Bull. Nov. Comp. Center, Num. Anal., 12 (2003), 1–12

Two solvers for nonsymmetric SLAEs*

M.Yu. Andreeva, V.P. Il'in, E.A. Itskovich

The implementation of the biconjugate and the squared conjugate gradient (BiCG and CGS) preconditioned iterative methods are described for solving nonsymmetric systems of linear algebraic equations (SLAEs) which arise when approximating multi-dimensional boundary value problems (BVPs) for diffusion-convection partial differential equations (PDEs), by finite eifference, finite element and finite volume methods (FDM, FEM and FVM). The results of numerical experiments are discussed.

1. Introduction

The numerical solution to nonsymmetric SLAEs is the subject of great practical importance and numerous publications, see, for example, [1] and references cited there. In recent decades, various versions of generalized conjugate gradient methods were proposed, such as GMRES, BiCGStab, and some others. Significance of this problem is explained, in particular, by the necessity of solving very large algebraic systems with sparse matrices occurring in FDM, FEM and/or FVM approximations of multidimensional BVPs for diffusion-convection PDEs [2, 3]. Such mathematical statements describe heat- and mass- transfer processes as well as hydro- and gasdynamic flows. In nonlinear and/or nonstationary systems of differential equations such SLAE have been multiply solved at every time step and at different stages of multilevel iterative processes.

Here we consider two iterative solvers and their applications for solution of three types of finite difference approximations (with central difference and one-side difference discretizations of first order derivatives as well as an exponential type scheme) for the two-dimensional and the three-dimensional (2D and 3D) diffusion-convection equations:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big(+\frac{\partial^2 u}{\partial z^2}\Big) + p\frac{\partial u}{\partial x} + q\frac{\partial u}{\partial y}\Big(+r\frac{\partial u}{\partial z}\Big) = f(x,d,z), \quad (x,y,z) \in \Omega, \qquad (1)$$
$$u|_\Gamma = g(x,y,z).$$

Here the coefficients p, q, and r have the physical sense of velocity and are for simplicity considered to be constants in the computational domain Ω.

*Supported by the Russian Foundation for Basic Research under Grants 02-01-01176, 01-07-90367 and NWO-RFBR under Grant 047-008-007.

Both presented algorithms are based on the preconditioning by multiparameter explicit incomplete factorization in the efficient Eusenstat modification [4] and acceleration by the BiCG and the CGS algorithms with multivariant restarts.

This paper is constructed as follows. In Section 2 we describe the systems of equations to be solved, and their algebraic properties. Section 3 presents implementation of the preconditioning procedure as well as biconjugate and conjugate squared algorithms. The last section is devoted to discussions of the results of numerous computational experiments.

2. Description of the algebraic systems

We consider three types of finite difference approximations of equation (1) in the unit cube (or the square in 2D) computational domain, under the Dirichlet boundary conditions $u|_\Gamma = g(x,y,z)$ on the uniform grid

$$\begin{aligned} &x_{i+1} = x_i + h_x, \quad i = 0,1,\ldots,I, \quad x_0 = 0, \quad x_{I+1} = 1, \\ &y_{j+1} = y_j + h_y, \quad j = 0,1,\ldots,J, \quad y_0 = 0, \quad y_{J+1} = 1, \\ &z_{k+1} = z_k + h_z, \quad k = 0,1,\ldots,K, \quad z_0 = 0, \quad z_{K+1} = 1. \end{aligned}$$

Two first approximations are based on the standard three-point discretizations of second order derivatives and on the presentations of first order derivatives by the central or one-side differences:

$$\left.\frac{\partial u}{\partial x}\right|_{x=x_i} = \frac{u_{i+1} - u_{i-1}}{2h_x} + O(h_x^2) = \frac{u_{i+1} - u_i}{h_x} + O(h_x). \tag{2}$$

The finite difference seven-point equations are written down in the following form:

$$\begin{aligned} (Au)_{i,j,k} = {} & a^0_{i,j,k}u_{i,j,k} - a^1_{i,j,k}u_{i-1,j,k} - a^2_{i,j,k}u_{i,j-1,k} - a^3_{i,j,k}u_{i+1,j,k} - \\ & a^4_{i,j,k}u_{i,j+1,k} - a^5_{i,j,k}u_{i,j,k-1} - a^6_{i,j,k}u_{i,j,k+1} = f_{i,j,k}, \qquad (3) \\ & a^1_{1,j,k} = a^2_{i,1,k} = a^3_{I,j,k} = a^4_{i,J+1,k} = a^5_{i,j,1} = a^6_{i,j,K} = 0, \\ & i = 1,2,\ldots,I, \quad j = 1,2,\ldots,J, \quad k = 1,2,\ldots,K. \end{aligned}$$

For approximation of first order derivatives by the central difference scheme (CD), nonzero coefficients are described by the formulas

$$\begin{gathered} a^{2\pm1}_{i,j,k} = \frac{2 \pm ph_x}{2h_x^2}, \quad a^{3\pm1}_{i,j,k} = \frac{2 \pm qh_y}{2h_y^2}, \quad a^{5.5\pm0.5}_{i,j,k} = \frac{2 \pm rh_z}{2h_z^2}, \\ a^0_{i,j,k} = \frac{2}{h_x^2} + \frac{2}{h_y^2} + \frac{2}{h_z^2}. \end{gathered} \tag{4}$$

For the one-(right-)side finite difference discretization (OS), we have

$$a^1_{i,j,k} = \frac{1}{h_x^2}, \quad a^2_{i,j,k} = \frac{1}{h_y^2}, \quad a^3_{i,j,k} = \frac{1+ph_x}{h_x^2}, \quad a^4_{i,j,k} = \frac{1+qh_y}{h_y^2},$$
$$a^5_{i,j,k} = \frac{1}{h_z^2}, \quad a^6_{i,j,k} = \frac{1+rh_z}{h_z^2}, \quad a^0_{i,j,k} = \frac{2+ph_x}{h_x^2} + \frac{2+qh_y}{h_y^2} + \frac{2+rh_z}{h_z^2}. \tag{5}$$

The third scheme is of the exponential type (ET) whose coefficients are presented as follows:

$$a^1_{i,j,k} = \frac{1}{h_x} e^{-ph_x/2}, \quad a^2_{i,j,k} = \frac{1}{h_y^2} e^{-qh_y/2}, \quad a^3_{i,j,k} = \frac{1}{h_x^2} e^{ph_x/2},$$
$$a^4_{i,j,k} = \frac{1}{h_y^2} e^{qh_y/2}, \quad a^5_{i,j,k} = \frac{1}{h_z^2} e^{-rh_z/2}, \quad a^6_{i,j,k} = \frac{1}{h_z^2} e^{rh_z/2}, \tag{6}$$
$$a^0_{i,j,k} = \frac{1}{h_x^2}(e^{ph_x/2} + e^{-ph_x/2}) + \frac{1}{h_y^2}(e^{qh_y/2} + e^{-qh_y/2}) + \frac{1}{h_z^2}(e^{rh_z/2} + e^{-rh_z/2}).$$

A short overview of peculiarities of these three finite difference systems of equations is the following:

- The CD and the ET schemes (4) and (6) have the second order accuracy $O(h^2)$ on the uniform grid, OS approximation has the first order accuracy $O(h)$, only;
- The CD scheme is of the positive type (by the Brambles definition this means positiveness of all $a_{i,j,k}$ in (3), the row diagonal dominance and monotonicity of A, i.e., $A^{-1} \geq 0$) under the conditions

$$|p| \leq \frac{h_x}{2}, \quad |q| \leq \frac{h_y}{2}, \quad |r| \leq \frac{h_z}{2}; \tag{7}$$

- The OS scheme is of the unconditionally positive type for the nonnegative velocities p, q, r, but in general it has diagonal dominance and monotonicity, if the following inequalities are fulfilled:

$$|p| \leq h_x, \quad |q| \leq h_y, \quad |z| \leq h_z; \tag{8}$$

- The ET scheme has the positive coefficients $a^l_{i,j,k}$, diagonal dominance and monotonicity properties for any value of the velocities p, q, r and the mesh steps h_x, h_y, h_z;
- The matrix A for the CD scheme has positive definite symmetric part, i.e.,

$$(A^s v, v) \equiv \frac{1}{2}((A + A^T)v, v) > 0 \tag{9}$$

for any vector v, even for the variable velocities p, q, r; the OS and the EP schemes have the same properties for constant velocities, but in general this is not valid.

3. The preconditioned BiCG and the CGS restarted methods

At first we consider the Eisenstat modification of the preconditioned form of the algebraic system

$$Au = f, \quad A = D - L - U, \tag{10}$$

where D, L, and U are diagonal, low and upper triangular parts of the matrix A.

We use the explicit incomplete factorization, for which the usual form of a preconditioner is the following:

$$\begin{gathered} B = (G - L)G^{-1}(G - U), \quad G = \frac{1}{\omega}D - \theta S, \\ Se = \Big(\frac{1-\omega}{\omega}D + LG^{-1}U\Big)e. \end{gathered} \tag{11}$$

Here ω, θ are the relaxation and the compensation parameters respectively, e is the vector with unit entries, G and S are diagonal matrices defined from the row sum criteria $Ae = Be$. If $A = \{a_{l,m},\ l,m = 1,\dots,N\}$ and the vector $\eta = \{\eta_l = (Ue)_l = \sum_{m>l} a_{l,m}\}$ is introduced, then $G = \{g_l\}$ is defined by the recurrent relations

$$g_1 = \frac{1-\omega}{\omega}a_{1,1}, \quad g_l = \frac{1-\omega}{\omega}a_{l,l} - \theta\sum_{m=5}^{l-1}\frac{a_{l,m}\eta_m}{g_m}, \quad l = 2,\dots,N. \tag{12}$$

The Eisenstat modification of incomplete factorization consists in solving the preconditioned system

$$\bar{A}\bar{u} = \bar{f} = (I - \bar{L})^{-1}G^{-1/2}f, \tag{13}$$

where the new vectors and matrices are defined as

$$\begin{aligned} &\bar{A} = (I - \bar{L})^{-1} + (I - \bar{U})^{-1} - (I - \bar{L})^{-1}(2I - \bar{D})(I - \bar{U})^{-1}, \\ &\bar{L} = G^{-1/2}LG^{-1/2}, \quad \bar{U} = G^{-1/2}UG^{-1/2}, \quad \bar{D} = G^{-1/2}DG^{-1/2}, \\ &\bar{u} = (I - \bar{U})G^{1/2}u. \end{aligned} \tag{14}$$

The matrix $\bar{A}$ is obtained by similarity transformation from the matrix product $B^{-1}A$:

$$\begin{gathered} \bar{A} = L_B^{-1}AU_B^{-1} = U_B(B^{-1}A)U_B^{-1}, \\ B = L_BU_B, \quad L_B = (G - L)G^{-1/2}, \quad U_B = G^{-1/2}(G - U), \end{gathered}$$

and has, presumably, the better condition number, as compared to the original matrix A. An advantage of formulation (13), (10) is due to the law computational costs needed for the matrix-vector multiplication

$$\bar{A}v = (I - \bar{L})^{-1}[v - (2I - \bar{D})w] + w, \quad w = (I - \bar{U})^{-1}v, \tag{15}$$

which is realized by a cheap solution of two auxiliary systems with the low and the upper triangular matrices.

So, implementation of the preconditioned biconjugate gradient method for solving the original system (10) can be done by "usual" BiCG formulas for the new (preconditioned) system (13):

$$\begin{aligned}
&r^0 = \bar{f} - \bar{A}\bar{u}^0, \quad \tilde{r}^0 = p^0 = \tilde{p}^0 = r^0, \\
&\sigma_n = (\bar{A}p^n, \tilde{p}^n), \quad \rho_n = (r^n, \tilde{r}^n), \\
&\alpha_n = \frac{\rho_n}{\sigma_n}, \quad \bar{u}^{n+1} = \bar{u}^n + \alpha_n p^n, \\
&r^{n+1} = r^n - \alpha_n \bar{A}p^n, \quad \tilde{r}^{n+1} = \tilde{r}^n - \alpha_n \bar{A}^t \tilde{p}^n, \\
&p^{n+1} = r^{n+1} + \beta_n p^n, \quad \tilde{p}^{n+1} = \tilde{r}^{n+1} + \beta_n \tilde{p}^n, \\
&\beta_n = \frac{(r^{n+1}, \tilde{r}^{n+1})}{(r^n, \tilde{r}^n)}, \quad n = 0, 1, 2, \dots .
\end{aligned} \tag{16}$$

In principle, the initial vectors $\tilde{r}^0 = \tilde{p}^0$ can be defined in an arbitrary way.

Here r^0 and $\tilde{r}^0$ are "real" and "virtual" residual vectors, p^n and $\tilde{p}^n$ are the respective correction vectors. Multiplication by the transposed matrix $\bar{A}^t$ is made by similar to (15) formulas:

$$\bar{A}^t v = (I - \bar{U}^t)^{-1}[v + (2I - \bar{D})w] + w, \quad w = (I - \bar{L}^t)^{-1}v. \tag{17}$$

In iterative process (16), under the conditions $\alpha_n > 0$, $n = 0, 1, \dots$, the calculated vectors satisfy the orthogonality properties

$$(r^n, \tilde{r}^k) = (\bar{A}p^n, \tilde{p}^k) = (\bar{A}^t p^n, \tilde{p}^k) = 0 \quad \text{for} \quad k \neq n. \tag{18}$$

These vectors can be defined by the matrix polynomials

$$r^n = \psi_n(\bar{A})r^0, \quad p^n = \varphi_n(\bar{A})r^0, \quad \tilde{r}^n = \psi_n(\bar{A}^t)\tilde{r}^0, \quad \tilde{p}^n = \psi_n(\bar{A}^t)\tilde{r}^0, \tag{19}$$

which satisfy the normalization conditions

$$\psi_n(0) = \psi_0(\xi) = \varphi_0(\xi) = 1, \quad \varphi_n(0) = n + 1. \tag{20}$$

From the polynomials $\varphi_n(\xi)$, $\psi_n(\xi)$, the following squared polynomials can be defined:

$$\Phi_n(\xi) = \psi_n^2(\xi), \quad \Psi_n(\xi) = \varphi_n^2(\xi), \quad X_n(\xi) = \psi_n(\xi)\varphi_{n-1}(\xi), \tag{21}$$

which satisfy the recurrent relations

$$Y_n = \Phi_n + \beta_n X_n, \quad \Psi_n = Y_n + \beta_n(X_n + \beta_n \Psi_{n-1}),$$
$$X_{n+1} = E_n - \alpha_n \xi \Psi_n, \quad \Phi_{n+1} = \Phi_n - \alpha_n \xi (Y_n + X_{n+1}). \tag{22}$$

From definitions (19)–(21), we have the following properties of the inner products:

$$\rho_n = (r^n, \bar{r}^n) = (\Phi_n(\bar{A})r^0, \bar{r}^0), \quad \sigma_n = (\bar{A}p^n, \bar{p}^n) = (\bar{A}\Psi_n(\bar{A})r^0, \bar{r}^0). \tag{23}$$

We can now define the vectors

$$r^n = \Phi_n(\bar{A})r^0, \quad p^n = \Psi_n(\bar{A})y^0, \quad v^n = X_n(\bar{A})r^0. \tag{24}$$

For their calculation from relations (21)–(23), the following preconditioned conjugate gradient squared (CGS) algorithm is obtained:

$$\begin{aligned}
&r^0 = \bar{f} - \bar{A}\bar{u}^0, \quad p^0 = w^0 = r^0, \\
&\sigma_n = (r^0, \bar{A}p^n), \quad \rho_n = (r^0, r^n), \quad \alpha_n = \frac{\rho_n}{\sigma_n}, \\
&v^{n+1} = w^n - \alpha_n \bar{A}p^n, \quad \bar{u}^{n+1} = \bar{u}^n + \alpha_n(w^n + v^{n+1}), \\
&r^{n+1} = r^n - \alpha_n \bar{A}(w^n + v^{n+1}), \quad \beta_{n+1} = (r^0, r^{n+1})/(r^0, r^n), \\
&w^{n+1} = r^{n+1} + \beta_{n+1} v^{n+1}, \quad p^{n+1} = w^{n+1} + \beta_{n+1}(v^{n+1} + \beta_{n+1} p^n).
\end{aligned} \tag{25}$$

This algorithm is transpose free, i.e., instead of multiplication by $\bar{A}^t$ in the BiCG, the second multiplication by $\bar{A}$ is presented in (27). The computational costs of both methods in terms of implementation of one iteration, for the considered SLAE are approximately the same.

Estimation of the iterative convergence rate for solution to nonsymmetric indefinite SLAEs by the BiCG and the CGS is open to questions. But if the BiCG is convergent, i.e.,

$$|\psi_n(\xi)| \le 1 - \delta_1, \quad \delta_1 \ll 1, \quad (r^n, r^n)^{1/2} \le (1 - \delta_1)(r^0, r^0)^{1/2},$$

then, at least, for symmetric positive definite (s.p.d.) matrices, the CGS has the twice better convergence rate, in the sense,

$$|\Phi_n(\xi)| = |\psi_n^2(\xi)| \le 1 - \delta_2, \quad \delta_2 \approx 2\delta_1, \quad (r^n, r^n)^{1/2} \le (1 - \delta_2)(r^0, r^0)^{1/2}.$$

A well-known drawback of the two considered methods is numerical instability which is the reason for development of various stabilized versions of algorithms, the BiCGStab, for example.

Let us use the multifold restarted form of iterative processes, by the following approach. If an algorithm fails due to a certain circumstance, then the recurrent computation of the vectors r^n, p^n and others are interrupted, and iterations follow from the recalculation of a current residual from the equation directly, $r^n = f - \bar{A}\bar{u}^n$, as $\bar{u}^n$ being the initial value. Such restart conditions are

$$\sigma_n \le \sigma_{\min}, \quad \rho_n < \rho_{\min}, \tag{26}$$

where $\sigma_{\min} = \rho_{\min} = 0$ for example, and σ_n, ρ_n are defined in (16), (25).

The other two restarts are motivated by the observation that convergence rate of the conjugate gradient method decreases (at least, locally) under under the conditions

$$\alpha_n < \alpha_{\min} \ll 1, \quad \beta_n > \beta_{\max} > 1. \tag{27}$$

So, implementation of the BiCG or the CGS iterative process, with four types of restarts, can be described by the following pseudocode:

```
n = 0
der = 1
restart:
r^0 = ..., p^0 = ...
if n = 0 then res = (r^0, r^0)^1/2
while (der ≥ eps & n < n_max)
   n = n + 1
   Beginning of iteration
   ...
   if σ_n < σ_min goto restart
   ...
   if σ_n ≤ σ_min goto restart
   ...
   if α_n < α_min goto restart
   ...
   if β_n > β_max goto restart
   ...
   End of iteration
   der = (r^n, r^n)^1/2 / res
end while
```

Here eps $= \varepsilon \ll 1$ is the iterative tolerance (stopping criteria, $(r^n, r^n)/(r^0, r^0) \le \varepsilon^2$), $n_{\max} \gg 1$ is the admissible number of iterations. The recommended values of six user parameters are the following:

$$\varepsilon = 10^{-6}, \quad n_{\max} = 1000, \quad \sigma_{\min} = \rho_{\min} = 0, \quad \alpha_{\min} = 0.05, \quad \beta_{\max} = 0.99.$$

However, for specific classes of the SLAE, these values can be optimized from the numerical experiments in terms of minimizing the number of iterations.

4. Description of the code

The presented algorithms are implemented into the two subroutines BiCG and CGS. The programming language is FORTAN. All arithmetic operations are implemented with double precision only. These subroutines use a special row sparse format for saving non-zero entries D, U, and L, where U is the upper triangular part of the matrix A and L is the low triangular part of the matrix $A = \{a_{i,j}\}$, because the matrix A is nonsymmetric. So, for each i-th row of the entries of the matrix U, the number `NE(`i`)` of the non-zero entries $a_{i,j}$ for $j > i$, their corresponding column numbers j and the real eigenvalues $a_{i,j}$ are given in the arrays `NEIB(NU)` and `AU(NU)`, where the `NU` is the total number of non-zero entries in the matrix U; for each i-th row of the entries of the matrix L, the number `NL(`i`)` of the non-zero entries $a_{i,j}$ for $j < i$, their corresponding column numbers j and the real eigenvalues $a_{i,j}$, are given in the arrays `NEIL(NL)` and `AL(NL)`, where the `NL` is the total number of non-zero entries in the matrix L. The iterations proceed until the convergence criteria $(r^n, r^n)/(r^0, r^0) \leq \varepsilon^2$ for a given tolerance or the condition `n=nmax` holds, where nmax is the given number. The call of the subroutines is identical for both cases, except its name, and has the form

```
call CGS(D, U, F, NE, N, NEIB, AU, NU, NEL, NEIL, AL, NL, EPS,
         NMAX, THETA, OM, amin, bmax)
```

Here the arguments are:

`N` – the order of the system $Au = f$;

`NU` – the number of non-zero entries of U – the upper triangular part of the matrix A;

`NL` – the number of non-zero entries of L – the low triangular part of the matrix A;

`EPS` – accuracy of an iterative solution (tolerance);

`NMAX` – a maximum number of iterations (input) and the resulting number of iterations (output);

`U(N)` – an initial value of solution (input) and the resulting solution (output);

`F(N)` – the right-hand side (input) and the preconditioned residual (output);

`D(N)` – an array of diagonal entries of the matrix A (input), the entries of the matrix (output);

`NE(N)` – the number of non-zero entries into the rows of the matrix U;

`NEL(N)` – the number of non-zero entries into the rows of the matrix L;

`NEIB(NU)` – contains a column of non-zero entries of U – the upper triangular part of the matrix A,

`NEIL(NU)` – contains a column of non-zero entries of L – the low triangular part of the matrix A;

`AU(NU)` – the values of non-zero entries of U (input), the corresponding entries of the matrix $\bar{U}$ (output);

`AL(NU)` – the values of non-zero entries of L (input), the corresponding entries of the matrix $\bar{L}$(output);

`THETA` – the compensation parameter;

`OM` – the relaxation parameter;

`EPS` – the accuracy of an iterative solution;

`NMAX` – a maximum number of iterations (input) and the resulting number of iterations (output);

`amin` – a minimum value of α_n for restarting;

`bmax` – a maximum value of β_n for restarting.

The subroutine uses, in addition, auxiliary real arrays `u1(n)`, `ap(n)`, `g(n)`, `p(n)`. The general structure and building the blocks of subroutines can be described by the following scheme:

1. Computing the matrix G by formulas (12) (call the auxiliary subroutine PRENS);
2. Implementation of the Eisenstat matrix-vector transformations (14);
3. Calculation of the initial residual by means of (15) and (for the BiCG) – the vector $\tilde{r}$;
4. Realization of the iterative process by formulas (16),(25);
5. Reconstruction of solution by formula (14).

The computational resources of the considered subroutines are defined by the number Q of multiplication at each iteration and the volume of the necessary operative memory P:

- for CGS: $P = 19N$ and $Q = 21N$;
- for BiCG: $P = 17N$ and $Q = 21N$.

The robustness of subroutines is provided by the full control of a possible division by zero and positive definiteness of the matrix.

5. Results of numerical experiments

In this section, we present and discuss the results of numerical experiments on the solution of SLAE (3), and the corresponding 5-point systems for a $2D$ case, by preconditioned restarted BiCG and CGS algorithms for the three considered types of approximation of diffusion-convection equation (1) with different constant coefficients $p = q = r = c$. In all the cases, the Dirichlet model BVP in the unit cube (and the unit square in the $2D$ case) with exact solution $u = 1$ is chosen. An initial value is defined as function $u^0(x, y, z) = x^2 + y^2 + z^2$ and the tolerance is $\varepsilon = 10^{-6}$. The iterative parameters in preconditioner (11) are $\omega = \theta = 1$ and the restart parameters are $\alpha_{\min} = 0.05$, $\beta_{\max} = 0.9$. The number of iterations n ($\varepsilon = 10^{-6}$) are presented for different cubic grids with the numbers of mesh steps $N = I + 1 = J + 1 = K + 1$. All computations are made with double precision.

In Table 1, the results of the BiCG method for CD scheme (4), OS scheme (5), and ET scheme (6) are presented for the 3D BVPs. The number of restarts are given in the parentheses. The symbol $G < 0$ implies that in this case the preconditioning matrix has negative diagonal entries and algorithm fails. It is a natural phenomenon for one-side difference scheme with a "bad" convective coefficient. Similar data on the solution of two-dimensional BVPs by the restarted BiCG method for three types of difference approximations are presented in Table 2.

The rest tables consist of the same results for the preconditioned and the restarted CGS methods. In Table 3 and Table 4, the data for 3D BVP and 2D BVP are respectively presented.

Table 1. BiCG method, 3D BVP

N \ c	−1024	−256	−64	16	−4	0	4	16	64	256	1024
					Central difference scheme						
8	463 (262)	176 (108)	22 (21)	7 (5)	6 (3)	9	6 (2)	7 (6)	19 (15)	51 (47)	115 (102)
16	95 (89)	36 (34)	18 (13)	1 (1)	13 (4)	14	11 (3)	1 (1)	15 (10)	35 (27)	97 (86)
32	71 (59)	24 (23)	9 (7)	12 (6)	18 (7)	20	16 (4)	8 (5)	8 (7)	22 (19)	60 (49)
64	45 (44)	15 (14)	1 (1)	16 (11)	25 (9)	29	22 (4)	16 (9)	1 (1)	11 (11)	34 (34)
					Right side difference scheme						
8	$G < 0$	$G < 0$	$G < 0$	$G < 0$	9 (2)	9	8 (2)	6 (2)	5 (3)	3 (2)	3 (2)
16	$G < 0$	$G < 0$	$G < 0$	1 (1)	13 (2)	14	12 (2)	9 (4)	6 (4)	4 (3)	3 (2)
32	$G < 0$	$G < 0$	$G < 0$	12 (6)	20 (4)	20	19 (3)	14 (5)	8 (5)	5 (4)	3 (2)
64	$G < 0$	$G < 0$	1 (1)	18 (10)	27 (4)	29	26 (3)	20 (7)	11 (6)	5 (2)	3 (2)
					Exponential type scheme						
8	1 (1)	1 (0)	2 (0)	5 (2)	8 (0)	9	8 (2)	6 (3)	2 (2)	1 (1)	1 (1)
16	1 (1)	1 (0)	3 (1)	8 (3)	14 (2)	14	13 (3)	8 (5)	3 (2)	1 (1)	1 (1)
32	1 (1)	2 (0)	5 (4)	15 (8)	18 (2)	20	20 (2)	14 (6)	5 (4)	2 (2)	1 (1)
64	1 (0)	3 (2)	8 (5)	23 (11)	27 (4)	29	26 (3)	20 (8)	8 (4)	3 (2)	1 (1)

Table 2. BiCG method, 2D BVP

N \ c	−1024	−256	−64	16	−4	0	4	16	64	256	1024
					Central difference scheme						
8	33 (31)	29 (29)	20 (18)	7 (5)	7 (2)	7 (0)	6 (3)	6 (5)	18 (15)	29 (26)	34 (28)
16	52 (51)	33 (31)	14 (10)	1 (1)	10 (4)	11 (0)	10 (3)	1 (1)	11 (9)	30 (24)	48 (48)
32	59 (53)	24 (20)	8 (6)	7 (3)	17 (6)	15 (0)	16 (4)	7 (3)	7 (6)	18 (16)	53 (47)
64	36 (35)	13 (10)	1 (0)	12 (7)	20 (5)	22 (0)	32 (4)	14 (6)	1 (1)	10 (9)	29 (28)
					Right side difference scheme						
8	$G<0$	$G<0$	$G<0$	$G<0$	7 (2)	7 (0)	7 (0)	6 (3)	5 (1)	4 (2)	3 (2)
16	$G<0$	$G<0$	$G<0$	1 (1)	12 (2)	11 (0)	13 (3)	9 (3)	5 (3)	4 (3)	3 (2)
32	$G<0$	$G<0$	$G<0$	11 (5)	16 (2)	15 (0)	16 (2)	14 (5)	7 (3)	4 (3)	3 (2)
64	$G<0$	$G<0$	1 (1)	23 (9)	25 (5)	22 (0)	27 (4)	16 (6)	11 (4)	5 (2)	3 (1)
					Exponential type scheme						
8	1 (1)	1 (0)	2 (0)	5 (3)	7 (0)	7 (0)	6 (0)	5 (1)	2 (1)	1 (0)	1 (1)
16	1 (1)	1 (0)	3 (2)	9 (4)	13 (2)	11 (0)	10 (2)	9 (3)	3 (2)	1 (1)	1 (1)
32	1 (0)	2 (0)	6 (3)	12 (5)	17 (2)	15 (0)	19 (3)	13 (5)	4 (3)	2 (1)	1 (0)
64	1 (0)	3 (2)	7 (4)	23 (9)	26 (5)	22 (0)	28 (4)	18 (7)	7 (3)	2 (1)	1 (1)

Table 3. CGS method, 3D BVP

N \ c	−1024	−256	−64	16	−4	0	4	16	64	256	1024
					Central difference scheme						
8	101 (65)	42 (31)	13 (12)	4 (2)	4 (2)	5 (0)	3 (1)	4 (4)	9 (9)	35 (27)	92 (61)
16	47 (43)	18 (17)	7 (6)	1 (1)	5 (3)	9 (0)	6 (2)	1 (1)	6 (6)	15 (15)	43 (42)
32	32 (30)	13 (12)	5 (3)	5 (3)	10 (4)	14 (0)	8 (3)	4 (2)	4 (3)	10 (10)	26 (26)
64	24 (23)	8 (7)	1 (1)	7 (5)	13 (5)	22 (0)	12 (3)	7 (5)	1 (1)	6 (6)	18 (18)
					Right side difference scheme						
8	$G<0$	$G<0$	$G<0$	$G<0$	5 (2)	5 (0)	5 (2)	3 (1)	3 (2)	2 (2)	2 (2)
16	$G<0$	$G<0$	$G<0$	1 (1)	7 (2)	9 (0)	8 (3)	5 (2)	3 (1)	2 (1)	2 (1)
32	$G<0$	$G<0$	$G<0$	5 (3)	12 (4)	14 (0)	10 (3)	7 (3)	4 (2)	3 (2)	2 (1)
64	$G<0$	$G<0$	1 (1)	14 (8)	16 (7)	22 (0)	14 (3)	11 (5)	6 (4)	3 (1)	2 (1)
					Exponential type scheme						
8	1 (0)	1 (1)	1 (0)	3 (1)	5 (0)	5 (0)	5 (2)	3 (2)	1 (1)	1 (1)	1 (1)
16	1 (0)	1 (0)	2 (0)	4 (3)	9 (4)	9 (0)	8 (3)	4 (2)	2 (1)	1 (1)	1 (1)
32	1 (1)	1 (0)	3 (2)	6 (4)	12 (4)	14 (0)	10 (3)	6 (3)	3 (2)	1 (1)	1 (1)
64	1 (0)	2 (1)	4 (3)	12 (7)	15 (6)	22 (0)	15 (3)	12 (6)	5 (3)	2 (1)	1 (1)

From the above-considered results of numerical experiments we can make the following conclusions:

- For all reasonable values of convective coefficients and all grids (except the cases when $G < 0$), the preconditioned restarted BiCG and CGS methods have a sufficient good iterative convergence rate;

Table 4. CGS method, 2D BVP

N \ c	−1024	−256	−64	16	−4	0	4	16	64	256	1024
					Central difference scheme						
8	18 (17)	17 (17)	11 (10)	4 (3)	4 (2)	4 (0)	4 (2)	3 (3)	8 (8)	16 (15)	21 (20)
16	26 (26)	16 (15)	7 (5)	1 (1)	5 (3)	6 (0)	6 (2)	1 (1)	6 (5)	14 (14)	26 (26)
32	29 (28)	11 (9)	4 (3)	4 (3)	7 (3)	10 (0)	7 (2)	4 (1)	4 (3)	9 (9)	24 (24)
64	19 (18)	7 (4)	1 (1)	9 (5)	10 (3)	16 (0)	10 (3)	6 (2)	1 (1)	6 (5)	15 (15)
					Right side difference scheme						
8	$G<0$	$G<0$	$G<0$	$G<0$	5 (2)	4 (0)	4 (0)	4 (2)	3 (1)	3 (1)	2 (1)
16	$G<0$	$G<0$	$G<0$	1 (0)	6 (2)	6 (0)	7 (2)	4 (1)	3 (1)	2 (1)	2 (1)
32	$G<0$	$G<0$	$G<0$	6 (4)	10 (3)	10 (0)	10 (2)	6 (2)	4 (1)	2 (1)	2 (1)
64	$G<0$	$G<0$	1 (0)	9 (5)	12 (3)	16 (0)	14 (3)	10 (4)	5 (1)	3 (1)	2 (1)
					Exponential type scheme						
8	1 (0)	1 (1)	1 (0)	3 (1)	4 (0)	4 (0)	4 (0)	3 (1)	1 (1)	1 (1)	1 (1)
16	1 (0)	1 (0)	2 (1)	5 (3)	7 (2)	6 (0)	7 (2)	4 (2)	2 (1)	1 (1)	1 (1)
32	1 (1)	1 (0)	3 (2)	7 (4)	14 (5)	10 (0)	12 (3)	6 (3)	2 (1)	1 (1)	1 (1)
64	1 (0)	2 (1)	4 (3)	10 (4)	12 (3)	16 (0)	14 (3)	10 (4)	4 (1)	1 (1)	1 (1)

- The exponential type scheme has a significant advantage, as compared to the rest schemes, for the large convective coefficients of both signs, in terms of the number of iterations;
- The number of iterations for 2D and 3D BVPs are of the same order for different grids and algorithms;
- The CGS algorithm has some advantage as compared to the BiCG method, in terms of the number of iterations.

In conclusion, we can say that both considered algorithms provide a fast iterative solution of the diffusion-convection difference algebraic systems of equations.

References

[1] Saad Y. Iterative Methods for Sparse Linear Systems. – New-York: PWS Publishing Co., 1996.

[2] Samarski A.A., Vabishchevich P.N. Numerical Methods for Solution of Convection-Diffusion Problems. – Moscow: Editorial UPSS Publ., 1999 (in Russian).

[3] Il'in V.P. Finite Difference and Finite Volume Methods for Elliptic Equations. – Novosibirsk: IM SB RAS Publ., 2000 (in Russian).

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

[5] Il'in V.P., Itskovich E.A. Two explicit incomplete factorization methods // NCC Bulletin. Series Num. Anal. – Novosibirsk: NCC Publisher, 2002. – Issue 11. – P. 51–60.