Journal of Algorithms and Computations

journal homepage: http://jac.ut.ac.ir

Metropolis-Hasting Idea for Approximating Matrix Inverse

N. Bagherpour[*1], N. Mahdavi Amiri[†2]

[1] School of Engineering Sciences, College of Engineering, University of Tehran

[2] Department of Mathematics, Sharif University of Technology, Tehran, Iran

ABSTRACT

Solving a linear system of equations is needed in many different applications and there exist many different techniques to solve such a system with no need to compute inverse matrix, as a costly and not stable computation. But the challenge is that in some other applications such as 3D prints, the goal is exactly computing the inverse of a matrix. In this paper, an optimization model equivalent to inverse matrix is introduced and an effective algorithm based on steepest-descent and Barzilai-Borwein step length is suggested. We also used conjugate gradient instead, to provide better numerical results. Finally, we used the Metropolis-Hastings algorithm to accelerate the convergence rate. The probabilities are chosen to have an inverse relation to remainder norm. A key point is that even a random step length is working for global convergence. Numerical results look promising based on stability and accuracy.

Keywords: Inverse matrix, Metropolis-Hastings, conjugate gradient, Barzilai-Borwein step length.

ARTICLE INFO

Article history:
Research paper
Received 07, September 2024
Accepted 05, December 2024
Available online 20, December 2024

AMS subject classification: 65F11

[*] Corresponding author: N. Bagherpour, Email: negin.bagherpour@ut.ac.ir
[†] nezamm@sharif.ir

1 Introduction

If the aim is to solve a linear system, computing the inverse matrix is not necessary. However, in some applications such as geometric design computing the inverse is of interest. Moreover, in usual algorithms for solving nonlinear matrix equations [5] a sequence converging to the inverse matrix is needed. Since computing the exact inverse matrix is complicated specially for large matrices, providing a reletively close approximation would be of high utility. Several algorithms exist based on optimization and numerical linear algebra. The iterative formula'

$$X_{n+1} = X_n(2I - AX_n), \qquad (1)$$

as Newton's iteration, to converge to the solution A^{-1} of $A - X^{-1} = 0$, is used to compute the inverse matrix approximately; e.g., see [2, 3]. We note that finding a suitable starting point which is relatively close to the inverse is not possible; hence, a global algorithm converging to the inverse matrix from an arbitrary starting point is necessary. Here, we present a novel global algorithm to approximate the inverse matrix.

The rest of our work is organized as follows. In Section 2, our newly defined optimization problem and some necessary optimality conditions are presented. In Section 3, two global R-linear algorithms converging to the inverse matrix are outlined. A randomized strategy based on the Metropolis-Hastings algorithm is presented in Section 4 for computing a proper inverse approximation without considerable sensitivity to starting point or extra computational cost for step length. Section 5 includes numerical results obtained using both sequential and parallel implementations of our presented algorithms. Comparisons with the existing methods confirm the efficiency of our algorithm in computing more accurate solutions in less computing times.

2 New Optimization Problem

The iterative formula (1) has been used to solve some nonlinear matrix equations iteratively; see [2, 3]. The Newton iteration (1) converges to the inverse matrix locally, meaning that a starting point sufficiently close to the inverse matrix is necessary for convergence. Here, we develop a globally convergent iterative algorithm to approximate the inverse matrix. The inverse of a non-singular matrix $A \in \mathbb{R}^{n\times n}$ is the solution of the following optimization problem:

$$\min \frac{1}{2} \| AX - I \|_F^2, \qquad (2)$$

where I is the $n \times n$ identity matrix and $\|\cdot\|_F$ denotes the Frobenius norm. In the following two theorems, we discuss some properties of the objective function of (2).

Theorem 1. If $A \in \mathrm{R}^{n\times n}$ is non-singular, then problem (2) has the unique solution $X^* = A^{-1}$. Otherwise, for a singular $A \in \mathrm{R}^{n\times n}$, the general solution of (2) is $A^+ + Z$, where $Z \in \mathrm{R}^{n\times n}$ is an arbitrary matrix with its columns in the null space of A^TA. Moreover, the solution of (2) with minimal Frobenius norm is A^+.

Proof. First, let A be a nonsingular matrix. The objective function $f(X) = \frac{1}{2} \| AX - I \|_F^2$ is strictly convex, because its Hessian, $H = A^T A$, is positive definite. Hence, the staitionary point of $f(X)$ is a global minimizer. We thus have:

$$\nabla f = A^T AX - A^T = 0 \rightarrow AX - I = 0 \rightarrow X^* = A^{-1}.$$

Now, letting A be an $n \times n$ singular matrix, it can be easily seen that the Hessian matrix is positive semi-definite and any stationary point would be a local minimizer. Moreover, $X^* = A^+$ is a solution of $\nabla f = A^T AX - A^T = 0$ and the general solution is $X = A^+ + Z$, where $Z \in \mathrm{R}^{n \times n}$ is an arbitrary matrix with its columns lying in the null space of $A^T A$. Now, we aim to compute the minimal norm solution of (2). Letting $A = U_r \Sigma V_r^T$ be the singular value decomposition (SVD) of A with rank r, we have

$$min \| V_r \Sigma U_r^T + V_{n-r} P \|^2, \tag{3}$$

with P being an arbitrary $(n - r) \times n$ matrix. It is clear that $P = 0$ is the global solution of (3); hence, the solution of (2) with minimal norm is A^+. Now, we are ready to outline the new algorithm for approximating the inverse matrix.

3 New Iterative Algorithm for Inverse

To solve the quadratic optimization problem (2), we first use the steepest descent algorithm. Letting $Q(X) = \frac{1}{2} \| AX - I \|_F^2$, the steepest descent direction would be $-\nabla Q(X) = -A^{\mathrm{T}}(AX - I)$. Moreover, the Barzilai-Borwein (BB) step length [1] is used to guarantee the global convergence of the algorithm. Hence, to compute the solution of (2), starting from an arbitrary matrix $X_0 \in \mathbb{R}^{n \times n}$, in each iteration $k \geq 1$, we set

$$X_{k+1} = X_k - \alpha_k A^{\mathrm{T}}(AX_k - I), \tag{4}$$

where α_k is the BB step length. Letting $S_k = X_k - X_{k-1}$, $Y_k = \nabla Q(X_k) - \nabla Q(X_{k-1})$ and $R_k = S_k - \alpha_{k-1} Y_k$, the BB step length is computed by

$$\alpha_k = \begin{cases} \dfrac{\mathrm{tr}(S_k^{\mathrm{T}} R_k)}{\mathrm{tr}(Y_k^{\mathrm{T}} R_k)}, & \mathrm{tr}(Y_k^{\mathrm{T}} R_k) > 0, \\ \dfrac{\mathrm{tr}(S_k^{\mathrm{T}} S_k)}{\mathrm{tr}(Y_k^{\mathrm{T}} Y_k)}, & \mathrm{tr}(Y_k^{\mathrm{T}} R_k) \leq 0. \end{cases} \tag{5}$$

Now, we give the steps of a steepest descent algorithm for computing the inverse matrix as Algorithm 1. We refer to this algorithm by SDBBI.

Algorithm 1: SDBBI.

(0) Choose an arbitrary matrix $X_0 \in \mathbb{R}^{n \times n}$ and set $k = 0$. let δ be the tolerance for machine zero.

(1) While $\| A * X_k - I \| > \delta$ Do

$$\text{Set } \alpha_k = \begin{cases} \frac{\mathrm{tr}(S_k^{\mathrm{T}} R_k)}{\mathrm{tr}(Y_k^{\mathrm{T}} R_k)}, & \mathrm{tr}(Y_k^{\mathrm{T}} R_k) > 0, \\ \frac{\mathrm{tr}(S_k^{\mathrm{T}} S_k)}{\mathrm{tr}(Y_k^{\mathrm{T}} Y_k)}, & \mathrm{tr}(Y_k^{\mathrm{T}} R_k) \le 0. \end{cases}$$

Set $X_{k+1} = X_k - \alpha_k A^{\mathrm{T}}(AX_k - I)$.
Set $k = k + 1$.

Next, we cite a theorem to establish the R-linear convergence of Algorithm 1.

Theorem 2. The BB step length leads to an R-linear global convergence.

Proof. See [1].

We note that the steepest descent method may produce dependent directions. To avoid this, the conjugate gradient method was introduced. Therefore, to achieve a more reliable performance, a combination of the conjugate gradient mehod with the BB step length was recommended; see [4]. Here, we also suggest using the conjugate gradient method in conjunction with the BB step length (CGBBI) for solving (2).

Algorithm 2: CGBBI.

(0) Choose an arbitrary matrix $X_0 \in \mathbb{R}^{n\times n}$, $\omega > 0$, $\Omega > 0$ and set $d_0 = -A^T(AX_0 - I)$ and $k = 0$. Let δ be the tolerance for machine zero.

(1) While $\| A * X_k - I \| > \delta$ Do

$$\text{Set } \alpha_k = \begin{cases} \frac{\mathrm{tr}(S_k^{\mathrm{T}} R_k)}{\mathrm{tr}(Y_k^{\mathrm{T}} R_k)}, & \mathrm{tr}(Y_k^{\mathrm{T}} R_k) > 0, \\ \frac{\mathrm{tr}(S_k^{\mathrm{T}} S_k)}{\mathrm{tr}(Y_k^{\mathrm{T}} Y_k)}, & \mathrm{tr}(Y_k^{\mathrm{T}} R_k) \le 0. \end{cases}$$

Set $X_{k+1} = X_k + \alpha_k d_k$,
If $Y_k = 0$, then let $t = \frac{2}{\Omega}$,
Elseif $\mathrm{tr}\left(Y_k^{\mathrm{T}} S_k\right) = 0$, then let $t = \frac{2}{\omega}$,
Else let $\hat{q} = \frac{\mathrm{tr}(Y_k^{\mathrm{T}} Y_k)}{\mathrm{tr}(S_k^{\mathrm{T}} Y_k)}$ and $\bar{q} = \frac{\mathrm{tr}(S_k^{\mathrm{T}} Y_k)}{\mathrm{tr}(S_k^{\mathrm{T}} S_k)}$,
If $\mathrm{tr}\left(Y_k^{\mathrm{T}} S_k\right) > 0$, $t = 2\hat{q}$
Else let $c_k = \frac{2(\bar{q} - \hat{q})}{\|S_k\|_F}$ and

$$t = \frac{2c_k \|\nabla Q(X_k)\|_F}{-\hat{q}_k + \sqrt{\hat{q}_k{}^2 + 2c_k \|\nabla Q(X_k)\|_F}}$$

$$t = \begin{cases} \omega & t < \omega \\ \Omega & t > \Omega \\ t & \text{otherwise} \end{cases},$$

$$\beta_k = \max\left(\frac{\mathrm{tr}\left(\nabla Q(X_{k+1})^T(Y_k - tS_k)\right)}{\mathrm{tr}(d_k^{\mathrm{T}} Y_k)}, 0\right)$$
$$d_{k+1} = -\nabla Q(X_{k+1}) + \beta_k d_k,$$

Set $k = k + 1$.

Additionally, the Newton iteration for approximating the inverse is faster than *R*-linear algorithms but does not converge globally. To overcome this deficiency, a natural idea is to use the SDBBI or the CGBBI algorithm in early iterations to compute a relatively good starting point for the Newton method and use the Newton algorithm to converge to the inverse matrix faster. The hybrid algorithm constructed from the SDBBI and Newton's method, HBBNI, is presented as Algorithm 3 below.

Algorithm3: HBBNI.

(0) Choose an arbitrary matrix $X_0 \in \mathbb{R}^{n\times n}$ and set $k = 0$. Let δ_1 be the tolerance for machine zero and $\delta_1 > \delta_2$ be a sufficiently small number.
(1) While $\| A * X_k - I \| > \delta_1$ Do

Set $\alpha_k = \begin{cases} \frac{\mathrm{tr}(S_k^{\mathrm{T}} R_k)}{\mathrm{tr}(Y_k^{\mathrm{T}} R_k)}, & \mathrm{tr}(Y_k^{\mathrm{T}} R_k) > 0, \\ \frac{\mathrm{tr}(S_k^{\mathrm{T}} S_k)}{\mathrm{tr}(Y_k^{\mathrm{T}} Y_k)}, & \mathrm{tr}(Y_k^{\mathrm{T}} R_k) \le 0. \end{cases}$

Set $X_{k+1} = X_k - \alpha_k A^{\mathrm{T}}(AX_k - I)$.
Set $k = k + 1$.
(2) Set $X_0 = X_k$ and $k = 0$.
(3) While $\| A * X_k - I \| > \delta_2$ Do
Set $X_{k+1} = X_k(2I - AX_k)$.
Set $k = k + 1$.

The most important challenge of the presented algorithms would be their computational complexity, specially in case of large matrices. So, in Section 4 we make use of the brilliant idea by Metropolis to discard with solutions and control the complexity.

4 Metropolis-Hastings Idea

In this section, a randomized approach will be defined to improve the computational cost and convergence properties of the outlined algorithm for approximating inverse matrix. There are some motivating issues to continue with randomized approach such as Metropolis-Hastings:

a) An extra cost is needed for computing the BB step length.
b) In first iterations, we do not necessarily see improvement in error function.
c) Performing a greater number of iterations although lead to a better result, increase the computational cost.

To overcome these challenges, we make use of Metropolis approach. We compute *k* different approximations $X_i, \ldots i = 1, \ldots, k$, each by performing *t* iterations with random step lengths. We

then generate a convex combination of X_is as a proper inverse approximation. We note that these k approximations might be computed in parallel. Moreover, a random step length is of more interest in comparison with previously used step length BB. Before specifying the convex combination, let review the Metropolis-Hastings algorithm.

To provide a guided sampling among a large number of samples, based on a desired probability distribution, the Metropolis-Hastings algorithm [6] as a Markov chain Monte Carlo (MCMC) method might be used. In this algorithm, new samples are added to the sequence in two steps: first a new sample is proposed randomly, then the proposed sample is either added to the sequence or rejected depending on the value of the probability distribution at that point. The resulting sequence can be used to approximate the distribution. Moreover, to follow a target distribution, the probability conditions for accepting a sample might be computed based on the target distribution. In Figure 1 the Metropolis Hastings scheme is showed.

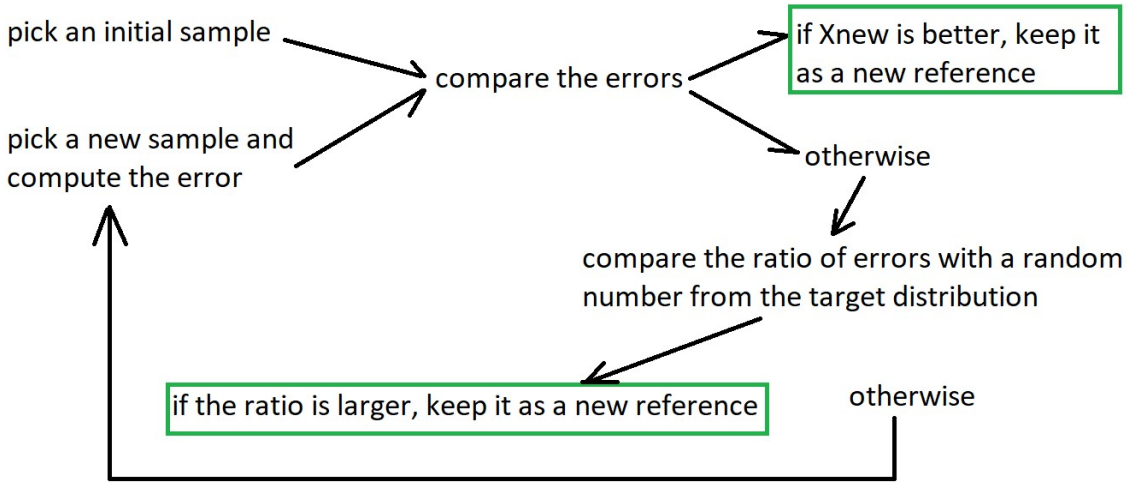

Figure 1: Metropolis Hastings Scheme

In inverse approximation, since our target is to minimize the norm $\|AX - I\|$, among k computed approximations each X_i with lower residue is a better approximation for inverse matrix and must have a larger weight in the desired convex combination. To this end, we define a target probability distribution

$$P(X_i) = \frac{\frac{1}{\|AX_i - I\|}}{\sum_{j=1}^{k} \frac{1}{\|AX_j - I\|}}$$

so that a reasonable combination would be constructed. So, the first approach for approximating inverse is

1) Compute $\widetilde{X}_i$s in parallel for $i = 1, \dots, k$ by performing the iteration below d times

$$X_{k+1} = X_k - \alpha_k A^{\mathrm{T}}(AX_k - I)$$

with α_k being a random positive number.

2) Compute $P(\widetilde{X}_i) = \frac{\frac{1}{\|A\widetilde{X}_i - I\|}}{\sum_{j=1}^{k} \frac{1}{\|A\widetilde{X}_j - I\|}}$ and let $X = \sum_{j=1}^{d} P(\widetilde{X}_j)\widetilde{X}_j$.

A more creative approach is to generate random matrices X_is and the convex combination based on their residues forms the approximation for the inverse matrix. The point here is that

- The computational cost for $X_{k+1} = X_k - \alpha_k A^{\mathrm{T}}(AX_k - I)$ is not needed any more,
- however, a larger number of iterations must be done based on the randomness of the original candidates.

To summerize the steps of our work, we start with designing an optimization problem whose solution is the inverse matrix which is of interest. We then, search for a proper optimization algorithm to approximate the solution. Finally, to improve the approximation and accelerate convergence we made use of the Metropolis-Hasting idea. These main steps are shown in Figure 2.

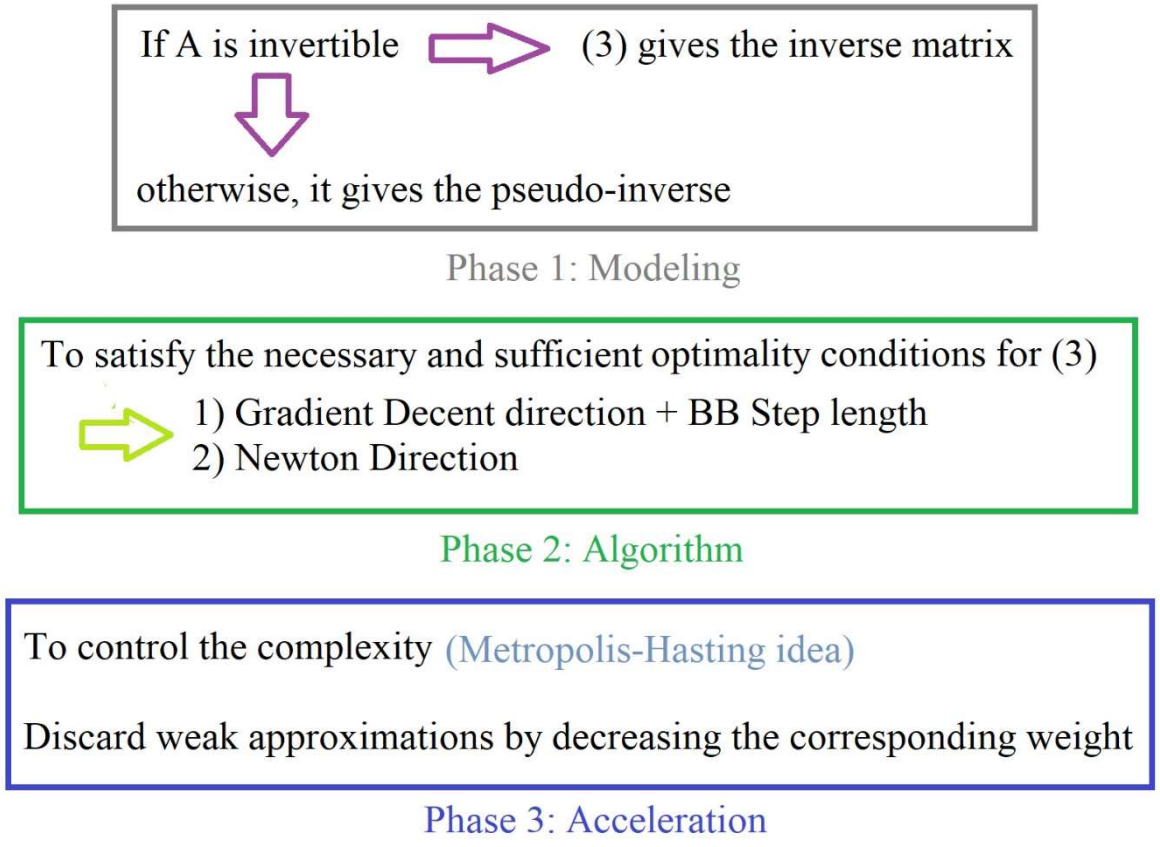

Figure 2: The overal scheme

Next, we present some numerical results to show the effectiveness of our proposed algorithm for computing the inverse matrix both in sequential and parallel implementations. According to the

numerical results, the Metropolis-Hasting idea outperforms the first approach in sparse matrices and other special structures.

5 Numerical Results

MATLAB 2024b in a Windows 11 machine with a 3.4 GHz CPU and a 16 GB RAM is used to implement the algorithms and compare the obtained results. We present the numerical results for the sequential and parallel implementations, respectively. We generate random $n \times n$ matrices of different sizes. These random matrices were produced using the rand command in MATLAB. The command rand(n) generates an $n \times n$ non-singular matrix with uniformly distributed random entries in the interval $[0, 1]$.

Sequential Implementation

In Table 1, the computing times for approximating the inverse matrix using implemntation of the SDBBI, CGBBI, HBBNI (sequential) algorithms, SVD and LU factorization are reported. Based on these results, the computing time of the sequential HBBNI algorithm is considerably less than those obtained by SVD and LU factorization. In tables below, n is the matrix size, δ_1 and δ_2 are the first and second error bounds in the HBBNI algorithm. The error bounds in SDBBI and CGBBI are assumed to be δ_2.

According to numerical results reported in Table 1, in almost all of the test problems the HBBNI algoritthm, as compared to the SDBBI and CGBBI algorithms, shows the best performance in computing a relatively accurate inverse approximation. This is indeed reasonable considering the following points:

- computational complexity is lower than exact matrix decompositions.
- Newton's method quadratic convergence is helpful to decrease the computing time.
- The need of Newton's algorithm for a proper starting point is fulfilled with the help of steepest descent method in early iterations.

Table 1. Computing times for approximating the inverse matrix by sequential SDBBI, CGBBI, HBBNI algorithms, SVD and LU factorization.

N	δ_1	δ_2	*CGBBI*	*SDBBI*	*HBBNI*	*SVD*	*LU*
50	$E-006$	$E-010$	$2.418E-006$	$1.032E-006$	$3.981E-007$	$2.091E-006$	$3.873E-006$
100	$E-006$	$E-010$	$8.794\mathrm{E}-006$	$5.763E-006$	$6.453E-007$	$4.985E-006$	$6.164E-006$
200	$E-006$	$E-010$	$3.118-005$	$1.172E-005$	$9.783E-006$	$1.093E-005$	$3.274E-005$
300	$E-006$	$E-010$	$4.916\mathrm{E}-005$	$4.872E-005$	$2.114E-005$	$6.574E-005$	$7.459E-005$
500	$E-006$	$E-010$	$6.384\mathrm{E}-005$	$7.661E-005$	$5.675E-005$	$1.334E-004$	$2.783E-004$

Parallel Implementation

The results obtained by parallel implementations are given in Table 2. We note that parallel HBBNI algorithm is faster than parallel implementation of SVD and LU factorization, on all the considered cases.

Based on the numerical results in tables 1 and 2, the hybrid algorithm is suggested in real applications for approximating the inverse matrix. We note that the most important benefit of matrix decompositions is that they provide the exact inverse matrix in case it is necessary.

Note: For large test problems, the CGBBI algorithm outperforms the SDBBI algorithm; hence, to implement the hybrid algorithm for large problems, the combination of conjugate gradient and Newton's methods is considered.

Table 2. Computing times for approximating the inverse matrix by parallel SDBBI, CGBBI, HBBNI algorithms, SVD and LU factorization.

N	δ_1	δ_2	*CGBBI*	*SDBBI*	*HBBNI*	*SVD*	*LU*
1000	$E-004$	$E-008$	$1.821\mathrm{E}-005$	$2.087E-005$	$1.653E-005$	$2.065E-005$	$3.115E-005$
3000	$E-004$	$E-008$	$4.053\mathrm{E}-005$	$4.116E-005$	$3.435E-005$	$3.875E-005$	$5.016E-005$
5000	$E-004$	$E-008$	$7.426\mathrm{E}-005$	$8.125E-005$	$6.773E-005$	$7.453E-005$	$9.342E-005$
7000	$E-004$	$E-008$	$1.275\mathrm{E}-004$	$1.984E-004$	$1.135E-004$	$1.675E-004$	$3.562E-004$
10000	$E-004$	$E-008$	$7.689\mathrm{E}-004$	$7.115E-004$	$8.474E-004$	$7.563E-004$	$1.105E-003$

Moreover, to provide a visual comparison of the computing times, we present the Dolan-More performance profiles. As shown in Figure 2, the time profile for the HBBNI algorithm appearing above the other profiles, the efficiency of the proposed algorithm is confirmed.

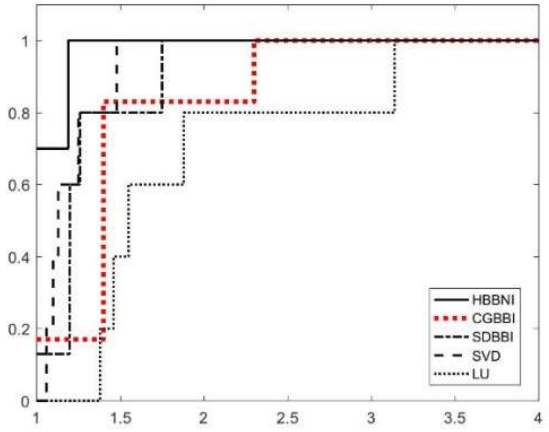

Figure 2: The Dolan-More Time Profile

Metropolis Hastings

In this section, the efficiency of Metropolis Hastings based approach for inverse approximation is discussed. In Table 3, the residue bound, *k* and *d* are reported and the computing time is compared with SDBBI and CGBBI.

Table 3: Metropolis Hasting approach based on steepest descent iteration

N	*k*	*d*	δ	*CGBBI*	*SDBBI*	*Metropolis Hastings*
1000	*10*	*10*	$E-008$	$2.087E-003$	$1.653E-003$	$*\ 1.281E-003$
3000	*20*	*15*	$E-008$	$4.116E-003$	$3.435E-003$	$*\ 2.354E-003$
5000	*25*	*20*	$E-008$	$8.125E-003$	$6.773E-003$	$*\ 4.327E-003$
7000	*25*	*20*	$E-008$	$1.984E-002$	$1.135E-002$	$*\ 1.032E-002$
10000	*30*	*25*	$E-008$	$7.115E-002$	$8.474E-002$	$*\ 5.463E-002$

We note that, the Metropolis Hasting algorithm leads to same error bound in lower time and with no need to specific step length. These properties make the idea a proper one in real applications where inverse matrix is needed. In Table 4, the computing time and number of randomized iterations are reported in approximating inverse of diagonal, tridiagonal and sparse matrices using the second approach.

Table 4: The randomized Metropolis Hasting approach for special structures

N	Structure	*n*	*Time*
1000	diagonal	*80*	$1.104E-003$
3000	tridiagonal	*200*	$1.435E-003$
5000	diagonal	*200*	$1.618E-003$
7000	tridiagonal	*300*	$6.253E-003$
10000	diagonal	*300*	$4.539E-003$
10000	Sparse	*500*	$6.472E-002$

Finally, the average computing time for computing the inverse of tridiagonal large matrices, as a well-known case, with residue norm less than $E-008$ is plotted in Figure 3.

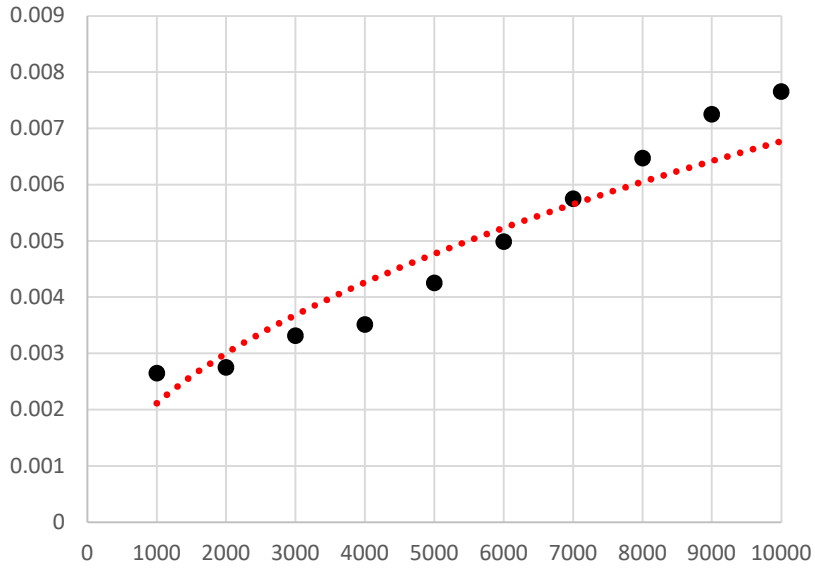

Figure 3: time vs size, tridiagonal inverse matrix approximation using Metropolis Hastings

The red plot in Figure 3, is a proper regressor for the computing time which shows a controlled growth of computing time whit increasing matrix size. This makes the algorithm reliable for larger matrices.

6 Concluding Remarks

We first defined an optimization problem with its solution being an inverse matrix. To solve the otimization problem, we outlined an efficient global algorithm, steepest descent method with Barzilai-Borwein step length, named as SDBBI, conevrging to the inverse matrix. Instead of steepest-descent, conjugate gradient method was also considered in the CGBBI algorithm. On large test problems, the CGBBI algorithm performs better than the SDBBI algorithm. Based on the obtained numerical results, CGBBI turns to be more efficient than SDBBI method, while all both proposed algorithms are faster than conventional decomposition methods such as the LU factorization and SVD in approximating the inverse matrix. Moreover, we introduced two approaches based on Metropolis Hastings algorithm to make the computations more efficient specially for large matrices. The target probability distribution is formed based on residue norms which provides a mathematically reasonale combination. The numerical results shows to be promising in computing inverse matrix faster and more accurate than SDBBI and CGBBI. The test where conducted both in general structures and special structures such as tridiagonal matrices.

Acknowledgement

The author would like to thank Professor Ali Moeini for his valuable comments to improve the randomized version of the algorithm.

References

[1] Barzilai J., Borwein J. M.: Two-point Step Size Gradient Method, *IMA J. Numer. Anal.*, Vol. 8, No.1, 1988, pp. 141-148.

[2] Ben-Israel A.: An iterative method for computing the generalized inverse of an arbitrary matrix, *Math. Comp.*, Vol. 19, 1965, pp. 452-455.

[3] Ben-Israel A., Cohen D.: On iterative computation of generalized inverses and associated projections, *SIAM J. Numer. Anal.*, Vol. 3, 1966, pp. 410-419.

[4] Momeni M., Peyghami M. R.: A new conjugate gradient algorithm with cubic Barzilaiâ€“Borwein stepsize for unconstrained optimization, *Optimization Methods and Software,* Vol. 34, No. 3, 2019, pp. 650-664.

[5] Zhou B., Cai G. B., Lam J.: Positive definite solutions of the nonlinear matrix equation $X + A^T \bar{X}^{-1} A = I$, *Appl. Math. Comput.,* Vol. 219, No. 14, 2013, pp.7377-7391.

[6] Brown A., Jones G. L.: Convergence rates of Metropolis–Hastings algorithms, WIREs Computational Statistics, Advanced Review, 2024, DOI: 10.1002/wics.70002.