Parallel Algorithms for Solving Linear Systems with Block-Fivediagonal Matrices on Multi-Core CPU

Elena Akimova* and Dmitry Belousov

[1] Krasovskii Institute of Mathematics and Mechanics, Yekaterinburg, Russia
[2] Ural Federal University, Yekaterinburg, Russia

Abstract. For solving systems of linear algebraic equations with block-fivediagonal matrices arising in geoelectrics and diffusion problems, the parallel matrix square root method, conjugate gradient method with preconditioner, conjugate gradient method with regularization, and parallel matrix sweep algorithm are proposed and some of them are implemented numerically on multi-core CPU Intel. Investigation of efficiency and optimization of parallel algorithms for solving the problem with quasi-model data are performed. The problem with quasi-model data is solved.

Keywords: parallel algorithms · block-fivediagonal SLAE · direct and iterative numerical methods · multi-core CPU

1 Introduction

Systems of linear algebraic equations (SLAE) with block-fivediagonal matrices arise when solving some mathematical modelling problems, in particular, geoelectrics and diffusion problems. The geoelectrics problems are very important for investigating the crust heterogeneity. This problem is described by a differential Poissons equation. After using a finite-difference approximation, the two- and three-dimensional lateral logging problems are reduced to solving ill-conditioned systems having large-scale block-five diagonal matrix [1]. Another important problem is the multicomponent diffusion problem when it is necessary to know concentration distribution of diffusing components at every moment of time. This problem is also reduced to solving a SLAE with block-fivediagonal matrix after approximating systems of differential equations [2]. These types of problems has the matrix with a special structure, we also describe.

In this paper, for solving SLAE with block-fivediagonal matrices, direct and iterative parallel algorithms are designed: conjugate gradient method with preconditioner (PCGM), conjugate gradient method with regularization (PCGR), square root method (PSRM), and parallel matrix sweep algorithm (PMSA).

* This work was partially supported by Program of UrB RAS (project no. 15-7-1-13) and by the Russian Foundation for Basic Research (project no. 15-01-00629 a).

The PCGM, PCGR, and PSRM algorithms are implemented on multi-core Intel processor. The parallel matrix sweep algorithm will be implemented on multi-core Intel processor later.

Investigation of efficiency and optimization of parallel algorithms are performed. The problem with quasi-model data is solved.

2 Parallel Algorithms for Solving SLAE

We consider the system:

$$\begin{cases} C_0\bar{Y}_0 - D_0\bar{Y}_1 + E_0\bar{Y}_2 = \bar{F}_0, \\ -B_1\bar{Y}_0 + C_1\bar{Y}_1 - D_1\bar{Y}_2 + E_1\bar{Y}_3 = \bar{F}_1, \\ A_i\bar{Y}_{i-2} - B_i\bar{Y}_{i-1} + C_i\bar{Y}_i - D_i\bar{Y}_{i+1} + E_i\bar{Y}_{i+2} = \bar{F}_i, \quad i = 2, ..., N-1 \ ; \\ A_N\bar{Y}_{N-2} - B_N\bar{Y}_{N-1} + C_N\bar{Y}_N - D_N\bar{Y}_{N+1} = \bar{F}_N, \\ A_{N+1}\bar{Y}_{N-1} - B_{N+1}\bar{Y}_N + C_{N+1}\bar{Y}_{N+1} = \bar{F}_{N+1}, \end{cases} \quad (1)$$

where $\bar{Y}_i$ are unknown n–vectors, $\bar{F}_i$ are given n–vectors, A_i, B_i, C_i, D_i, E_i are given $n \times n$ – matrices.

As mentioned in introduction, after a finite-difference approximation, the geoelectrics problems and diffusion problems can be reduced to solving SLAE with block-fivediagonal matrices with the structure presented in Fig. 1. This special case arising when we are solving diffusion problems for higher order scheme and geoelectric problems.

Figure 1 show the special case of SLAE (1) where central diagonal blocks have three non zero diagonals.

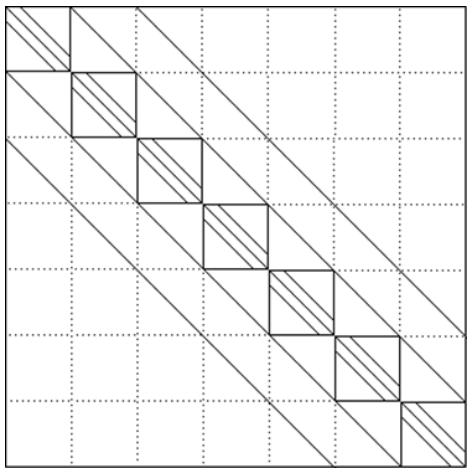

Fig. 1. Block-fivediagonal matrix

For solving SLAE with block-fivediagonal matrices, parallel algorithms based on conjugate gradient method with preconditioner (PCGM), conjugate gradient

method with regularization (PCGR), parallel square root method (PSRM), and parallel matrix sweep algorithm (PMSA) are proposed.

2.1 Parallel Conjugate Gradient Method with Regularization

One of the fast iterative algorithms for solving a SLAE with a symmetric positive definite matrix is the conjugate gradient method (CGM) [3].

We write the system (1) in the form

$$Ax = b, \tag{2}$$

When we are solve geoelectric problems with non-uniformly scaled coefficients and when the system is ill-conditioned, we consider a regularization to prevent effect of rounding errors. We add regularization with alpha parameter of regularization to ensure stability of the method with the following equation:

$$\widetilde{A}x = b, \ \widetilde{A} = A + \alpha E, \tag{3}$$

where α is the parameter of regularization.

The CGM method has the following form:

$$\begin{aligned} &r^0 = b - Ax^0, \qquad p^0 = r^0, \\ &x^{k+1} = x^k + \alpha_k p^k, \ \ \alpha_k = \frac{\left\|r^k\right\|^2}{(Ap^k, p^k)}, \quad r^{k+1} = r^k - \alpha_k \widetilde{A} p^k, \\ &p^{k+1} = r^{k+1} + \beta_k p^k, \ \ \beta_k = \frac{\left\|r^{k+1}\right\|^2}{\left\|r^k\right\|^2}. \end{aligned} \tag{4}$$

The condition for stopping the CGM iterative process is $\left\|Ar^k - b\right\| / \|b\| < \varepsilon$.

2.2 Parallel Conjugate Gradient Method with Preconditioner

One of the fast iterative algorithms for solving a SLAE with a symmetric positive definite matrix is the conjugate gradient method (CGM). The introduction of a preconditioner is applied to accelerate the convergence of the iterative process. Preconditioning consists in the fact that the initial system of equations $Ax = b$ is replaced by the system

$$C^{-1}Ax = C^{-1}b, \tag{5}$$

for which the iterative method converges essentially faster. The condition for the choice of the preconditioner C is the following:

$$cond(\tilde{A}) << cond(A), \quad cond(\tilde{A}) = \frac{\tilde{\lambda}_{\max}}{\tilde{\lambda}_{\min}}, \quad cond(A) = \frac{\lambda_{\max}}{\lambda_{\min}}, \tag{6}$$

where $cond(A)$ and $cond(\tilde{A})$ are the condition numbers of the matrices A and $\tilde{A}$; $\lambda_{\max}$, $\tilde{\lambda}_{\max}$ and $\lambda_{\min}$, $\tilde{\lambda}_{\min}$ are the largest and smallest eigenvalues of the matrices A and $\tilde{A}$, respectively.

For system of equations (1), the conjugate gradient method with preconditioner C has the form:

$$\begin{aligned} &r^0 = b - Ax^0, \quad p^0 = C^{-1}r^0, \quad z^0 = p^0, \\ &x^{k+1} = x^k + \alpha_k p^k, \quad \alpha_k = \frac{(r^k, z^k)}{(Ap^k, p^k)}, \quad r^{k+1} = r^k - \alpha_k A p^k, \\ &z^{k+1} = C^{-1}r^{k+1}, \quad p^{k+1} = z^{k+1} + \beta_k p^k, \quad \beta_k = \frac{(r^{k+1}, z^{k+1})}{(r^k, z^k)}. \end{aligned} \tag{7}$$

The condition for stopping the CGM iterative process with preconditioner is $\left\|Az^k - b\right\| / \|b\| < \varepsilon$.

Parallelization of the PCGR and PCGM is based on splitting the matrix A by horizontal lines into m blocks, and the solution vector z and the vector of the right-hand part $b$of the SLAE are divided into m parts such that $n = m \times L$, where n is the dimension of the system of equations, m is the number of processors, and $L$is the number of lines in the block (Fig. 2). At every iteration, each processor computes its part of the solution vector. In the case of the matrix-vector multiplication (A by z), each processor multiplies its part of the lines of the matrix A by the vector z. The host-CPU (master) is responsible for transferring data and, also, calculates its part of the solution vector.

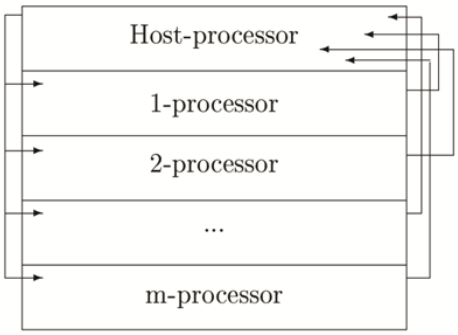

Fig. 2. Distribution of data among processors

2.3 Parallel Square Root Method

For solving SLAE (1), it is also possible to use the square root method [5]. This method is based on decomposition of the symmetric positive definite matrix A into the product $A = S^T S$, where S is an upper triangular matrix with positive elements on the main diagonal, $S^T$is the transposed matrix. The square root method consists in the sequential solution of two systems of equations with the triangular matrices

$$S^T y = b, \quad Sz = y. \tag{8}$$

To solve systems (8), we use the recursion formulas

$$\begin{cases} y_1 = b_1/s_{11}, \\ y_i = \dfrac{b_i - \sum\limits_{k=1}^{i-1} s_{ki} y_k}{s_{ii}}, \\ i = 2, 3, ..., n\,; \end{cases} \qquad \begin{cases} z_n = y_n/s_{nn}, \\ z_i = \dfrac{y_i - \sum\limits_{k=i+1}^{n} s_{ik} z_k}{s_{ii}}, \\ i = n-1, n-2, ..., 1. \end{cases} \tag{9}$$

The main idea of parallelization of the square root method for multiprocessor computers with shared memory is based on parallel computing the elements s_{ij},$j = i, ..., n$, of each i-th line of the matrix S. The i-th line is divided into m parts so that $n - i = m \times L_i$, where i is the line number, n is the dimension of the system, m is the number of processors, and L_i is the number of line items that are evaluated by each processor (Fig. 3) described in [4].

1 – Processor	2 – processor	…	m – processor

Fig. 3. Distribution of the i-th line among processors

2.4 Constructing the Parallel Matrix Sweep Algorithm

Let us consider system (1). When constructing the parallel algorithm, we choose the unknown vectors $\bar{Y}_K$, $\bar{Y}_{K+1}$, $K = 0, M, ..., N$, as parameters connecting vertically the unknown values on the grid in Lsubdomains (Fig. 4) described in [5].

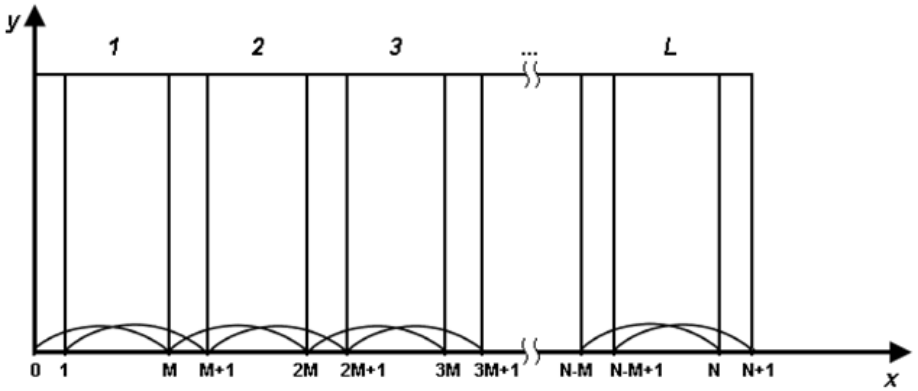

Fig. 4. Dividing the initial domain P into L subdomains

Let us introduce the operator

$$\Delta\bar{Y}_i \equiv A_i\bar{Y}_{i-2} - B_i\bar{Y}_{i-1} + C_i\bar{Y}_i - D_i\bar{Y}_{i+1} + E_i\bar{Y}_{i+2}\,.$$

Similarly to $\Delta\bar{Y}_i$, we define the operators $\Delta\bar{U}_i$, $\Delta\bar{V}_i$, $\Delta\bar{P}_i$, $\Delta\bar{Q}_i$, $\Delta\bar{W}_i$.

A reduced system of equations with respect to the parameters $\bar{Y}_K$, $\bar{Y}_{K+1}$ is constructed as follows. In L subdomains defined by the intervals $(K, K+M+1)$, $K = 0, M, ..., N-M$, we consider the following problems:

$$\begin{cases} \Delta\bar{U}_i^1 = 0,\ \bar{U}_K^1 = \begin{pmatrix}1\\0\\..\\0\end{pmatrix},\ \bar{U}_{K+1}^1 = \begin{pmatrix}0\\0\\..\\0\end{pmatrix}, \bar{U}_{K+M}^1 = \begin{pmatrix}0\\..\\0\\0\end{pmatrix}, \bar{U}_{K+M+1}^1 = \begin{pmatrix}0\\..\\0\\0\end{pmatrix}, \\ \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \\ \Delta\bar{U}_i^n = 0,\ \bar{U}_K^n = \begin{pmatrix}0\\..\\0\\1\end{pmatrix},\ \bar{U}_{K+1}^n = \begin{pmatrix}0\\..\\0\\0\end{pmatrix}, \bar{U}_{K+M}^n = \begin{pmatrix}0\\..\\0\\0\end{pmatrix}, \bar{U}_{K+M+1}^n = \begin{pmatrix}0\\..\\0\\0\end{pmatrix}, \end{cases} \tag{10}$$

$$\begin{cases} \Delta\bar{V}_i^1 = 0,\ \bar{V}_K^1 = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{V}_{K+1}^1 = \begin{pmatrix}1\\0\\..\\0\end{pmatrix},\ \bar{V}_{K+M}^1 = \begin{pmatrix}0\\..\\0\\0\end{pmatrix},\ \bar{V}_{K+M+1}^1 = \begin{pmatrix}0\\..\\0\\0\end{pmatrix}, \\ \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \\ \Delta\bar{V}_i^n = 0,\ \bar{V}_K^n = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{V}_{K+1}^n = \begin{pmatrix}0\\0\\..\\1\end{pmatrix},\ \bar{V}_{K+M}^n = \begin{pmatrix}0\\..\\0\\0\end{pmatrix},\ \bar{V}_{K+M+1}^n = \begin{pmatrix}0\\..\\0\\0\end{pmatrix}, \end{cases} \tag{11}$$

$$\begin{cases} \Delta\bar{P}_i^1 = 0,\ \bar{P}_K^1 = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{P}_{K+1}^1 = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{P}_{K+M}^1 = \begin{pmatrix}1\\..\\0\\0\end{pmatrix},\ \bar{P}_{K+M+1}^1 = \begin{pmatrix}0\\..\\0\\0\end{pmatrix}, \\ \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \\ \Delta\bar{P}_i^n = 0,\ \bar{P}_K^n = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{P}_{K+1}^n = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{P}_{K+M}^n = \begin{pmatrix}0\\..\\0\\1\end{pmatrix},\ \bar{P}_{K+M+1}^n = \begin{pmatrix}0\\..\\0\\0\end{pmatrix}, \end{cases} \tag{12}$$

$$\begin{cases} \Delta\bar{Q}_i^1 = 0,\ \bar{Q}_K^1 = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{Q}_{K+1}^1 = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{Q}_{K+M}^1 = \begin{pmatrix}0\\..\\0\\0\end{pmatrix},\ \bar{Q}_{K+M+1}^1 = \begin{pmatrix}1\\..\\0\\0\end{pmatrix}, \\ \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \\ \Delta\bar{Q}_i^n = 0,\ \bar{Q}_K^n = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{Q}_{K+1}^n = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{Q}_{K+M}^n = \begin{pmatrix}0\\..\\0\\0\end{pmatrix},\ \bar{Q}_{K+M+1}^n = \begin{pmatrix}0\\..\\0\\1\end{pmatrix}, \end{cases} \tag{13}$$

$$\Delta\bar{W}_i = \bar{F}_i,\ \bar{W}_K = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{W}_{K+1} = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{W}_{K+M} = \begin{pmatrix}0\\0\\..\\0\end{pmatrix},\ \bar{W}_{K+M+1} = \begin{pmatrix}0\\0\\..\\0\end{pmatrix}, \tag{14}$$

where $i = K + 2,\ ...,\ K + M - 1$.

The following theorem is proved in [5].

Theorem. If $\bar{U}_i^1, ..., \bar{U}_i^n$ are solutions of (10), $\bar{V}_i^1, ..., \bar{V}_i^n$ are solutions of (11), $\bar{P}_i^1, ..., \bar{P}_i^n$ are solutions of (12), $\bar{Q}_i^1, ..., \bar{Q}_i^n$ are solutions of (13), $\bar{W}_i$ are solutions of (14), and $\bar{Y}_i$ are solutions of given problem (1) on $(K, K + M + 1)$, then, by the superposition principle [5], we have

$$\begin{aligned}\bar{Y}_i = \left(\bar{U}_i^1 \bar{U}_i^2 ... \bar{U}_i^n\right) \bar{Y}_K + \left(\bar{V}_i^1 \bar{V}_i^2 ... \bar{V}_i^n\right) \bar{Y}_{K+1} + \left(\bar{P}_i^1 \bar{P}_i^2 ... \bar{P}_i^n\right) \bar{Y}_{K+M} + \\ + \left(\bar{Q}_i^1 \bar{Q}_i^2 ... \bar{Q}_i^n\right) \bar{Y}_{K+M+1} + \bar{W}_i\,.\end{aligned} \tag{15}$$

Substituting relations (15) into the given system (1) at the points $K,\ K+1$, $K = 0, M, ..., N$, we get the reduced system with respect to the vector-parameters $\bar{Y}_K,\ \bar{Y}_{K+1}$. This reduced system has a smaller dimension.

$$\tilde{A}\bar{Y} = \bar{F}. \tag{16}$$

Reduced system (16) is one of vector equations with block-sevendiagonal matrices of coefficients; in each line, one of the seven vector elements $\bar{Y}_K$ being on the left or right from the main diagonal is zero. After finding vector-parametres $\bar{Y}_K,\ \bar{Y}_{K+1}$, other required unknown vectors are expressed through vector-parameters and are found in each subdomain L independently by formula (16).

The direct parallel matrix sweep algorithm for solving vector system (1) with block-fivediagonal matrices consists in solving the following equations:

$$(10) \longrightarrow (14) \longrightarrow (16) \longrightarrow (15). \tag{17}$$

Problems (10)–(14) and system of equations (16) can be solved by the matrix sweep algorithms (Gaussian elimination methods) for solving systems of equations with block-fivediagonal and block-sevendiagonal matrices, respectively. The formulas of the matrix sweep algorithms for solving SLAEs with block-fivediagonal and block-sevendiagonal matrices are deduced similarly to formulas of the corresponding scalar sweep algorithms [6].

The stable parallel matrix sweep algorithm for solving SLAE with block-fivediagonal matrices can be effectively implemented on parallel computing systems with distributed memory with L processors (the number is equal to the number of subdomains). Problems and unknown vector-parameters $\bar{Y}_i$ ($i = K+2,\ ...,\ K+M-1$) inside of each subdomain L are computed on L processors independently. In addition, the parallel matrix sweep algorithm can be effectively implemented on multi-core processor and graphic processors.

3 Numerical Experiments

The parallel algorithms were implemented on a multi-core Intel processor using the technology of OpenMP, and the Windows API development tools.

With the help of the parallel preconditioned conjugate gradient method, and square root method, we solved the problem of finding a potential distribution in a conducting medium with known quasi-model solution.

The source data and quasi-model solution of the problem were provided by the Department of Borehole Geophysics, Institute of Geology and Geophysics, Siberian Branch of RAS (Novosibirsk).

After discretization the problem is reduced to solving a SLAE with an ill-conditioned symmetric positive defined block-fivediagonal matrix of dimension 134862×134862 with 247 square blocks.

The numerical solution of the problem is compared with the quasi-model solution by means of calculating the relative error

$$\sigma = \left\|\bar{Y}^M - \bar{Y}^N\right\| / \left\|\bar{Y}^M\right\| , \tag{18}$$

where $\bar{Y}^M$is the quasi-model solution of the problem, $\bar{Y}^N$is the numerical solution of the problem.

The condition is chosen as a stopping criterion for the iterative PCGM.

A priori we find the condition number of the original matrix:

$$cond(A) = \frac{\lambda_{\max}}{\lambda_{\min}} \approx 1.3 \cdot 10^{11}, \quad \lambda_{\max} = 1.4 \cdot 10^{6}\,, \quad \lambda_{\min} = 1.1 \cdot 10^{-5} > 0, \tag{19}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of the matrix A.

In the case of solving the problem by the preconditioned PCGM, to verify conditions (11), we find the condition number of matrix $\tilde{A}$

$$cond(\tilde{A}) = \frac{\tilde{\lambda}_{\max}}{\tilde{\lambda}_{\min}} \approx 4.1 \cdot 10^{9} < cond(A). \tag{20}$$

This problem was solved by the parallel conjugate gradient method with preconditioner, parallel conjugate gradient method with regularization, and parallel square root method. The numerical solution of the problem coincides with the quasi-model solution with accuracy

$$\sigma_{PCGM} \approx 10^{-7},\ \sigma_{PMGR} \approx 2 \cdot 10^{-7},\ \sigma_{PSRM} \approx 6 \cdot 10^{-7}.$$

For the quasi-model data the numerical solution of the problem is presented in Fig. 5.

The computation times of solving the SLAE in the potential distribution problem on the hybrid computing system are presented in Table 1. This system is installed in the Department of Ill-posed Problems of Analysis and Applications of the Institute of Mathematics and Mechanics UrB RAS. The computing system consists of 4-core processor Intel Core I5-750.

Note that the time for solving the problem by PCGM without preconditioner on one core Intel Core I5-750 for $\sigma_{PCGM} = 10^{-3}$ was 30 minutes.

Table 1 shows that the preconditioner decreases essentially the time of solving the problem.

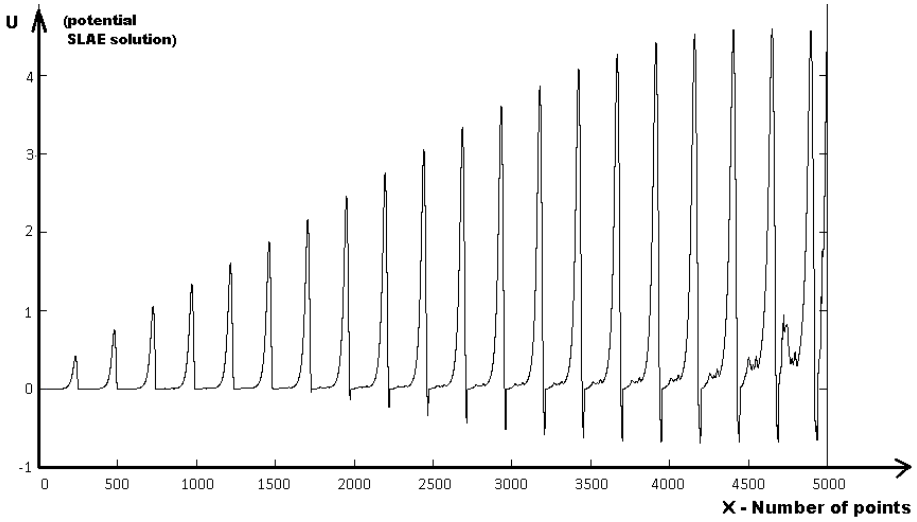

Fig. 5. Numerical solution of the problem

Table 1. Computation times of solving SLAE

Method	Number of cores Intel Core I5	T_m, sec. Windows API	T_m, sec. OpenMP
PCGM	1	165	145
	2	142	127
	4	118	101
PCGR	1	1805	1804
	2	1490	1450
	4	1288	1230
PSRM	1	26	18
	2	18	13
	4	9	7

At the beginning, for a multi-core processor with shared memory, the PCGM, PCGR, and PSRM parallelization was implemented by means of the operating system (OS) threads by the development tools Windows API. For parallel implementation of the computing block of the program, the threads were created. Each of these threads executed on the OS "logical processor" and computed the data portion. At the end of each computing block, the barrier synchronization of the threads was perfomed.

In order to optimize the program and reduce the computation time, the OpenMP technology was used. Automatic parallelization of loops was carried out by the OpenMP library using special compiler directives.

The interval of size L of the loop variable i was divided into m parts. Each thread of the process calculated its p-th part of the data, where $p = L/m$ (Fig. 4).

So, the PSRM is the fastest method. The computation time for solving the SLAE on a 4-core CPU Intel is reduced to several seconds (in comparison with 30 min by CGM without preconditioner).

4 Conclusion

For solving SLAE with block-fivediagonal matrices arising in geoelectrics and diffusion problems, the parallel conjugate gradient method and parallel square root method with preconditioner are proposed and numerically implemented on a multi-core processor Intel. Investigation of efficiency and optimization of parallel algorithms for solving the problem with quasi-model data are performed.

The calculation results show that the use of parallel PCGR, PSRM, and PCGM with preconditioner allows us to solve rather effectively problems with ill-conditioned matrices on multi-core CPU.

For the future work we want to compare parallel matrix sweep algorithm with others and to implement and optimize for GPU.

References

1. Dashevsky, J.A., Surodina, I.V., Epov, M.I.: Quasi-three-dimensional mathematical modelling of diagrams of axisymmetric direct current probes in anisotropic profiles. Siberian J. of Industrial Mathematics. Vol. 5, No. 3, 76–91 (2002)
2. Gorbachev, I.I., Popov V.V., Akimova, E.N.: Computer simulation of the diffusion interaction between carbonitride precipitates and austenitic matrix with allowance for the possibility of variation of their composition, The Physics of Metals and Metallography. Vol. 102, No. 1, 18–28 (2006). http://link.springer.com/article/10.1134/S0031918X06070039
3. Faddeev, V.K., Faddeeva, V.N.: Computational methods of linear algebra. M. Gos. Isdat. Fizmat. Lit. (1963)
4. Akimova, E.N., Belousov, D.V.: Parallel algorithms for solving linear systems with block-tridiagonal matrices on multi-core CPU with GPU, Journal of Computational Science. Vol. 3, No. 6, 445–449 (2012). http://www.sciencedirect.com/science/article/pii/S1877750312000932
5. Akimova, E.N.: A parallel matrix sweep algorithm for solving linear system with block-fivediagonal matrices. AIP Conf. Proc. 1648, 850028. 2015. Rhodes, Greece, 22–28 Sept. (2014). http://scitation.aip.org/content/aip/proceeding/aipcp/10.1063/1.4913083
6. Samarskii, A.A., Nikolaev, E.S.: Methods for solving the grid equations. M. Nauka. (1978)