A RANDOMIZED ALGORITHM FOR ESTIMATING THE CONDITION NUMBER OF MATRICES

FARSHID MEHRDOUST

Communicated by the former editorial board

In this paper, we propose a randomized algorithm based on quasi– Monte Carlo method for estimating the condition number of a given matrix. Computational results on various problems prove that on especially large scale matrix the proposed algorithm is much faster than the MATLAB **cond** function.

AMS 2010 Subject Classification: 65C05, 35P15, 34L16.

Key words: randomized algorithm, quasi–Monte Carlo method, Markov chain, condition number.

1. INTRODUCTION

The condition number of matrices is a widely used matrix feature in many areas, such as in numerical analysis, linear algebra etc. [2]. In numerical analysis, the condition number is basically a measure of stability or sensitivity of a matrix to numerical operations. Suppose that A be a nonsingular matrix then, the linear system, $Ax = b$, has a unique solution, $x = A^{-1}b$. Now, if we perturb the matrix A while keeping the vector b fixed, then we have [4]

$$\frac{\|\delta x\|}{\|x + \delta x\|} \leq \|A\|\|A^{-1}\|\frac{\|\delta A\|}{\|A\|}. \tag{1}$$

The quantity $\|A\|\|A^{-1}\|$ which appears in (1) reflects the maximum possible relative change in the exact solution of a linear system induced by a change in the data. The condition number of a nonsingular matrix A is defined by

$$\kappa_p(A) = \|A\|_p\|A^{-1}\|_p, \tag{2}$$

where p is usually $1, 2, \infty$. Throughout this paper we consider 2-norm, however we note that there are some relationship among $\kappa_p(A)$. For example [4]

$$n^{-3}\kappa_1(A) \leq n^{-1}\kappa_\infty(A) \leq \kappa_2(A) \leq n\kappa_\infty(A) \leq n^3\kappa_1(A) \tag{3}$$

The matrix A is called a well-conditioned matrix, if $\kappa_p(A)$ is relatively small and A is called ill-conditioned matrix if $\kappa(A)$ is large. Since $I = AA^{-1}$ and $\|AA^{-1}\| \leq \|A\|\|A^{-1}\|$, we can see $\kappa(A) \geq 1$.

Suppose that the values λ_1 and λ_n are largest and smallest eigenvalue of matrix A^TA, respectively. Then we have [4]

$$\kappa(A) = \left(\frac{\lambda_1}{\lambda_n}\right)^{\frac{1}{2}}, \tag{4}$$

where $A_{n\times n}$ is a given nonsingular matrix.

There are several methods for estimating the condition number of large matrices [1, 3]. In this paper, we present a new algorithm based on quasi Monte Carlo algorithm to estimate the condition number of large matrices. The computational cost of proposed algorithm on large dimensions is significantly lower than MATLAB **cond** function.

2. RESOLVENT QUASI–MONTE CARLO METHOD

Here, we describe some results based on quasi Monte Carlo methods using resolvent matrix and then extend it for evaluating the eigenvalue problem. Now, we consider computing the eigenvalues of the matrix A

$$Ax = \lambda x, \tag{5}$$

where the matrix A is symmetric. Let us order all the eigenvalues of A in decreasing order, *i.e.*

$$\lambda_{max} = \lambda_1 > \lambda_2 \geq \ldots > \lambda_n = \lambda_{min}.$$

Now, consider an algorithm based on Monte Carlo iterations using resolvent operator $R_q = [I - qA]^{-1}$, where q is a parameter that can be chosen such that $|q| < \frac{1}{\|R\|}$.

It is known that [6]

$$R_q^m = [I - qR]^{-m} = \sum_{i=0}^{\infty} q^i C_{m+i-1}^i A^i. \tag{6}$$

The eigenvalues of the operators $[I - qA]^{-1}$ and R are connected with the equality $\mu = \frac{1}{1-q\lambda}$. If $q > 0$, then the largest eigenvalue of the resolvent matrix R_q^m corresponds to the largest eigenvalue of the matrix A, but if $q < 0$, then it corresponds to the smallest eigenvalue of the matrix A. Also, as $m \to \infty$ we have [6]

$$\mu^{(m)} = \frac{([I - qA]^{-m} f, h)}{([I - qA]^{-(m-1)} f, h)} \longrightarrow \mu = \frac{1}{1 - q\lambda}, \qquad f, h \in R^n. \tag{7}$$

Here, we apply continuous Markov chain showed by T_l of length l starting at state x_0

$$T_l : x_0 \to x_1 \to \cdots \to x_l$$

where $x_j \in D_j$ show the chosen state , for each $j = 1, 2, \ldots, l$, also $D_j = [j-1, j)$ and $D = \bigcup_{j=1}^{n} D_j = [0, n)$. Assume that

$$p(x) = p_i, \quad x \in G_i, \quad \text{for a given} \quad h \in R^n \tag{8}$$

and

$$p(x, y) = p_{ij}, \quad x \in G_i,\ y \in G_j, \tag{9}$$

are the probability of starting chain at x_0 and transition probability from state x_i to y_j, respectively.

Now, define the random variable W_j^Q based on quasi Monte Carlo methods by the following recursion equation

$$W_0 = \frac{h(x_0)}{p(x_0)}, \quad W_j^Q = W_{j-1}^Q \frac{a(x_{j-1}, x_j)}{p(x_{j-1}, x_j)}, \quad j = 1, 2, \ldots, l. \tag{10}$$

It is shown [2] that

$$E[W_i f_{k_i}] =< A^i f, h >, \quad i = 1, 2, \ldots. \tag{11}$$

Thus, we can write

$$< R_q^m f, h >= E[\sum_{i=0}^{\infty} q^i C_{m+i-1}^i (A^i f, h)], \quad m = 1, 2, \ldots. \tag{12}$$

Using the Rayleigh quotient [3] we have

$$\lambda \approx \frac{E \sum_{i=0}^{l} q^i C_{m+i-1}^i W_{i+1}^Q}{E \sum_{i=0}^{l} q^i C_{m+i-1}^i W_i^Q}. \tag{13}$$

Therefore, we have

$$\lambda \approx \frac{\sum_{i=0}^{l} q^i C_{m+i-1}^i \int_{D_0} \int_{D_1} \cdots \int_{D_i} W_{i+1}^Q dx_0 \cdots dx_{i+1}}{\sum_{i=0}^{l} q^i C_{m+i-1}^i \int_{D_0} \cdots \int_{D_i} W_i^Q dx_0 \cdots dx_i}. \tag{14}$$

Now, by some simple calculations we get

$$\lambda \approx \frac{\sum\limits_{i=0}^{l} q^i C_{m+i-1}^i \int\limits_{D_0} \cdots \int\limits_{D_{i+1}} h(x_0) a(x_0, x_1) a(x_1, x_2) \cdots a(x_i, x_{i+1}) \mathrm{d}x_0 \cdots \mathrm{d}x_{i+1}}{\sum\limits_{i=0}^{l} q^i C_{m+i-1}^i \int\limits_{D_0} \cdots \int\limits_{D_i} h(x_0) a(x_0, x_1) a(x_1, x_2) \cdots a(x_{i-1}, x_i) \mathrm{d}x_0 \cdots \mathrm{d}x_i}. \tag{15}$$

In (15), the multi-dimensional integral can be efficiently estimated by quasi Monte Carlo methods [7].

3. THE PROPOSED RANDOMIZED ALGORITHM

Given an $m \times n$ matrix A, we seek to approximate its condition number. We note that for computing the largest and smallest eigenvalue of $A^T A$ we have to obtain the multi-dimensional integral [4]. This leads to the following algorithm for estimating $\kappa(A)$.

1. **Compute** $C = A^T A$.
2. **Call** QMC algorithm for computing $\lambda_1(C)$ and $\lambda_n(C)$ [2, 6].
3. **Return** $\kappa(A) = \frac{\lambda_1(C)}{\lambda_n(C)}$.
4. End of algorithm.

4. IMPLEMENTATION DETAILS AND RUNNING TIME OF ALGORITHM

A special property of our presented algorithm is that it is very simple and can be easily implemented. The important ingredient of this algorithm is to computing the eigenvalues of the matrix $A^T A$ by QMC algorithm. The computational complexity for QMC is $\mathcal{O}((l+1)N)$, where N is the number of chains, $l+1$ is the length of a single Markov chain [3]. Therefore, the proposed algorithm has the linear complexity. In other hand, the function **cond** in MATLAB software uses the SVD. There are many algorithms that either exactly compute the SVD of a matrix in $\mathcal{O}(mn^2 + m^2 n)$ time or using Lanczos methods to approximate it faster [3]. Then, if we want to approximate the condition number of a given matrix by direct method, the computational cost for computing $\|A^{-1}\|$ is too high and thus, not applicable for large scale problems.

5. NUMERICAL RESULTS

In this section, we present some numerical results to show the performance of quasi Monte Carlo algorithm to MATLAB **cond** function. We run our results on workstation, Intel(R) 1.83 GHz Dual CPU, 2.00 GB RAM using MATLAB software 7.6. The number of Markov chains, N, and length of a single Markov chain, l, in these experiments are 1000 and 10, respectively.

Estimation of condition number for the following matrix is considered.

$$A = \begin{pmatrix} 3n+1 & 1 & \cdots & & 1 \\ 1 & 3n+1 & 1 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \\ 1 & \cdots & 1 & 3n+1 & 1 \\ 1 & \cdots & & 1 & 3n+1 \end{pmatrix}_{3n\times 3n}$$

Results are summarized in Table 1 and Figure 1 for different values of n.

Table 1
Comparison of QMC algorithm and cond function in MATLAB

n	QMC-cond	Time (s)	MATLAB-cond	Time (s)
100	1.9998	2.16	2.0000	0.06
200	1.9998	2.28	2.0000	0.40
400	1.9998	2.61	2.0000	3.46
800	1.9998	4.77	2.0000	25.44
1600	1.9998	20.23	2.0000	194.14
2000	1.9998	36.15	2.0000	372.04

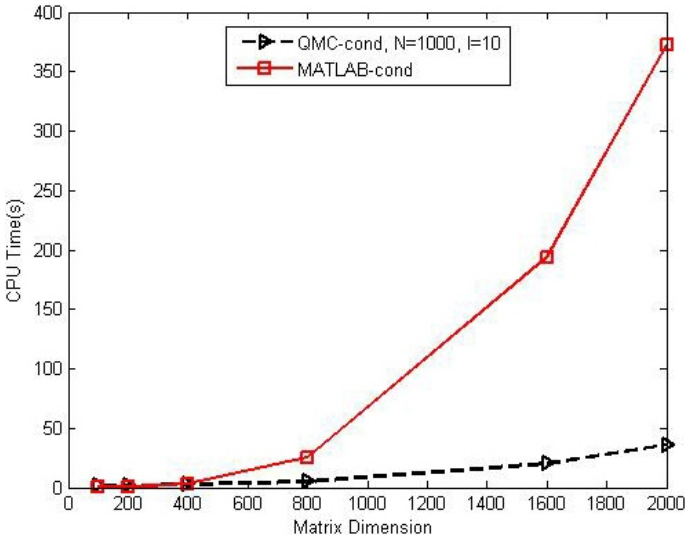

Fig. 1 – Time comparison for QMC and MATLAB procedures.

In this example, three different classes of matrices are considered. Results are outlined in Table 2.

Table 2
Comparison of QMC algorithm and cond function in MATLAB for some matrices, $n = 2000$

Type of matrix	$A = rand(n)$	A=$gallery('tridiag', n)$	$A = hilb(n)$
$\lambda_{max}(A^TA), \lambda_{min}(A^TA)$	$1.0007\times10^{6}, 1.8685\times10^{-4}$	$16, 6.2754 \times 10^{-12}$	$6.8567, 2.4921\times10^{-22}$
QMC-cond	7.3180×10^{4}	1.5968×10^{6}	2.7514×10^{22}
Time (s)	3.79	4.19	3.76
MATLAB-cond	7.3180×10^{4}	1.6228×10^{6}	2.0691×10^{22}
Time (s)	17.87	33.11	17.44

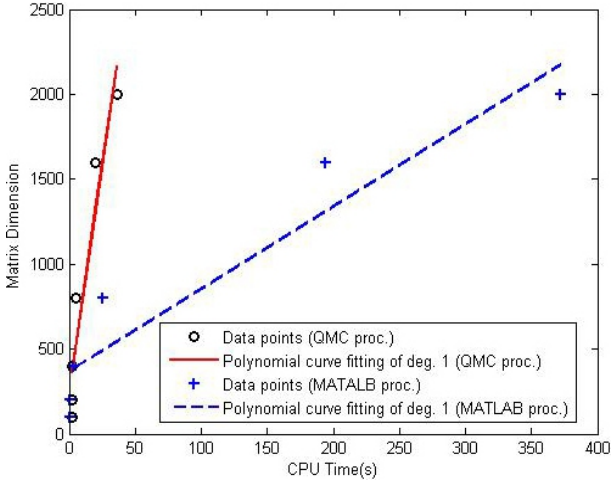

Fig. 2 – Time comparison and polynomial curves fitting for QMC and MATLAB procedures.

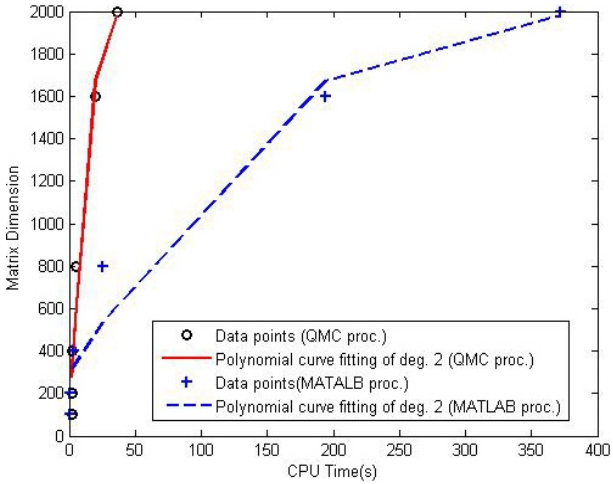

Fig. 3 – Time comparison and polynomial curves fitting for QMC and MATLAB procedures.

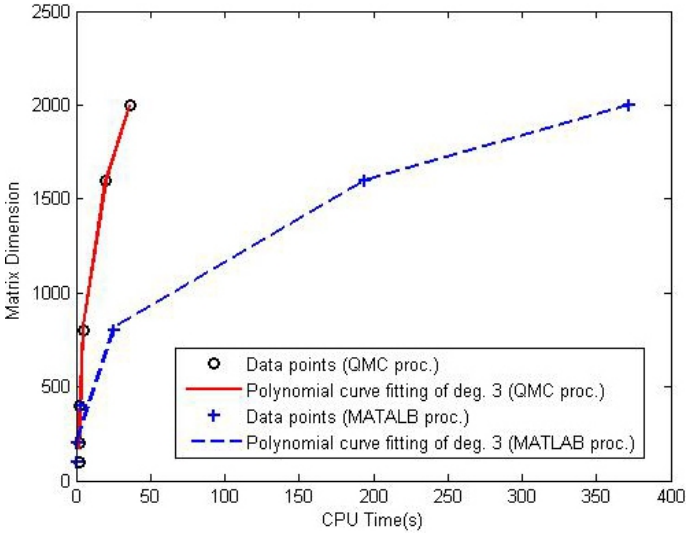

Fig. 4 – Time comparison and polynomial curves fitting for QMC and MATLAB procedures.

6. **CONCLUSION**

In this paper, we explain a randomized algorithm for estimating the condition number of matrices. Our computational results show that QMC algorithm is extremely faster than MATLAB cond function, while resulting to very good approximation of the condition number. Also, in Figs. 2–4 we see that the correlation of computational time and dimension of matrix for QMC procedure and MATLAB procedure is quadratic and cubic, respectively. Roughly, we can say that in QMC algorithm, the computational time and dimension of matrix have linear correlation (see Fig. 2).

REFERENCES

[1] Q. Fang, *A note on the condition number of a matrix.* J. Comput. Appl. Math. **157** (2003), 231–234.

[2] B. Fathi Vajargah and F. Mehrdoust, *New Monte Carlo algorithm for obtaining three dominant eigenvalues.* Internat. J. Appl. Mech **22** (2009), 553–559.

[3] G.H. Golub and C.F. Van Loan, *Matrix Computations.* The Johns Hopkins University Press, 1996.

[4] N.J. Higham, *Efficient algorithm for computing the condition number of a tridiagonal matrix.* SIAM J. Sci. Stat. Comput. **7** (1986), 150–165.

[5] C. Lemieux, *Monte Carlo and Quasi-Monte Carlo Sampling.* Springer Science, 2009.

[6] M. Mascagni and A. Karaivanova, *A Parallel Quasi-Monte Carlo Method for Computing Extremal Eigenvalues.* Monte Carlo and Quasi-Monte Carlo Methods, Springer, 2000, 369–380.
[7] I.M. Sobol, *On quasi-Monte Carlo integrations.* Math. Comput. Simulation **47** (1998), 103–112.

Received 29 June 2011

University of Guilan,
Faculty of Mathematical Science,
Department of Applied Mathematics,
Rasht, Iran
fmehrdoust@guilan.ac.ir