Performance evaluation of preconditioning method for in-body current density analysis using numerical human model

Masamune Nomura¹, Hiroo Tarao², Noriyuki Hayashi³, Amane Takei^{3*}

¹ Graduate Course in Engineering, Graduate School of Engineering, University of Miyazaki

² Department of Electrical and Computer Engineering, National Institute of Technology, Kagawa College

³ Department of Electrical and System Engineering, Faculty of Engineering, University of Miyazaki

* `takei@cc.miyazaki-u.ac.jp`

Received: April 18, 2018; Accepted: March 10, 2019; Published: April 2, 2019

Abstract. Iterative methods such as the COCG method and the COCR method take too long to solve large-scale linear equations obtained by the SPFD method, which is a current density analysis method that uses a numerical human model. Therefore, in the present study, in order to accelerate these methods, we introduce iterative methods with various preconditioning methods. The results indicate that the block decomposition method can improve the parallelism of the IC and SSOR preconditioning. In addition, SSOR preconditioning and block SSOR preconditioning for a single thread and block IC decomposition preconditioning for multiple threads are the fastest methods used in the present study.

Keywords: Numerical human model, Current density analysis, COCG method, COCR method, Preconditioning, Block decomposition

1. Introduction

Currently, electric shock accidents kill several dozen people in Japan each year [1]. In order to prevent these accidents or minimize the degree of injury, a number of studies have been conducted, including studies to clarify the current density path in the human body during electric shock, which is particularly important for evaluating the impact of electricity on the human body. These studies have evaluated the impedance of a human body and a current density distribution using experiments and a numerical simulation [2, 3]. In addition, a current is induced in the body when a human is exposed to electromagnetic fields. These currents induce electrical

stimulation at low frequencies and heat at high frequencies. In order to clarify these mechanisms and estimate the current density induced in the body directly and indirectly by electromagnetic fields, analysis has been performed using a numerical model based on a human body of anatomical characteristic [4, 5]. Development of high-precision numerical human models has progressed based on numerical data obtained by computed tomography and magnetic resonance imaging. Based on this background, the development of numerical analysis methods using these human models has progressed. However, it is difficult to apply commercially available general software because of the large scale and complicated nature of the numerical model. Therefore, we need analysis methods in order to efficiently process numerical models [6, 7, 8].

In this analysis, simultaneous equations for which the coefficient matrix is a large-scale complex symmetric matrix must be solved. Krylov subspace methods are used as a solver to solve these simultaneous equations. The conjugate gradient (CG) method is a widely used Krylov subspace method. However, the CG method is not appropriate as a solver for a non-Hermitian matrix used in this analysis. The biconjugate gradient stabilized (BiCGSTAB) method, the conjugate gradient squared (CGS) method, and the generalized minimal residual (GMRES) method, for example, have been proposed as appropriate solvers for a non-Hermitian matrix. However, most of these solvers require the calculation of matrix-vector products more than twice in one iteration and do not take advantage of the complex symmetry in the coefficient matrix. The conjugate orthogonal conjugate gradient (COCG) method, the computational complexity of which is the similar to that of the CG method, is an effective solver for this matrix [9]. Therefore, we use the COCG method and the conjugate orthogonal conjugate residual (COCR) method [10], the convergence of which is similar to that of the COCG method, as a solver for the large-scale complex symmetric matrix in the present study.

However, when this matrix is solved by the COCG method or the COCR method, a large amount of computation is needed because this matrix is large. Moreover, the computational complexity of the complex matrix is higher than that of a real matrix. Therefore, in order to efficiently calculate the matrix, it is necessary to improve the convergence by preconditioning and shorten the computation time by parallelization. However, improvement of the convergence of these iterative methods and the computation time are regarded as future tasks in studies [6, 7, 8] of the current density analysis with a numerical human model. Therefore, in the present study, we focus on the application of optimum preconditioning and parallelization methods to the iterative solver for high performance for large-scale current density analysis using a numerical human body model. We use a numerical human model consisting of 2-mm voxels and apply the scalar potential finite-difference (SPFD) method [11], which is known for being capable of efficiently solving problems related to this model. In the SPFD method, the coefficient matrix of the simultaneous linear equations to be solved is a symmetric positive-definite matrix with approximately eight million degrees of freedom. In addition, we implement the COCG and the COCR method with a compressed row storage (CRS) data structure, which is suitable for sparse matrix storage because this matrix is a sparse complex matrix. As an API for parallelization, we use OpenMP on a shared-memory parallel computer because the memory requirements are only up to 4 GB. We apply diagonal scaling preconditioning, incomplete Cholesky (IC) decomposition preconditioning, and symmetric successive over-relaxation (SSOR) preconditioning to the COCG method and the COCR method. In order to search for an optimal method, we compare these methods from the viewpoints of convergence, computa-

tion time, required memory, and parallelization efficiency. We also apply the block decomposition method, which is one of the methods for parallelizing forward and backward substitutions to the IC decomposition and SSOR preconditioning. In general, the multicolor ordering method [12, 13] has been proposed as a parallelization method for forward and backward substitution calculations during preconditioning. However, in this method, as the number of colors increases, the degree of parallelism decreases. Moreover, application of this method requires rewriting many contents of the SPFD method. In contrast, application of block decomposition to the SPFD method is easy, and preconditioning with the block decomposition in this matrix might be effective in a parallel environment with several tens of threads [14] because influence to the convergence by block decomposition is relatively small.

The present study shows that SPFD analysis using block decomposition in a parallel computing environment of approximately 64 parallel threads does not significantly deteriorate convergence. In addition, in a parallel computer using 64 threads, the current density analysis using a large human body model with approximately eight million degrees of freedom can be completed in several minutes using the proposed method.

2. Formulation and numerical analysis method

2.1. Formulation

First, the following equations are obtained when a displacement current can be considered:

$$\mathbf{J} = (\sigma + j\omega\epsilon)\mathbf{E} \quad (1)$$

$$\text{div}\mathbf{J} = 0 \quad (2)$$

$$\mathbf{E} = -\text{grad}V \quad (3)$$

where $\mathbf{J}$ is the current density (A/m²), σ is the conductivity (S/m), j is the imaginary unit, ω is the angular frequency (rad/s), ϵ is the permittivity (F/m), $\mathbf{E}$ is the electric field (V/m), and V is the potential (V). Equations (1), (2), and (3) show Ohm's law, the continuous current condition, and the relationship of the electric field and potential established in the static or quasi-static field, respectively. We can substitute Eq. (3) into Eq. (1) and substitute the result for Eq. (2), which yields the following partial differential equation:

$$-\text{div}((\sigma + j\omega\epsilon)\text{grad}V) = 0 \quad (4)$$

Equation (4) is the basic equation [8] used in the present study.

2.2. Numerical analysis method

The numerical human body model used herein has the average body shape (height: 173.2 cm, weight: 67 kg) of a Japanese adult male and was developed by the National Institute of Information and Communications Technology [15]. In this model, 51 tissues and organs obtained from whole-body magnetic resonance images are represented using approximately eight million 2-mm cubic voxels. Conductivity and permittivity are imparted to each tissue or organ using values normally used in the commercial frequency (50 Hz-60 Hz) domain [16, 17]. The

SPFD method is a numerical analysis method for solving an equation assuming a static or quasi-static field of an electromagnetic field, such as that indicated by Eq. (4). This method is suitable for analysis of electric field and current density in the commercial frequency domain. In the present study, we perform discretization using the SPFD method with the human body voxel model. The unknown to be solved is the potential V at the center of gravity of each voxel. Figure 1 shows the biological model used herein divided into lattice-like voxels [7].

Figure 1: Equivalent circuit of the human body with electrodes

As shown in Figure 1, nodes are set at the center of the lattice in the SPFD method. In addition, nodes (E_1 , E_2 , and E_3) are also set at the boundary of the model because this analysis assumes direct contact with charged objects. In addition, the impedance Z between the nodes is already known from the conductivity and the permittivity given to each lattice. Here, when considering nodes i and j in Figure 1, the following equation is established from Kirchhoff's current law:

$$I_1 + I_2 + I_3 + I_4 = 0 \quad (5)$$

where current I_1 can be obtained by using Ohm's law as follows:

$$I_1 = \frac{V_{i-1,j} - V_{i,j}}{Z_{i-1,j} + Z_{i,j}} \quad (6)$$

Currents I_2 , I_3 , and I_4 are obtained in the same way, and by substituting the four current values into Eq. (5), an equation in which the unknown is the potential is obtained. By applying the above process to all lattices, simultaneous equations, the unknown of which is the potential, are created. In addition, the current densities can be obtained after the potentials are obtained from these simultaneous equations.

In the present study, electrodes are attached to the palms of both hands of the numerical human model shown in Fig. 2, and the numerical analysis is performed for a potential difference of 1 V (60 Hz) between the electrodes. The contact area (number of voxels) on the palms

of the right and left hands are 169.8 cm² (2,326) and 157.9 cm² (2,253), respectively.

Figure 2: Numerical human model used in the present study [7]

3. Solution for speed up of simultaneous linear equations

3.1. Iterative solution of simultaneous linear equations

3.1.1. Algorithm of preconditioning COCG and COCR methods

In general, when the condition number (maximum and minimum eigenvalue ratio) of solving matrix A is closer to 1 (unity matrix), the convergence of iterative methods such as the COCG and COCR methods should be better. Therefore, using a preconditioning matrix M similar to solving matrix A , we convert

$$Ax = b \quad (7)$$

to

$$M^{-1}Ax = M^{-1}b \quad (8)$$

where x and b are unknowns and the right-hand side vector, respectively. This preconditioning reduces the condition number and improves the convergence of the iterative methods. Figure 3 shows each algorithm of the preconditioning COCG and COCR methods [18].

The convergence of the iterative methods with preconditioning matrix M depends on the degree of approximation of matrix M of coefficient matrix A . Therefore, it is important to choose a more correct preconditioning matrix M . In the present study, we introduce and compare diagonal scaling preconditioning, IC decomposition preconditioning, and SSOR preconditioning, which are common methods for matrix preconditioning. In addition, these preconditioning methods and the COCG and COCR methods are implemented so as to correspond to the CRS data structure.

Figure 3: Algorithm of iterative methods with preconditioning

3.1.2. Diagonal scaling preconditioning matrix

The diagonal scaling preconditioning matrix is shown in Fig. 4.

$$M = \begin{bmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & a_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{n,n} \end{bmatrix}$$

$a_{i,i} : \text{Diagonal components of } A$

Figure 4: Diagonal scaling preconditioning matrix

This preconditioning matrix is obtained by extracting only the diagonal elements of the original matrix, A . In the calculation of $Mz^{(i-1)} = r^{(i-1)}$ of Fig. 3, since M is a diagonal matrix, its inverse matrix is easily obtained. Therefore, if M^{-1} is evaluated before each iteration of the COCG and COCR methods, then diagonal scaling preconditioning can be easily parallelized.

3.1.3. IC decomposition preconditioning matrix

The original matrix A is decomposed as follows:

$$A = \tilde{L}\tilde{D}\tilde{L}^T \quad (9)$$

where $\tilde{L}$, $\tilde{D}$, and $\tilde{L}^T$ are the lower triangular matrix, the diagonal matrix, and the upper triangular matrix, respectively. This decomposition takes more time as matrix A becomes larger. Therefore, we ignore the fill-in occurring during the modified Cholesky decomposition of matrix A and calculate $\tilde{L}$ and $\tilde{D}$. The decomposition that ignores this fill-in is called incomplete Cholesky (IC) decomposition. Thus, the IC preconditioning matrix is expressed as follows:

$$M = \tilde{L}\tilde{D}\tilde{L}^T \quad (10)$$

In the calculation of $Mz^{(i-1)} = r^{(i-1)}$, the vector $z^{(i-1)}$ is obtained by forward and backward substitution using $\tilde{L}\tilde{D}\tilde{L}^T$. However, calculation by forward and backward substitution includes data dependency, which is a situation in which an instruction is dependent on a result from a sequentially previous instruction before execution can be completed. Therefore, parallelization is generally difficult. Figure 5 shows the algorithm of IC decomposition used in the present study. In Fig. 5, d , a , and l are the elements of the diagonal matrix $\tilde{D}$, the original matrix A , and the lower triangular matrix $\tilde{L}$, respectively.

$$\begin{array}{l} i = 1, 2, \dots, n \\ \left\{ \begin{array}{l} j = 1, 2, \dots, i-1 \\ \text{If } a_{i,j} \neq 0 \\ l_{i,j} = a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} \cdot d_k \cdot l_{j,k} \end{array} \right. \\ d_i = \left[a_{i,i} - \sum_{k=1}^{i-1} l_{i,k}^2 \cdot d_k \right]^{-1} = l_{i,i}^{-1} \end{array}$$

Figure 5: Algorithm of IC decomposition

Here, when IC decomposition is applied to the matrix actually appearing in the SPFD method, the second term on the right-hand side of the calculation formula for updating $l_{i,j}$ is zero. Therefore, the calculation formula for updating $l_{i,j}$ is $l_{i,j} = a_{i,j}$. Thus, the IC decomposition preconditioning matrix for the matrix appearing in the SPFD method is as follows:

$$M = \tilde{L}\tilde{D}\tilde{L}^T \quad (11)$$

where L and L^T are the lower triangular matrix and the upper triangular matrix, respectively, of the original matrix.

3.1.4. SSOR preconditioning matrix

The SSOR preconditioning matrix is easier to prepare compared to the IC decomposition preconditioning. The original symmetric matrix A can be decomposed by the following equation using the diagonal matrix D , the lower triangular matrix L , and the upper triangular matrix L^T :

$$A = D + L + L^T \quad (12)$$

The SSOR preconditioning matrix (acceleration factor $\Omega = 1$) is defined in the following equation:

$$M = (D + L)D^{-1}(D + L)^T \quad (13)$$

In addition, in the calculation of $Mz^{(i-1)} = r^{(i-1)}$, the vector $z^{(i-1)}$ is obtained by forward and backward substitution, as in the IC decomposition preconditioning.

3.2. Parallelization of forward substitution and backward substitution

Figure 6 shows the calculation procedure for forward and backward substitution in the IC decomposition preconditioning and SSOR preconditioning.

Figure 6: Calculation procedure of forward and backward substitution

As a characteristic of this calculation, in the i -th calculation of the forward substitution, the reference of y_j obtained by the calculation before the i -th result is entered. In the i -th calculation of the backward substitution, the reference of z_j obtained by calculation after the i -th result is entered. Since this method has data dependency, it is not possible to parallelize forward and backward substitution by dividing the matrix in the row direction in the same manner as a matrix-vector product. Multicolor ordering [11, 12] is a method that can completely parallelize the calculations of forward and backward substitution. In this method, it is necessary to classify elements that are not dependent on each other as the same “color” and to

re-assign the number of elements and nodes according to the classification. Therefore, the program becomes complicated, and the calculation overhead increases greatly due to the addition of a color number loop outside of the forward and backward substitution loops.

In the present study, we apply the block decomposition method [19], which is a simple parallelization approach for forward and backward substitution. In the block decomposition method, virtual blocks for each parallel thread are arranged diagonally, and non-zero elements leaking from the block are excluded from preconditioning. By this measure, since there is no data dependency between blocks, it is possible to parallelize forward and backward substitution for each block. Figure 7 shows the lower triangular matrix L and upper triangular matrix L^T . The areas filled with black in Fig. 7 indicate non-zero elements of L and L^T . The area filled with gray is an area in which non-zero elements of L and L^T become zero by block decomposition.

In general, as shown in Fig. 7, in the preconditioning matrix M obtained by block decomposition, as the number of parallel thread increases, the area that becomes zero elements by block decomposition also increases. Therefore, the degree of approximation of the preconditioning matrix M to the original matrix A becomes weak, and the convergence deteriorates. However, in a relatively small-scale computing environment of several tens of parallel threads, it can be expected to be effective for matrices with very narrow bandwidths based on orthogonal differential lattices.

Figure 7: Examples of split matrices (p = number of parallel threads)

4. Numerical experiment

4.1. Calculation condition

In the present study, we calculate a large sparse complex matrix obtained by discretization of the SPFD method by using the CG method with preconditioning such as diagonal scaling pre-

conditioning, IC decomposition preconditioning, SSOR preconditioning, and block IC decomposition preconditioning and block SSOR preconditioning incorporating block decomposition into the IC decomposition preconditioning and the SSOR preconditioning. Then, we compare the results of each method. In addition, in order to compare the difference between the complex matrix (a problem with displacement) and the real matrix (a problem without displacement), we calculate the real matrix using CG methods with these preconditioning methods. Table 1 shows the computer environment used in the present study. The convergence criterion for the COCG method, the COCR method, and the CG method is 10^{-10} .

Table 1: Computing environment

CPU	Intel Xeon Phi 7210, 1.30 GHz
Memory	16 GB (multichannel DRAM)
Operating system	CentOS 7.3
Compiler	GCC version 4.9.4
Number of cores	64
Parallelization	OpenMP API 4.0

4.2. Visualization

Figure 8 shows the internal current density distribution (A/m^2) and internal potential distribution (V) as determined by analysis.

Figure 8: Visualization results

In addition, the arrows in Fig. 8(a) indicate the direction of the current. Based on this visualization, since the analysis is performed by giving a potential difference of 1 V between the two hands, the current density between the two hands is larger than those between other parts, and the potential in the right hand is higher than that in the left hand. Therefore, the current density and potential in the body during electric shock are reasonably calculated.

4.3. Performance evaluation

4.3.1. Performance evaluation in diagonal scaling, IC, and SSOR

Figures 9, 10, and 11 show the convergence history with no preconditioning, diagonal scaling preconditioning, IC decomposition preconditioning, and SSOR preconditioning for the COCG, COCR, and CG methods, respectively. The results for parallel execution are omitted because they are similar to those for sequential execution.

Figure 9 : Convergence history of the relative residual norm for each preconditioning for the COCG method (sequential execution)

Figure 10: Convergence history of the relative residual norm for each preconditioning for the COCR method (sequential execution)

Figure 11: Convergence history of the relative residual norm for each preconditioning for the CG method in the real symmetric case (sequential execution)

The horizontal and vertical axes indicate the number of iterations and the relative residual norm ($\|r_e\|/\|r_0\|$), respectively. These figures show that convergence is greatly improved by applying a preconditioning matrix when solving simultaneous linear equations with the SPFD method. Comparison of the preconditioning methods shows that convergence is better in the order of SSOR preconditioning, IC decomposition preconditioning, diagonal scaling preconditioning, and no preconditioning.

Tables 2 and 3 compare the memory requirement and the number of iterations for each preconditioning method in the COCG, COCR, and CG methods. These results are obtained by sequential execution.

Table 2: Memory requirement and number of iterations for each COCG and COCR preconditioning method (sequential execution)

Preconditioning	Memory requirement (MB)		Number of iterations	
	COCG	COCR	COCG	COCR
None	1,873	1,995	23,888	23,044
Diagonal	2,119	2,364	4,843	4,725
IC	2,119	2,364	1,831	1,825
SSOR	2,119	2,364	1,744	1,702

Table 3: Memory requirement and number of iterations for each CG preconditioning method (sequential execution)

Preconditioning	Memory requirement (MB)	Number of iterations
None	1,074	32,540
Diagonal	1,197	4,847
IC	1,197	1,865
SSOR	1,197	1,709

The number of iterations in Table 2 shows that diagonal scaling preconditioning, IC decomposition preconditioning, and SSOR preconditioning decrease the number of iterations by approximately 80%, 92%, and 93%, respectively, compared to no preconditioning. In addition, in terms of two solvers (COCG and COCR with no preconditioning), the COCR method is better. Moreover, from Tables 2 and 3, the memory requirements of the complex matrix increase by 800 to 1,100 (MB) compared with the real matrix, and the effect of preconditioning for COCG and COCR is weaker than that for CG.

Table 4 (COCG and COCR) and Table 5 (CG) show the computation times for no preconditioning, diagonal scaling preconditioning, IC decomposition preconditioning, and SSOR preconditioning for various degrees of parallelization. From the computation time for each method using a single thread (Table 4), as compared with no preconditioning, it is found that diagonal scaling preconditioning, IC decomposition preconditioning, and SSOR preconditioning decrease the time by approximately 77 to 78%, 88 to 89%, and 89%, respectively. In addition, comparison of computation times of a single thread and 64 parallel threads for each method shows that the times for no preconditioning, diagonal scaling preconditioning, IC decomposition preconditioning, and SSOR preconditioning decrease by approximately 97%, 96%, 66 to

67%, and 66 to 68%, respectively. Parallel performance of IC decomposition preconditioning and SSOR preconditioning is much lower than those of the other two preconditioning methods. This is because there are processes (forward substitution and backward substitution) that cannot be parallelized in the loop process of these iterative methods with IC decomposition preconditioning or SSOR preconditioning.

Moreover, Table 4 and 5 indicate that the computation times of the complex matrix (Table 4) are longer than those of the real matrix (Table 5).

Table 4: Computation time for each COCG and COCR preconditioning method (s)

No. of threads	None		Diagonal		IC		SSOR	
	COCG	COCR	COCG	COCR	COCG	COCR	COCG	COCR
1	77,929	77,992	16,869	17,666	8,909	9,334	8,486	8,692
2	39,643	39,142	8,563	8,954	5,871	6,094	5,605	5,696
4	19,951	19,770	4,420	4,621	4,378	4,477	4,165	4,192
8	10,054	10,082	2,366	2,477	3,625	3,649	3,456	3,442
16	5,301	5,376	1,338	1,467	3,252	3,276	3,107	3,070
32	2,971	3,254	896	977	3,121	3,091	2,933	2,909
64	2,124	2,176	698	735	3,023	3,045	2,886	2,828

Table 5: Computation time for each CG preconditioning method (s)

No. of threads	None	Diagonal	IC	SSOR
1	15,257	2,579	1,781	1,650
2	7,713	1,363	1,322	1,223
4	3,919	762	1,097	1,011
8	2,048	460	986	912
16	1,133	303	921	854
32	651	239	896	829
64	489	219	888	826

4.3.2. Performance evaluation in block IC and block SSOR

Figures 12 (COCG), 13 (COCR), and 14 (CG) and Figures 15 (COCG), 16 (COCR), and 17 (CG) compare the convergence histories for block IC decomposition preconditioning and block SSOR preconditioning, respectively. These figures show that the convergence under preconditioning methods with block decomposition worsens as the number of threads increase.

Figure 12: Convergence history of the relative residual norm for each number of parallel threads for the block ICCOCG method

Figure 13: Convergence history of the relative residual norm for each number of parallel threads for the block ICCOCR method

Figure 14: Convergence history of the relative residual norm for each number of parallel threads for the block ICCG method in the real symmetric case

Figure 15: Convergence history of the relative residual norm for each number of parallel threads for the block SSORCOG method

Figure 16: Convergence history of the relative residual norm for each number of parallel threads for the block SSORCOCR method

Figure 17: Convergence history of the relative residual norm for each number of parallel threads for the block SSORCG method in the real symmetric case

Table 6 shows the memory requirements and the number of iterations for IC decomposition preconditioning and block IC decomposition preconditioning for each number of threads. Table 7 shows these results for SSOR preconditioning and block SSOR preconditioning. Here, these calculations are performed by securing the same memory requirement as the coefficient matrix to the split matrix obtained by block decomposition because estimating the degree of decrease in the number of non-zero elements with the increase in the number of parallels is difficult. Therefore, Tables 6 and 7 show that these memory requirements do not decrease due to parallelization. In addition, these memory requirements increase by approximately 0.7 to 1.1 (GB) compared with the IC decomposition preconditioning and SSOR preconditioning. This is

because block IC and block SSOR must store the split matrix obtained by block decomposition compared with the IC and SSOR. Table 6 shows that the number of iterations of the block IC decomposition preconditioning with 64 threads increases by approximately 25 to 26%, as compared with the number of iterations for the IC decomposition preconditioning. Moreover, Table 7 indicates that the number of iterations of the block SSOR decomposition preconditioning with 64 threads increases by approximately 50 to 53%, as compared with the number of iterations in the SSOR preconditioning. Although the convergence with the SSOR preconditioning in sequential execution is better than that with the IC preconditioning, in parallel execution, the convergence with the block SSOR is worse than that with the block IC.

Table 6: Memory requirement and number of iterations for IC and block IC

Preconditioning	Memory (MB)			No. of iterations		
	COCG	COCR	CG	COCG	COCR	CG
IC	2,119	2,364	1,197	1,831	1,825	1,865
Block IC (2 threads)	3,255	3,500	1,903	1,913	1,868	1,923
Block IC (4 threads)	3,255	3,500	1,903	1,928	1,887	1,940
Block IC (8 threads)	3,255	3,500	1,903	1,967	1,920	1,970
Block IC (16 threads)	3,255	3,500	1,903	2,049	1,995	2,105
Block IC (32 threads)	3,255	3,500	1,903	2,140	2,115	2,270
Block IC (64 threads)	3,255	3,500	1,903	2,307	2,279	2,333

Table 7: Memory requirement and number of iterations for SSOR and block SSOR

Preconditioning	Memory (MB)			No. of iterations		
	COCG	COCR	CG	COCG	COCR	CG
SSOR	2,119	2,364	1,197	1,744	1,702	1,709
Block SSOR (2 threads)	3,225	3,500	1,903	2,163	2,141	2,182
Block SSOR (4 threads)	3,225	3,500	1,903	2,223	2,168	2,231
Block SSOR (8 threads)	3,225	3,500	1,903	2,229	2,207	2,284
Block SSOR (16 threads)	3,225	3,500	1,903	2,386	2,326	2,381
Block SSOR (32 threads)	3,225	3,500	1,903	2,482	2,411	2,478
Block SSOR (64 threads)	3,225	3,500	1,903	2,622	2,557	2,614

Next, Tables 8 and 9 shows the computation time for each number of parallel threads for IC preconditioning, block IC preconditioning, SSOR preconditioning, and block SSOR preconditioning. The computation time of IC and SSOR on the sequential execution is used as the computation time of the block IC and block SSOR. Comparing times for the single thread and 64 parallel threads in Table 8 shows that the reduction rate of computation time of IC is 66 to 67%, whereas that of block IC is 94%. On the other hand, the reduction rate of the computation time of SSOR is 66 to 68%, whereas that of block SSOR is 93%. Therefore, parallelization using block decomposition shortens the computation time of both IC decomposition preconditioning and SSOR preconditioning. Moreover, Tables 8 and 9 show that the computation time of the complex matrix (Table 8) is longer than that of the real matrix (Table 9) because the computational complexity of the complex matrix is greater than that of the real matrix.

Table 8: Computation time for each parallel of IC and SSOR with block decomposition (COCG and COCR)

No. of threads	IC		Block IC		SSOR		Block SSOR	
	COCG	COCR	COCG	COCR	COCG	COCR	COCG	COCR
1	8,909	9,334	8,909	9,334	8,486	8,692	8,486	8,692
2	5,871	6,094	4,767	4,883	5,605	5,696	5,347	5,561
4	4,378	4,477	2,539	2,622	4,165	4,192	2,888	2,952
8	3,625	3,649	1,436	1,468	3,456	3,442	1,589	1,659
16	3,252	3,276	891	946	3,107	3,070	1,010	1,070
32	3,121	3,091	686	703	2,933	2,909	737	765
64	3,023	3,045	561	587	2,886	2,828	604	620

Table 9: Computation time for each parallel of IC and SSOR with block decomposition (CG)

No. of threads	IC	Block IC	SSOR	Block SSOR
1	1,781	1,781	1,650	1,650
2	1,322	995	1,223	1,115
4	1,097	601	1,011	653
8	986	384	912	416
16	921	280	854	298
32	896	240	829	249
64	888	236	826	249

5. Conclusion

In the present study, we calculated a symmetric positive-definite matrix with approximately eight million degrees of freedom obtained by the SPFD method, which is a current density analysis method that uses a numerical human model. In order to improve the computational performance in a parallel computing environment, we evaluated the performance of the COCG method and the COCR method with diagonal scaling preconditioning, IC decomposition preconditioning, and SSOR preconditioning from the perspectives of the number of iterations, computation time, and required memory. All preconditioning methods were able to decrease the number of iterations and the computation time. Moreover, analysis could be performed with a memory requirement of 4 GB or less by using the CRS format. In addition, the SSOR preconditioning shows the best convergence among the preconditioning methods. Furthermore, we evaluated the performance of these iterative methods with block IC decomposition preconditioning and block SSOR preconditioning, which can parallelize the calculation of the forward and backward substitution. As a result, these methods were found to be able to address the problem due to parallelizing the IC decomposition preconditioning and SSOR preconditioning.

Using a computing environment with up to 64 parallel threads, we examined the effectiveness of each method to solve the matrix obtained by the SPFD method in the numerical human model. Among the preconditioning methods used in this analysis, the block SSOR and SSOR

methods in the case of a single thread and the block IC method in the case of multiple threads were the fastest calculation methods for the complex matrix. Moreover, when these preconditioning methods were used, the COCG method is more effective than the COCR method.

In the present study, we examined iterative methods using various preconditioning methods. In the future, we intend to investigate the matrix storage format, such as the diagonal storage format, and the parallelization method of preconditioning, such as multi-color ordering. In addition, in addition to OpenMP, we intend to investigate MPI, which is an inter-node parallelization method. As solvers capable of calculating the current analysis more quickly, we also intend to investigate multi-grid methods with constant convergence, regardless of the size of the matrix.

Acknowledgments

The present study was subsidized in part by a JSPS Grant-in-Aid for Research (Basic Research B, No. 17H03256), and the computing environment used in the present study was funded in part by a JKA Research Assistance Project Grant (FY2016) from the Public Interest Foundation Corporation.

References

- [1] Ministry of Economy, Trade and Industry, Commercial Distribution Security Group, Power Safety Division: *Heisei 27 Year Electrical Safety Statistics*, 2016. (in Japanese)
- [2] T. D. Bracken, G. G. Sias, C. Kim, R. S. Senior, and R. M. Patterson: Survey of electrical utility worker body impedance, *IEEE Trans. Power Delivery*, 23 (2008), pp.1251-1259.
- [3] P. J. Dimbylow: Induced current densities from low-frequency magnetic fields in a 2mm resolution, anatomically realistic model of the body, *Phys. Med. Biol.*, 43 (1998), pp.221-230.
- [4] H. Tarao, H. Miyamoto, L. Korpinen, N. Hayashi, and K. Isaka: Simple estimation of induced electric fields in nervous system tissues for human exposure to non-uniform electric fields at power frequency, *Phys. Med. Biol.*, 61 (2016), pp.4438-4451.
- [5] P. J. Dimbylow: Development of the female voxel phantom, NAOMI, and its application to calculations of induced current densities and electric fields from applied low frequency magnetic and electric fields, *Phys. Med. Biol.*, 50 (2005), pp.1047-1070.
- [6] I. Hamamoto, N. Hayashi, and H. Tarao: Study on contact current and human body resistance in commercial frequency using human detail model, *Bull. Faculty of Eng., Univ. Miyazaki* ., 41 (2012), pp.41-47. (in Japanese)
- [7] H. Tarao, N. Hayashi, and I. Hamamoto: Comparison of numerical analysis results and measurement results of human body internal resistance at commercial frequency, *J. Inst. Elec. Eng. Japan A*, 131:3 (2011), pp.178-184. (in Japanese)
- [8] A. H. Tarao, H. Kusuti, L. Korpien, N. Hayashi, and K. Isaka: Effects of tissue conductivity and electrode area on internal electric fields in a numerical human model for ELF contact current exposures, *Phys. Med. Biol.*, 57 (2012), pp.2981-2996.

- [9] H. A. van der Vorst, J. B. M. Melissen: A Petrov-Galerkin type Method for Solving $Ax=b$, Where A is Symmetric Complex, *IEEE Transaction on Magnetics*, 26:2, (1990), pp. 706-708.
- [10] T. Sogabe and S.-L. Zhang: A COCR Method for Solving Complex Symmetric Linear Systems, *J. Comput. Appl. Math.*, 199:2, pp.297-303, 2007.
- [11] Tokyo Metropolitan University: Numerical Dosimetry Evaluation Report on indirect bonding between electromagnetic field and human body in the intermediate frequency band, 2012.(in Japanese)
- [12] T. Iwashita, Y. Nakanishi, and M. Shimasaki: Comparison criteria for parallel orderings in ILU preconditioning, *SIAM J. Sci. Comput.*, 26:4 (2005), pp.1234-1260.
- [13] T. Tsuburaya, Y. Okamoto, and S. Sato: Parallelization of preconditioned MRTR method with Eisenstat's technique by means of algebraic multicolor ordering, *IEEE Trans. on Magn.*, 51:3 (2015), DOI: 10.1109/TMAG.2014.2360036.
- [14] M. Nomura, H. Tarao, A. Takei: Performance Evaluation of Numerical Current Analysis with Anatomical Human Model, *The 36th JSST Annual International Conference on Simulation Technology*, USB.
- [15] T. Nagaoka, S. Watanabe, K. Sakurai, E. Kunieda, S. Watanabe, M. Taki, and Y. Yamana: Development of realistic high-resolution whole-body voxel models of Japanese adult males and females of average height and weight, and application of models to radio-frequency electromagnetic dosimetry, *Phys. Med. Biol.*, 49 (2004), pp.1-15.
- [16] A. Hirata, K. Yamazaki, S. Hamada, Y. Kamimura, H. Tarao, K. Wake, Y. Suzuki, and O. Fujiwara: Intercomparison of induced fields in Japanese male model for ELF magnetic field exposures, *Rad. Prot. Dos.*, 138 (2010), pp.237-244.
- [17] It is foundation. (n.d.). *TISSUE PROPERTIES*, Retrieved March 8, 2018, from <https://www.itis.ethz.ch/virtual-population/tissue-properties/database/tissue-frequency-chart/>
- [18] A. R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, U.S.A., 1994.
- [19] T. Iwasaki, M. Kanazawa, and M. Shimasaki: Development of Parallelized ICCG Solver for Finite Element Analysis, *Information technology center's annual report*, 2(2002), pp.55-62.(in Japanese)