Computer Science Journal of Moldova, vol.**18**, no.3(54), 2010

Matrix balancing and robust Monte Carlo algorithm for evaluating dominant eigenpair

Behrouz Fathi Vajargah Farshid Mehrdoust

Abstract

Matrix balancing may effect the stability of algorithms in matrix computations and the accuracy of computed solutions. In this paper, we first introduce an algorithm for matrix balancing. Then, using Monte Carlo method we propose a robust algorithm to evaluate dominant eigenpair of a given matrix. Finally, several randomly generated examples are presented to show the efficiency of the new method.

Keywords: Monte Carlo algorithms; Robust Monte Carlo algorithm; Markov chain; Balancing; Eigenpair; Large scale matrices

1 Introduction

The need to compute dominant eigenpair of matrices arises frequently in scientific and engineering applications with the solution being useful either by itself or as an intermediate step in solving a larger problem. There are many different algorithms presently used to obtain eigenpair of a matrix, among them are the Householder method, the QR method and subspace iteration [6, 7]. Many of these algorithms are inefficient when applied to very large structural systems. Krylov subspace Lanczos method is widely appreciated by the numerical analysis community [6, 7]. The problem of using Monte Carlo and quasi Monte Carlo methods for finding an eigenpair has been extensively studied, for example [2-7]. In this paper, we study the Monte Carlo approach to obtain the dominant eigenpair of matrices with an emphasis on preconditioning

implementation of the corresponding algorithm which is called matrix balancing. We employ a special balancing procedure as a preprocessing step before running the Monte Carlo procedure. Such a balancing procedure ensures robustness of the Monte Carlo algorithm and therefore relatively small values for the stochastic error.

Let A be an $n \times n$ real matrix whose eigenvalues we seek. The pair (λ, x) is called an eigenpair of A if

$$Ax = \lambda x, \quad x \neq 0. \tag{1}$$

In equation (1) the scaler λ and the vector x are called an eigenvalue and eigenvector, respectively. Throughout the paper we suppose that the matrix $A \in \mathbb{R}^{n\times n}$ is diagonalizable with eigenvalues

$$\lambda_{max} = |\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_k|.$$

Note that this implies λ_1 is real, otherwise $\overline{\lambda}_1$(conjugate of λ_1) is another eigenvalue with the same magnitude as λ_1 [7].

2 Monte Carlo approach for computing (λ, x)

Consider the following Markov chain with length i

$$T_i : k_0 \to k_1 \to \cdots \to k_i \tag{2}$$

where $k_j \in \{1, 2, ..., n\}$ for $j = 1, 2, ..., i$.
The following choice of p_α and $p_{\alpha,\beta}$, for constructing $T_i's$, is considered:

$$p(k_0 = \alpha) = p_\alpha, \qquad p(k_j = \beta | k_{j-1} = \alpha) = p_{\alpha\beta}, \tag{3}$$

where p_α and $p_{\alpha\beta}$ show the probability of starting chain at α and transition probability from state α to β, respectively. We further should have that

$$\sum_{\alpha=1}^{n} p_\alpha = 1 \tag{4}$$

and

$$\sum_{\beta=1}^{n} p_{\alpha\beta} = 1 \tag{5}$$

for each $\alpha = 1, 2, ..., n$. Probabilities $p_{\alpha\beta}$ define the transition matrix P.

Let matrix $A \in \mathbb{R}^{n\times n}$ and two vectors $f, h \in \mathbb{R}^n$ are given. Further suppose that the distributions created from the density probabilities p_α and $p_{\alpha\beta}$ are acceptable according to the following definition [2]

$$p_\alpha > 0 \quad \text{when} \quad h_\alpha \neq 0, \qquad p_\alpha \geq 0 \quad \text{when} \quad h_\alpha = 0 \tag{6}$$

and

$$p_{\alpha\beta} > 0 \quad \text{when} \quad a_{\alpha\beta} \neq 0, \qquad p_{\alpha\beta} \geq 0 \quad \text{when} \quad a_{\alpha\beta} = 0. \tag{7}$$

We define the random variable W_j using the following recursive equation

$$W_j = W_{j-1} \frac{a_{k_{j-1}k_j}}{p_{k_{j-1}k_j}}, \quad j = 1, 2, ..., i, \tag{8}$$

where $W_0 = \frac{h_{k_0}}{p_{k_0}}$. Then we may apply the following probability structures

$$\begin{aligned} p_\alpha &= \frac{|h_\alpha|}{\sum_{\beta=1}^{n} |h_\beta|} \\ p_{\alpha\beta} &= \frac{|a_{\alpha\beta}|}{\sum_{\beta=1}^{n} |a_{\alpha\beta}|} \qquad \alpha, \beta = 1, 2, ..., n. \end{aligned} \tag{9}$$

Thus, from (8) and (9) we have

$$\begin{aligned} W_i &= \sum_{\beta=1}^{n} |h_\beta| \sum_{\beta=1}^{n} |a_{k_0\beta}| \sum_{\beta=1}^{n} |a_{k_1\beta}| ... \sum_{\beta=1}^{n} |a_{k_{i-1}\beta}| \prod_{j=1}^{i} \frac{a_{k_{j-1}k_j}}{|a_{k_{j-1}k_j}|} \frac{h_{k_0}}{|h_{k_0}|} \\ &= \sum_{\beta=1}^{n} |h_\beta| \prod_{j=0}^{i-1} \sum_{\beta=1}^{n} |a_{k_j\beta}| sign\{h_{k_0} \prod_{j=1}^{i} a_{k_{j-1}k_j}\}. \end{aligned} \tag{10}$$

Now, let us define the random variable $\Gamma^{(i)}$ as

$$\Gamma^{(i)} = W_i f_{k_i} = \sum_{\beta=1}^{n} |h_\beta| \prod_{j=0}^{i-1} \sum_{\beta=1}^{n} |a_{k_j\beta}| sign\{h_{k_0} \prod_{j=1}^{i} a_{k_{j-1}k_j}\} f_{k_i}, \quad (11)$$

where $\Gamma^{(i)}$ will be employed for evaluating $< h, A^i f >$, which is used for estimating λ_{max}. We follow the above discussions in the next theorem.

Theorem 1. *Under assumptions 2 to 9, the random variable $\Gamma^{(i)}$ is an unbiased estimator for $< h, A^i f >$ i.e.,*

$$E[\Gamma^{(i)}] =< h, A^i f > . \quad (12)$$

Proof. *We have*

$$\begin{aligned} E[\Gamma^{(i)}] &= E[\frac{h_{k_0}}{p_{k_0}} \frac{a_{k_0k_1}...a_{k_{i-1}k_i}}{p_{k_0k_1}...p_{k_{i-1}k_i}} f_{k_i}] \\ &= \sum_{k_0,...,k_i=1}^{n} \frac{h_{k_0}}{p_{k_0}} \frac{a_{k_0k_1}...a_{k_{i-1}k_i}}{p_{k_0k_1}...p_{k_{i-1}k_i}} f_{k_i} p_{k_0} p_{k_0k_1}...p_{k_{i-1}k_i} \\ &= \sum_{k_0=1}^{n} h_{k_0} \sum_{k_1=1}^{n} a_{k_0k_1}... \sum_{k_{i-1}=1}^{n} a_{k_{i-2}k_{i-1}} \sum_{k_i=1}^{n} a_{k_{i-1}k_i} f_{k_i} \\ &= \sum_{k_0=1}^{n} h_{k_0} \sum_{k_1=1}^{n} a_{k_0k_1}... \sum_{k_{i-1}=1}^{n} a_{k_{i-2}k_{i-1}} (Af)_{k_{i-1}} \\ &= \sum_{k_0=1}^{n} h_{k_0} (A^i f)_{k_0} =< h, A^i f > . \quad \blacksquare \end{aligned}$$

Now, let us simulate N random paths as T_i defined in (2), (3) and suppose $\Gamma_s^{(i)}$ is the s^{th} realization of random variable $\Gamma_s^{(i)}$ i.e.,

$$\Gamma_s^{(i)} = \sum_{\beta=1}^{n} |h_\beta| \{\prod_{j=0}^{i-1} \sum_{\beta=1}^{n} |a_{k_j\beta}| sign(\prod_{j=1}^{i} a_{k_{j-1}k_j} f_{k_i})\}_s, \qquad s = 1, ..., N.$$

Then the value

$$\bar{\Gamma}^{(i)} = \frac{1}{N} \sum_{s=1}^{N} \Gamma_s^{(i)}, \qquad i \geq 1 \quad (13)$$

is considered as a Monte Carlo approximation of $< h, A^i f >$.

Based on the power method [7], Monte Carlo method for evaluating the dominant eigenvalue as $i \to \infty$, is

$$\lambda_{max} \approx \frac{\overline{\Gamma}_s^{(i)}}{\overline{\Gamma}_s^{(i-1)}}. \tag{14}$$

Let the goal is to find the eigenvector x that corresponds to the eigenvalue λ, then we set $h = e(j) = (0, ..., 0, 1, 0, ..., 0)$, where $e(j)$ is the j^{th} unit vector in $\mathbb{R}^n$, i.e. $(e(j))_\alpha = \delta_{j\alpha}$. It follows that

$$< h, x >= \sum_{\alpha=1}^{n} (e(j))_\alpha x_\alpha$$

and the j^{th} component of eigenvector x using Monte Carlo method is

$$x_j \approx \frac{1}{N} \sum_{s=1}^{N} \{\Gamma^{(i)}[e(j)]\}_s. \tag{15}$$

Theorem 2. *The stochastic error for calculating the dominant eigenvalue of a given matrix $A \in \mathbb{R}^{n\times n}$, based on the Monte Carlo algorithm is minimized if for each i and for some $L > 0$, we have $\sum_{j=1}^{n} |a_{ij}| = L$, i.e. the absolute rowsums of A be a constant number.*

Proof. *Consider the following random variable*

$$\Theta_l(f) = \sum_{i=0}^{l} W_i f_{k_i}, \tag{16}$$

where $f \in \mathbb{R}^n$ is an arbitrary vector and W_i is the variable, defined in (8). It is easy to see that

$$\begin{aligned} Var[\Theta_l(f)] &= \sum_{i=0}^{l} Var[W_i f_{k_i}] + 2 \sum_{i} \sum_{j>i} Cov(W_i f_{k_i} W_j f_{k_j}) \\ &\leq \sum_{i=0}^{l} Var[W_i f_{k_i}] + 2 \sum_{i} \sum_{j>i} \sigma(W_i f_{k_i}) \sigma(W_j f_{k_j}). \end{aligned}$$

Therefore, it is sufficient to minimize $Var[W_i f_{k_i}]$.
We further have

$$\begin{aligned} E[(W_i f_{k_i})^2] &= \sum_{k_0 k_1 \cdots k_i=1}^{n} \frac{h_{k_0}^2}{p_{k_0}} \frac{a_{k_0 k_1}^2 \cdots a_{k_{i-1} k_i}^2}{p_{k_0 k_1} \cdots p_{k_{i-1} k_i}} f_{k_i}^2 \\ &= \prod_{k=0}^{i-1} \sum_{j=1}^{n} |a_{kj}| \sum_{k_0 k_1 \cdots k_i=1}^{n} |h_{k_0}||a_{k_0 k_1}| \ldots |a_{k_{i-1} k_i}| f_{k_i}^2 \\ &= \prod_{k=0}^{i-1} \sum_{j=1}^{n} |a_{kj}| \sum_{k_0=1}^{n} |h_{k_0}| (|A|^i, f_{k_i}^2)_{k_0} \\ &= \prod_{k=0}^{i-1} \sum_{j=1}^{n} |a_{kj}| (|h|, |A|^i f^2). \end{aligned}$$

Now, since f *and* h *are arbitrary vectors, without loss of generality assume that* $f = (1, \cdots, 1)^T$, $h = (\frac{1}{n}, \cdots, \frac{1}{n})$. *Now, we suppose that* $A = |A| = (|a_{i,j}|)_{i,j=1,\ldots,n}$, *thus from the previous equality it follows*

$$\prod_{k=0}^{i-1} \sum_{j=1}^{n} |a_{kj}| = (h, A^i f),$$

hence we easily conclude that $Var[W_i f_{k_i}]$ *vanishes when* $\sum_{j=1}^{n} |a_{kj}|$ *is equal to a constant value* L, *for* $k = 1, 2, \ldots, n$. ■

The above theorem shows that to reduce the Monte Carlo error it is important to deal with balanced matrices. In the next section, we introduce matrix balancing procedure based on the condition given in Theorem 2.

3 Balancing matrices

Balancing is a preprocessing step, which may produce positive effects on the accuracy and performance of numerical methods for computing eigenvalues. A matrix $A \in \mathbb{R}^{n \times n}$ with a norm that is several orders

of magnitude larger than the modules of its eigenvalues typically has eigenvalues that are sensitive to perturbations in the entries of A. One of the main methods is Sinkhorn-Knopp algorithm [1]. There are other algorithms for balancing that can converge faster than the Sinkhorn-Knopp algorithm, for example, Parlett and Landis [6]. In this paper, we have used a Krylov-based balancing algorithm proposed in [1].

Definition 1. An $n \times n$ matrix A with nonnegative entries is said to be balanced if for each $i = 1, ..., n$, the sum of the entries of its i^{th} row is equal to the sum of the entries of its i^{th} column. In other words,

$$A\mathbf{e} = A^T\mathbf{e}, \tag{17}$$

where $\mathbf{e}$ is the n-vector of all ones.
More generally, an $n \times n$ matrix with arbitrary real entries is said to be balanced in $l^p - norm$ if for each $i = 1, ..., n$ its i^{th} row and column have the same $l^p - norm$. Our employed balancing algorithm is as follows:

Algorithm 1. (Balancing algorithm)

1. **Input** $A \in R^{n\times n}$, t (number of iterations)
2. **for** s=1 : t
2.1 **Set** vector $z_{n\times 1}$ of random numbers $1, -1s$
2.2 **Compute** $p = Az$
2.3 **for** $i = 1 : n$
2.4 **if** $(p(i) = 0)$ then
2.4.1 **Set** $D(i) = 1$
2.4.2 **else** $D(i) = \frac{1}{p(i)}$
2.4.3 **end** for
2.5 **Set** $A = D * A * D^{-1}$
2.6 **end** for
3. **end of** algorithm 1

Now, we present the robust Monte Carlo algorithm based on the balanced matrix. In this algorithm, we use partitioned random number generator for generating Markov chains (random trajectories). In fact, we may divide the interval $(0, 1)$ to r subintervals with equal length $\frac{1}{r}$,

where r is a natural number greater than 1. We use it in rand function of MATLAB software for generating random numbers with more uniformity. We note that in the following algorithm each row of the $N \times i$ matrix ZZ is a Markov chain with the length i. According to Theorem 2, we further consider the vector $h = (\frac{1}{n}, \ldots, \frac{1}{n})^T$.

Algorithm 2. (Robust Monte Carlo algorithm)

1. **Input** the matrix $A_{n\times n}$, the number of Markov chains N and the length of Markov chains i
2. **Call** algorithm 1 for balancing matrix A
3. **Generate** transition probability matrix $P = (p_{ij})_{i,j=1,\ldots,n}$ according to the equation (9)
4. **Generate** an $N \times i$ random matrix ZZ
5. **Set** D1=0; D2=0
6. **for** $s = 1 : N$
7. **Set** $z = ZZ(s, :)$ ▶ s^{th} row of the matrix ZZ
8. **Set** $v = A(z(i-1), :)$
9. **if** $A(z(i-1), z(i)) \neq 0$ **then**
10. **Set** $D1 = D1 + sign(A(z(i-1), z(i))) * norm1(v) * \frac{1}{n}$
11. **Set** $D2 = D2 + h(z(i-1))$
12. **End if**
13. **End for**
14. Approximation of dominant eigenvalue is $D1/D2$
15. **End** algorithm 2

4 Computational results

In the following tables we compare the precision of the computed dominant eigenpair without balancing and after applying balancing algorithm. All test problems were run in MATLAB software on a PC with Intel(R) 1.83 GHz Dual CPU processor. Moreover, the Monte Carlo relative error was computed by the following formula

$$MC\ relative\ error = \frac{|MC\ result - exact\ result|}{|exact\ result|}.$$

First, let us consider the following test matrix

$$A = \begin{pmatrix} 0.6716 & 0.2417 & 0.2461 & 0.4788 & 0.1615 \\ 0.7003 & 0.1290 & 0.3725 & 0.3583 & 0.9478 \\ 0.9097 & 0.3089 & 0.9758 & 0.8556 & 0.5761 \\ 0.2902 & 0.5112 & 0.0766 & 0.6052 & 0.1597 \\ 0.7199 & 0.3323 & 0.2303 & 0.7844 & 0.8600 \end{pmatrix}.$$

Basing on the Theorem 2, the rowsums vector for matrix A before balancing is

$$rowsums = (1.7997, 2.5079, 3.6261, 1.6429, 2.9269)^T,$$

and after balancing is

$$rowsums = (2.3035, 2.0470, 2.4542, 2.1937, 2.5738)^T,$$

where the balanced matrix is

$$A = \begin{pmatrix} 0.6716 & 0.4076 & 0.5187 & 0.4671 & 0.2385 \\ 0.4153 & 0.1290 & 0.4655 & 0.2073 & 0.8299 \\ 0.4316 & 0.2472 & 0.9758 & 0.3960 & 0.4036 \\ 0.2975 & 0.8837 & 0.1655 & 0.6052 & 0.2418 \\ 0.4875 & 0.3795 & 0.3287 & 0.5182 & 0.8600 \end{pmatrix}.$$

As we see, the rowsums after applying balancing algorithm are more centralized. In Table 2, we have compared the precision of the computed eigenvalues without balancing and after applying balancing algorithm for several randomly generated matrices.

Table 1. MC relative error for calculating dominant eigenpair $(\hat{\lambda}, \hat{v})$

Number of trajectories	$\hat{\lambda}, \hat{v}$ (without bal.)	Time (s)	$\hat{\lambda}, \hat{v}$ (with bal.)	Time (s)
100	$0.0797, 0.2504$	0.03	$0.0069, 0.1637$	0.05
1000	$0.0727, 0.1340$	0.14	$0.0041, 0.0571$	0.16

Table 2. MC relative error in computing the dominant eigenvalue for several randomly generated matrices

Dimension	MC method	
	$\widehat{\lambda}_1$ (with bal.)	$\widehat{\lambda}_1$ (without bal.)
100	4.1289×10^{-4}	0.1600×10^{-2}
200	5.4924×10^{-5}	3.0526×10^{-4}
400	1.7083×10^{-4}	0.1100×10^{-2}
800	2.9271×10^{-5}	0.1900×10^{-2}
1600	1.3916×10^{-5}	3.4172×10^{-4}
3200	5.1900×10^{-6}	3.7581×10^{-4}

5 Concluding remarks

In this paper, we have proposed a new algorithm for obtaining dominant eigenpair of desired matrices. By this algorithm, we are able to reduce the stochastic error in Monte Carlo method. The numerical experiments have shown that the Algorithm 1 not only makes the matrix more balanced, but also balances the sum of absolute values in the rows which is desirable according to the Theorem 2. The proposed algorithm has been implemented for large scale matrices in computing dominant eigenvalue and we can conclude that the balancing of the input matrix is very important for improving the accuracy of Monte Carlo algorithm (Figure 1 and Figure 2).

References

[1] Chen T. Y. and Demmel J. W., *Balancing sparse matrices for computing eigenvalues, Linear algebra and its applications,* 309(2000)261-287.

[2] Dimov I.T. and Alexandrov V.N., *A new highly convergent Monte Carlo method for matrix computations.* Mathematics and Computers in Simulation, 47(1998)165-181.

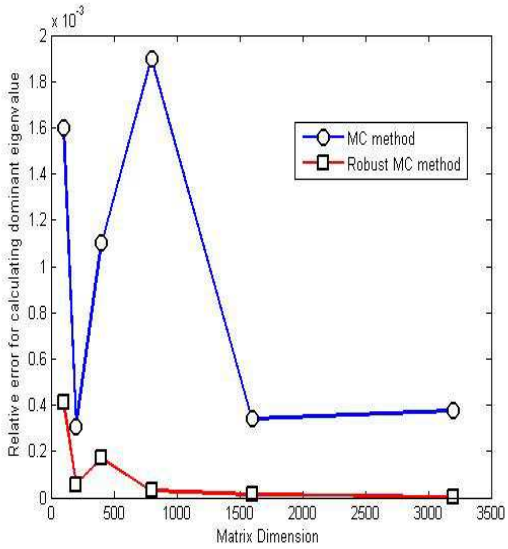

Figure 1. Comparison of relative error between Monte Carlo and robust Monte Carlo methods for various matrices

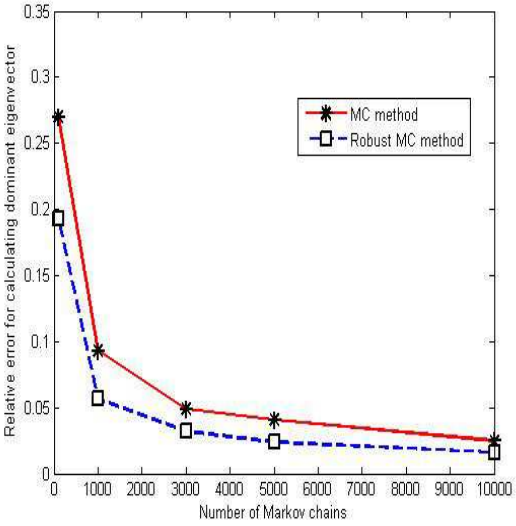

Figure 2. Comparison of relative error between Monte Carlo and robust Monte Carlo methods for various Markov chains

[3] Fathi Vajargah B. and Mehrdoust F., *New Monte Carlo algorithm for obtaining three dominant eigenvalues,* IJAM, 22(2009)553-559.

[4] Fathi Vajargah B. and Mehrdoust F., *New hybrid Monte Carlo methods and computing the dominant generalized eigenvalue*, International journal of computer mathematics, Taylor & Francic, 2009.

[5] Hammersley J. M. and Handscomb D. C., *Monte Carlo Methods.* Methuen, London, (1964).

[6] Kressner D., *Numerical Methods for General and Structured Eigenvalue Problems,* Springer-Verlag Berlin Heidelberg, (2005).

[7] Meyer K. D., *Matrix analysis and applied linear algebra,* SIAM, (2000).

Behrouz Fathi Vajargah, Farshid Mehrdoust

Received June 17, 2010
Revised January 31, 2011

Faculty of Mathematical Sciences,
University of Guilan,
Rasht, P.O. Box: 1914, Iran
E–mail: *fathi@guilan.ac.ir*,
fmehrdoust@guilan.ac.ir