ISE 407 Project Report

Numerical Analysis for various Cholesky Factorization Methods

Baoyu Zhou

December 9, 2018

1 Introduction

In this project, we try to figure out the best method for doing Cholesky factorization. Cholesky factorization is an important mathematical technique in linear algebra, by which we can use an upper triangular matrix R to represent a positive definite symmetric matrix A, where $A = R^T R$. Cholesky factorization could also be widely applied to various optimization problems. For example, as [3] shown, we can solve a Quadratic Programming by the help of Cholesky factorization process.

We know there are at least 3 ways to get R by given A: (1) by basic linear algebra process (2) by Cholesky update process and (3) by *chol()* function in MATLAB. So in this project, I'll try to implement all three methods for realizing Cholesky factorization process in MATLAB, and comparing their performance in multiple ways. I would also try to test the same method under different architectures to check whether the same performance still remains.

2 Mathematical Background

In this section, we introduce the mathematical background of algorithms we try to implement.

From linera algebra, we know that for a given symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$, we can always calculate an upper triangular matrix $R \in \mathbb{R}^{n\times n}$ such that $A = R^T R$. And there are at least two ways to get R from A: Direct Calculation process and Cholesky Update process. We will talk about both methods in detail.

2.1 Direct Calculation Process

Because of the special architecture of matrix R, we would have

$$\begin{cases} R_{(1,1)}^2 = A_{(1,1)},\ R_{(1,2:n)} = \frac{A_{(1,2:n)}}{R_{(1,1)}}; \\ R_{(2,2)}^2 = A_{(2,2)} - \|R_{(1,2)}\|^2,\ R_{(2,3:n)} = \frac{A_{(2,3:n)} - R_{(1,2)}\cdot R_{(1,3:n)}}{R_{(2,2)}}; \\ R_{(3,3)}^2 = A_{(3,3)} - \|R_{(1:2,3)}\|^2,\ R_{(3,4:n)} = \frac{A_{(3,4:n)} - R_{(1:2,3)}^T R_{(1:2,4:n)}}{R_{(3,3)}}; \\ \vdots \\ R_{(n-1,n-1)}^2 = A_{(n-1,n-1)} - \|R_{(1:n-2,n-1)}\|^2,\ R_{(n-1,n)} = \frac{A_{(n-1,n)} - R_{(1:n-2,n-1)}^T R_{(1:n-2,n)}}{R_{(n-1,n-1)}}; \\ R_{(n,n)}^2 = A_{(n,n)} - \|R_{(1:n-1,n)}\|^2, \end{cases}$$

where $\|\cdot\|$ means the l_2 norm.

By observing the system above, we know that every variable is on the left hand side of "=" and we can get R directly from A by solving the system above from up to down.

2.2 Cholesky Update Process

Assume we already know a symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$, and an upper triangular matrix $R \in \mathbb{R}^{n\times n}$ satisfying $A = R^T R$, then for the known $u \in \mathbb{R}^n$

and $v \in \mathbb{R}$, we try to find an upper triangular matrix $R' \in \mathbb{R}^{(n+1)\times(n+1)}$ satisfies $A' = R'^T R'$, where $A' = \begin{bmatrix} A & u \\ u^T & v \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$ is a symmetric positive definite matrix as well.

Let $R' = \begin{bmatrix} R_1 & R_2 \\ 0 & R_3 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$, where $R_1 \in \mathbb{R}^{n\times n}$ being an upper triangular matrix, $R_2 \in \mathbb{R}^n$ and $R_3 \in \mathbb{R}$. Then from $A' = R'^T R'$, we would have

$$\begin{cases} A = R_1^T R_1, \\ u = R_1^T R_2, \\ v = R_2^T R_2 + R_3^T R_3. \end{cases}$$

So we can simply let $R_1 = R$, then we only need to solve for R_2 and R_3, where

$$u = R^T R_2 \text{ and } v = R_2^T R_2 + R_3^T R_3.$$

Because A is positive definite and $A = R^T R$, then R is invertible. We can solve the system above for R_2 and R_3 and get

$$\begin{aligned} R_2 &= R^{-T} u, \\ R_3 &= \sqrt{v - R_2^T R_2}. \end{aligned} \tag{1}$$

In our case, we can solve for R from A by using the method above recursively. Let's define $R = \mathit{recur}(A)$, where $A \in \mathbb{R}^{n\times n}$. When $n = 1$, we can simply get $R = \sqrt{A}$. When $n \geq 2$, we can write matrix A as $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix}$, where $A_{11} \in \mathbb{R}^{(n-1)\times(n-1)}$, $A_{12} \in \mathbb{R}^{(n-1)}$ and $A_{22} \in \mathbb{R}$. Because A is a symmetric positive definite matrix, we would know A_{11} is a symmetric positive definite matrix as well. We can also write R

as $R = \begin{bmatrix} R_1 & R_2 \\ 0 & R_3 \end{bmatrix}$, where $R_1 \in \mathbb{R}^{(n-1)\times(n-1)}$, $R_2 \in \mathbb{R}^{(n-1)}$ and $R_3 \in \mathbb{R}$. From method above, we would know that

$$R_2 = R_1^{-T} A_{12},$$
$$R_3 = \sqrt{A_{22} - R_2^T R_2},$$

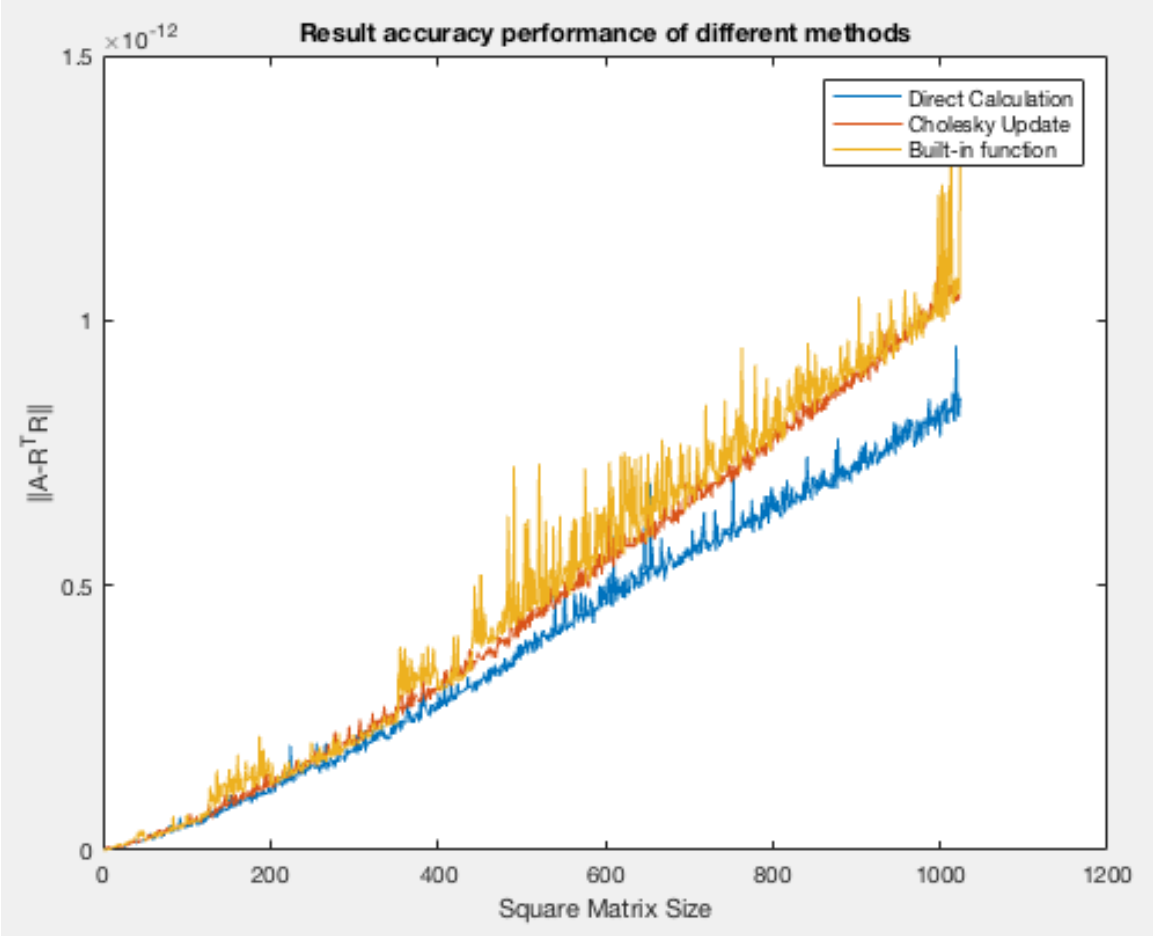
where $R_1 = recur(A_{11})$.

3 Numerical Analysis

In addition to the two methods we mentioned above for calculating Cholesky factorization matrix, there is also a built-in function in MATLAB called *chol()*. For the aim of numerical analysis, we compare the accuracy and time-cost performance of all the three methods.

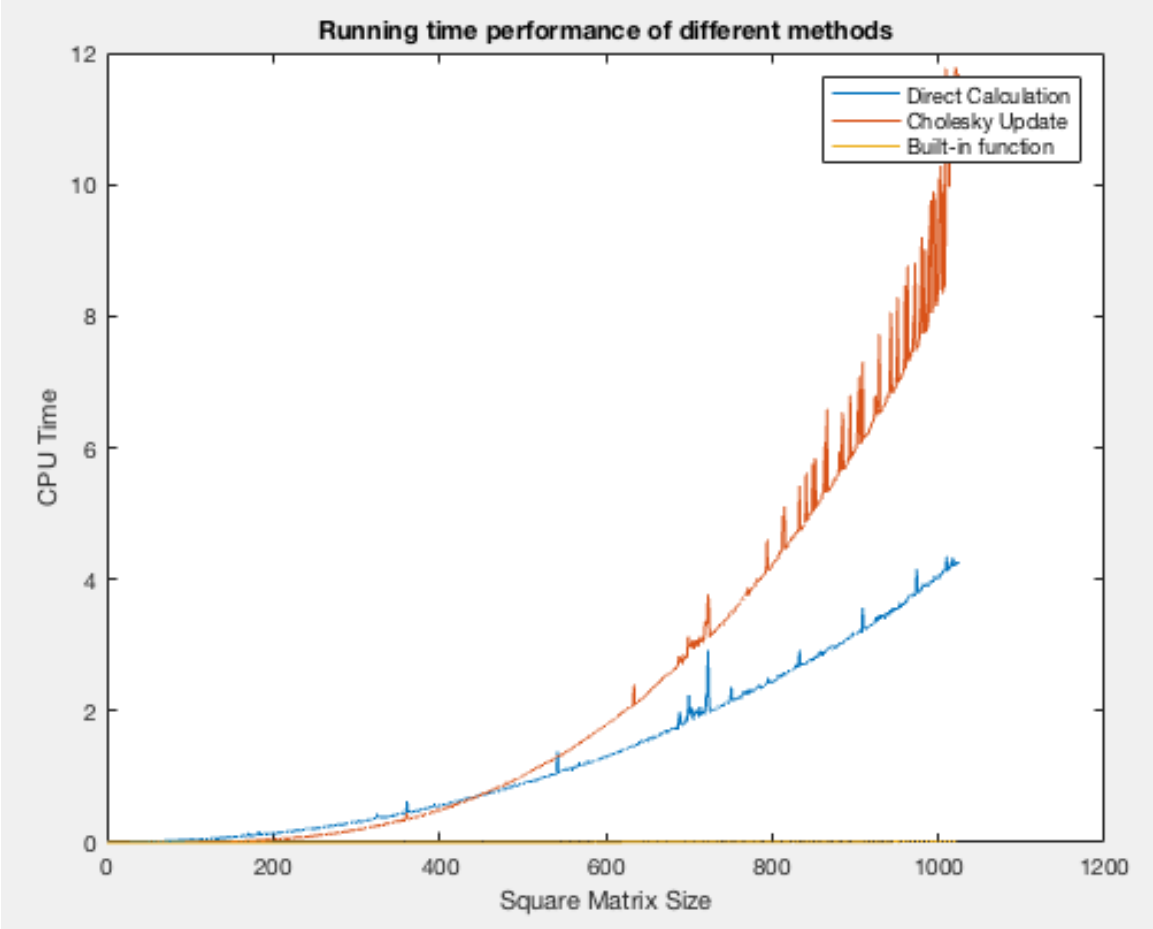
For the square matrix $A \in \mathbb{R}^{n\times n}$, we vary n from 1 to 1024, and implement all three methods to get the Cholesky matrix R (which is an upper triangular matrix)

such that $R^T R = A$. We get accuracy results as below:



From the graph above, we can see all three methods provide stable correct results, because the spectral norm of $(A - R^T R)$ always maintains fewer than 10^{-11}. However, comparing to other two methods, the direct calculation process provides a relatively more accurate result. The cholesky update process also has a better performance than the MATLAB built-in function, in terms of the result accuracy.

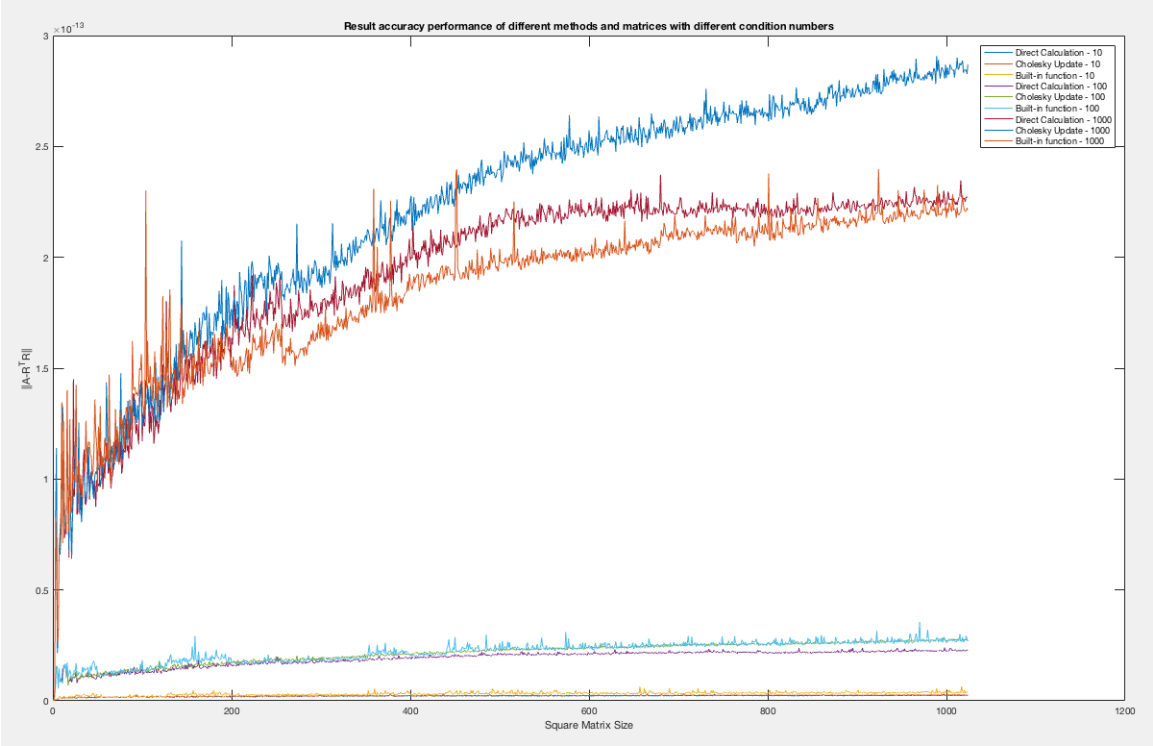
We can also get the time-cost performance results as below:



We can see that the MATLAB built-in function can solve all sizes problem really quickly, which is much better than other two methods' performance. When the matrix size is relatively small (smaller than 400×400), cholesky update process has a better performance than direct calculation process. However, direct calculation process performs better than cholesky update process when the problem has a larger size of matrix.

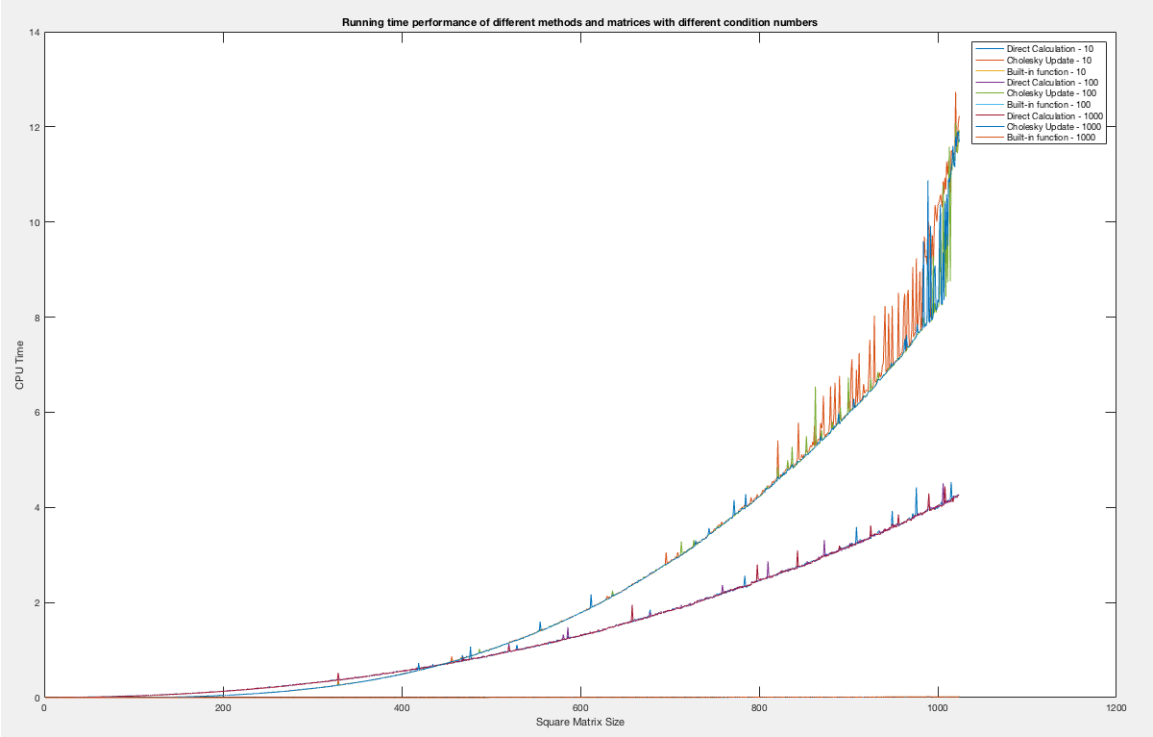
We also take the condition number of matrices into consideration. We generate A as random positive symmetric matrices with fixed condition numbers: 10, 100, 1000. And try to see the time cost and result accuracy of all three methods by solving matrices with different condition numbers.

The accuracy results is shown below as:



We can see that all methods provide a correct solution when solving matrices with various condition numbers. The largest spectral norm of $(A - R^T R)$ is even smaller than 3×10^{-13}. When condition number is small, the direct calculation process performs better, and when the condition number turns to 1000 (which is relatively large), the other two methods have a better performance.

The time-cost performance results performs as:



From the result above, we can see that no matter how condition number varies. the MATLAB built-in function can always solve the problem a small amount of time. When the size of matrix is small, cholesky update process has a better performance than direct calculation process, in terms of the running time. However, when the size of matrix is large, direct calculation process has a much better performance than cholesky update process. Problem for solving A with a large condition number would make direct calculation process and cholesky update process be unstable in terms of performance of running time, which makes sense to us.

4 Conclusion

After comparing all three methods' performance in terms of running time and result accuracy, we can conclude the MATLAB built-in function has the best performance

among all three methods. For a small size problem (where size of square matrix A is smaller than 400×400), the cholesky update process has a better performance than direct calculation process. However, the direct calculation process performs better when we try to solve a problem with matrix A being with a larger size than 400×400.

References

[1] S.J. Wright, J. Nocedal. *Numerical Optimization*. Springer, Berlin, 1999.

[2] S. Boyd, L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] K.C. Kiwiel. *A Method for Solving Certain Quadratic Programming Problems Arising in Nonsmooth Optimization*. IMA Journal of Numerical Analysis, 6 (1986), pp. 137 - 152