Numerical Investigation of Preconditioning for Iterative Methods in Linear Systems Obtained by Extended Element-Free Galerkin Method

Taku Itoh¹*, Ayumu Saitoh², Soichiro Ikuno³, Atsushi Kamitani²

¹College of Industrial Technology, Nihon University, Japan
²Graduate School of Science and Engineering, Yamagata University, Japan
³School of Computer Science, Tokyo University of Technology, Japan

*itoh.taku@nihon-u.ac.jp

Received: February 1, 2016; Accepted: June 22, 2016; Published: April 7, 2017

Abstract. In an extended element-free Galerkin method (X-EFG), the essential and natural boundary conditions can be imposed by the collocation based method. However, a coefficient matrix of linear systems obtained by X-EFG become asymmetric, although a symmetric structure exists in a part of the coefficient matrix. In fact, the structure of the coefficient matrices almost becomes symmetric when the size of linear systems is large. Hence, efficient effects may be obtained by using preconditioning for symmetric matrices. The purpose of the present study is to investigate effects of preconditioning for symmetric matrices to linear systems obtained by X-EFG. To this end, the incomplete Cholesky factorization (IC) is applied to the linear systems by regarding the coefficient matrix as symmetric one. In numerical experiments, it is found that the linear systems obtained by X-EFG can be efficiently solved by using IC as preconditioning for GMRES(m) and Bi-CGSTAB. In some cases, the efficiency of IC is superior than that of the incomplete LDU factorization as preconditioning for these iterative methods.

Keywords: Meshless methods, Element-free Galerkin method, Preconditioning for iterative methods, Asymmetric coefficient matrix

1. Introduction

In meshless methods as typified by the element-free Galerkin method (EFG) [1] and the meshless local Petrov-Galerkin method (MLPG) [2], elements of a geometrical structure are no longer necessary. In EFG, to impose the essential boundary condition, the method of Lagrange multipliers is usually employed [1].

Recently, EFG has been reformulated together with a new imposing method for the essential boundary condition [3]. Since this method is based on the collocation method,
implementation of a code of the one is easier than that of Lagrange multipliers. In addition, by another reformulation of EFG [4], the natural boundary condition can also be imposed by the collocation based method. We refer to this as an extended EFG (X-EFG).

As mentioned above, in X-EFG, the essential and natural boundary conditions can be imposed by the collocation based method. However, coefficient matrices of linear systems obtained by X-EFG become asymmetric. Note that a symmetric structure exists as a part of the coefficient matrix. In fact, the structure of the coefficient matrices almost becomes symmetric in large linear systems. By utilizing a part of the symmetric structure for solving linear systems, an incomplete Cholesky factorization (IC) is applied as preconditioning for iterative methods in [4]. Note that an IC is always applied to linear systems whose coefficient matrix is symmetric. Hence, a coefficient matrix of linear systems obtained by X-EFG has to be regarded as symmetric matrix to apply the IC. Note that there are some patterns for regarding the coefficient matrix as symmetric one. Although one of the patterns was investigated in [4], more efficient preconditioning may be found in the other patterns.

The purpose of the present study is to investigate effects of preconditioning for symmetric matrices to linear systems obtained by X-EFG. To this end, the IC is applied to the linear systems by regarding the coefficient matrix as symmetric one.

2. Extended Element-Free Galerkin Method

Recently, some kinds of formulations for X-EFGs have been proposed [3, 4]. In this paper, we employ an X-EFG based on [4]. In what follows, we describe a formulation for the X-EFG [4]. Linear systems obtained by the X-EFG will be shown as (15).

For simplicity, we consider a 3-dimensional (3D) Poisson problem:

\[ -\Delta u = f \quad \text{in} \quad V, \]  \hspace{1cm} (1)
\[ u = \bar{u} \quad \text{on} \quad S_D, \] \hspace{1cm} (2)
\[ \frac{\partial u}{\partial n} = \bar{q} \quad \text{on} \quad S_N, \] \hspace{1cm} (3)

where \( V \) is a region bounded by a simple closed surface \( \partial V \) that consists of both \( S_D \) and \( S_N \). Here, \( S_D \) and \( S_N \) satisfy \( S_D \cup S_N = \partial V \) and \( S_D \cap S_N = \emptyset \). In addition, \( \bar{u} \) and \( \bar{q} \) are known functions on \( S_D \) and \( S_N \), respectively, and \( n \) is an outward-unit-normal on \( \partial V \). Furthermore, \( f(x) \) is a given function on \( V \), and \( x = [x, y, z]^T \).

From (1), the weak form is derived as

\[ \text{Find } w \text{ s.t. } w|_{\partial V = 0} : \int_V \nabla w \cdot \nabla u \, d^3x = \int_V w f \, d^3x. \] \hspace{1cm} (4)

where \( w(x) \) is a test function. To discretize (4), nodes, \( x_1, x_2, \ldots, x_N \), are first placed both in \( V \) and on \( \partial V \). In addition, shape functions \( \phi_i(x) \) corresponding to \( x_i \) \((i = 1, 2, \ldots, N)\) are determined from the nodes [1, 2]. Here, \( N \) is the number of nodes in \( V \cup \partial V \).

Let us first discretize the weak form (4). To this end, we assume that \( u \) and \( w \) can be expanded with \( \phi_i(x) \) \((i = 1, 2, \ldots, N)\) as follows:

\[ u(x) = \sum_{i=1}^{N} \hat{u}_i \phi_i(x), \quad w(x) = \sum_{i=1}^{N} \hat{w}_i \phi_i(x). \] \hspace{1cm} (5)
By substituting (5) into (4), we obtain \((\hat{\omega}, B\hat{u} - f) = 0\), where \(\hat{\omega} = [\hat{\omega}_1, \hat{\omega}_2, \ldots, \hat{\omega}_N]^T\), and \(\hat{u} = [\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_N]^T\). In addition, \(B\) and \(f\) are defined as

\[
B \equiv \sum_{i=1}^{N} \sum_{j=1}^{N} \int_V \nabla \phi_i \cdot \nabla \phi_j \, d^3 x \, e_j^T, \quad f \equiv \sum_{i=1}^{N} \int_V \phi_i \, f \, d^3 x \, e_i, \tag{6}
\]

where \(\{e_1, e_2, \ldots, e_N\}\) denotes the \(N\)-dimensional orthonormal system.

Next, the constraint \(u|_{\partial V} = 0\) in (4) is discretized. To this end, the constraint is rewritten as the equivalent proposition:

\[
\int_{\partial V} \beta(s, t) : \int_{\partial V} \beta(s, t) w(x(s, t)) \, dS = 0, \tag{7}
\]

and an arbitrary function \(\beta(s, t)\) is assumed to be contained in \(\text{span}(N_1, N_2, \ldots, N_M)\), where \(N_1(s, t), N_2(s, t), \ldots, N_M(s, t)\) are linearly independent functions on \(\partial V\), and \(M\) denotes the number of nodes on \(\partial V\). Here, \(s\) and \(t\) are parameters for representing \(\partial V\). By using \(N_1(s, t), N_2(s, t), \ldots, N_M(s, t)\), (7) can be discretized as

\[
(\hat{\omega}, e_k) = 0 \quad (k = 1, 2, \ldots, M), \tag{8}
\]

where \(e_k\) \((k = 1, 2, \ldots, M)\) are defined as

\[
e_k \equiv \sum_{i=1}^{N} \int_{\partial V} N_k(s, t) \phi_i(x(s, t)) \, dS \, e_i. \tag{9}
\]

Note that (8) indicate \(\hat{\omega} \in \mathcal{V}^\perp\), where \(\mathcal{V} = \text{span}(e_1, e_2, \ldots, e_M)\). Hence, the weak form (4) can be discretized as

\[
\hat{\omega} \in \mathcal{V}^\perp : (\hat{\omega}, B\hat{u} - f) = 0. \tag{10}
\]

Since \((\mathcal{V}^\perp)^\perp = \mathcal{V}\), (10) can be written as \(B\hat{u} - f \in \mathcal{V}\). Therefore there exists a \(M\)-dimensional vector \(\hat{b}\) that satisfies

\[
B\hat{u} + C\hat{b} = f, \tag{11}
\]

where \(C \equiv [e_1, e_2, \ldots, e_M]\) and \(\hat{b} \equiv [\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_M]^T\).

Finally, the essential and natural boundary conditions, (2) and (3), are simultaneously discretized. By the similar procedures for discretizing the constraint \(u|_{\partial V} = 0\), (2) and (3) can be discretized as

\[
D^T \hat{u} = g, \tag{12}
\]

where \(D \equiv [d_1, d_2, \ldots, d_M]\), and \(g \equiv [g_1, g_2, \ldots, g_M]^T\). In addition, \(d_k\) and \(g_k\) \((k = 1, 2, \ldots, M)\) are defined as

\[
d_k \equiv \begin{cases} \sum_{i=1}^{N} \int_{\partial V} N_k(s, t) \phi_i(x(s, t)) \, dS \, e_i, & \text{for } x_k \in S_D, \\ \sum_{i=1}^{N} \int_{\partial V} N_k(s, t) \frac{\partial \phi_i}{\partial n} (x(s, t)) \, dS \, e_i, & \text{for } x_k \in S_N, \end{cases} \tag{13}
\]
and

\[
g_k = \begin{cases} 
\int_{S_D} N_k(s, t) \bar{u}(s, t) \, dS, & \text{for } x_k \in S_D, \\
\int_{S_N} N_k(s, t) \bar{q}(s, t) \, dS, & \text{for } x_k \in S_N,
\end{cases}
\]

(14)

respectively. Note that, for the case where \( x_k \in S_D \), \( d_k \) is exactly the same as \( c_k \). Equations (11) and (12) can be written in the form,

\[
\begin{bmatrix} B & C \\
D^T & 0 \end{bmatrix} \begin{bmatrix} \hat{u} \\
\hat{v} \end{bmatrix} = \begin{bmatrix} f \\
g \end{bmatrix}.
\]

(15)

Equation (15) is a discretized form of the Poisson problem by the X-EFG.

Note that, in this paper, the \( \delta \)-functions defined on \( \partial V \) are employed as \( N_k \) \((k = 1, 2, \ldots, M)\) so that essential and natural boundary conditions are satisfied exactly [4]. The explicit form of \( N_k(s, t) \) is given as

\[
N_k(s, t) = \frac{\delta(s-s_k)\delta(t-t_k)}{\left| \frac{\partial x}{\partial s} \times \frac{\partial x}{\partial t} \right|} \quad (k = 1, 2, \ldots, M).
\]

(16)

Note that, on \( \partial V \), the \( k \)th boundary node \( x_k \) is represented by \( s_k \) and \( t_k \), i.e., \( x_k = x(s_k, t_k) \). By using (16), \( C, d_k \) and \( g_k(k = 1, 2, \ldots, M) \) can be rewritten as

\[
C = \sum_{i=1}^{N} \sum_{k=1}^{M} \phi_i(x(s_k, t_k)) e_i \bar{e}_k^T,
\]

(17)

\[
d_k = \begin{cases} 
\sum_{i=1}^{N} \phi_i(x(s_k, t_k)) e_i, & \text{for } x_k \in S_D, \\
\sum_{i=1}^{N} \frac{\partial \phi_i}{\partial n}(x(s_k, t_k)) e_i, & \text{for } x_k \in S_N,
\end{cases}
\]

(18)

\[
g_k = \begin{cases} 
\bar{u}(s_k, t_k), & \text{for } x_k \in S_D, \\
\bar{q}(s_k, t_k), & \text{for } x_k \in S_N,
\end{cases}
\]

(19)

where \( \{\bar{e}_1, \bar{e}_2, \ldots, \bar{e}_M\} \) is the \( M \)-dimensional orthonormal system.

### 3. Structure of Linear Systems Obtained by X-EFG

In this paper, we consider solving the linear systems (15). Here, we rewrite (15) as

\[
Ax = b,
\]

(20)

where

\[
A \equiv \begin{bmatrix} B & C \\
D^T & 0 \end{bmatrix}, \quad x \equiv \begin{bmatrix} \hat{u} \\
\hat{v} \end{bmatrix}, \quad \text{and } b \equiv \begin{bmatrix} f \\
g \end{bmatrix}.
\]
In the coefficient matrix in the linear systems, \( B \) is a symmetric matrix, and a part of \( C \) and \( D \) is exactly the same. Namely, although the coefficient matrix \( A \) is asymmetric, almost all part of \( A \) is symmetric.

We consider solving the linear systems by using a preconditioned iterative method. In addition, to utilize the almost symmetric structure, we apply an IC as preconditioning by regarding \( A \) as a symmetric matrix. There are some patterns for regarding the coefficient matrix as symmetric one, e.g.,

\[
A \approx \begin{bmatrix} B & D \\ D^T & O \end{bmatrix}, \quad (21)
\]

\[
A \approx \begin{bmatrix} B & C \\ C^T & O \end{bmatrix}, \quad (22)
\]

\[
A \approx \begin{bmatrix} B & O \\ O & I \end{bmatrix}, \quad (23)
\]

and

\[
A \approx \frac{1}{2}(A + A^T) = \begin{bmatrix} B & \frac{(C + D)/2}{(C^T + D^T)/2} \\ \frac{(C + D)/2}{(C^T + D^T)/2} & O \end{bmatrix}. \quad (24)
\]

Here, \( I \) is a unit matrix. By selecting a symmetric matrix from (21)–(24) as an approximation of the coefficient matrix \( A \), we can obtain preconditioning matrices \( \tilde{L}\tilde{D}\tilde{L}^T \), although the exact structure of \( A \) is asymmetric. Here, \( \tilde{L} \) is a lower triangular matrix, and \( \tilde{D} \) is a diagonal matrix. The matrices \( \tilde{L} \) and \( \tilde{D} \) are generated by IC.

4. Numerical Experiments

4.1. Problem Setting

In this section, some numerical experiments are conducted to investigate the performance of preconditioned iterative methods for solving (15). To apply an IC as preconditioning, the coefficient matrix in (15) is regarded as a symmetric matrix. We adopt four-types of symmetric structures shown in (21)–(24). In addition, we adopt an IC algorithm shown in [5]. In this algorithm, a lower triangular matrix \( \tilde{L} \) and a diagonal matrix \( \tilde{D} \) are generated so that

\[
\tilde{l}_{ii}\tilde{d}_i = 1 \quad (25)
\]

is satisfied, where \( \tilde{l}_{ii} \) is the \((i,i)\)-element of \( \tilde{L} \) and \( \tilde{d}_i \) is the \(i\)th diagonal part of \( \tilde{D} \). Note that the non-zero structure of \( \tilde{L} \) is the same as that of each of the symmetric matrices in (21)–(24) except for the diagonal part. Even if the diagonal part of each of the symmetric matrices in (21)–(24) is zero, (25) is satisfied, i.e., \((i,i)\)-elements of \( \tilde{L} \) always become non-zero values.

Although we employ IC as preconditioning, iterative methods for asymmetric linear systems must be chosen. This is because the coefficient matrix in (15) is asymmetric. As iterative methods for asymmetric linear systems, we employ GMRES\((m)\) [6] and Bi-CGSTAB [7]. Here, \( m \) is some fixed integer parameter and GMRES\((m)\) restarts every \( m \) steps [6]. In what follows, we refer to GMRES\((m)\) with IC and Bi-CGSTAB with IC as IC-GMRES\((m)\) and IC-Bi-CGSTAB, respectively.
In (1)–(3), we assume \( V = (-0.5, 0.5) \times (-0.5, 0.5) \times (-0.5, 0.5) \). In addition, the natural boundary condition is imposed on \( S_N \). As \( S_N \), we consider the following three cases.

**Case 1:** \( S_N \equiv S_N^1 \).

**Case 2:** \( S_N \equiv S_N^1 \cup S_N^2 \).

**Case 3:** \( S_N \equiv S_N^1 \cup S_N^2 \cup S_N^3 \).

Here, \( S_N^1, S_N^2, \) and \( S_N^3 \) are defined as follows (see Figs. 1(a)–(c)):

\[
S_N^1 : -0.25 \leq x \leq 0.25, -0.25 \leq y \leq 0.25 \text{ and } z = 0.5,
\]

\[
S_N^2 : -0.25 \leq x \leq 0.25, -0.25 \leq y \leq 0.25 \text{ and } z = -0.5,
\]

\[
S_N^3 : -0.25 \leq z \leq 0.25, -0.25 \leq x \leq 0.25 \text{ and } y = 0.5.
\]
The essential boundary condition is imposed on \( S_D = \partial V - S_N \). Here, the functions \( f(x) \), \( \bar{u} \) and \( \bar{q} \) are determined so that the analytic solution of the 3D Poisson problem may be \( u = \exp(-x^2 - y^2 - z^2) \).

In this paper, the shape functions \( \phi_i(x)(i = 1, 2, \ldots, N) \) in (5) are determined by the moving least-squares (MLS) approximation [1, 2]. Namely, the shape functions are represented as

\[
\phi_i(x) = p^T(x)A^{-1}(x)b_i(x),
\]

where \( p^T(x) = [1 \ x \ y \ z] \), and the matrix \( A(x) \) and the vector \( b_i(x) \) are defined as

\[
A(x) = \sum_{k=1}^N w_k(x)p(x_k)p^T(x_k),
\]

\[
b_i(x) = w_i(x)p(x_i),
\]

where \( w_i(x) = w(|x - x_i|) \) is a weight function. We adopt a spline weight function [2],

\[
w(r) = \begin{cases} 
1 - 6\left(\frac{r}{R}\right)^2 + 8\left(\frac{r}{R}\right)^3 - 3\left(\frac{r}{R}\right)^4 & (r \leq R), \\ 0 & (r > R), 
\end{cases}
\]

where \( R \) denotes a support radius. In addition, the support radius is determined as

\[
R = \alpha d_{\text{min}},
\]

where \( \alpha \) is a parameter, and \( d_{\text{min}} \) is the minimum distance between two nodes. Here, \( \alpha \) is set so that the domain \( V \) is completely filled by shape functions.

The partial derivative of \( \phi_i(x) \) with respect to \( X(=x, y \text{ or } z) \) can be obtained as

\[
\phi_{i,X}(x) = p^T_X(x)A^{-1}(x)b_i(x) + p^T(x)[A^{-1}_X(x)b_i(x) + A^{-1}(x)b_{i,X}(x)],
\]

where \( A^{-1}_X(x) \) represents the derivative of the inverse of \( A \) with respect to \( X \) [2]. \( A^{-1}_X(x) \) is obtained by

\[
A^{-1}_X(x) = -A^{-1}(x)A_X(x)A^{-1}(x).
\]

To construct the linear systems (15), the integration for \( B \) and \( f \) in (6) must be evaluated. To this end, a cubic cell structure being independent of the nodes is used [1], and the Gauss-Legendre quadrature is employed. The number \( N_G \) of quadrature points depends on the number \( m_{\text{in}} \) of nodes in a cell. To handle \( N_G \), we adopt a similar criterion in [1], i.e., \( N_G = n^3_G \), where \( n_G = \lfloor \sqrt{m_{\text{in}}} + 0.5 \rfloor + 2 \). In addition, the number \( N_C \) of cells is set as \( N_C = m^3_C \), where \( m_C = \lfloor N^{1/3} \rfloor \).

### 4.2. Performance Evaluation

In what follows, the number \( N \) of nodes is fixed at 274625(= 65^3), unless otherwise noted. The nodes, \( x_1, x_2, \ldots, x_N \), are uniformly placed both in \( V \) and on \( \partial V \). In this case, the number \( M \) of boundary nodes is 24578. In addition, \( M_N = 1089, 2178, \text{ and } 3267 \) for Cases 1, 2, and 3, respectively. Here, \( M_N \) denotes the number of boundary nodes contained in \( S_N \).
Note that, in (15), we construct the coefficient matrix so that the last $M_N$ columns of $D$ are different from those of $C$.

Computations were performed on a computer equipped with a 3.4 GHz Intel Core i7 4930 processor, 32 GB RAM, CentOS Linux ver. 6.5, and g++ ver. 4.4.7 with double-precision arithmetic. Note that we only used a single core of this processor in the following experiments, and compiler options were set as “-03 -Wall -m64.” In addition, the elements of matrices related for solving the linear systems (15) were stored by the compressed row storage (CRS) method [8].

### 4.2.1. Dependence of Accuracy on Parameter $\alpha$

To determine appropriate values of $\alpha$ in (30), let us first investigate the dependence of the accuracy of $u$ and $q$ on the parameter $\alpha$. To investigate the accuracy of $u$ and $q$, we employ relative errors, $\varepsilon_u$ and $\varepsilon_q$, calculated by the following equations:

$$
\varepsilon_u = \frac{\|u^e - u^n\|}{\|u^e\|}, \quad \varepsilon_q = \frac{\|q^e - q^n\|}{\|q^e\|}.
$$

(33)

Here, $\|f\|$ denotes the maximum norm of $f$, and $\varepsilon_u$ and $\varepsilon_q$ are calculated in $V \cup \partial V$ and on $S_N$, respectively. In addition, $u^e$ and $u^n$ denote exact and numerical solutions for $u$, respectively, and $q^e$ and $q^n$ denote exact and numerical solutions for $q$, respectively.

In Figs. 2(a) and (b), the relative errors, $\varepsilon_u$ and $\varepsilon_q$, are plotted as a function of the parameter $\alpha$ for Cases 1–3. The range is $1.3 \leq \alpha \leq 1.9$. Note that, to obtain $u^n$ and $q^n$, GMRES($m$) with an incomplete LDU factorization (ILDU) is employed, since the linear systems (15) cannot be solved by direct solvers such as the LU factorization because of memory limits of the computer. As the ILDU, we employ the same algorithm shown in Section 4.2.3. It must be noted here that, for $\alpha = 1.8$, the linear systems (15) cannot be solved in all cases. For this reason, we do not plot $\varepsilon_u$ and $\varepsilon_q$ for $\alpha = 1.8$ in Figs. 2(a) and (b).
Figure 3: The relative residual norms for Case 3. The results are obtained by (a) IC-GMRES\((m)\) with \(\alpha = 1.5\), (b) IC-Bi-CGSTAB with \(\alpha = 1.5\), (c) IC-GMRES\((m)\) with \(\alpha = 1.9\), and (d) IC-Bi-CGSTAB with \(\alpha = 1.9\).

We see from Fig. 2(a) that values of \(u\) for Cases 1–3 are almost the same for each \(\alpha\), and \(u^\alpha\) is the most accurate for \(\alpha = 1.9\) in this range. Similarly, in Fig. 2(b), values of \(q\) for Cases 1–3 are almost the same for each \(\alpha\), and \(q^\alpha\) is the most accurate for \(\alpha = 1.9\) in this range. From these results, we adopt \(\alpha = 1.9\). In addition, to investigate the performance in the other sparseness of the coefficient matrix in (15), we also adopt \(\alpha = 1.5\). This is because \(u^\alpha\) for \(\alpha = 1.5\) is the second most accurate in this range.

4.2.2. Selection of Preconditioning Matrices from (21)–(24)

To select appropriate symmetric preconditioning matrices from (21)–(24), we here investigate the performance of IC-GMRES\((m)\) and IC-Bi-CGSTAB with the preconditioning matrices. To this end, relative residual norms \(\|r_k\|/\|b\|\) for both methods are plotted as a function of the number \(k\) of iterations and are depicted in Fig. 3 for Case 3. In the figure, pairs of a
solver and $\alpha$ for (a)–(d) are as follows:

(a) IC-GMRES($m$) and $\alpha = 1.5$, 
(b) IC-Bi-CGSTAB and $\alpha = 1.5$, 
(c) IC-GMRES($m$) and $\alpha = 1.9$, 
(d) IC-Bi-CGSTAB and $\alpha = 1.9$.

In all cases, the initial vector $x_0$ required in iterative methods is guessed as $x_0 = 0$, and we set $m = 100$. Note that the iterations of IC-GMRES($m$) and IC-Bi-CGSTAB are repeated until

$$\|r_k\|/\|b\| \leq \epsilon_{\text{tol}}$$

is satisfied. Here, $\epsilon_{\text{tol}}$ is fixed at $\epsilon_{\text{tol}} = 10^{-8}$. Here, $r_k$ is the $k$th residual vector that can be obtained in the algorithms of both iterative methods [6, 7], and $b$ is the right-hand side vector of the linear systems in (20). Note that the same parameters and settings as described in this paragraph are used in all numerical experiments.

We see from Figs. 3(a)–(d) that the relative residual norms obtained with (23) satisfy (34) at last in all cases. In addition, in Figs. 3(a) and (b), the performance of (21) and (24) is particularly efficient as preconditioning matrices for IC-GMRES($m$) and IC-Bi-CGSTAB. On the other hand, Figs. 3(c) and (d) show that the performance of (22) is the most efficient. From these results, we consider that (23) is not appropriate as preconditioning matrix of GMRES($m$) and Bi-CGSTAB for solving the linear systems (15) in Cases 1–3. Hence, in what follows, we show the results obtained with (21), (22) and (24).

### 4.2.3. Dependence of Computation Time and Number of Iterations on the Size of Coefficient Matrix

To investigate the performance of IC-GMRES($m$) and IC-Bi-CGSTAB with the symmetric matrices shown in (21), (22) and (24), the computation time and number of iterations for solving the linear systems by IC-GMRES($m$) or IC-Bi-CGSTAB are plotted as a function of the size of coefficient matrix in (15) and are depicted in Figs. 4, 5 and 6 for Cases 1, 2, and 3, respectively. In these figures, pairs of a solver and $\alpha$ for (a)–(d) are the same as shown in Section 4.2.2. Note that performing IC or ILDU is contained in the computation time.

In Figs. 4, 5 and 6, the computation time and number of iterations obtained by GMRES/Bi-CGSTAB with an ILDU as preconditioning are also shown. As the ILDU, we adopt an algorithm shown in [9]. In this algorithm, a lower triangular matrix $\hat{L}$, a diagonal matrix $\hat{D}$, and an upper triangular matrix $\hat{U}$ are generated so that

$$\hat{d}_i = 1/\hat{l}_{ii} = 1/\hat{u}_{ii}$$

is satisfied, where $\hat{l}_{ii}$ and $\hat{u}_{ii}$ denote the $(i, i)$-elements of $\hat{L}$ and $\hat{U}$, respectively. In addition, $\hat{d}_i$ is the $i$th diagonal part of $\hat{D}$. Note that the non-zero structure of $\hat{L} + \hat{U}$ is the same as that of the coefficient matrix $A$ in (20) except for the diagonal part. Even if the diagonal part of $A$ is zero, (35) is satisfied, i.e., $(i, i)$-elements of $\hat{L}$ and $\hat{U}$ always become non-zero values. In what follows, we refer to GMRES($m$) with ILDU and Bi-CGSTAB with ILDU as ILDU-GMRES($m$) and ILDU-Bi-CGSTAB, respectively.

We see from Figs. 4, 5 and 6 that, for $N + M < 10^5$, the computation times of ILDU-based methods are less than those of IC-based methods in almost all cases. However, the
computation times of IC-based methods are close to those of ILDU-based methods. In addition, for \( N + M > 10^5 \), the computation times of IC-based methods are less than those of ILDU-based methods in a lot of cases. In particular, IC-Bi-CGSTAB based on (24) and IC-GMRES(m) based on (22) are efficient for \( \alpha = 1.5 \) and \( \alpha = 1.9 \), respectively. In the most efficient case, the computation time of IC-GMRES(m) based on (22) is about 4.2 times faster than that of ILDU-GMRES(m) as shown in Fig. 5(c).

From the above results, it is found that the linear systems (15) can be efficiently solved by using IC-GMRES(m) and IC-Bi-CGSTAB together with employing an appropriate symmetric matrix for IC. In some cases, the efficiency of IC-based iterative methods is superior than that of ILDU-based ones. In particular, the performance of IC may be efficient for \( N + M \gtrsim 10^5 \).

As described above, for \( N + M > 10^5 \), there are many cases that the computation times of IC-based methods are less than those of ILDU-based methods. However, the numbers \( k \) of iterations of IC-based methods are greater than those of ILDU-based methods in almost
all cases as shown in Figs. 4, 5 and 6. This is because the computational cost of ILDU is about 2 times larger than that of IC. In fact, for $\alpha = 1.5$, the average of computation times for ICs is about 13.1 (s), and the computation time of ILDU is about 26.0 (s). In addition, for $\alpha = 1.9$, the average of computation times for ICs is about 15.1 (s), and the computation time of ILDU is about 28.1 (s). Hence, the total computation times of IC-based methods tend to be less than those of ILDU-based methods for the case where the number of iterations of IC-based method is slightly greater than that of ILDU-based method. This is also shown in Figs. 7 and 8.

4.2.4. Influence of the Number $M_N$ of Nodes Contained in $S_N$ to Efficiency of IC-based Iterative Methods

To investigate the influence of the number $M_N$ of nodes contained in $S_N$ to the computation time and number of iterations, the computation time and number of iterations for solving
linear systems by IC-GMRES(\(m\)) or IC-Bi-CGSTAB are plotted as a function of the ratio \(M_N / (N + M)\) and are depicted in Figs. 7(a)–(d), i.e., the horizontal axes in these figures are

\[
R_N \equiv 100 \frac{M_N}{N + M} \% \tag{36}
\]

and \(N + M\) is fixed at 299203 (\(N = 274625\) and \(M = 24578\)). To increase the number \(M_N\) of nodes contained in \(S_N\), we consider the following three cases similarly with Cases 1–3.

**Case 4:** \(S_N \equiv S^1_N \cup S^2_N \cup S^3_N \cup S^4_N\).

**Case 5:** \(S_N \equiv S^1_N \cup S^2_N \cup S^3_N \cup S^4_N \cup S^5_N\).

**Case 6:** \(S_N \equiv S^1_N \cup S^2_N \cup S^3_N \cup S^4_N \cup S^5_N \cup S^6_N\).
Figure 7: Computation time and number of iterations as a function of the ratio of the number $M_N$ of nodes contained in $S_N$ to the size $N + M$ of coefficient matrix in (15). The results are obtained by (a) IC-GMRES($m$) with $\alpha = 1.5$, (b) IC-Bi-CGSTAB with $\alpha = 1.5$, (c) IC-GMRES($m$) with $\alpha = 1.9$, and (d) IC-Bi-CGSTAB with $\alpha = 1.9$.

Here, $S_N^4$, $S_N^5$, and $S_N^6$ are defined as follows:

- $S_N^4 : -0.25 \leq z \leq 0.25, -0.25 \leq x \leq 0.25$ and $y = -0.5$.
- $S_N^5 : -0.25 \leq y \leq 0.25, -0.25 \leq z \leq 0.25$ and $x = 0.5$.
- $S_N^6 : -0.25 \leq y \leq 0.25, -0.25 \leq z \leq 0.25$ and $x = -0.5$.

In Cases 4, 5 and 6, $M_N = 4356, 5445$ and 6534, respectively. In each of Figs. 7(a)–(d), the results for Cases 1–6 are plotted from left to right. In Fig. 7, top ((a) and (b)) and bottom ((c) and (d)) figures are results for $\alpha = 1.5$ and $\alpha = 1.9$, respectively. In addition, left ((a) and (c)) and right ((b) and (d)) figures are results obtained by IC-GMRES($m$) and IC-Bi-CGSTAB, respectively.

For $\alpha = 1.5$, we see from Figs. 7(a) and (b) that the computation time and number of iterations obtained with (24) are less than those obtained with (21) and (22). In Fig. 7(a), the results of (24) are similar with those of ILDU. In addition, in Fig. 7(b), the results of (24) are
less than those of ILDU. In Figs. 7(a) and (b), the computation time and number of iterations obtained by IC-Bi-CGSTAB tend to be less than those obtained by IC-GMRES(m). Hence, for \( \alpha = 1.5 \), we consider that IC-Bi-CGSTAB based on (24) is the best choice for solving the problem. Furthermore, the computation time of IC-Bi-CGSTAB based on (24) is less than that of ILDU-Bi-CGSTAB in all cases as shown in Fig. 7(b). Note that, when the ratio \( M_N / (N + M) \) is increased, the difference between the computation time of IC-Bi-CGSTAB based on (24) and that of ILDU-Bi-CGSTAB is decreased.

For \( \alpha = 1.9 \), we see from Figs. 7(c) and (d) that the computation time and number of iterations obtained with (22) are less than those obtained with (21) and (24). In Fig. 7(c), for \( R_N < 2.0 \), the results of (22) are less than those of ILDU. In addition, in Fig. 7(d), the results of (22) are similar with those of ILDU. For \( R_N > 1.5 \), the computation times obtained with (22) are larger than those obtained with ILDU. From these results, we consider that IC-GMRES(m) based on (22) is the best choice for solving the problem for \( \alpha = 1.9 \). However, when the ratio \( M_N / (N + M) \) is increased, the efficiency of IC-GMRES(m) based on (22) is lower than that of ILDU-GMRES(m). For \( R_N \leq 2.0 \), IC-GMRES(m) based on (22) may be efficient.

### 4.2.5. Influence of Sparseness of the Coefficient Matrix to Efficiency of IC-based Iterative Methods

To investigate the influence of sparseness of the coefficient matrix in (15) to the efficiency of IC-based iterative methods, the computation time and number of iterations are plotted as a function of the parameter \( \alpha \), and are depicted in Fig. 8 for \( N + M = 299203 \). In this figure, top ((a) and (b)), middle ((c) and (d)), and bottom ((e) and (f)) figures are results for Cases 1, 2, and 3, respectively. In addition, left ((a), (c) and (e)) and right ((b), (d) and (f)) figures are results obtained by IC-GMRES(m) and IC-Bi-CGSTAB, respectively. In each of Figs. 8(a)–(f), the results for \( \alpha = 1.3, 1.4, 1.5, 1.6, 1.7 \) and 1.9 are plotted from left to right. The numbers \( n_{nz} \) of non-zero elements in the coefficient matrix for \( \alpha = 1.3, 1.4, 1.5, 1.6, 1.7 \) and 1.9 are 15369419, 21562571, 25047023, 38905187, 44930147 and 53287323, respectively. For these \( \alpha \), the ratios \( R_{nz} \) of non-zero elements,

\[
R_{nz} \equiv 100 \frac{n_{nz}}{(N + M)^2} \quad (\%)
\]

are about \( 1.72 \times 10^{-2}, 2.41 \times 10^{-2}, 2.80 \times 10^{-2}, 4.35 \times 10^{-2}, 5.02 \times 10^{-2} \) and \( 5.95 \times 10^{-2} \), respectively. It must be noted here that, in all cases for \( \alpha = 1.8 \), the linear systems (15) can not be solved by all iterative methods employed in this paper. For this reason, we do not plot the computation time and number of iterations for \( \alpha = 1.8 \) in Fig. 8.

We see from Figs. 8(a)–(f) that, for \( \alpha \leq 1.7 \), the computation times and numbers of iterations obtained with (24) are less than those obtained with (21) and (22) in almost all cases. In addition, the computation times obtained with (24) are sometimes less than those obtained with ILDU for \( \alpha \leq 1.5 \). In particular, the results of IC-Bi-CGSTAB based on (24) are better than those of ILDU-Bi-CGSTAB in many cases for \( \alpha \leq 1.5 \) as shown in Figs. 8(b), (d) and (f). For \( \alpha = 1.9 \), we see from Figs. 8(a)–(f) that the computation times and numbers of iterations obtained with (22) are always less than those obtained with (21).
Figure 8: Dependence of computation time and number of iterations on the parameter $\alpha$.
Top ((a) and (b)), middle ((c) and (d)), and bottom ((e) and (f)) figures are results for Cases 1, 2, and 3, respectively. In addition, left ((a), (c) and (e)) and right ((b), (d) and (f)) figures are results obtained by IC-GMRES($m$) and IC-Bi-CGSTAB, respectively.
and (24). In addition, the computation times of IC-based iterative methods based on (22) are less than those of ILDU-based ones except for Fig. 8(f). In particular, the efficient results are obtained by IC-GMRES(m) with (22) for \( \alpha = 1.9 \) as shown in Figs. 8(a), (c) and (e). From these results and the fact that \( R_{nz} = 5.02 \times 10^{-2} \) (%) for \( \alpha = 1.7 \) and \( N + M = 299203 \), we consider that IC-Bi-CGSTAB based on (24) and IC-GMRES(m) based on (22) may be employed for \( R_{nz} \lesssim 5.0 \times 10^{-2} \) and \( R_{nz} \gtrsim 5.0 \times 10^{-2} \), respectively.

5. Conclusion

To investigate effects of preconditioning for symmetric matrices to asymmetric linear systems obtained by X-EFG, the IC-based iterative methods have been applied for solving the linear systems. To apply IC as preconditioning, the coefficient matrix of the linear systems is regarded as a symmetric matrix. In numerical experiments, four-types of symmetric structures shown in (21)–(24) are employed, and the performance of ICs based on (21)–(24) has been investigated. Conclusions obtained in the present study are summarized as follows:

1. It is found that the linear systems obtained by X-EFG can be efficiently solved by using IC-GMRES(m) and IC-Bi-CGSTAB together with employing an appropriate symmetric matrix for IC. In particular, the performance of IC may be efficient for \( N + M \gtrsim 10^5 \).

2. In some cases, the efficiency of IC-based iterative methods is superior than that of ILDU-based iterative methods. At most, the computation time of IC-GMRES(m) based on (22) is about 4.2 times faster than that of ILDU-GMRES(m).

3. When \( R_N \) is increased, the efficiency of IC-based iterative methods is decreased, especially for the case where \( \alpha \) is relatively large. For \( \alpha = 1.9 \), IC-GMRES(m) based on (22) may be efficient for \( R_N \lesssim 2.0 \) in comparison with ILDU-GMRES(m).

4. IC-Bi-CGSTAB based on (24) and IC-GMRES(m) based on (22) may be candidates for solving the linear systems in \( R_{nz} \lesssim 5.0 \times 10^{-2} \) and \( R_{nz} \gtrsim 5.0 \times 10^{-2} \), respectively.

As described in Sections 4.2.1 and 4.2.5, in all cases for \( \alpha = 1.8 \), the linear systems (15) can not be solved by all iterative methods employed in this paper. This will be investigated in future work.

Acknowledgement

This work was supported in part by JSPS KAKENHI Grant Number 15K00177.

References

[1] T. Belytschko, Y. Y. Lu, L. Gu: Element-Free Galerkin Methods, *Int. J. Numer. Methods Eng.*, 37 (1994), 229–256.
[2] S. N. Atluri and T. Zhu: A New Meshless Local Petrov-Galerkin (MLPG) Approach in Computational Mechanics, *Comput. Mech.*, 22 (1998), 117-127.

[3] A. Kamitani, T. Takayama, T. Itoh, H. Nakamura: Extension of Meshless Galerkin/Petrov-Galerkin Approach without Using Lagrange Multipliers, *Plasma and Fusion Research*, 6:2401074 (2011).

[4] T. Itoh, S. Ikuno, A. Kamitani: Numerical Investigation of Linear Systems Obtained by Extended Element-Free Galerkin Method, In *IGTE 2012 proceedings*, (2012), 47–52.

[5] M. Mori: Numerical Methods and FORTRAN 77 Programming, Iwanami Shoten, Tokyo (1986) (in Japanese).

[6] Y. Saad and M. H. Schultz: GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, *SIAM J. Sci. Stat. Comput.*, 7:3 (1986), 856–869.

[7] H. A. van der Vorst: Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems, *SIAM J. Sci. Stat. Comput.*, 13:2 (1992), 631–644.

[8] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, et al.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods (2nd Edition), SIAM, Philadelphia (1994).

[9] K. Murata, M. Natori, Y. Karaki: Large-scale Numerical Simulations, Iwanami Shoten, Tokyo (1990) (in Japanese).