1} + f_{i+1,j,k} + f_{i+1,j,k-1})\Big]. \qquad (5)
\end{aligned}
$$

For the weight $w = 32/31$, this equation turns into 19-point one with the zero coefficients $p^{13} = p^{14} = p^{15} = p^{16} = 0$, and for $w = 16/15$ the scheme is 7-point one with the zero coefficients $p^7, p^8, \ldots, p^{16}$, see [1–3] for details.

The order of accuracy differs for different values of w and mesh types:

$O(h)$ for $w = 16/15$ and general non-uniform grid,

$O(h^2)$ for $w = 16/17$ and general non-uniform grid or for any $0 \le w \le 1$ and piece-wise uniform grid (finite number of grid zones with constant mesh steps in each one),

$O(h^4)$ for $w = 32/31$, uniform grid, Dirichlet boundary conditions and constant coefficient λ, μ.

We should make several important remarks on the mentioned accuracy.

Remark 1. For general non-uniform grid with any mesh-size ratio the mentioned result is guaranteed for $w = 16/15$ only (7-point scheme);

Remark 2. The accuracy $O(h^4)$ for $w = 32/31$, $\mu = \text{const}$, $\lambda = \text{const}$, and uniform grid is guaranteed for the Dirichlet BVP only and under the condition

$$\frac{h_x^2}{h_y^2 + h_z^2} \le 4, \quad \frac{h_y^2}{h_x^2 + h_z^2} \le 4, \quad \frac{h_z^2}{h_y^2 + h_x^2} \le 4.$$

Remark 3. For $w = 16/17$, the accuracy $O(h^2)$ on non-uniform grid is guaranteed under the conditions

$$\begin{aligned}
&\alpha \le h_k h_{k-1}\left(\frac{1}{h_j^2} + \frac{1}{h_i^2}\right), && h_k^2\left(\frac{1}{h_i h_{i-1}} + \frac{1}{h_j h_{j-1}}\right) \le \beta, \\
&\alpha \le h_i h_{i-1}\left(\frac{1}{h_k^2} + \frac{1}{h_j^2}\right), && h_i^2\left(\frac{1}{h_k h_{k-1}} + \frac{1}{h_j h_{j-1}}\right) \le \beta, \\
&\alpha \le h_j h_{j-1}\left(\frac{1}{h_i^2} + \frac{1}{h_k^2}\right), && h_j^2\left(\frac{1}{h_i h_{i-1}} + \frac{1}{h_k h_{k-1}}\right) \le \beta, \\
&\alpha = \frac{32 - 31p}{32 - 29p}, && \beta = \frac{32 - 23p}{32 - 29p}.
\end{aligned} \tag{6}$$

The latter inequalities are sufficient for existence and convergence of grid solution, as well as for applicability of iterative processes.

Remark 4. The above mentioned accuracy is valid for $\lambda > 0$, $\mu \ge 0$, $æ \ge 0$ and smooth enough functions $u(x, y, z)$, $f(x, y, z)$, $\mu(x, y, z)$ in each subdomain Ω_k, see [1] for more details.

The solution of algebraic system (5) is computed on the basis of iterative incomplete factorization methods accelerated by the conjugate gradient algorithm. Namely, the Eisenstadt modification of generalized symmetric successive over relaxation (SSOR) algorithm with the compensation principle (row sum criteria, see [4]) is used. This method includes iterative compensation parameter $0 \le \theta \le 1$ and relaxation parameter $0 \le \omega \le 1$ for

possibility to minimize the number of iterations. For the Stiltjes matrices, the values $\theta = \omega = 1$ are recommended. The iterative process stops when one of the following conditions holds:

$$\|r^0\| \leq \frac{\varepsilon}{LMN}, \quad \frac{\|r^n\|}{\|r^0\|} \leq \varepsilon, \quad n \leq n_{\max}. \tag{7}$$

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

Let us note that the termination of iterations does not guarantee the accuracy of solution $\delta_n = \max_{i,j,k}\{|u_{i,j,k} - u^n_{i,j,k}|\} \leq \varepsilon$ (usually $\delta_n \approx 10\varepsilon$). To understand the real error, the reasonable analysis of u^n is recommended for different ε and n.

3. Program and data

The presented algorithms are the basis for the program package to solve the problem under consideration. The programming language of the code is Fortran-77. The package consists of several subroutines which read the input data, construct the internal data structures and solves iteratively the resulting linear system of grid equations.

The program runs in the manner when regardless some error in the data it will end presenting some information describing the emergency cause. The usage of the code is the following:

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

Here the arguments are

`fname` – name of the input file (input);

`u` – array of size *lmn* for the solution values at the grid points (output);

`x` – array of size l containing the x-coordinates of the grid nodes (output);

`y` – array of size m containing the y-coordinates of the grid nodes (output);

`z` – array of size n containing the z-coordinates of the grid nodes (output);

`nmax` – maximum admissible number of iterations (input) and real number of iterations (output);

`l, m, n` – number of mesh-points in x-, y- and z- direction (output);

`rh` – user-supplied function for the right-hand side of the PDE. The form is `rh(isr, x, y, z)`, where

`x` – x-coordinate value (input),
`y` – y-coordinate value (input),
`z` – z-coordinate value (input),
`isr` – number of subregion which contains the point (x, y, z) (input),
`rh` – value of right-hand side at the point (x, y, z) (output);

`dfun` – user-supplied function for the right side function g in the Dirichlet boundary condition (3). The form is `dfun(nbcond, x, y, z)`, where

`nbcond` – number of boundary condition (input),
`x` – x-coordinate value (input),
`y` – y-coordinate value (input),
`z` – z-coordinate value (input),
`dfun` – the value of the function g at the point (x, y, z) (output);

`cfun` – user-supplied function to evaluate the Helmholtz coefficient μ in the PDE (1). The form is `cfun(isr, x, y, z)`, where input parameters are the same as in `rh(isr, x, y, z)`; cfun returns the value of the function μ at the point (x, y, z) (output);

`userk` – user-supplied function for the coefficient æ in Newton boundary condition (3). The form is `userk(nbcond, x, y, z)`, where input parameters are the same as in `dfun`; userk returns the value of the coefficient æ at the point (x, y, z) (output);

`userg` – user-supplied function for the right side of Newton condition (3). The form is `userg(nbcond, x, y, z)`, where input parameters are the same as in `dfun`; userg returns the value of the function γ in (3) at the point (x, y, z) (output);

`ido` – flag indicating the state of computations (input/output); non-zero value of the flag indicates some error in the data or error arising during the iterative solution.

We will now give some details of geometric modeling for the problem under consideration to understand then the input data on geometry of the computational domain.

The computational region is constructed from rectangular parallelepipeds in accordance with the following rules:

- each parallelepiped is described by two corner points: bottom-left-front (minimal coordinates) and top-right-back (maximal coordinates);
- firstly, a user should define the largest parallelepiped which includes the computational domain with holes, subdomains, etc.;
- then define smaller embedded parallelepipeds with different media (a hole is a subregion with zero medium coefficient);

- computational domain is conjunction and/or embedding of its subregions;
- only simple embedding is admissible (any subregion in a subregion being already embedded into another one is prohibited);
- conjunction and embedding of subregions is given by the incidence matrix (see comments on it after description of the input file structure);
- the boundaries of mesh zones in each dimension must be the value from the set formed from all different projections of all faces of all subdomains onto corresponding direction.

The data for the program are given in the input file with the fixed format, and, in several user functions, describing some coefficients of equation (1). The data file includes the input data on the geometry of the computational domain, the differential statement of the BVP, and computational parameters of the algorithm. File consists of the following integer and real values given in the fixed order:

`ncub` – number of parallelepipeds;

`x1, y1, z1` – real arrays of size ncub containing coordinates of first corner point of parallelepipeds, with minimal coordinates;

`x2, y2, z2` – real arrays of size ncub containing coordinates of second corner point of parallelepipeds, with maximal coordinates;

`incid` – incidence matrix to determine the topology of the region;

`valmed` – real array of size ncub containing the values of the diffusion coefficients $\lambda > 0$ from equation (1) in each subregion;

`helm` – real array of size ncub containing the values of the Helmholtz coefficient $\mu(x, y) \geq 0$ from equation (1) in each subregion; $\texttt{helm}(k) \geq 0$ (real value of μ) means the constant value of μ in k-th subregion and $\texttt{helm}(k) < 0$ means that μ is defined by user-supplied function `cfun`;

`rhscon` – real array of size ncub containing the value of right side function f from equation (1) in each subregion; if $|\texttt{rhs}(k) - 3.1415| < 10^{-5}$, it means that in k-th subregion $f(x, y)$ is defined by a user-supplied function `rh`, in other cases rhscon(k) is the constant value of the function f in k-th subregion;

`nzx` – number of mesh zones (integer) in x-direction;

`xz` – integer array of size nzx+1 containing x-coordinates of mesh zone boundaries;

`nsx` – integer array of size nzx containing the numbers of mesh steps in x-zones;

`tx` – real array of size `nzx` containing the the types of x-zones: $\mathtt{tx}(k) = 1$ (or, more precisely, $|\mathtt{tx}(k) - 1| \leq 10^{-5}$) means that k-th zone is of u-type, i.e., uniform grid in k-th zone; $\mathtt{tx}(k) < 0$ means that k-th zone is of a-type and $-\mathtt{tx}(k)$ is the initial mesh step h_k^x; otherwise $\mathtt{tx}(k)$ is the mesh ratio of k-th g-type zone; $\mathtt{tx}(k) = 0$ is prohibited;

`nzy` – number of mesh zones (integer) in y-direction;

`yz` – integer array of size `nzy+1` containing y-coordinates of mesh-zone boundaries;

`nsy` – integer array of size `nzy` containing the numbers of mesh steps in y-zones;

`ty` – real array of size `nzy` containing the types of zones (with different values as for x-zones);

`nzz` – number of mesh zones (integer) in z-direction;

`zz` – integer array of size `nzz+1` containing z-coordinates of mesh-zone boundaries;

`nsz` – integer array of size `nzz` containing the numbers of mesh steps in z-zones;

`tz` – real array of size `nzz` containing the types of zones (with different values as for x-zones);

`ndir1` – number of the standard Dirichlet boundary conditions with different constant values;

`ndir3` – number of the Dirichlet boundary conditions with different user supplied functions `dfun`;

`nnei2` – number of different constant Newton boundary conditions ($æ$ = const, γ = const);

`nnei3` – number of different Newton conditions defined by user-supplied functions `userk` and `userg`.

`bd` – real array of size `ndir1` containing the values of parameter g for each standard Dirichlet condition (g = const);

`bnk` – real array of size `nnei2` for constant $æ$;

`bng` – real array of size `nnei2` for constant γ for Newton conditions;

`nbc` – integer array of size `ncub*6` with numbers of boundary conditions on all faces of the region: first, six numbers for the faces of the first subregion, then six numbers for the second subregion, etc. The zero value of nbc(i) means standard Neumann condition or "internal boundary" (without any condition);

`iorder` – order of accuracy of finite volume scheme (admissible values are 1, 2, 4);

θ, ω, ε, `nmax` – values of iterative compensation and relaxation parameter of incomplete factorization methods ($0 \leq \theta \leq 1, 1 \leq \omega \leq 2$), accuracy ε of iterative process in (7); maximum number of iterations in (7).

The incidence matrix needs some explanations. As the computational region is composed of non-intersecting parallelepipedoidal subregions, it is described by the surfaces of its subregions and defined by their topological neighbourhood. So, let the computational region consist of m parallelepipeds such that any face of every parallelepiped cannot intersect any edge of any other parallelepiped, but the parallelepipeds can touch each other by the whole faces, or by their parts, or can be located thoroughly one in another. Then the computational region can be represented by a parallelepiped (let us call it the "maximal" one) in which another parallelepipeds of smaller size with different physical characteristics are distinguished or from which another parallelepipeds of smaller volume are "cut out" if any.

Let us define the origin of the Cartesian coordinate system in one of the corners of the maximal parallelepiped. Then every subregion can be defined via six real numbers which are the coordinates of two corners - one with the minimal coordinates and another with the maximal ones. The order of representation of the subregions defines their numbering. This is precisely topological adjacency which defines the shape of the computational region. To set the adjacency for the parallelepipeds, the incidence matrix is introduced. It is square $m \times m$ matrix $\{m_{ij}\}$ with the integer entries of the following values:

$6 \geq m_{i,j} > 0$ – the number of a face of parallelepiped number i touching from outside parallelepiped number j;

$-6 \leq m_{i,j} < 0$ – the number of a face of the parallelepiped number i touching from inside parallelepiped number j;

7 – parallelepiped number i contains entirely parallelepiped number j;

-7 – parallelepiped number i is contained entirely in parallelepiped number j.

Two last lines of this description mean that for the intersection of two parallelepipeds, the value of $\lambda(x, y, z)$ is taken from the smaller parallelepiped. Here we also assume that the faces of every parallelepiped is numbered $1, \ldots, 6$ in a fixed order, say, left, near, right, far, bottom, and top.

Let us also make some explanations on the boundary conditions and their posing on the external boundaries of the computational region. Different boundary conditions are successfully numbered in the following order: constant Dirichlet conditions, nonstandard Dirichlet conditions (g is defined by the user-supplied function `dfun`), Newton conditions with constant coefficients and, at last, nonstandard Newton conditions ($æ$ and γ are defined by user-supplied functions `userk` and `userg`).

Faces of each parallelepiped are given numbers $1, \ldots, 6$ in the following order: left, front, right, back, bottom, and top.

Each face is given one number of the boundary condition form the range $0, \ldots,$ `ndir1` $+$ `ndir3` $+$ `nnei2` $+$ `nnei3`.

Below is the text of the input data file for Example 3:

```
3                       number of cubes;
0. 0. 0. 1. 1. 1.       two corner points of 1st cube;
0. 0. 0. .25 .25 .25    two corner points of 2nd cube;
.25 .25 .25 .75 .75 .75 two corner points of 3d cube;
1 7 7                   1st row of incidence matrix;
-7 2 0                  2nd row of incidence matrix;
-7 0 3                  3rd row of incidence matrix;
2. 1. 0.                media in subregions;
0. 0. 0.                the Helmholtz coefficient in subregions;
3.1415 3.1415 0.        right-hand side as function in 1st and 2nd subregions;
3                       number of x-zones;
0. .25 .75 1.           boundaries of x-zones;
5 10 5                  number of steps in x-zones;
1. 1. 1.                step types in x-zones are uniform;
3                       number of y-zones;
0. .25 .75 1.           boundaries of y-zones;
5 10 5                  number of steps in y-zones;
1. 1. 1.                step types in y-zones are uniform;
3                       number of z-zones;
0. .25 .75 1.           boundaries of z-zones;
5 10 5                  number of steps in z-zones;
1. 1. 1.                step types in z-zones are uniform;
0 2 0 3                 boundary conditions: two user Dirichlet and three user
                        Newton;
1 1 3 4 1 5 2 2 0 0 2 0 1 1 1 1 1 1 numbers of conditions on all
                        faces;
1                       order of approximation;
1. 1. 1.D-11 90         θ, ω, ε and maximum number of iterations.
```

4. Examples of program applications

Example 1. The Laplace equation (1) with $\lambda \equiv 1$, $f = \mu \equiv 0$ is solved in the unit square with the Dirichlet boundary conditions which correspond to two kinds of harmonic polynomials: $u(x,y,z) = x^3 - 3xy^2$, $u(x,y,z) = x^4 - 6x^2y^2 + y^4$.

The numerical solutions are obtained using different stencils on two sets of embedded grids: uniform and nonuniform. The uniform grids are cubic ones with the number of nodes $L = 11, 21, 41$ in each coordinate direction. The second grid set has the same constant mesh steps in z-direction and $h_i^x = h_i^y = h_0 q^{i-1}$, $i = 1, \ldots, L$ with $q = 0.95$. Tables 1, 2 give the mean square errors $\delta = \|u - u^n\|_2 / L^3$ of numerical solutions for different weight parameters, for iterative accuracy $\varepsilon = 10^{-11}$.

Table 1 presents the errors δ for the problem with the exact solution in the form of the fourth order polynomial for the uniform grid. It is easy to see that two first approximations have the error $O(h^2)$. For this test with the Mikeladze parameter $w = 32/31$ we obtain the exact solution, i.e., $\delta \approx \varepsilon$. And for the exact solution in the form of the cubic harmonic polynomial all these schemes provide $\delta \approx \varepsilon$ in accordance with theoretical estimates. It is interesting to note that in this example the scheme with the optimal parameter $w = 16/17$ (in the sense of the estimate on nonuniform grid) is almost the worst scheme experimentally.

Table 1. The error δ on uniform grids

w \ L	11	21	41
16/15	0.00085	0.00023	0.000062
16/17	0.00088	0.00023	0.000060
32/31	$2.8 \cdot 10^{-9}$	$1.5 \cdot 10^{-9}$	$1.6 \cdot 10^{-9}$

Table 2 presents the similar tests but for the nonuniform grids. The upper error value in each cell of the table corresponds to the cubic, and the low value – to the fourth order polynomial. One can make several conclusions from these data.

Table 2. The error δ on nonuniform grids

w \ L	11	21	41
16/15	0.00022 0.00020	0.00012 0.00022	0.000061 0.00014
16/17	10^{-8} 0.00091	10^{-8} 0.00027	10^{-8} 0.000092
32/31	0.00011 0.00044	0.000061 0.00023	0.000033 0.00012

The unique scheme of the second order on the nonuniform grid is that corresponding to $w = 16/17$. Firstly, the errors for the test with the cubic solution only in this case give $\delta \approx \varepsilon$. Secondly, for harmonic polynomial of the fourth order the error dependence on the mesh size is close to $O(h^2)$.

The error dependence on h both for $w = 16/15$ and $w = 32/31$ is close to the linear one. It is interesting to mention that the best scheme on the uniform grid – the scheme with $w = 32/31$ – is the worst in this case.

Example 2. The Helmholtz equation (1) with $\lambda \equiv 1$ and three variants of the coefficient μ and the function f:

a) $\mu = f = 0$,

b) $\mu = 0, \quad f = u(x, y, z)$,

c) $\mu = 1, \quad f = 3\pi^2 \sin \pi x \sin \pi y \sin \pi z$,

is solved in the unit square on the uniform grids ($L = M = N = 11, 21, 41$) with the Dirichlet boundary conditions which provide the exact solution in the form

$$u(x, y, z) = \sin \pi x \sin \pi y \left[\frac{\mathrm{sh}(\pi\sqrt{2}z)}{\mathrm{sh}(\pi\sqrt{2})} + \rho \right],$$

where $\rho = 0$ for $f = 0$ and $\rho = \sin \pi z$ for $f \neq 0$.

Table 3 presents the errors of the test results for 7-point scheme ($w = 16/15$), 19–point scheme ($w = 32/31$) and 27–point scheme ($w = 16/17$). The upper value in each cell corresponds to the variant "a", the middle – to "b" and low- to "c". There also are the numbers of iteration in the brackets.

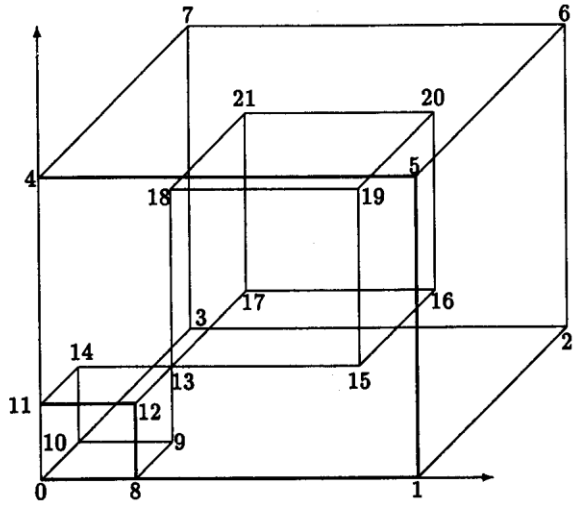
Table 3. The error δ for Example 2, uniform grids

w \ L	10		20		40	
16/17	0.0045	(13)	0.0011	(23)	0.00028	(41)
	0.0117	(13)	0.00288	(23)	0.000717	(42)
	0.00446	(13)	0.00110	(23)	0.000275	(41)
16/15	0.0044	(13)	0.0011	(34)	0.00028	(61)
	0.0109	(18)	0.00280	(34)	0.00071	(61)
	0.00431	(19)	0.00109	(34)	0.00275	(60)
32/31	$5.87 \cdot 10^{-5}$	(16)	$3.68 \cdot 10^{-6}$	(29)	$2.32 \cdot 10^{-7}$	(50)
	$1.39 \cdot 10^{-4}$	(16)	$8.63 \cdot 10^{-6}$	(29)	$5.39 \cdot 10^{-7}$	(50)
	$5.74 \cdot 10^{-5}$	(16)	$3.60 \cdot 10^{-6}$	(29)	$2.26 \cdot 10^{-7}$	(49)

These results confirm the second order accuracy on the uniform grid for 7- and 27-point schemes ($w = 16/15$ and $w = 16/17$), with the same errors approximately for all variants. The 19-point scheme ($w = 32/31$) has considerably higher accuracy and the fourth order accuracy, in accordance with theoretical estimates.

Example 3. The goal of this experiment is to demonstrate the applicability of the algorithms and the program for solution of complicate boundary problem, described in the figure.

The computational domain is the unit cube $\Omega_1 = \{0 < x, y, z < 1\}$ without the cubic hole $1/4 < x, y, z < 3/4$, with two subdomains of different constant diffusion coefficient $\lambda : \lambda = \lambda_2 = 1$ in the small cube $\Omega_2 = \{0 < x, y, z < 1/4\}$ and $\lambda = \lambda_1 = 2$ in the rest part of Ω_1.



Geometry of computational domain in Example 3

The boundary conditions and the right side function f are defined by selected exact solution

$$u(x,y,z) = \begin{cases} u_1, & (x,y,z) \in \Omega_1 \setminus \Omega_2, \\ u_2, & (x,y,z) \in \Omega_2; \end{cases}$$

$$u_m = 1 - \left(x - \frac{1}{4}\right)^2 - \left(y - \frac{1}{4}\right)^2 - \left(z - \frac{1}{4}\right)^2 + \left(x - \frac{1}{4}\right)\left(y - \frac{1}{4}\right)\left(z - \frac{1}{4}\right)\lambda_m, \quad m = 1,2;$$

$$u|_{x=0} = \begin{cases} u_2(0,y,z), & (0,y,z) \in S_2, \\ u_1(0,y,z), & (0,y,z) \in S_1 \setminus S_2, \end{cases}$$

$$u|_{y=0} = \begin{cases} u_2(x,0,z), & (x,0;z) \in S_2, \\ u_1(x,0,z), & (x,0;z) \in S_1 \setminus S_2, \end{cases}$$

$$u|_{z=0} = \begin{cases} u_2(x,y,0), & (x,y,0) \in S_2 \\ u_1(x,y,0), & (x,y,0) \in S_1 \setminus S_2, \end{cases}$$

$$\left.\frac{\partial u}{\partial x}\right|_{x=1} = -2\left(x - \frac{1}{4}\right) + \left(y - \frac{1}{4}\right)\left(z - \frac{1}{4}\right)/\lambda_2,$$

$$\left.u + \frac{\partial u}{\partial n}\right|_{y=1} = -2\left(y - \frac{1}{4}\right) + \left(x - \frac{1}{4}\right)\left(z - \frac{1}{4}\right)/\lambda_2 + u_2(x,1,z),$$

$$\left.u + \frac{\partial u}{\partial n}\right|_{z=1} = -2\left(z - \frac{1}{4}\right) + \left(x - \frac{1}{4}\right)\left(y - \frac{1}{4}\right)/\lambda_2 + u_2(x,y,1).$$

The maximum error δ for this solution on the uniform grid $20 \times 20 \times 20$ is 0.00028 (7-point scheme for $w = 16/15$ was used).

User supplied functions and main program for Example 3

```
      program head
      implicit real*8(a-h,o-z)
      integer idimx, idimy, idimz
      parameter (idimx=41, idimy=41, idimz=41)
      dimension x(idimx), y(idimy), z(idimz),
                u(idimx*idimy*idimz)
      character*10 fname
      external rh, dfun, userk, userg, cfun
      real*8 rh, dfun, userk, userg, cfun
      fname='cubex3.dat'
      call fvsde3(fname,u,x,y,znmax,l,m,k,ido,rh,dfun,
                  userk,userg,cfun,ido)
      if(ido.ne.0) then
        print *,' Abnormal state: ido = ',ido
        stop
      end if
      print 100,' number of iterations =',nmax
      call deferr(l,m,k,u,x,y,z)
 100  format (a23,i3)
      stop
      end

      real*8 function rh(isr,x,y,z)
      implicit real*8(a-h,o-z)
      rh=12.d0
      if(isr.eq.2) rh=6.d0
      return
      end

      real*8 function dfun(nbcond,x,y,z)
      implicit real*8(a-h,o-z)
      x2=x-.25d0
      y2=y-.25d0
      z2=z-.25d0
      if(nbcon.eq.2) dfun=1.d0+x2*y2*z2-x2*x2-y2*y2-z2*z2
      if(nbcon.eq.1) dfun=1.d0+x2*y2*z2*.5d0-x2*x2-y2*y2-z2*z2
      return
      end

      real*8 function userk(nbcond,x,y,z)
      implicit real*8(a-h,o-z)
      userk=1.d0
```

```
if(nbcond.eq.3) userk=0.d0
return
end

real*8 function userg(nbcond,x,y)
implicit real*8(a-h,o-z)
nbs=1
if(nbcond.eq.3) userg=-1.5d0+(y-.25d0)*(z-.25d0)*.5d0
    +dfun(nbs,x,y,z)
if(nbcond.eq.4) userg=-1.5d0+(x-.25d0)*(z-.25d0)*.5d0
    +dfun(nbs,x,y,z)
if(nbcond.eq.5) userg=-1.5d0+(y-.25d0)*(x-.25d0)*.5d0
    +dfun(nbs,x,y,z)
return
end

real*8 function cfun(isr,x,y,z)
implicit real*8(a-h,o-z)
integer*2 isr
cfun=0.d0
return
end
```

Output, Example 3

```
number of iterations = 25
max error is .281E-03 at the point  19  21  21
```

5. Recommendations for user

- If you are interested in a robust computations mainly and not in some special optimization of the algorithm, use iterative parameter $\omega = \theta = 1$ and 7-point scheme (first order of accuracy in general), which is applicable for any non-uniform grid.
- To obtain a very high accuracy or a very fast solving process for the usual accuracy for simple problem, use uniform grid and $O(h^4)$ scheme taking into account Remark 2 on the accuracy.
- If you need to solve many variants of the similar problems, try to experimentally optimize iterative parameter θ, ω.
- If you cannot apply uniform grid and use slightly non-uniform one, try to experimentally optimize the mesh under conditions (6) and use $O(h^2)$ scheme.

- If you have non-successfully run (`ido` $\neq 0$), check the input data or try to change the algorithm parameters (grid, order of approximation, iterative parameters).

References

[1] Gurieva Y.L., Il'in V.P. On the finite volume technology for mixed boundary value problems // Proceedings of the International Conference AMCA-95. – Novosibirsk: NCC Publisher, 1995. – P. 650–655.

[2] Gurieva Y.L., Il'in V.P. On second order finite volume approximations for 3D mixed boundary value problems // NCC Bulletin, Series Num. Anal. – Novosibirsk: NCC Publisher, 1996. – Issue 7. – P. 51–70.

[3] Il'in V.P. Finite Difference and Finite Volume Methods for Elliptic Equations. – Novosibirsk: ICMMG publ., 2001.

[4] Il'in V.P. Iterative Incomplete Factorization Methods. – Singapore: World Scientific Publishing Co., 1992.