Bull. Nov. Comp. Center, Comp. Science, 46 (2022), 57–70

Randomized vector Gauss-Seidel algorithm with black-red ordering for solving the elastostatics Lamé equation*

Karl K. Sabelfeld, Anastasiya Kireeva

Abstract. A stochastic iterative algorithm for solving the elastostatics Lamé equation in a two-dimensional domain is suggested. The Dirichlet boundary value problem for a system of two coupled second order elliptic equations for the displacement vector components is considered. We approximate the Lamé equations with finite differences and transform the resulting system of linear algebraic equations using the red-black ordering of the Gauss-Seidel method. We solve this system using a vector randomized algorithm. The idea behind these stochastic methods is a randomized vector representation of matrix iterations that are performed by sampling random columns only, avoiding matrix by matrix and matrix by vector multiplications. We test the developed iterative algorithm by a comparison of the simulation results with the exact solution of the Lamé equation.

Keywords: elastostatics Lamé equation, matrix iterations, linear algebraic system, Monte Carlo method, red-black ordering.

Introduction

Systems of linear algebraic equations arise in many scientific and engineering fields. Some practical problems require solving systems of very high dimension. Therefore, fast efficient methods for solving large linear systems are of high demand. Stochastic and randomization algorithms are currently widely used in large-scale computing [1, 2], including numerical methods for solving linear systems [3, 4]. Monte Carlo algorithms are very efficient and scalable on parallel platforms [5].

In [6], a new vector randomized algorithm for solving systems of linear algebraic equations is suggested. The algorithm is based on iteration methods, for instance, a simple iteration represented as a von Neumann series for matrix iterations. The calculation of the matrix iteration is based on a special representation of the matrix through a special stochastic matrix and sampling random columns instead of multiplying matrix by matrix and matrix by vector.

In this paper, we apply the vector randomized algorithm to solve the Lamé equation governing the static elasticity problem. Many of continuum

*Supported by the Russian Science Foundation under Grant 19-11-00019.

mechanics problems are related to the elasticity theory [7]. Finite difference and finite element methods solving the elasticity equations face some problems related to the fact that the components of the displacement vector are strongly coupled [8–12].

We study the efficiency of the randomization algorithm by solving the stationary Lamé equation for a two-dimensional square domain. Thus we deal in this paper with a system of two coupled second order elliptic equations for the displacement vector components as functions of variables x and y. Each equation involves a mixed second order partial derivative of another component. The idea of the iterative algorithm we suggest in this paper can be shortly described as follows: in the first equation, the mixed partial derivative of the second component is moved to the right-hand side of the equation. The same is done with the second equation. In this form, we deal with two anisotropic diffusion equations where in the right-hand-sides mixed partial derivatives are placed.

We approximate the Lamé equations with finite differences and transform the resulting system of linear algebraic equations using the red-black ordering of the Gauss-Seidel method. Red-black ordering increases the convergence of the iterative method, and is also well suited for vector and parallel computations [13, 14]. As mentioned above, the solution of the Lamé equations is calculated using an iteration algorithm. At each iteration, systems of linear algebraic equations for both components are solved by the vector randomized algorithm, then the right-hand sides of the equations are corrected for new computed solutions. In addition, we construct a parallel implementation of the vector randomized algorithm.

The paper is organized as follows. Section 1 presents the Lamé equation and the iterative algorithm for solving it, and also shows the systems of linear algebraic equations obtained for this equation in the two-dimensional case. Section 2 describes the vector randomized algorithm for solving systems of linear algebraic equations. Section 3 presents the simulation results.

1. Iteration randomized algorithm for solving the Lamé equation

1.1. Lamé equation. We consider the stationary Lamé equation with the Dirichlet boundary condition:

$$\begin{gathered}\mu\Delta\boldsymbol{u}(\boldsymbol{r}) + (\lambda+\mu)\nabla(\nabla\cdot\boldsymbol{u}(\boldsymbol{r})) = 0, \quad \boldsymbol{r}\in G, \\ \boldsymbol{u}(\boldsymbol{r})|_\Gamma = \boldsymbol{g}(\boldsymbol{r}),\end{gathered} \tag{1}$$

where the Lamé coefficients $\mu \in R^+$ and $\lambda \in R^+$ are positive real numbers, the vector $\boldsymbol{u}(\boldsymbol{r})$ is the displacement at the point $\boldsymbol{r}$, the components of the vector $\boldsymbol{g}(\boldsymbol{r})$ are continuous functions on the boundary Γ.

For simplicity, we solve the problem in a square domain. The equation (1) in the two-dimensional case has the following form

$$\begin{cases} \mu\Big(\dfrac{\partial^2 u_1(\boldsymbol{r})}{\partial x^2} + \dfrac{\partial^2 u_1(\boldsymbol{r})}{\partial y^2}\Big) + (\lambda+\mu)\Big(\dfrac{\partial^2 u_1(\boldsymbol{r})}{\partial x^2} + \dfrac{\partial^2 u_2(\boldsymbol{r})}{\partial x \partial y}\Big) = 0, \\ \mu\Big(\dfrac{\partial^2 u_2(\boldsymbol{r})}{\partial x^2} + \dfrac{\partial^2 u_2(\boldsymbol{r})}{\partial y^2}\Big) + (\lambda+\mu)\Big(\dfrac{\partial^2 u_2(\boldsymbol{r})}{\partial y^2} + \dfrac{\partial^2 u_1(\boldsymbol{r})}{\partial x \partial y}\Big) = 0, \\ u_1(\boldsymbol{r})|_\Gamma = g_1(\boldsymbol{r}), \\ u_2(\boldsymbol{r})|_\Gamma = g_2(\boldsymbol{r}). \end{cases} \tag{2}$$

1.2. Discretization of the Lamé equation. The Lamé equation (2) is discretized into a linear system of algebraic equations by approximating derivatives through a central finite difference of the second order. To discretize the domain G of size $L \times L$, we introduce a uniform grid of points of size $N \times N$.

Let $\{x_i\}$, $i = 0, \ldots, N$, and $\{y_j\}$, $j = 0, \ldots, N$, be uniform partitions of the interval $[0, L]$ in the x and y directions, respectively, such that $x_i = ih$, $i = 0, \ldots, N$, and $y_j = jh$, $j = 0, \ldots, N$, where the spatial step size is $h = L/(N+1)$. The continuous functions $u_1(\boldsymbol{r})$, $u_2(\boldsymbol{r})$ and $g_1(\boldsymbol{r})$, $g_2(\boldsymbol{r})$ are replaced by the grid functions $\bar{u}_1(i,j)$, $\bar{u}_2(i,j)$ and $\bar{g}_1(i,j)$, $\bar{g}_2(i,j)$.

For all internal points, i.e. points that do not belong to the boundary of the domain: $i, j = 1, \ldots, N-1$, the finite difference scheme for the Lamé equation (2) is defined as follows.

$$c_1(\bar{u}_1(i-1,j) + \bar{u}_1(i+1,j)) + b_1(\bar{u}_1(i,j-1) + \bar{u}_1(i,j+1)) + a_1\bar{u}_1(i,j) = f_1(i,j), \tag{3}$$

$$c_1 = \frac{(2\mu+\lambda)}{h^2}, \quad b_1 = \frac{\mu}{h^2}, \quad a_1 = \frac{(6\mu+2\lambda)}{h^2}, \tag{4}$$

$$f_1(i,j) = \frac{-(\lambda+\mu)}{h^2}(\bar{u}_2(i,j) - \bar{u}_2(i,j+1) - \bar{u}_2(i+1,j) + \bar{u}_2(i+1,j+1)), \tag{5}$$

$$c_2(\bar{u}_2(i-1,j) + \bar{u}_2(i+1,j)) + b_2(\bar{u}_2(i,j-1) + \bar{u}_2(i,j+1)) + a_2\bar{u}_2(i,j) = f_2(i,j), \tag{6}$$

$$c_2 = \frac{\mu}{h^2}, \quad b_2 = \frac{(2\mu+\lambda)}{h^2}, \quad a_2 = \frac{(6\mu+2\lambda)}{h^2}, \tag{7}$$

$$f_2(i,j) = \frac{-(\lambda+\mu)}{h^2}(\bar{u}_1(i,j) - \bar{u}_1(i,j+1) - \bar{u}_1(i+1,j) + \bar{u}_1(i+1,j+1)). \tag{8}$$

For the points belonging to the boundary Γ, the values of the functions $\bar{u}_1(\boldsymbol{r})$, $\bar{u}_2(\boldsymbol{r})$ are determined as follows

$$\bar{u}_1(i,j) = \bar{g}_1(i,j), \quad \bar{u}_2(i,j) = \bar{g}_2(i,j),$$
$$(i,j) \in \{i = 0, N,\ j = 0, \ldots, N \text{ or } i = 0, \ldots, N,\ j = 0, N\},$$

Equations (3), (6) yield the $(N-1)^2 \times (N-1)^2$ linear systems:

$$\begin{cases} A_1 \bar{\boldsymbol{u}}_1 = \boldsymbol{f}_1, \\ A_2 \bar{\boldsymbol{u}}_2 = \boldsymbol{f}_2, \end{cases} \tag{9}$$

where the solution vector and the right-hand side vector are as follows:

$$\bar{\boldsymbol{u}}_k = [u_k(1,1), \ldots, u_k(1,N-1), u_k(2,1), \ldots, u_k(2,N-1), \ldots,$$
$$u_k(N-1,1), \ldots, u_k(N-1,N-1)]^T, \quad k = 1,2,$$
$$\bar{\boldsymbol{f}}_k = [f_k(1,1), \ldots, f_k(1,N-1), f_k(2,1), \ldots, f_k(2,N-1), \ldots,$$
$$f_k(N-1,1), \ldots, f_k(N-1,N-1)]^T, \quad k = 1,2.$$

The matrices A_1 and A_2 are symmetric five-diagonal matrices of size $M \times M$, where $M = (N-1)^2$. The structure of the matrices A_1 and A_2 is shown in Figure 1. The coefficients a_k, b_k, c_k, $k = 1, 2$, are given by the formulae (4), (7).

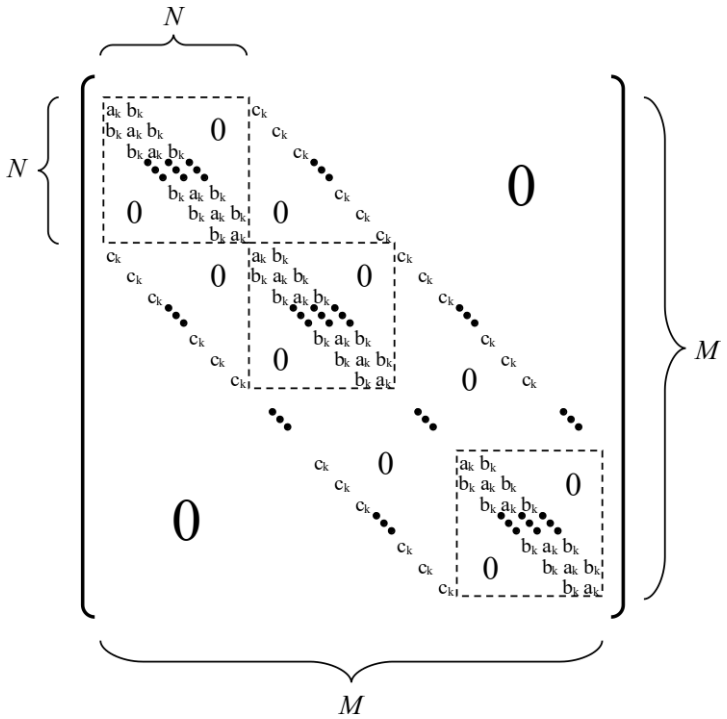

Figure 1. Structure of matrix A_k, $k = 1, 2$

1.3. The red-black ordering of the Gauss-Seidel method. To improve the spectral properties of the matrices A_k, we use the red-black ordering of the Gauss-Seidel method.

The Gauss-Seidel method is an iterative method used to solve a system of linear equations. However, we use this method not to solve the linear systems (9), but only to transform the matrices A_1 and A_2.

The red-black ordering implies that, first, nodes are numbered, whose sum of indices $i+j$ is equal to an even number:

$$\begin{aligned} n_R = \{&(1,1),(1,3),\ldots,(2,2),(2,4),\ldots,\\ &(N-1,1+odd),(N-1,2+odd),\ldots\}, \end{aligned} \tag{10}$$

$$odd = \begin{cases} 1, & \text{if } N \text{ is odd number,} \\ 0, & \text{otherwise.} \end{cases}$$

These nodes are called red nodes.

Then the nodes are numbered, the sum of indices $i+j$ of which gives an odd number:

$$\begin{aligned} n_B = \{&(1,2),(1,4),\ldots,(2,1),(2,3),\ldots,\\ &(N-1,1+even),(N-1,2+even),\ldots\}, \end{aligned} \tag{11}$$

$$even = \begin{cases} 1, & \text{if } N \text{ is even number,} \\ 0, & \text{otherwise.} \end{cases}$$

These nodes are called black nodes.

The matrix A_k and the vectors $\bar{\boldsymbol{u}}_k$, $\boldsymbol{f}_k$ are reordered according to the new node numbering. For example, let us consider the matrix A for the grid size $N=3$:

$$\begin{pmatrix} a & b & 0 & c & 0 & 0 & 0 & 0 & 0 \\ b & a & b & 0 & c & 0 & 0 & 0 & 0 \\ 0 & b & a & 0 & 0 & c & 0 & 0 & 0 \\ c & 0 & 0 & a & b & 0 & c & 0 & 0 \\ 0 & c & 0 & b & a & b & 0 & c & 0 \\ 0 & 0 & c & 0 & b & a & 0 & 0 & c \\ 0 & 0 & 0 & c & 0 & 0 & a & b & 0 \\ 0 & 0 & 0 & 0 & c & 0 & b & a & b \\ 0 & 0 & 0 & 0 & 0 & c & 0 & b & a \end{pmatrix} \tag{12}$$

The matrix A, transformed according to the red-black ordering, looks like this

$$\begin{pmatrix} a & 0 & 0 & 0 & 0 & b & c & 0 & 0 \\ 0 & a & 0 & 0 & 0 & b & 0 & c & 0 \\ 0 & 0 & a & 0 & 0 & c & b & b & c \\ 0 & 0 & 0 & a & 0 & 0 & c & 0 & b \\ 0 & 0 & 0 & 0 & a & 0 & 0 & c & b \\ b & b & c & 0 & 0 & a & 0 & 0 & 0 \\ c & 0 & b & c & 0 & 0 & a & 0 & 0 \\ 0 & c & b & 0 & c & 0 & 0 & a & 0 \\ 0 & 0 & c & b & b & 0 & 0 & 0 & a \end{pmatrix} \tag{13}$$

For any value of size N, the matrix has the following block form [17]:

$$\begin{pmatrix} aE & C \\ C^T & aE \end{pmatrix} \begin{pmatrix} \bar{u}_R \\ \bar{u}_B \end{pmatrix} = \begin{pmatrix} f_R \\ f_B \end{pmatrix}, \tag{14}$$

where aE is a diagonal matrix in which all diagonal entries are equal to a. The matrix C is a block of 5×4 elements in the upper right corner of the matrix (13). The matrix C^T is the transposed matrix C. The vectors $\bar{u}_R$ and f_R contain elements of the vectors $\bar{u}$ and f corresponding to the nodes of red color. The vectors $\bar{u}_B$ and f_B contain elements of the vectors $\bar{u}$ and f corresponding to the nodes of black color.

The Gauss-Seidel iterations for the system (14) have the following form:

$$\begin{cases} \bar{u}_R^{(t+1)} = \dfrac{1}{a}(f_R - C\bar{u}_B^{(t+1)}), \\ \bar{u}_B^{(t+1)} = \dfrac{1}{a}\Big(f_B - \dfrac{1}{a}C^T(f_R - C\bar{u}_B^{(t)})\Big). \end{cases} \tag{15}$$

Thus, at each iteration, the values of the vector $\bar{u}$ are calculated first in nodes of black color ($\bar{u}_B$), then using these values in nodes of red color ($\bar{u}_R$).

To write the Gauss-Seidel method in matrix form, we represent the matrix A_k as $A_k = D_k - L_k - U_k$, where D_k is a diagonal matrix containing diagonal elements of the matrix A_k, L_k is a lower triangular matrix, and U_k is an upper triangular matrix. Then the Gauss-Seidel method is written as follows:

$$\bar{\boldsymbol{u}}_k^{t+1} = \tilde{A}_k \bar{\boldsymbol{u}}_k^t + \boldsymbol{b_k}, \tag{16}$$

$$\tilde{A}_k = (E - D_k^{-1}L_k)^{-1} D_k^{-1} U_k, \tag{17}$$

$$\boldsymbol{b_k} = (E - D_k^{-1}L_k)^{-1} D_k^{-1} \boldsymbol{f_k}, \tag{18}$$

where E is an identity matrix.

In [16, chapter 9, page 176], it is shown that for the red-black ordering of the matrix $A = L + U$, the inverse matrix $(E - L)^{-1}$ can be calculated as $(E-L)^{-1} = (E+L)$. Based on this property, we calculate $(E - D_k^{-1}L_k)^{-1} = (E + D_k^{-1}L_k)$ and obtain the matrix $\tilde{A}_k$ and the vector $\boldsymbol{b}$. The matrix $\tilde{A}_k$ is

a singular matrix, it contains zero elements corresponding to nodes of red color. This is a consequence of the red-black ordering and corresponds to the fact that first the system is solved for the black nodes, and then the solution for the red nodes is calculated from these values.

1.4. Solving the Lamé equation. Thus, we have moved from the original system (9) to the new system:

$$\begin{cases} \bar{\boldsymbol{u}}_1^{t+1} = \tilde{A}_1 \bar{\boldsymbol{u}}_1^t + \boldsymbol{b}_1^t, \\ \bar{\boldsymbol{u}}_2^{t+1} = \tilde{A}_2 \bar{\boldsymbol{u}}_2^t + \boldsymbol{b}_2^t. \end{cases} \tag{19}$$

The right-hand side $\boldsymbol{b}_1^t$ of the first system of linear equations is a function of the solution $\bar{\boldsymbol{u}}_2^t$ of the second system of linear equations. Similarly, the right-hand side $\boldsymbol{b}_2^t$ is a function of the solution $\bar{\boldsymbol{u}}_1^t$.

Therefore, we solve the system (19) by iterative method. At each iteration, the solution $\bar{\boldsymbol{u}}_1^t$ and the right-hand side $\boldsymbol{b}_2^t$ are calculated first, then the solution $\bar{\boldsymbol{u}}_2^t$ and the right-hand side $\boldsymbol{b}_1^t$ are computed. In addition, each iteration is performed in two stages. In the first step, the solution $\bar{\boldsymbol{u}}_k$ is calculated in nodes of black color using the vector randomized algorithm given in [6] (it will be given below in Section 2). In the second step, the solution $\bar{\boldsymbol{u}}_k$ is calculated in nodes of red color using the newly calculated values in the black nodes.

Let us describe the iterative algorithm for solving the system (19) in more detail:

1.5. Iterative algorithm for solving the Lamé equation

1. At the beginning, initialize the right-hand side with any non-zero values, for example, $\boldsymbol{b}_1^0 = 1$. Initialize $t = 0$.
2. Calculate the solution $\bar{\boldsymbol{u}}_{1\_B}^{t+1}$ for black nodes using the vector randomized algorithm.
3. Calculate the solution $\bar{\boldsymbol{u}}_{1\_R}^{t+1}$ for red nodes using the formula $\bar{\boldsymbol{u}}_{1\_R}^{t+1} = \frac{1}{a_1}(\boldsymbol{b}_{1\_R}^t - C_1 \bar{\boldsymbol{u}}_{1\_B}^{t+1})$.
4. Calculate the right-hand side $\boldsymbol{b}_2^{t+1}$ by the formulae (8) and (18) using the new values of the vector $\bar{\boldsymbol{u}}_1^{t+1}$.
5. Calculate the solution $\bar{\boldsymbol{u}}_{2\_B}^{t+1}$ for black nodes using the vector randomized algorithm.
6. Calculate the solution $\bar{\boldsymbol{u}}_{2\_R}^{t+1}$ for red nodes using the formula $\bar{\boldsymbol{u}}_{2\_R}^{t+1} = \frac{1}{a_2}(\boldsymbol{b}_{2\_R}^t - C_2 \bar{\boldsymbol{u}}_{2\_B}^{t+1})$.
7. Calculate the right-hand side $\boldsymbol{b}_1^{t+1}$ by the formulae (8) and (18) using new values of the vector $\bar{\boldsymbol{u}}_2^{t+1}$.

8. Calculate the errors $er_k = \frac{|\bar{\boldsymbol{u}}_k^{t+1} - \bar{\boldsymbol{u}}_k^t|}{|\bar{\boldsymbol{u}}_k^{t+1}|}$, $k = 1, 2$.
9. If $er_1 < \epsilon$ and $er_2 < \epsilon$, where $\epsilon \in (0, 1)$ is the required accuracy of the solution, then stop the iterations.
10. Otherwise, increase the iteration number $t := t + 1$ and continue from step 2.

2. The vector randomized algorithm for solving system of linear algebraic equations

2.1. Random vector estimators for stochastic matrix iterations. Let $\boldsymbol{b} \in R^n$ be a nonnegative stochastic vector, which means, $\sum_{i=1}^n b_i = 1$. Let A be a nonnegative $n \times n$ column stochastic matrix, which means, $\sum_{i=1}^n a_{ij} = 1$ for $j = 1, \ldots, n$. Further, we denote by A_k the kth column of the matrix A.

In [6], the random unbiased vector estimators for the products $A^k\boldsymbol{b}$ are presented. In what follows the products $A^k\boldsymbol{b}$ are called iterations.

For a stochastic matrix A and stochastic vector $\boldsymbol{b}$ an unbiased estimator ξ_{j_1} for $A\boldsymbol{b}$ is as follows.

$$A\boldsymbol{b} = E\xi_{j_1} = E\{A_k \mid \boldsymbol{p} = \boldsymbol{b}\}, \tag{20}$$

where, $E\{A_k \mid \boldsymbol{p} = \boldsymbol{b}\}$ means that the expectation is taken over random columns chosen at random from the distribution $\boldsymbol{p} = \boldsymbol{b}$.

Next step, an unbiased estimator ξ_{j_2,j_1} for the second iteration $A^2\boldsymbol{b}$ is constructed as follows:

$$A^2\boldsymbol{b} = A\xi_{j_1} = EE\{A_j \mid \boldsymbol{p}_2 = A_k;\ \boldsymbol{p}_1 = \boldsymbol{b}\} = E\xi_{j_2,j_1}. \tag{21}$$

This means that in the double expectation we first choose a random column A_k from the distribution $\boldsymbol{p}_1 = \boldsymbol{b}$, and then choose a random column A_j from the distribution $\boldsymbol{p}_2 = A_k$. The unbiased estimator for the third iteration is obtained by the next random sampling of a random column A_i from the distribution $\boldsymbol{p}_3 = A_j$, etc., and for the kth iteration we get

$$\xi_{j_k,j_{k-1},\ldots,j_1} = A_{j_k}, \tag{22}$$

where each next random column is sampled from the previously sampled column.

In the case of an arbitrary vector $\boldsymbol{b}$, we construct the unbiased estimator for the kth iteration $A^k\boldsymbol{b}$ as follows. Let $\boldsymbol{p}(j)$ be an arbitrary probability distribution of indices $j = 1, 2, \ldots, n$ satisfying the condition $\boldsymbol{p}(j) \neq 0$ if $\boldsymbol{b}_j \neq 0$, $j = 1, \ldots, n$. Then

$$E\frac{b_{j_1}}{p(j_1)}\xi_{j_k,j_{k-1},\ldots,j_1} = A^k\boldsymbol{b}, \quad k = 1, 2, \ldots, \tag{23}$$

where j_1 is a random index sampled from the density $\boldsymbol{p}$.

2.2. Random vector estimators for iterations of positive or irreducible nonnegative matrix. In [6] the randomized vector algorithm is extended to positive or irreducible nonnegative matrices using transformation of nonnegative matrix A to a column stochastic matrix S.

Assume that a nonnegative matrix A has a positive eigenvalue λ and a positive eigenvector z of the transpose matrix A^T ($A^T z = \lambda z$) with its positive components $z_1, \ldots, z_n$. Let Z be a diagonal matrix defined by $Z = \mathrm{diag}\{z_1, \ldots, z_n\}$. Then the matrix

$$S = \frac{1}{\lambda} Z A Z^{-1}$$

is a column stochastic matrix.

This transformation allows to reduce the calculation of iterations A^k to calculation of iterations S^k:

$$A^k = \lambda^k Z^{-1} S^k Z, \quad k = 1, 2, \ldots$$

We take the maximum eigenvalue of the matrix A^T as a positive eigenvalue λ. It is calculated by the power method [18].

2.3. Solving of a linear system. According to [6], let us describe the algorithm for calculating the solution of a system of linear algebraic equation

$$\boldsymbol{x} = A\boldsymbol{x} + \boldsymbol{b}. \tag{24}$$

In the case when spectral radius of matrix A is less than unity $\rho(A) < 1$, the solution of the system (24) can be represented as a Neumann series

$$\boldsymbol{x} = \sum_{k=0}^{\infty} A^k \boldsymbol{b}.$$

Transform (24) for a column stochastic matrix $A = \lambda Z^{-1} S Z$ multiplying it by Z and introducing $\boldsymbol{y} = Z\boldsymbol{x}$, $\tilde{\boldsymbol{b}} = Z\boldsymbol{b}$. Then we arrive to the equation

$$\boldsymbol{y} = \lambda S\boldsymbol{y} + \tilde{\boldsymbol{b}}. \tag{25}$$

The solution of the system (25) can be represented as a Neumann series

$$\boldsymbol{y} = \tilde{\boldsymbol{b}} + \sum_{k=1}^{\infty} (\lambda S)^k \tilde{\boldsymbol{b}}.$$

To achieve the target accuracy ϵ of the solution, a finite number of iterations K should be taken as $K \sim |\log \epsilon|$. Thus, we calculate $\boldsymbol{y}_K$, the approximation of $\boldsymbol{y}$, from

$$\boldsymbol{y}_K = \tilde{\boldsymbol{b}} + \sum_{k=1}^{K} (\lambda S)^k \tilde{\boldsymbol{b}}. \tag{26}$$

The iterations $(\lambda S)^k \tilde{\boldsymbol{b}}$ are calculated using the random vector estimators described in Section 2.1 for the stochastic matrix and arbitrary vector (23). We use Walker's alias method [19] to sample from the discrete distributions given by the columns of a stochastic matrix and vectors.

We introduce a vector $\boldsymbol{V}$ where we store the contributions for the iterations $S^k \boldsymbol{b}$ along the Markov trajectories. The contribution corresponding to the kth iteration is calculated as follows

$$q_k = w \lambda^k S_{j_k}. \tag{27}$$

Here, $w = \frac{\tilde{b}_{j_1}}{p_{j_1}}$ is a weight that occurs when using the formula (23), where j_1 is a random element of the vector $\tilde{b}$; λ^k comes from the representation (26); j_k is a random column of the matrix S.

The Markov trajectories are constructed as follows:

2.4. Vector randomized algorithm for solving linear system

1. Choose the density $\boldsymbol{p}$ as follows: $p_i = \frac{|\tilde{b}_i|}{\sum_{j=0}^{n} \tilde{b}_j}$, $i = 1, 2, \ldots, n$.
2. A random index j_1 is sampled from the density $\boldsymbol{p}$ and contribution $q_1 = \frac{b_{j_1}}{p_{j_1}} \lambda S_{j_1}$ is added to the j_1th element of the vector $\boldsymbol{V}$: $V_{j_1} := V_{j_1} + q_1$.
3. A random index j_2 is sampled from the column S_{j_1} and contribution $q_2 = \frac{b_{j_1}}{p_{j_1}} \lambda^2 S_{j_2}$ is added to the j_2th element of the vector $\boldsymbol{V}$: $V_{j_2} := V_{j_2} + q_2$,
4. etc., and the last state, a random index j_K is sampled from the column $S_{j_{K-1}}$ and contribution $q_K = \frac{b_{j_1}}{p_{j_1}} \lambda^K S_{j_K}$ is added to the j_Kth element of the vector $\boldsymbol{V}$: $V_{j_K} := V_{j_K} + q_K$.
5. Repeat the steps 2–4 for M independent trajectories, and calculate the result as

$$\boldsymbol{y}_K = \tilde{\boldsymbol{b}} + \frac{1}{M} \sum_{i=1}^{n} V_i.$$

3. Simulation results

In the Monte Carlo methods, the statistical error ϵ_s is measured by the standard deviation $\sqrt{D}/\sqrt{M}$, where D is the variance, and M is the number of trajectories [6]. Hence, the number of trajectories behaves like $M \sim D/\epsilon_s^2$. The trajectories are statistically independent and can be calculated in parallel. The traditional method of parallel implementation of Monte Carlo algorithms is the distribution of random trajectories among the available supercomputer's cores [5].

The vector randomized algorithm 2.4 takes up most of the computation of the iterative algorithm 1.5 for solving the Lamé equation. Therefore, we implement the vector randomized algorithm in parallel using the MPI standard. In addition, the compressed sparse column format is used to store the matrices C_1, C_2 (14) and $\tilde{A}_1$, $\tilde{A}_2$ (17), which reduces the amount of memory needed to store data and the amount of calculations.

To verify the iterative algorithm 1.5, we solve the Lamé equation with a known solution:

$$\begin{cases} u_1(x,y) = x(x-1)y(y-1), \\ u_2(x,y) = 2x(x-1)y(y-1). \end{cases} \tag{28}$$

After substituting the solution (28) into the equations (2), we get the following system:

$$\begin{cases} \mu\Big(\dfrac{\partial^2 u_1(\boldsymbol{r})}{\partial x^2} + \dfrac{\partial^2 u_1(\boldsymbol{r})}{\partial y^2}\Big) + (\lambda+\mu)\Big(\dfrac{\partial^2 u_1(\boldsymbol{r})}{\partial x^2} + \dfrac{\partial^2 u_2(\boldsymbol{r})}{\partial x \partial y}\Big) = \tilde{\boldsymbol{f}}_1(\boldsymbol{r}), \\ \tilde{\boldsymbol{f}}_1(\boldsymbol{r}) = 2(2\mu+\lambda)y(y-1) + 2\mu x(x-1) + 2(\lambda+\mu)(2x-1)(2y-1), \\ \mu\Big(\dfrac{\partial^2 u_2(\boldsymbol{r})}{\partial x^2} + \dfrac{\partial^2 u_2(\boldsymbol{r})}{\partial y^2}\Big) + (\lambda+\mu)\Big(\dfrac{\partial^2 u_2(\boldsymbol{r})}{\partial y^2} + \dfrac{\partial^2 u_1(\boldsymbol{r})}{\partial x \partial y}\Big) = \tilde{\boldsymbol{f}}_2(\boldsymbol{r}), \\ \tilde{\boldsymbol{f}}_2(\boldsymbol{r}) = 4(2\mu+\lambda)x(x-1) + 4\mu y(y-1) + (\lambda+\mu)(2x-1)(2y-1). \end{cases} \tag{29}$$

We solve the Lamé equation in a unit square. In this case, we deal with zero boundary conditions:

$$u_1(\boldsymbol{r})|_\Gamma = 0, \quad u_2(\boldsymbol{r})|_\Gamma = 0. \tag{30}$$

After discretization of the equations (29), we obtain linear systems (9) with the same matrices A_1 and A_2 as given in Section 1.3, and the following right-hand sides:

$$\begin{aligned} \boldsymbol{v}_1(\boldsymbol{r}) &= \boldsymbol{f}_1(\boldsymbol{r}) + \tilde{\boldsymbol{f}}_1(\boldsymbol{r}), \\ \boldsymbol{v}_2(\boldsymbol{r}) &= \boldsymbol{f}_2(\boldsymbol{r}) + \tilde{\boldsymbol{f}}_2(\boldsymbol{r}), \end{aligned}$$

where $\boldsymbol{f}_1(\boldsymbol{r})$, $\boldsymbol{f}_2(\boldsymbol{r})$ depend on the mixed derivatives of $u_1(\boldsymbol{r})$, $u_2(\boldsymbol{r})$, and are calculated by the formulae (5), (8).

We calculate the solution of the system for the grid size $N = 100$, the space step $h = 9.9 \cdot 10^{-3}$. The spectral radius of the matrices $\tilde{A}_1$, $\tilde{A}_2$ (17) for this space step is $\rho(\tilde{A}_k) = 0.999$, $k = 1, 2$. Due to the fact that the spectral radius is close to unity, the vector randomized algorithm 2.4 converges slowly, and the number of iterations K must be large enough to obtain the solution of linear systems (9) with sufficient accuracy. We take K equal to $5 \cdot 10^3$. The maximum variance of the estimator for the solution obtained by the algorithm 2.4 for $N = 100$ is 0.013 for the first component $u_1(\boldsymbol{r})$ and 0.05 for the second component $u_2(\boldsymbol{r})$. We simulate $M = 10^7$ trajectories to get the solution smooth enough to compute mixed derivatives. These parameter values provide an average absolute error of the linear system solution equal to 0.0002. The iterative algorithm of solving the Lamé equation 1.5 converges in four iterations and gives the solution with an absolute error of 0.0003 and a relative error of 0.01.

The partition "Cascade_lake" of the cluster "MVS-10P" of the Joint Supercomputer Center of RAS [20] is employed for calculations. The "Cascade_lake" node consists of two processors Intel Xeon Gold 6248R CPU 3.00 GHz, each containing 24 cores with 2 threads.

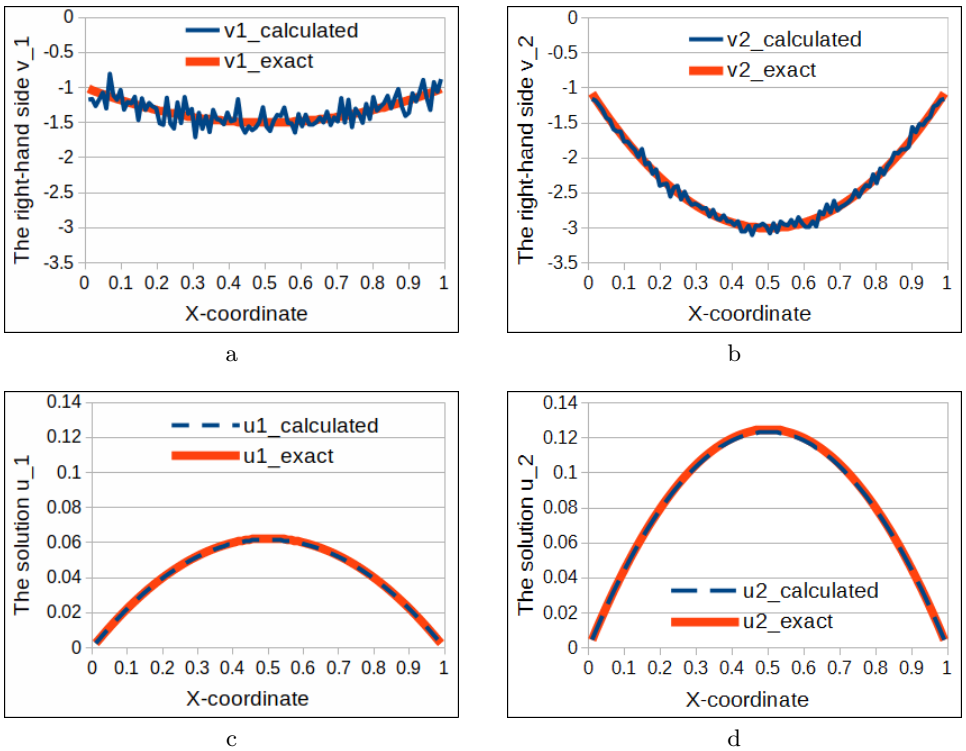

Figure 2. The right-hand sides (a, b) and solutions (c, d) of the Lamé equation obtained using the iterative algorithm and calculated using the exact formulae

The calculation time for solving one linear system is 13 seconds, and the entire time for solving the Lamé equation is 102 seconds when using a single node and 48 MPI processes.

Figure 2 shows the right-hand sides $\boldsymbol{v}_1(\boldsymbol{r})$, $\boldsymbol{v}_2(\boldsymbol{r})$ and solutions $u_1(\boldsymbol{r})$, $u_2(\boldsymbol{r})$ obtained by the iterative algorithm (1.5) and calculated using exact formulae (28), (29). As shown, the approximation of mixed derivatives, which is included in the right part, is not smooth. However, the solution of the Lamé equation is calculated with good accuracy.

Conclusion

A stochastic iterative algorithm for solving the Lamé equation in a two-dimensional domain is presented. We deal with a system of two coupled second order elliptic equations for the displacement vector components. Each equation includes a mixed second order partial derivative of another component. The iterative algorithm is based on the following idea: in the first equation, the mixed partial derivative of the second component is moved to the right-hand side of the equation. The same is done with the second equation. We approximate the Lamé equations by finite differences and transform the resulting system of linear algebraic equations using the red-black ordering of the Gauss-Seidel method. The linear system is solved using the vector randomized algorithm. We check the iterative algorithm for solving the Lamé equation with a known solution. The simulation results show that the algorithm allows solving the Lamé equation with good accuracy.

References

[1] Nemec N. Diffusion Monte Carlo: Exponential scaling of computational cost for large systems // Phys. Rev. B. — 2010. — Vol. 81. — 035119.

[2] Brown F.B. Recent advances and future prospects for Monte Carlo // Progress in Nuclear Science and Technology. — 2011. — Vol. 2. — P. 1–4.

[3] Dimov I., Maire S., Sellier J.M. A new walk on equations Monte Carlo method for solving systems of linear algebraic equations // Applied Mathematical Modelling. — 2015. — Vol. 39, Iss. 15. — P. 4494–4510.

[4] Magalhẽs F., Monteiro J., Acebrón J.A., Herrero J.R. A distributed Monte Carlo based linear algebra solver applied to the analysis of large complex networks // Future Generation Computer Systems. — 2022. — Vol. 127. — P. 320–330.

[5] Bhavsar V.C., Isaac J.R. Design and analysis of parallel Monte Carlo algorithms // SIAM J. on Scientific and Statistical Computing. — 1987. — Vol. 8, Iss. 1. — P. 73–95.

[6] Sabelfeld K.K. A new randomized vector algorithm for iterative solution of large linear systems // Applied Mathematics Letters. — 2022. — Vol. 126. — 107830.

[7] Landau L.D., Lifshitz E.M. Theory of Elasticity. — Pergamon Press, 1984.

[8] Koubaiti O., El-Mekaoui J., Elkhalfi A. Complete study for solving Navier-Lamé equation with new boundary condition using mini element method // Intern. J. Mechanics. — 2019. — Vol. 12. — P. 46–58.

[9] Bialecki B., Karageorghis A. Finite difference schemes for the Cauchy-Navier equations of elasticity with variable coefficients // J. Scientific Computing. — 2014. — Vol. 62. — P. 78–121.

[10] Slak J., Kosec G. Refined Meshless local strong form solution of Cauchy-Navier equation on an irregular domain // Engineering Analysis with Boundary Elements. — 2019. — Vol. 100. — P. 3–13.

[11] Ciarlet P.G. The Finite Element Method for Elliptic Problems. — North-Holland, Amsterdam, 1978.

[12] Lisbona F.J., Vabishchevich Pert.N. Operator-splitting schemes for solving unsteady elasticity problems // Computational Methods in Applied Mathematics. — 2001. — Vol. 1, Iss. 2. — P. 188–198.

[13] Zhang J. Acceleration of five-point red-black Gauss-Seidel in multigrid for Poisson equation // Applied Mathematics and Computation. — 1996. — Vol. 80, Iss. 1. — P. 73–93.

[14] Gjesdal T. Analysis of a New Red-Black Ordering for Gauss-Seidel Smoothing in Cell-Centred Multigrid. — 1993. — Ref. No. CMR-93-A200007.

[15] Eremeev V.A. Reshenie sistem linejnyh algebraicheskih uravnenij dlya bol'shih razrezhennyh matric: Uchebno-metodicheskoe posobie. — Rostov-na-Donu, 2008 (In Russian).

[16] Sabelfeld K.K., Shalimova I.A. Spherical and Plane Integral Operators for PDEs // Construction, Analysis and Aapplications. — Berlin-Boston: De Gruyter, 2013.

[17] Starchenko A.V., Bercun V.N. Metody parallel'nyh vychislenij: Uchebnik. — Tomsk: Izdatel'stvo Tomskogo universiteta. Seriya Superkomp'yuternoe obrazovanie, 2013 (In Russian).

[18] O'Leary D.P., Stewart G.W., Vandergraft J.S. Estimating the Largest Eigenvalue of a Positive Definite Matrix // Mathematics of Computation. — 1979. — Vol. 33, Iss. 148. — P. 1289–1292.

[19] Walker A.J. New fast method for generating discrete random numbers with arbitrary frequency distributions // Electr. Lett. — 1974. — Vol. 10. — P. 127–128.

[20] Joint Supercomputer Center of the Russian Academy of Sciences. — http://www.jscc.ru/ (Accessed December 10, 2022).