Journal of Applied Mathematics and Physics, 2016, 4, 16-20
Published Online January 2016 in SciRes. http://www.scirp.org/journal/jamp
http://dx.doi.org/10.4236/jamp.2016.41003

Efficient Generalized Inverse for Solving Simultaneous Linear Equations

S. Kadiam Bose[1], D. T. Nguyen[2]

[1]Structural Technologies Strong Point LLC, Baltimore, MD, USA
[2]Department of Civil and Environmental Engineering, Old Dominion University, Norfolk, VA, USA
Email: skadi002@odu.edu, dnguyen@odu.edu

Received 22 November 2015; accepted 5 January 2016; published 12 January 2016

Abstract

Solving large scale system of Simultaneous Linear Equations (SLE) has been (and continue to be) a major challenging problem for many real-world engineering and science applications. Solving SLE with singular coefficient matrices arises from various engineering and sciences applications [1]-[6]. In this paper, efficient numerical procedures for finding the generalized (or pseudo) inverse of a general (square/rectangle, symmetrical/unsymmetrical, non-singular/singular) matrix and solving systems of Simultaneous Linear Equations (SLE) are formulated and explained. The developed procedures and its associated computer software (under MATLAB [7] computer environment) have been based on "special Cholesky factorization schemes" (for a singular matrix). Test matrices from different fields of applications have been chosen, tested and compared with other existing algorithms. The results of the numerical tests have indicated that the developed procedures are far more efficient than the existing algorithms.

Keywords

Generalized Inverse Algorithms, Simultaneous Linear Systems, Matrix Inverse, Singular Matrix, Pseudo Inverse, Cholesky Factorization

1. Introduction

In scientific computing, most computational time is spent on solving system of Simultaneous Linear Equations (SLE) which can be represented in matrix notations as

$$Ax = b \tag{1.1}$$

where $A \in R^{n \times n}$ is a singular/non-singular matrix, and b is a given vector in R^n. For practical engineering/science applications, matrix A can be either sparse (for most cases), or dense (for some cases). For a non-singular coefficient matrix A, direct methods (Cholesky factorization, LDL^T algorithm, LU decomposition, etc) or iterative methods (Conjugate Gradient algorithm, Bi-Conjugate Stabilization, GMRES, etc.) are used to solve Equation (1.1). If the coefficient matrix is singular or rectangular, the above mentioned direct and iterative methods cannot be used to solve Equation (1.1) and thus generalized inverse is needed to solve the unknown solution vector x in Equation (1.1).

How to cite this paper: Bose, S.K. and Nguyen, D.T. (2016) Efficient Generalized Inverse for Solving Simultaneous Linear Equations. *Journal of Applied Mathematics and Physics*, **4**, 16-20. http://dx.doi.org/10.4236/jamp.2016.41003

The generalized (or pseudo) inverse of a matrix is an extension of the ordinary/regular square (non-singular) matrix inverse, which can be applied to any matrix (such as singular, rectangular etc.). The generalized inverse has numerous important engineering and sciences applications. Over the past decades, generalized inverses of matrices and its applications have been investigated by many researchers [1]-[6]. Generalized inverse is also known as "Moore-Penrose inverse" or "g-inverse" or "pseudo-inverse" etc.

In this paper we introduce an efficient (in terms of computational time and computer memory requirement) generalized inverse formulation to solve SLE with full or deficient rank of the coefficient matrix. The coefficient matrix can be singular/non-singular, symmetric/unsymmetric, or square/rectangular. Due to popular MATLAB software, which is widely accepted by researchers and educators worldwide, the developed code from this work is written in MATLAB language.

The rest of this paper is organized as follows. In Section 2, we discuss background of generalized inverse. In Section 3, we give a description of the algorithm. This section also describes the efficient generalized inverse formulation (which uses modified Cholesky factorization). In Section 4, we present comparison of numerical performances of the proposed algorithm with other existing algorithms. Extensive set of coefficient matrices (including rectangular, square, symmetrical, non-symmetrical, singular, non-singular matrices) obtained from well-established/popular websites [8] [9] were tested and the numerical performance in terms of timings, error norm were compared with other algorithms. Finally, conclusions are drawn in Section 5.

2. Singular Value Decomposition (SVD) and the Generalized Inverse

A general (square or rectangular) matrix $A \in R^{n\times n}$ can be decomposed as

$$A = U\Sigma V^H \tag{2.1}$$

where

$$\Sigma = \text{a diagonal matrix(does NOT have to be a square matrix)} = \begin{cases} \Sigma_{ij} = 0, \ for\ i \neq j \\ \Sigma_{ij} \geq 0, \ for\ i = j \end{cases} \tag{2.2}$$

$$[U] \ and \ [V] = \text{unitary matrices}$$

$$\text{and} \begin{Bmatrix} U^H = U^T (\text{for real matrices}) \\ U^H = U^{-1} \end{Bmatrix} \tag{2.3}$$

Let A be a singular matrix of size $m \times n$ and let k be the rank of the matrix. Based on Equation (2.1), one has

$$A = U\Sigma V^H;$$

$$where\ \Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_k \end{bmatrix}$$

with

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0; \tag{2.4}$$

and $\sigma_i = \sqrt{\text{Eigen} - \text{Values of } A^T A \text{ (or } AA^T)}$

Note: Eigen-values of $A \times A^H$ and Eigen-values of $A^H \times A$ are the same. However, the Eigen-vectors of $A \times A^H$ and Eigen-vectors of $A^H \times A$ are "NOT" the same.

Then, the generalized inverse A^+ of A is the $n \times m$ matrix and is given as

$$A^+ = V\Sigma^+ \mathrm{U}^{\mathrm{H}} \tag{2.5}$$

where

$\Sigma^+ = \begin{bmatrix} [E] & [0] \\ [0] & [0] \end{bmatrix}$ and E is the $k \times k$ diagonal matrix, with $E_{ii} = \Sigma_i^{-1}$ $for\ 1 \leq i \leq k$.

3. Efficient Generalized Inverse Algorithms [1]-[3] [5] [6]

Moore-Penrose inverse can be computed using Singular Value Decomposition (SVD), Least Squares Method, QR factorizations, Finite Recursive Algorithm [2] [3], etc. In this work, our numerical algorithms have been based on:

(a) The "special Cholesky factorization" (for symmetrical/singular coefficient matrix), and

(b) The generalized inverse of a product of 2 matrices [6] and can be described in the following paragraphs.

The Moore-Penrose inverse (or generalized inverse or pseudo inverse) of a $m \times n$ matrix K (not necessarily a square matrix) is the unique $n \times m$ matrix K^+ which satisfies the following four conditions:

1. General condition: $KK^+K = K$,

2. Reflexive condition: $K^+KK^+ = K^+$,

3. Normalized condition: $\left(KK^+\right)' = KK^+$,

4. Reverse normalized condition: $\left(K^+K\right)' = K^+K$

Consider $[G]\vec{x} = \vec{b}$, with a square coefficient $n \times n$ matrix, and let the rank be less than the size of the matrix (if r is the rank of the matrix, then $r \le n$). Let the size of the known right-hand-side vector $\vec{b}$ be $n \times 1$. Consider a symmetric positive $n \times n$ matrix $G'G$, with rank $r \le n$ (here, the matrix $[G]$ plays the same role as matrix $[A]$ in Equation (1.1)), then based on the theorem presented in [6], there exists a unique $[M]$ such that:

$$G'G = M'M \tag{3.1}$$

In Equation (3.1), matrices $[G']$ and $[G]$ have the dimensions $n \times m$ and $m \times n$, respectively.

M is the upper triangular (special) Cholesky factorized matrix and contains exactly $n - r$ zero rows. Removing the zero rows from M, one obtains a $r \times n$ (upper, rectangular) matrix L'.

$$A \equiv M'M = LL' \tag{3.2}$$

In this work, the upper triangular (special) Cholesky factorized matrix $[M]$ can be obtained by the regular/standard Cholesky factorization, with the following modifications:

a) When the diagonal term of the current i^{th} row is very close to zero, then factorization of this dependent row is skipped.

b) When the current i^{th} row is factorized, all previous rows $k = 1, 2, \ldots, i-1$ were used except those dependent row(s).

Consider the generalized inverse of a matrix product AB [1] [6]

$$\left(AB\right)^+ = B'\left(A'ABB'\right)^+ A' \tag{3.3}$$

From Equation (3.3), if $B = I$ then

$$A^+ = \left(A'A\right)^+ A' \tag{3.4}$$

If $B = A'$ and A is a $n \times r$ matrix of rank r, then one obtains from Equation (3.3)

$$\left(AA'\right)^+ = \left(A'\right)'\left(A'AA'\left(A'\right)'\right)^+ A' \tag{3.5}$$

Let us consider regular inverse in Equation (3.5) in place of generalized inverse

$$\left(AA'\right)^+ = A\left(A'AA'A\right)^{-1}A' = A\left(A'A\right)^{-1}\left(A'A\right)^{-1}A' \tag{3.6}$$

Using Equation (3.4),

$$G^+ = \left(G'G\right)^+ G' \tag{3.7}$$

From Equations (3.1)-(3.2) and Equation (3.6) one obtains,

$$\left(G'G\right)^{+} = \left(LL'\right)^{+} = L\left(L'L\right)^{-1}\left(L'L\right)^{-1}L' \tag{3.8}$$

Thus, Equation (3.7) becomes

$$G^{+} = \left(G'G\right)^{+}G' = L\left(L'L\right)^{-1}\left(L'L\right)^{-1}L'G' \tag{3.9}$$

While MATLAB solution can be obtained by $\vec{x} = pinv(G)\times\vec{b}$, implying the generalized inverse G^{+} [see Equation (3.9)] to be formed explicitly, our main idea is to solve SLE where $\vec{b}$ is a known right-hand-side vector.

4. Numerical Performance of ODU Generalized Inverse Solver

Based on the detailed algorithms explained in Section 3, the numerical performance of our proposed procedures are evaluated in this section. The known RHS vector $\{b\}$ can be random vector, or can be chosen such a way that the unknown solution vector $\{x\} = \{1,1,\ldots,1\}$.

We also compared the performance of our algorithm with the efficient algorithm described in [6] and also with MATLAB built-in function $pinv(\,)$ [7] for computing the generalized inverse explicitly. We use MATLAB version 7.6.0.324 (R2008a) on Intel Core 2 CPU, 2.13 GHZ, 2GB RAM, Windows XP Professional SP3 for numerical comparisons.

Table 1 and **Table 2** records the times (in seconds) taken by our proposed algorithm, the algorithm mentioned in [6] and MATLAB built-in function [7] $pinv(\,)$. For our convenience, we represent our algorithm with $ODU-ginverse(\,)$, algorithm in [6] with $geninv$ and MATLAB built-in function with $MATLAB-pinv(\,)$. In addition, we have also presented the error norm for all the test matrices.

Table 1. Computational times (in seconds) for symmetric rank-deficient test matrices with RHS Vector as linear combination of columns of coefficient matrix.

Sl. No.	Name	Size	Rank	ODU-ginverse Error Norm	geninv Error Norm	MATLAB-pinv ( ) Error Norm
1	lock_700	700 × 700	165	0.1514 1.033×10^{-8}	0.3446 1.1399×10^{-6}	1.2967 2.215×10^{-11}
2	dwt_1005	1005 × 1005	995	2.6634 7.1302×10^{-9}	4.2889 6.764×10^{-6}	14.0320 4.5736×10^{-12}
3	bcspwr06	1454 × 1454	1446	8.5029 1.477×10^{-8}	13.3176 1.829×10^{-5}	40.3646 2.7131×10^{-12}
4	bcsstm13	2003 × 2003	1241	11.5997 6.7629×10^{-9}	19.1901 1.826×10^{-8}	36.3413 5.6493×10^{-13}
5	lock2232	2232 × 2232	368	5.5518 7.9519×10^{-9}	10.8755 2.5797×10^{-7}	40.7582 1.0761×10^{-11}
6	cegb2802	2802 × 2802	289	8.9571 9.7558×10^{-9}	18.6816 3.7220×10^{-7}	69.9847 1.7532×10^{-11}

Table 2. Computational times (in seconds) for rectangular rank-deficient test matrices (Tall type: Rows >> Cols) with RHS Vector as linear combination of columns of coefficient matrix.

Sl. No.	Name	Size	Rank	ODU-ginverse Error Norm	geninv Error Norm	MATLAB-pinv ( ) Error Norm
1	D_6	970 × 435	339	0.1347 1.216×10^{-11}	0.2809 1.333×10^{-11}	1.3240 7.403×10^{-13}
2	mk9-b2	1260 × 378	343	0.1162 5.950×10^{-14}	0.2478 1.018×10^{-13}	0.6098 1.681×10^{-13}
3	Franz1	2240 × 768	755	1.3077 1.457×10^{-13}	2.3649 1.290×10^{-13}	6.0490 2.806×10^{-13}
4	mk10-b2	3150 × 630	586	0.8094 1.599×10^{-13}	1.5776 2.057×10^{-13}	3.2363 2.573×10^{-13}

5. Conclusion

In this paper, various efficient algorithms for solving SLE with full rank, or rank deficient have been reviewed, proposed and tested. The developed numerical procedures can be applied to solve "general" SLE (in the form $[G]\{x\}=\{b\}$, where the coefficient matrix $[G]$ could be square/rectangular, symmetrical/unsymmetrical, non-singular/singular). The users have option to choose either a direct solver or an iterative solver inside the generalized inverse to solve for SLE. Numerical results have shown that the proposed algorithms are highly efficient as compared to existing algorithms [6] (including the popular MATLAB built-in function $pinv(G)*b$) [7].

Acknowledgements

The authors would like to acknowledge Gelareh Bakhtyar for her useful discussions.

References

[1] Nguyen, D.T. (2006) Finite Element Methods: Parallel-Sparse Statics and Eigen-Solutions. Springer Publisher.

[2] Golub, G.H. and Loan, C.F.V. (1996) Matrix Computations. The John Hopkins University Press.

[3] Heath, M.T. (1997) Scientific Computing: An Introductory Survey. McGraw Hill Publisher.

[4] Hou, G. and Wang, Y. (2004) A Substructuring Technique for Design Modifications of Interface Conditions. *Structural Dynamics & Materials Conference*, Palm Springs, California, 19-22 April 2004. http://dx.doi.org/10.2514/6.2004-2010

[5] Farhat, C. and Roux, F.X. (1994) Implicit Parallel Processing in Structural Mechanics. Computational Mechanics Advances, **2**, Elsevier Publisher.

[6] Pierre, C. (2005) Fast Computation of Moore-Penrose Inverse Matrices. *Neural Information Processing—Letters and Reviews*, **8**.

[7] MATLAB, MATLAB—The Language of Technical Computing.

[8] Davis, T. University of South Florida Matrix Collection.

[9] SJSU, SJSU—Singular Matrix Database.