PLOS | ONE

RESEARCH ARTICLE

Condition Number Estimation of Preconditioned Matrices

Noriyuki Kushida[1] *

1 International Data Centre, the Preparatory Commission for the Comprehensive Nuclear-Test-Ban Treaty Organization, Vienna, Austria

* noriyuki.kushida@ctbto.org

Abstract

The present paper introduces a condition number estimation method for preconditioned matrices. The newly developed method provides reasonable results, while the conventional method which is based on the Lanczos connection gives meaningless results. The Lanczos connection based method provides the condition numbers of coefficient matrices of systems of linear equations with information obtained through the preconditioned conjugate gradient method. Estimating the condition number of preconditioned matrices is sometimes important when describing the effectiveness of new preconditionerers or selecting adequate preconditioners. Operating a preconditioner on a coefficient matrix is the simplest method of estimation. However, this is not possible for large-scale computing, especially if computation is performed on distributed memory parallel computers. This is because, the preconditioned matrices become dense, even if the original matrices are sparse. Although the Lanczos connection method can be used to calculate the condition number of preconditioned matrices, it is not considered to be applicable to large-scale problems because of its weakness with respect to numerical errors. Therefore, we have developed a robust and parallelizable method based on Hager's method. The feasibility studies are curried out for the diagonal scaling preconditioner and the SSOR preconditioner with a diagonal matrix, a tridaigonal matrix and Pei's matrix. As a result, the Lanczos connection method contains around 10% error in the results even with a simple problem. On the other hand, the new method contains negligible errors. In addition, the newly developed method returns reasonable solutions when the Lanczos connection method fails with Pei's matrix, and matrices generated with the finite element method.

Citation: Kushida N (2015) Condition Number Estimation of Preconditioned Matrices. PLoS ONE 10 (3): e0122331. doi:10.1371/journal.pone.0122331

Academic Editor: Rodrigo Huerta-Quintanilla, Cinvestav-Merida, MEXICO

Received: September 15, 2014

Accepted: February 10, 2015

Published: March 27, 2015

Data Availability Statement: All relevant data are within the paper.

Funding: The authors have no support or funding to report.

Competing Interests: The authors have declared that no competing interests exist.

Introduction

Solving the linear equation is considered to be the most time consuming component of simulation computation. As a result, a number of linear equation solvers have been developed in order to realize more efficient simulation. Linear equation solvers can be roughly categorized into two types: direct solvers and iterative solvers. Direct solvers have been used for ill

conditioned problems, because of the robustness of the solution. However, the iterative method has become mainstream in recent years for the following reasons:

- The iterative method requires less memory than the direct method, if the coefficient matrix of the equation is sparse. Most simulation methods, as is the case with the finite element method and the finite difference method, use sparse matrices.
- The iterative method is more suitable for distributed memory type parallel computers than the direct method. Recently, the trend in super computers has been toward distributed memory computers.

As a result, iterative solvers have generated a great deal of interest. Iterative methods are of two types: the stationary method and the Krylov sub-space type method. Generally, the Krylov sub-space type method is more commonly used than the stationary method because the stationary method, is in most cases, slower and more suited to parallel computers. The conjugate gradient method is representative of the Krylov sub-space type method, and the Gauss-Seidel method is a typical representative example of the stationary method. The Krylov sub-space type method is faster than the stationary method, but is sometimes unstable. Furthermore, rapid convergence is always desired. In order to address such considerations, preconditioning is applied to the Krylov sub-space type method. In particular, the conjugate gradient method with preconditioning, sometimes called the preconditioned conjugate gradient method (PCG), is one of the most well known iterative solvers [1].

Evaluating the convergence rate of an iterative solver is sometimes important in order to demonstrate the effectiveness of new methods or to select an adequate method. The convergence rate of the conjugate gradient method (CG) and other Krylov sub-space type solvers strongly depends on the eigenvalue distribution of coefficient matrix, that is to say, the convergence rate is better if the eigenvalues are concentrated. Complete knowledge of the eigenvalue distribution enables the complete prediction of the convergence behavior of the PCG. However, obtaining all of the eigenvalues is difficult, and so another simple indicator, called condition number, is used. Generally, the condition number is easy to calculate when the coefficient matrix is explicitly obtained. As described above, parallel computers, especially distributed memory type parallel computers, are becoming increasingly important for engineers or physicists. Obtaining preconditioned matrices using a parallel computer is extremely difficult and an evaluation method is required. One way to evaluate preconditioned matrices is by the Lanczos connection method [2], [3]. However, the Lanczos connection method is weak with respect to round-off errors and therefore is not suitable for large matrices. Moreover, the applicability of the Lanczos connection method has not yet been discussed.

In the present paper, we introduce a new evaluation method of preconditioned matrices based on Hager's method, and verify the robustness of the new method by comparing their precisions.

Methods

Preconditioning of the conjugate gradient method

Preconditioning. When using iterative solvers, an acceleration method called preconditioning is performed:

Let the linear equation be written as

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

if a preconditioning matrix $\mathbf{M}$, which is similar to $\mathbf{A}$ in some sense, is operated on Equation 1,

then Equation 1 can be written as

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{2}$$

where

$$\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1} = \tilde{\mathbf{A}}, \tag{3}$$

$$\mathbf{M}_1\mathbf{M}_2 = \mathbf{M}, \tag{4}$$

$$\mathbf{M}_2\mathbf{x} = \tilde{\mathbf{x}}, \tag{5}$$

$$\mathbf{M}_1^{-1}\mathbf{b} = \tilde{\mathbf{b}}. \tag{6}$$

In particular, the following property is required to maintain the symmetry of the preconditioned matrix:

$$\mathbf{M}_1^T = \mathbf{M}_2. \tag{7}$$

The coefficient matrix of a transformed system of linear equations (Equation 3) should be more favorable in nature than Equation 1, and the number of itrations to convergence of the CG on the transformed system should be less than that on the original system. One can show that the number of iterations to convergence of the CG is proportional to $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$, where κ is the 2-norm condition number of a coefficient matrix (defenition of the 2-norm condition number is given in the section "Condition number" in this article) [4]. Thus, the smaller κ becomes, the less iterations is required to convergence. Therefore, **M** should be selected to decrease κ of the transformed system. Such a transformation is not operated in practical programming, since the matrix—matrix product is needed, which requires much computational effort. The PCG algorithm is shown in Table 1. In the algorithm, Solve **Mz** = **r** is the preconditioning (line 5 in the algorithm), where **r** is the residual norm of the PCG, and **z** is the residual norm in the transformed space by **M**. Although this equation is usually difficult to solve, such difficulty can be avoided by selecting matrix **M** to be a diagonal matrix or products of triangular matrices. Such matrices are described in the following two sections.

Diagonal scaling

When the preconditioning matrix **M** is expressed as

$$\mathbf{M} = diag\,(\mathbf{A}), \tag{8}$$

where the function *diag*(**A**) extract the diagonal component of a matrix **A**. This preconditioning is called the diagonal scaling [5].

SSOR

Assuming that the coefficient matrix is symmetric and is decomposed as

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{L}^T \tag{9}$$

the SSOR preconditioning matrix is defined as

$$\mathbf{M}(\omega) = \frac{1}{2-\omega}\left(\frac{1}{\omega}\mathbf{D} + \mathbf{L}\right)\left(\frac{1}{\omega}\mathbf{D}\right)^{-1}\left(\frac{1}{\omega}\mathbf{D} + \mathbf{L}\right)^T \tag{10}$$

Table 1. Preconditioned Conjugte Gradient Algorithm.

1:	**procedure** PCG(**A**, **b**)
2:	Let $\mathbf{x}_0$ be the initial approximation
3:	$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
4:	**for** i = 0, 1, 2, 3, . . ., until convergence do **do**
5:	solve $\mathbf{M}\mathbf{z}_i = \mathbf{r}_i$
6:	$\rho_i = \langle \mathbf{r}_i, \mathbf{z}_i \rangle$
7:	**If** $i = 0$ **then**
8:	$\mathbf{p}_{i+1} = \mathbf{z}_i$
9:	**else**
10:	$\beta_i = \frac{\rho_i}{\rho_{i-1}}$
11:	$\mathbf{p}_{i+1} = \mathbf{z}_i + \beta_i \mathbf{p}_i$
12:	**end if**
13:	$\mathbf{q}_{i+1} = \mathbf{A}\mathbf{p}_{i+1}$
14:	$\alpha_i = \frac{\rho_i}{\langle \mathbf{p}_{i+1}, \mathbf{q}_{i+1} \rangle}$
15:	$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_{i+1}$
16:	$\mathbf{r}_{i+1} = \mathbf{r}_i + \alpha_i \mathbf{q}_{i+1}$
17:	**end for**
18:	return $\mathbf{x}_{i_{last}}$, where i_{last} is the last iteration
19:	**end procedure**

Pseudocode of the PCG. In the algorithm, $\langle \cdot, \cdot \rangle$ denotes vector inner product.

doi:10.1371/journal.pone.0122331.t001

where, **D**, **L**, and ω are the diagonal component of **A**, the lower triangular component of **A**, and the relaxation parameter, respectively. Usually ω is set to 1.0 [6]. Note that the diagonal scaling and SSOR preconditioning matrices are symmetric positive definite (SPD), if **A** is SPD. This is because, those preconditioning matrices can be rewritten as,

$$\mathbf{M} = \hat{\mathbf{L}}\hat{\mathbf{L}}^T, \tag{11}$$

where $\hat{\mathbf{L}}$ is a solvable lower triangular matrix, and then, Equation 11 is the definition of the Cholesky decomposition. Thus **M** is SPD [7], [8].

Condition number

The convergence rate of the CG method depends on the distribution of eigenvalues of a coefficient matrix **A** or a preconditioned matrix $\tilde{\mathbf{A}}$. Although the complete information of eigenvalues enables prediction of the exact convergence behavior, a simpler indicator, known as the condition number, exists. The condition number of a matrix is given by

$$cond(\mathbf{A}) = \| \mathbf{A} \| \left\| \mathbf{A}^{-1} \right\| \tag{12}$$

where $\|\cdot\|$ denotes the norm of a matrix [8]. Since there are various definitions of the norm, we can define various condition numbers. The condition numbers given by the 1-norm or

2-norm, as defined below, are commonly used.

$$\| \mathbf{A} \|_1 = \max_j \sum_{i=1}^{n} |a_{ij}| \tag{13}$$

$$\| \mathbf{A} \|_2 = \left(\sqrt{maximum\ eigenvalue\ of\ \mathbf{A}^T\mathbf{A}} \right) \tag{14}$$

Furthermore, if the matrix **A** is SPD, then the condition number given by the 2-norm can be written as

$$cond(\mathbf{A}) = \frac{maximum\ eigenvalue\ of\ \mathbf{A}}{minimum\ eigenvalue\ of\ \mathbf{A}} \tag{15}$$

Lanczos connection

The Lanczos connection method is a method of solving the eigensystem and is strongly related to the CG method [2]. Using such a relationship, the condition number of **A** can be calculated through the CG procedure. Now, we define the $n \times k$ matrix $\mathbf{R}_k$ as

$$\mathbf{R}_k = [\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{k-1}] \tag{16}$$

where n is the matrix size of **A**, k is the iteration number of the PCG and $\mathbf{r}_i$, $(i = 0, \ldots, k-1)$ are the residual vectors and can be found in the PCG algorithm (Table 1).

Next, we define $k \times k$ matrix $\mathbf{B}_k$ as follows:

$$\mathbf{B}_k = \begin{bmatrix} 1 & -\beta_1 & & \cdots & 0 \\ & 1 & -\beta_2 & & \vdots \\ & & \ddots & \ddots & \\ & & & \ddots & -\beta_{k-1} \\ & & & & 1 \end{bmatrix}, \tag{17}$$

where β_k are the scalars shown in the PCG algorithm. Using the matrix $\mathbf{P}_k = [\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{k-1}]$,

$$\mathbf{R}_k = \mathbf{P}_k \mathbf{B}_k. \tag{18}$$

where $\mathbf{p}_i$, $(i = 0, \ldots, k-1)$ are the modification direction vectors in the PCG algorithm. Since vectors $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k$ are **A**-orthogonal,

$$\hat{\mathbf{T}}_k = \mathbf{R}_k^T \mathbf{A} \mathbf{R}_k = \mathbf{B}_k^T \Lambda_k \mathbf{B}_k, \tag{19}$$

where Λ_k is a matrix having components that are given by

$$\Lambda_k = \begin{bmatrix} \mathbf{p}_1^T\mathbf{A}\mathbf{p}_1 & & & \\ & \mathbf{p}_2^T\mathbf{A}\mathbf{p}_2 & & \\ & & \ddots & \\ & & & \mathbf{p}_k^T\mathbf{A}\mathbf{p}_k \end{bmatrix}. \tag{20}$$

Here, we consider the $n \times k$ matrix $\mathbf{Q}_k$,

$$\mathbf{Q}_k = \mathbf{R}_k \Delta_k^{-1}. \tag{21}$$

where Δ_k is a $k \times k$ matrix having the following components:

$$\Delta_k = \begin{bmatrix} \| \mathbf{r}_1 \|_2 & & & \\ & \| \mathbf{r}_2 \|_2 & & \\ & & \ddots & \\ & & & \| \mathbf{r}_k \|_2 \end{bmatrix}, \tag{22}$$

where $\|\mathbf{r}_i\|_2$, $(i = 0, \ldots, k)$ are the 2-norms of the residual vectors. The columns of $\mathbf{Q}_k$ are the Lanczos vectors, for which the associated projections of $\mathbf{A}$ form a tridiagonal matrix, thus

$$\mathbf{T}_k = \Delta_k^{-1} \mathbf{B}_k^T \Lambda_k \mathbf{B}_k \Delta_k^{-1}. \tag{23}$$

Since external eigenvalues of $\mathbf{T}_k$ approximate those of matrix $\mathbf{A}$, we can approximate the condition number of matrix $\mathbf{A}$ using this method. Furthermore, if the above parameters are obtained from the PCG, we can obtain the condition number of the preconditioned matrix [3]. The advantages of this method are ease of implementation and ease of evaluating the condition number of preconditioned matrices. However, this method is not applicable for large systems or ill-conditioned systems, due to the fact that this method requires the strict orthogonality of basis vector used in the CG aglgorithm. However, such orthogonality tends to be broken for the large system or ill conditioned system.

Hager's method

Hager's method gives the 1-norm of $\mathbf{A}^{-1}$ [9]. Originally, the condition number is defined as Equation 12. Therefore, we need the 1-norm of $\mathbf{A}$. Fortunately, the 1-norm of $\mathbf{A}$ is easy to obtain if the components of $\mathbf{A}$ are given explicitly. The algorithm of Hager's method is described in Table 2. Although Hager's method requires the transposition of matrix $\mathbf{A}$, we assume that $\mathbf{A}^T = \mathbf{A}$, because we are using the PCG, therefore symmetric positive definite matrices can be solved. Hager's method requires the solution of a linear equation during iteration of the procedure for finding the 1-norm condition number. In the present study, they are solved using the PCG.

Hager's method for preconditioned matrices

Here, we develop a new condition number estimation method which overcomes the downside of the Lanczos connection method. That is to say, new method must have such features:

- Applicable for large system. Especially, it must work on distributed memory computers.
- Robust for ill conditioned system.

In order to develop such new method, we employ Hager's method which gives the 1-norm of inverse matrices as a basic algorithm. In the rest of this sections, we describe the algorithm of our new method.

As introduced in the previous section, we can obtain the 1-norm condition number using Hager's method. There is a simple method of extending Hager's method to preconditioned matrices without explicit matrix production. That is, in Hager's method, we simply replace matrix $\mathbf{A}$ with $\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}$. It can be easily achieved if the PCG is employed to solve equations that appear in Hager's method (lines 4 and 11 in Table 2). This is because, matrices are used as the

Table 2. The algorithm of Hager's method.

1:	**procedure** HAGER'S METHOD(**A**)
2:	Set $\mathbf{b} = (\frac{1}{n}, \cdots, \frac{1}{n})^T$, where n is the size of the matrix **A**, and $\rho = 0$
3:	**for do**
4:	Solve $\mathbf{Ax} = \mathbf{b}$
5:	**if** $\|\mathbf{x}\|_1 \leq \rho$ **then**
6:	return ρ as $\|\mathbf{A}^{-1}\|_1$, and exit
7:	**else**
8:	$\rho = \|\mathbf{x}\|_1$
9:	**end if**
10:	$\mathbf{y}[i] = \text{sgn}(\mathbf{x}[i])$ $(i = 1, 2, \ldots, n)$
11:	Solve $\mathbf{A}^T\mathbf{z} = \mathbf{y}$
12:	$i_{\max} = i$ which satisfies $max_i \|\mathbf{z}[i]\|$
13:	**if** $\|\mathbf{z}[i_{max}]\| < \mathbf{z}^T\mathbf{b}$ **then**
14:	return ρ as $\|\mathbf{A}^{-1}\|_1$, and exit
15:	**else**
16:	$\mathbf{b}[i] = \begin{cases} 1 & (i = i_{\max}) \\ 0 & (i \neq i_{\max}) \end{cases}$
17:	**end if**
18:	**end for**
19:	**end procedure**

Pseudocode of Hager's method. In the algorithm, [*i*] denotes the *i*th vector element of a vector.

doi:10.1371/journal.pone.0122331.t002

form of $\mathbf{p} = \mathbf{Aq}$, where $\mathbf{p}$ and $\mathbf{q}$ are vectors. Therefore, we can solve the system of $\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{x} = \mathbf{b}$ by replacing $\mathbf{A}$ with $\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}$ in the PCG algorithm. The algorithm is named "calc 1-norm of inverse preconditioned matrix" and is shown in Table 3. In addition, the PCG algorithm that solves $\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{x} = \mathbf{b}$ is shown in Table 4.

Since the Hager's method gives us only $\|\left(\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\right)^{-1}\|_1$, a method by which to calculate $\|\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\|_1$ must be constructed. It is not straightforward, because we have no explicit $\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}$. In the present study, we develop such a method using Hager's method by replacing the solution of the two linear equations in Table 2 (lines 4 and 11) with $\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{b} = \mathbf{x}$ and $\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{y} = \mathbf{z}$, respectively. We thus obtain $\|\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\|_1$. The algorithm which calculates the 1-norm of preconditioned matrix is named "calc 1-norm of preconditioned matrix" and is shown in Table 3. The entire algorithm to calculate 1-norm condition number is referred to as the modified Hager's method in the present article.

These developed algorithms for preconditioned matrices always converge if coefficient matrices and preconditioning matrices are positive definite. This is because, Hager's method always converges for positive definite matrices [10]. In addition, one can show that the inverse matrices of positive definite matrices are also positive definite, and the product of two positive definite matrices are also positive definite.

Results

In this section, we carry out the feasibility test of our new method by comparing the performance of both our method and the Lanczos connection method. In the following sections, the

Table 3. The algorithm of the modified Hager's method.

1:	**procedure** MODIFIED HAGER'S METHOD ($\mathbf{M}_1^{-1}$, $\mathbf{A}$, $\mathbf{M}_2^{-1}$)
2:	ρ_{inv} = call procedure calc 1-norm of inverse preconditioned matrix ($\mathbf{M}_1^{-1}$, $\mathbf{A}$, $\mathbf{M}_2^{-1}$)
3:	$\rho_{forward}$ = call procedure calc 1-norm of preconditioned matrix($\mathbf{M}_1^{-1}$, $\mathbf{A}$, $\mathbf{M}_2^{-1}$)
4:	return $\rho_{inv} \cdot \rho_{forward}$ as the condition number of $\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}$
5:	**end procedure**
6:	**procedure** CALC 1-NORM OF INVERSE PRECONDITIONED MATRIX ($\mathbf{M}_1^{-1}$, $\mathbf{A}$, $\mathbf{M}_2^{-1}$)
7:	Set $\mathbf{b} = \left(\frac{1}{n}, \cdots, \frac{1}{n}\right)^T$ and $\rho_{inv} = 0$
8:	**for do**
9:	$\mathbf{x}$ call procedure PCG in modified Hager ($\mathbf{M}_1^{-1}$, $\mathbf{A}$, $\mathbf{M}_2^{-1}$, $\mathbf{b}$)
10:	**if** $\|\mathbf{x}\|_1 \leq \rho_{inv}$ **then**
11:	return ρ_{inv} as $\|(\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1})^{-1}\|_1$, and exit
12:	**else**
13:	$\rho_{inv} = \|\mathbf{x}\|_1$
14:	**end if**
15:	$\mathbf{y}[i]$ = sgn ($\mathbf{x}[i]$) (i = 1, 2, . . ., n)
16:	$\mathbf{z}$ = call procedure PCG in modified Hager ($\mathbf{M}_1^{-1}$, $\mathbf{A}$, $\mathbf{M}_2^{-1}$, $\mathbf{y}$)
17:	$i_{\max} = i$ which satisfies max_i $\|\mathbf{z}[i]\|$
18:	**if** $\|\mathbf{z}[i_{max}]\| < \mathbf{z}^T\mathbf{b}$ **then**
19:	return ρ_{inv} as $\|(\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1})^{-1}\|_1$, and exit
20:	**else**
21:	$\mathbf{b}[i] = \begin{cases} 1 & (i = i_{\max}) \\ 0 & (i \neq i_{\max}) \end{cases}$
22:	**end if**
23:	**end for**
24:	**end procedure**
25:	**procedure** CALC 1-NORM OF PRECONDITIONED MATRIX ($\mathbf{M}_1^{-1}$, $\mathbf{A}$, $\mathbf{M}_2^{-1}$)
26:	Set $\mathbf{b} = \left(\frac{1}{n}, \cdots, \frac{1}{n}\right)^T$ and $\rho_{forwad} = 0$
27:	**for do**
28:	$\mathbf{x} = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{b}$
29:	**if** $\|\mathbf{x}\|_1 \leq \rho_{forward}$ **then**
30:	return $\rho_{forward}$ as $\|\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\|_1$, and exit
31:	**else**
32:	$\rho_{forward} = \|\mathbf{x}\|_1$
33:	**end if**
34:	$\mathbf{y}[i]$ = sgn ($\mathbf{x}[i]$) (i = 1, 2, . . ., n)
35:	$\mathbf{z} = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{y}$
36:	$i_{\max} = i$ which satisfies max_i $\|\mathbf{z}[i]\|$
37:	**if** $\|\mathbf{z}[i_{max}]\| < \mathbf{z}^T\mathbf{b}$ **then**
38:	return $\rho_{forward}$ as $\|\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\|_1$, and exit
39:	**else**
40:	$\mathbf{b}[i] = \begin{cases} 1 & (i = i_{\max}) \\ 0 & (i \neq i_{\max}) \end{cases}$
41:	**end if**
42:	**end for**
43:	**end procedure**

Pseudocode of the modified Hager's method. The modified Hager's method consists of two major parts: one is for calculating $\|(\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1})^{-1}\|_1$ and the other is for $\|\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\|_1$. In the algorithm, [$i$] denotes the ith vector element of a vector.

doi:10.1371/journal.pone.0122331.t003

Table 4. Preconditioned Conjugte Gradient Algorithm in the modified Hager's method.

1:	**procedure** PCG IN MODIFIED HAGER ($\mathbf{M}_1^{-1}$, $\mathbf{A}$, $\mathbf{M}_2^{-1}$, $\mathbf{b}$)
2:	Let $\mathbf{x}_0$ be the initial approximation
3:	$\mathbf{r}_0 = \mathbf{b} - \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{x}_0$
4:	**for** i = 0, 1, 2, 3, . . ., until convergence do **do**
5:	$\rho_i = \langle \mathbf{r}_i, \mathbf{r}_i \rangle$
6:	**if** i = 0 **then**
7:	$\mathbf{p}_{i+1} = \mathbf{r}_i$
8:	**else**
9:	$\beta_i = \frac{\rho_i}{\rho_{i-1}}$
10:	$\mathbf{p}_{i+1} = \mathbf{r}_i + \beta_i \mathbf{p}_i$
11:	**end if**
12:	$\mathbf{q}_{i+1} = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{p}_{i+1}$
13:	$\alpha_i = \frac{\rho_i}{\langle \mathbf{p}_{i+1}, \mathbf{q}_{i+1} \rangle}$
14:	$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_{i+1}$
15:	$\mathbf{r}_{i+1} = \mathbf{r}_i + \alpha_i \mathbf{q}_{i+1}$
16:	**end for**
17:	return $\mathbf{x}_{i_{last}}$, where i_{last} is the last iteration
18:	**end procedure**

Pseudocode of the PCG in the modified Hager's method. In the algorithm, $\langle \cdot, \cdot \rangle$ denotes vector inner product.

doi:10.1371/journal.pone.0122331.t004

computational environment, and the matrices which we employ will be explained, and the performances are also described.

Computational environment

Here, we describe the computational environment in order to determine the precision of the numerical calculation. All of the code was written with Fortran90, and the eigenvalues of the tridiagonal matrix obtained from the Lanczos connection was calculated using the Intel Math Kernel library. The machine architecture and compiler environment are listed in Table 5. In addition, the condition number, which should be compared between the Lanczos connection method and our modified Hager's method, was calculated using Octave [11].

Table 5. Computational Environment.

Architecture	Intel Pentium4 2.8 GHz
Operating System	Vine linux 3.1
Compiler	Intel Fortran Compiler 8.1
Compiler options	-O3
Math library	Intel Math Kernel library
Float number precision	real(8)

Details of the computational environment used for the precision verification.

doi:10.1371/journal.pone.0122331.t005

Sample matrices

Diagonal matrix. In order to verify the precision of both of these methods, we first employ a simple diagonal matrix for a test problem. The test matrix $\mathbf{T}_{diag}$ is given by

$$\mathbf{T}_{diag} = \begin{bmatrix} 1 & & & \\ & 2 & & \\ & & \ddots & \\ & & & n \end{bmatrix}, \tag{24}$$

where n is the matrix size. The eigenvalues of this matrix are $(1, 2, \ldots, n)$, and the condition number of these eigenvalues should be n. Note that, the condition number of both diagonal scaled and SSOR preconditioned $\mathbf{T}_{diag}$ are 1.0.

Tridiagonal matrix. The $n \times n$ tridiagonal matrix we employed in the present studies is given as

$$\mathbf{T}_{Tri} = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}. \tag{25}$$

Pei's matrix. We employ Pei's matrix [12] as a test matrix, because one can control the condition number of the matrix quite easily. Pei's matrix is defined as follows:

$$\mathbf{T}_{Pei} = \begin{bmatrix} 1+d & 1 & \cdots & \cdots & 1 \\ 1 & 1+d & 1 & \cdots & 1 \\ \vdots & 1 & 1+d & \ddots & 1 \\ \vdots & \vdots & \ddots & \ddots & 1 \\ 1 & 1 & 1 & 1 & 1+d \end{bmatrix}, \tag{26}$$

where d is the parameter by which to determine the condition of matrix $\mathbf{T}_{Pei}$, which becomes increasingly ill conditioned as d approaches zero. If d is zero, matrix $\mathbf{T}_{Pei}$ is singular.

Poission's equation with FEM. The above sample matrices were artificially created and have simple structures. In this section, we will introduce matrices that are created with the finite element method (FEM) in order to investigate the feasibility of both methods with realistic problems. The equation which employed was Poission's equation. The analysis domain was a cube with edge lengths of 1.0. In the present study, three types of finite element meshes were prepared and each of three types had a different degree of freedom (DOF) on x, y, and z axes: $20 \times 20 \times 20$, $10 \times 10 \times 80$, and $5 \times 5 \times 320$. The total DOFs were the same. Dirichlet boundary conditions were applied to both faces at $z = 0$, and 1.0.

Estimated condition numbers

Diagonal matrix. First we discuss the feasibility of the Lanczos connection. The eigenvalues of $\mathbf{T}_{diag}$ calculated with the Lanczos connection is listed in the first row in Table 6. Here we set $n = 10$. The largest eigenvalue is unreasoably large, and therefore, obtained condition number is unreasonable. We can remove this unresonably large eigenvalue, by ignoring the first step information of the PCG. This treatment leads one-rank smaller tri-diagonal matrix

Table 6. Eigenvalues of a 10 × 10 diagonal matrix calculated using the Lanczos connection method.

Ideal	1	2	3	4	5	6	7	8	9	10
Lanczos connection	1.00	2.02	3.04	4.08	5.13	6.17	7.28	8.37	9.51	3.0×10^6
Without 1st step	1.00	2.02	3.04	4.08	5.13	6.20	7.28	8.37	9.51	NA

In the table, the eigenvalues of a 10 × 10 diagonal matrix calculated using the Lanczos connection method are listed. The first row indicates the eigenvalues with all information, and the second row indicates the eigenvalues without the first CG step information. In the full information case, the condition number becomes unreasonable, while the second one shows a reasonable result. In the table, NA means "Not Available".

doi:10.1371/journal.pone.0122331.t006

associated with the Lanczos connection. However, the resulted condition number is more reasonable. The eigenvalues without the first step information is list in the second row in Table 6. Simply, the largest eigenvalue is removed. This treatment is applied to the remaining of this article.

Next, we check the feasibility of both the Lanczos connection and Hager's method with $\mathbf{T}_{diag}$. The condition numbers calculated using both methids are listed in Table 7. The obtained condition numbers are reasonable.

Tridiagonal matrix. The condition numbers of $\mathbf{T}_{Tri}$ are listed in Table 8. Here we employ 10 × 10, 100 × 100, 200 × 200 and 500 × 500 matrices. In addition, those of diagonally scaled matrices of the same size are listed in Table 9. Comparison of Table 8. Table 9 reveals that the condition number is not improved by diagonal scaling in this case. Moreover, the number of iterations to convergence was also the same. We first discuss Hager's method, which corresponds to Octave completely, for every size in both cases. Higham reported that Hager's method gives the correct value for positive definite matrices [10]. As we discussed, the diagonal scaled matrices are also SPD. Therefore, these results can be expected. On the other hand, although the Lanczos connection method provides less favorable results for smaller matrices, the results improved for larger problems. This behavior results from the modification described in the previous section, whereby we removed the information of the first conjugate gradient step. The effects of the modification became smaller for larger problems. Finally, it can be said that software developed in the present studies of both the Lanczos connection method and the modified Hager's method work well with preconditioned matrices.

Pei's matrix. In the present paper, we fixed the matrix size as 100 × 100 and set d as $d = 0.5, 0.25$ and 0.125. Calculated condition numbers and errors with respect to the condition number from Octave are listed in Table 10. An error is defined as

$$Error = \frac{|\text{Calculated condition number} - \text{Octave}|}{\text{Octave}} \times 100(\%). \tag{27}$$

Table 7. Condition numbers of $\mathbf{T}_{diag}$ by the Lanczos connection method and Hager's method.

Matrix Size	10	100	200	500
Lanczos connection	9.49	99.74	199.78	499.81
Octave (2-norm cond. num.)	10	100	200	500
Hager's method	10.00	100.00	200.00	500.00
Octave (1-norm cond. num.)	10	100	200	500

In the table, condition numbers calculated using both the Lanczos connection method and Hager's method are listed with numbers with Octave. Both method show good performances, but the Lanczos method includes some error, although the test matrix is simple.

doi:10.1371/journal.pone.0122331.t007

Table 8. Condition numbers: Tridiagonal matrix.

Matrix Size	10	100	200	500
Lanczos connection	41.8	4.13×10^3	1.64×10^4	1.02×10^5
Octave (2-norm cond. num.)	48.0	4.13×10^3	1.64×10^4	1.02×10^5
Hager's method	60.0	5.10×10^3	2.02×10^6	1.26×10^6
Octave (1-norm cond. num.)	60.0	5.10×10^3	2.02×10^6	1.26×10^6

The condition numbers of $\mathbf{T}_{Tri}$ using the Lanczos connection method, the modified Hager's method, and Octave are listed.

doi:10.1371/journal.pone.0122331.t008

Table 9. Condition numbers: Diagonal scaled tridiagonal matrix.

Matrix Size	10	100	200	500
Lanczos connection	41.8	4.13×10^3	1.64×10^4	1.02×10^5
Octave (2-norm cond. num.)	48.0	4.13×10^3	1.64×10^4	1.02×10^5
Modified Hager's method	60.0	5.10×10^3	2.02×10^6	1.26×10^6
Octave (1-norm cond. num.)	60.0	5.10×10^3	2.02×10^6	1.26×10^6

The condition numbers of diagonal scaled $\mathbf{T}_{Tri}$ using the Lanczos connection method, the modified Hager's method, and Octave are listed.

doi:10.1371/journal.pone.0122331.t009

The condition numbers estimated by the Lanczos connection method are 64,851($d = 0.5$), 178,699($d = 0.25$) and 338,463($d = 0.125$), and those estimated by Octave are 1,365.6($d = 0.5$), 3,259.8($d = 0.25$) and 7,224.7($d = 0.125$). Thus, in the Lanczos connection method case, the error is always larger than 4,000%, and the results diverge greatly from the correct answers. Because the Lanczos connection showed the reasonable results on SSOR preconditioned matrices in the article [3], it can be considered that these results are caused by the weakness with respect to numerical errors of the CG method, and great care in usage is required. On the other hand, the condition numbers estimated by the modified Hager's method are 1,684.08($d = 0.5$), 4,020.75($d = 0.25$) and 8,911.86($d = 0.125$), and those estimated by Octave 1,684.1($d = 0.5$), 4,020.8($d = 0.25$) and 8,911.9($d = 0.125$). That is to say, the modified Hager's method provided the best results for all cases.

Poission's equation with FEM. The condition numbers of preconditioned matrices of Poisson's equation are listed in Table 11, and 12. The errors are also shown in the tables. In diagonal scaled cases, the Lanczos connection gives results with over 1,000% errors. Particularly, in the $20 \times 20 \times 20$ case, it shows 3.03×10^{27}% error. Reasonable results can be obtained by

Table 10. Condition numbers: SSOR preconditioned Pei's matrix for various *d*.

	Lanczos	Octave	Error(%)	Modified Hager	Octave	Error (%)
$d = 0.5$	64,851	1,365.6	4,648.9	1,684.08	1,684.1	0.0012
$d = 0.25$	178,699	3,259.8	5,387.0	4,020.75	4,020.8	0.0012
$d = 0.125$	338,463	7,224.7	4,584.8	8,911.86	8,911.9	0.0004

The condition numbers of SSOR preconditioned $\mathbf{T}_{Pei}$ using the Lanczos connection method, the modified Hager's method, and Octave are listed. In this table, error values defined as $Error = \frac{|\text{Calculated condition number} - \text{Octave}|}{\text{Octave}} \times 100(\%)$ are also indicated.

doi:10.1371/journal.pone.0122331.t010

Table 11. Condition numbers: Diagonal scaled Poisson–FEM matrix.

	Lanczos	Octave	Error(%)	Modified Hager	Octave	Error(%)
$20 \times 20 \times 20$	2.35×10^{28}	7.75×10^{2}	3.03×10^{27}	2.98×10^{2}	2.98×10^{2}	0.00
$10 \times 10 \times 80$	5.82×10^{4}	2.79×10^{3}	1.98×10^{3}	3.63×10^{3}	3.87×10^{3}	6.43
$5 \times 5 \times 320$	5.98×10^{5}	4.73×10^{4}	1.16×10^{3}	6.16×10^{4}	6.59×10^{4}	6.46

The condition numbers of the diagonal scaled Poisson–FEM matrices using the Lanczos connection method, the modified Hager's method, and Octave are listed. In this table, error values defined as $Error = \frac{|\text{Calculated condition number} - \text{Octave}|}{\text{Octave}} \times 100(\%)$ are also indicated.

doi:10.1371/journal.pone.0122331.t011

removing the last CG step information from the Lanczos connection procedure. The condition number moves from 2.35×10^{28} to 6.94×10^{2}. However, such procedures can only be done when the true result is already known, and therefore, the Lanczos connection cannot be used with ease. On the other hand, using the modified Hager's method gives the results with an error rate under 6.46%.

In SSOR preconditioned cases, the Lanczos connection gives more accurate results than diagonal scaled cases, however, the results still include over 50% errors. In addition, when the condition number of the original matrix becomes large, the error becomes more extreme. Estimations may become unreasonable in cases where problems are ill-conditioned. The modified Hager's method gives results with under 3% errors. The results in this section show that the modified Hager's method can be used to estimate the condition numbers of practical problems. Finally, for reference, the condition numbers of each problem based on each norm without preconditioning are indicated below: 1-norm $20 \times 20 \times 20$: 1.02×10^{3}, $10 \times 10 \times 80$: 3.73×10^{3}, and $5 \times 5 \times 320$: 7.22×10^{4}, 2-norm $20 \times 20 \times 20$: 7.75×10^{2}, $10 \times 10 \times 80$: 2.79×10^{3}, and $5 \times 5 \times 320$: 4.73×10^{4}. The condition numbers without preconditioning were calculated with Octave. Sample programs of both the Lanczos connection and the modified Hager's method will be uploaded on the Zenodo site for readers' convenience [13].

Discussion

Parallel implementation

In this section, the research examines how to implement the PCG, and the modified Hager's method. First, the algorithm of the PCG (Table 1) is considered. The PCG algorithm consists of just three components: (1) a matrix–vector product (lines 3, and 13), (2) a vector inner product (lines 6, and 14), and (3) a scalar–vector product (lines 11, 15, and 16). By taking into account the above point, the PCG on distributed memory parallel computers is usually

Table 12. Condition numbers: SSOR preconditioned Poisson–FEM matrix.

	Lanczos	Octave	Error(%)	Modified Hager	Octave	Error(%)
$20 \times 20 \times 20$	4.12×10^{2}	8.22×10^{2}	49.84	1.05×10^{3}	1.07×10^{3}	2.37
$10 \times 10 \times 80$	4.82×10^{2}	2.47×10^{3}	80.49	8.48×10^{3}	8.53×10^{3}	0.64
$5 \times 5 \times 320$	1.22×10^{5}	7.61×10^{5}	84.03	1.50×10^{6}	1.50×10^{6}	0.00

The condition numbers of SSOR preconditioned Poisson–FEM matrices using the Lanczos connection method, the modified Hager's method, and Octave are listed. In this table, error values defined as $Error = \frac{|\text{Calculated condition number} - \text{Octave}|}{\text{Octave}} \times 100(\%)$ are also indicated.

doi:10.1371/journal.pone.0122331.t012

implemented with row-wise matrix decompoistion manner [6], [14]. Here we assume that there are two processors, an 8×8 matrix, and an 8 elements vector. The first processor has rows from one to four of the matrix, and vector elements from one to four. The other processor has the remaining matrix rows and vector elements. This memory allocation enables a user to compute all the above three operations with minimal (or even zero) communication, which deteriorates parallel efficiency. Namely, (1) matrix–vector products can be performed if necessary vector elements are obtained, (2) vector inner products can be performed by exchanging the results of partial vector inner products (which are just scalars) on each processor, and (3) scalar–vector products can be performed without any communication. One drawback of this memory allocation is that some preconditioning techniques, like SSOR, cannot be implemented on parallel computers as they are on sequential computers. One of the ways to accomplish such a preconditioning technique is localization, which uses only the information that is on each processor, and ignores the information on the other processors [6]. This ignoring of information degrades the effect of preconditioning; however the entire PCG algorithm still works successfully.

Hager's method, and therefore the modified Hager's method as well, can be implemented with the memory allocation techniques explained above. There are several additional operations to PCG (e.g. lines 10, 12, and 16 in Table 2), but they can be implemented without any difficulties. In addition, it should be noted that there is no extra memory space required for parallel computing, except the vector elements which must be obtained from other processors.

Finally, the parallel version of the modified Hager's method was implemented with the above technique. In addition, it was confirmed that the same results as the sequential version were obtained with the diagonal scaling preconditioning.

Computational effort and memory usage

Here, the research discusses the computational effort and the memory usage of the Lanczos connection, the modified Hager's method, and the naive implementation that computes preconditioned matrices explicitely. First, the computational effort, and the memory usage of the naive implementation are proportional to n^3, and n^2 respectively, where n is the size of a matrix. Here, we assume that the Householder transformation is used to calculate the eigenvalues. Thus, both the computational effort and the memory usage are colossal when n is large and it is applicable only when n is sufficiently small.

Second, the computational effort and the memory usage of the Lanczos connection are proportional to $m \times n$ and n respectively, where m is the number of iterations required to achieve convergence of the PCG algorithm. Here, we assume that the matrix we consider has the same matrix structure as the matrices used in the section "Poisson's equation with FEM". In addition, we can assume that $m << n$, in other words, the tridiagonal matrix associated with the Lanczos connection is small and the computational effort required to solve is negligible. The matrix has 10 matrix elements in each row, therefore, the entire size becomes $10 \times n$. The PCG algorithm requires six vectors. In the end, $16 \times n$ elements must be stored. The number of floating point operations of an iteration of the PCG algorithm is approximately twice (add and multiply) the number of matrix and vector elements (excluding preconditioning). The diagonal scaling requries n floating point operations, and the SSOR preconditioning requries $20 \times n$ floating point operations. As a result, roughly $50 \times m \times n$ floating point operations are required.

Finally, we consider the modified Hager's method. The memory space requirement of the modified Hager's method is almost the same as the PCG, but additional four vectors must be stored. Experiments in this study indicate that Hager's method converges within four

iterations. In addition, Higham also reported that Hager's method converges within four iterations [10]. Using Hager's method, solving linear equations requires the most computational effort and must be done twice par iteration. Other operations, however, require negligible effort. Thus, the total number of floating point operations can be written as 4 iterations ×2 PCGs ×50 $\times m \times n$. The computational effort required of the modified Hager's method can be as much as ten times larger than effort required of the Lanczos connection. Nevertheless, the memory usage required for each method is almost identical, and the modified Hager's method is more reliable than the Lanczos connection.

Conclusions

In present paper, we developed a condition number estimation method for the preconditioned matrix and verified the accuracy of the newly developed method. Although the Lanczos connection method can be considered as a method by which to estimate the condition number of the preconditioned matrix, this method is weak with respect to numerical errors. Therefore, we compared the condition numbers estimated using both the Lanczos connection method and the newly developed method with Octave's result. The newly developed method provided better results, whereas the Lanczos connection method failed. In addition, the newly developed method can be applied without difficulty for parallel computing and large-scale problem because this new method does not require explicit matrix operation.

Acknowledgments

I would like to thank the CTBTO for giving me the opportunity to publish the article. However, the views expressed are those of the author and do not necessarily reflect the view of CTBTO Preparatory Commission.

Author Contributions

Performed the experiments: NK. Analyzed the data: NK. Contributed reagents/materials/analysis tools: NK. Wrote the paper: NK.

References

1. Saad Y.Iterative Methods for Sparse Linear Systems. 2nd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics; 2003.
2. Barrett R, Berry M, Chan TF, Demmel J, Donato J, Dongarra J, et al. Templates for the solution of Linear Systems: Building Blocks for Iterative Methods. 2nd ed. Philadelphia, PA: SIAM; 1994.
3. Kushida N, Okuda H. Convergence Acceleration of Parallel CG-FEM with Controlled Domain Decomposition for Singularity Problems. Journal of Computational Science and Technology. 2007; 1(1):2–13.
4. Meurant G.The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations (Software, Environments, and Tools). Philadelphia, PA: SIAM; 2006.
5. Garatani K, Nakamura H, Okuda H, Yagawa G. GeoFEM: High performance parallel FEM for solid earth. In: Sloot P, Bubak M, Hoekstra A, Hertzberger B, editors. High-Performance Computing and Networking. vol. 1593 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 1999. p. 133–140. Available from: http://dx.doi.org/10.1007/BFb0100574.
6. Kushida N, Okuda H. Optimization of the Parallel Finite Element Method for the Earth Simulator. Journal of Computational Science and Technology. 2008; 2(1):81–91. Available from: http://ci.nii.ac.jp/naid/130000079323/ doi: 10.1299/jcst.2.81
7. Tatebe O. MGCG METHOD: A ROBUST AND HIGHLY PARALLEL ITERATIVE METHOD. The Graduate School of The University of Tokyo; 1996.
8. Golub GH, Van Loan CF. Matrix Computations. 3rd ed. Baltimore, MD, USA: Johns Hopkins University Press; 1996.
9. Hager WW.Condition estimates. SIAM Journal on Scientific and Statistical Computing. 1984; 5(2):311–316. doi: 10.1137/0905023

10. Higham NJ.FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation. ACM Trans Math Softw. 1988 Dec; 14(4):381–396. Available from: http://doi.acm.org/10.1145/50063.214386.
11. Eaton JW, Bateman D, Hauberg S. GNU Octave version 3.0.1 manual: a high-level interactive language for numerical computations. CreateSpace Independent Publishing Platform; 2009. ISBN 1441413006. Available from: http://www.gnu.org/software/octave/doc/interpreter.
12. Pei ML.A Test Matrix for Inversion Procedures. Commun ACM. 1962 Oct; 5(10):508. Available from: http://doi.acm.org/10.1145/368959.368975.
13. Kushida N. Modified Hager's method; 2015. Available from: http://dx.doi.org/10.5281/zenodo.14912.
14. Balay S, Abhyankar S, Adams MF, Brown J, Brune P, Buschelman K, et al. PETSc Users Manual. Argonne National Laboratory; 2014. ANL-95/11—Revision 3.5. Available from: http://www.mcs.anl.gov/petsc.