Warsaw, 23rd May 2022

Name: Yusupov Yuldashbek
ID: 317304

Cholesky decomposition on tridiagonal matrices and its usage in XA=B

1. **Task Description**

Given tridiagonal matrix A and arbitrary sized matrix B, we had to find Cholesky (LL^T) decomposition. Using this decomposition we had to find matrix X such that the following equation holds: XA=B.

However, usage of n by n matrices for A and L was not efficient, so instead it was proposed to use 3 by n matrix for A and 2 by n matrix for L. The algorithm behind this will be explained in Method Description section.

2. Method Description

Cholesky decomposition for finding unique lower matrix.

$$L = \begin{bmatrix} \sqrt{A_{11}} & 0 & 0 & \dots & 0 \\ A_{21}/L_{11} & \sqrt{A_{22} - L_{21}^2} & 0 & \dots & 0 \\ A_{31}/L_{11} & (A_{32} - L_{31}L_{21})/L_{22} & \sqrt{A_{33} - L_{31}^2 - L_{32}^2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{n1}/L_{11} & (A_{n2} - L_{n1}L_{21})/L_{22} & (A_{n3} - L_{n1} - L_{n2}L_{32})/L_{33} & \dots & \sqrt{A_{nn} - L_{n1}^2 - \dots - L_{n(n-1)}^2} \end{bmatrix}$$

Since we have tridiagonal matrix A, the above matrix L can be simplified to the following:

$$L = \begin{bmatrix} \sqrt{A_{11}} & 0 & 0 & \dots & 0 \\ A_{21}/L_{11} & \sqrt{A_{22} - L_{21}^2} & 0 & \dots & 0 \\ 0 & A_{32}/L_{22} & \sqrt{A_{33} - L_{32}^2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sqrt{A_{nn} - L_{n(n-1)}^2} \end{bmatrix}$$

Now we see that Cholesky decomposition gives us simple 2 by n matrix, which can be represented as following:

$$L = \begin{bmatrix} \sqrt{A_{11}} & \sqrt{A_{22} - L_{21}^2} & \sqrt{A_{33} - L_{32}^2} & \dots & \sqrt{A_{nn} - L_{n(n-1)}^2} \\ 0 & A_{21}/L_{11} & A_{32}/L_{22} & \dots & \frac{A_{n(n-1)}}{L_{(n-1)(n-1)}} \end{bmatrix}$$

Usually Cholesky decomposition would be found using following formulas:

$$L_{j,j} = \sqrt{A_{j,j} - \sum_{k=1}^{j-1} L_{j,k}^2}$$

$$L_{i,j} = \frac{A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k}}{L_{j,j}} \quad (for \ i > j)$$

However, since we have tridiagonal matrix A and 2 by n matrix L, the above formulas would be simplified to:

$$L_{1,j} = \sqrt{A_{j,j} - L_{j,j-1}^2}$$

$$L_{2,j} = \frac{A_{j,j-1}}{L_{j-1,j-1}}$$

Moreover, the given matrix A is tridiagonal, we can represent it by the following matrix:

$$A = \begin{bmatrix} A_{12} & A_{23} & A_{34} & \dots & 0 \\ A_{11} & A_{22} & A_{33} & \dots & A_{nn} \\ 0 & A_{21} & A_{32} & \dots & A_{(n-1)n} \end{bmatrix}$$

And now the formula for computing Cholesky decomposition would be represented as:

$$L_{1,j} = \sqrt{A_{2,j} - L_{j,j-1}^2}$$

$$L_{2,j} = \frac{A_{3,j-1}}{L_{j-1,j-1}}$$

Now, we have to solve the equation XA = B.

1) Let's begin by replacing A with our lower matrix L: $X * L * L^T = B$
2) We can say that $X * L = y$, where y is an arbitrary matrix. Now we have $y * L^T = B$

3) Now we have to solve $y * L^T = B$ and then $X * L = y$
(We know that A is n by n and B is m by n, so X should be m by n)

For now let us represent L^T as n by n matrix, so that we could determine an algorithm, which after will lead for modification of this algorithm to work on 2 by n matrix.

$$L^T = \begin{bmatrix} L_{11}^T & L_{12}^T & 0 & \dots & 0 \\ 0 & L_{22}^T & L_{23}^T & \dots & 0 \\ 0 & 0 & L_{33}^T & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & L_{nn}^T \end{bmatrix} => L^T = \begin{bmatrix} L_{12}^T & L_{23}^T & L_{34}^T & \dots & 0 \\ L_{11}^T & L_{22}^T & L_{33}^T & \dots & L_{nn}^T \end{bmatrix}$$

$$B = \begin{bmatrix} B_{11} & B_{12} & B_{13} & \dots & B_{1n} \\ B_{21} & B_{22} & B_{23} & \dots & B_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & B_{m3} & \dots & B_{mn} \end{bmatrix}$$

$$\text{y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & \dots & y_{1n} \\ y_{21} & y_{22} & y_{23} & \dots & y_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & y_{m3} & \dots & y_{mn} \end{bmatrix}$$

From here we can see certain technique we can use to solve the $y * L^T = B$ equation. Let's plug in the matrices and see:

$$\begin{bmatrix} y_{11} & y_{12} & y_{13} & \dots & y_{1n} \\ y_{21} & y_{22} & y_{23} & \dots & y_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & y_{m3} & \dots & y_{mn} \end{bmatrix} * \begin{bmatrix} L_{11}^T & L_{12}^T & 0 & \dots & 0 \\ 0 & L_{22}^T & L_{23}^T & \dots & 0 \\ 0 & 0 & L_{33}^T & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & L_{nn}^T \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} & B_{13} & \dots & B_{1n} \\ B_{21} & B_{22} & B_{23} & \dots & B_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & B_{m3} & \dots & B_{mn} \end{bmatrix}$$

Now we can represent the whole equation with following matrix:

$$\begin{bmatrix} y_{11} * L_{11}^T = B_{11} & y_{11} * L_{12}^T + y_{12} * L_{22}^T = B_{12} & \dots & y_{1(n-1)} * L_{(n-1)n}^T + y_{1n} * L_{nn}^T = B_{1n} \\ y_{21} * L_{11}^T = B_{21} & y_{21} * L_{12}^T + y_{22} * L_{22}^T = B_{22} & \dots & y_{2(n-1)} * L_{(n-1)n}^T + y_{2n} * L_{nn}^T = B_{2n} \\ y_{31} * L_{11}^T = B_{31} & y_{31} * L_{12}^T + y_{32} * L_{22}^T = B_{32} & \dots & y_{3(n-1)} * L_{(n-1)n}^T + y_{3n} * L_{nn}^T = B_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} * L_{11}^T = B_{m1} & y_{m1} * L_{12}^T + y_{m2} * L_{22}^T = B_{m2} & \dots & y_{m(n-1)} * L_{(n-1)n}^T + y_{mn} * L_{nn}^T = B_{mn} \end{bmatrix}$$

Now we can construct our algorithm:

$$y_{ij} = \frac{B_{ij} - y_{i,j-1} * L_{j-1,j}^T}{L_{j,j}}$$

However, now we need to modify it, so that it works on 2 by n matrix L^T:

$$y_{ij} = \frac{B_{ij} - y_{i,j-1} * L^T_{1,j}}{L_{2,j}}$$

Let's take same steps for X and try to determine the algorithm.

$$L = \begin{bmatrix} L_{11} & 0 & 0 & \dots & 0 \\ L_{21} & L_{22} & 0 & \dots & 0 \\ 0 & L_{32} & L_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & L_{nn} \end{bmatrix} => L = \begin{bmatrix} L_{11} & L_{22} & L_{33} & \dots & L_{nn} \\ 0 & L_{21} & L_{32} & \dots & L_{n,n-1} \end{bmatrix}$$

$$y = \begin{bmatrix} y_{11} & y_{12} & y_{13} & \cdots & y_{1n} \\ y_{21} & y_{22} & y_{23} & \cdots & y_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & y_{m3} & \cdots & y_{mn} \end{bmatrix}$$

$$X = \begin{bmatrix} X_{11} & X_{12} & X_{13} & \cdots & X_{1n} \\ X_{21} & X_{22} & X_{23} & \cdots & X_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_{m1} & X_{m2} & X_{m3} & \cdots & X_{mn} \end{bmatrix}$$

Now if we plug it into the $X * L = y$ equation we will get:

$$\begin{bmatrix} X_{11} * L_{11} + X_{12} * L_{21} = y_{11} & X_{12} * L_{22} + X_{13} * L_{32} = y_{12} & \dots & X_{1n} * L_{nn} = y_{1n} \\ X_{21} * L_{11} + X_{22} * L_{21} = y_{21} & X_{22} * L_{22} + X_{23} * L_{32} = y_{22} & \dots & X_{2n} * L_{nn} = y_{2n} \\ X_{31} * L_{11} + X_{32} * L_{21} = y_{31} & X_{32} * L_{22} + X_{33} * L_{32} = y_{32} & \dots & X_{3n} * L_{nn} = y_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1} * L_{11} + X_{m2} * L_{21} = y_{m1} & X_{m2} * L_{22} + X_{m3} * L_{32} = y_{m2} & \dots & X_{mn} * L_{nn} = y_{mn} \end{bmatrix}$$

Now we are able to construct an algorithm:

$$X_{ij} = \frac{y_{ij} - X_{i,j+1} * L_{j+1,j}}{L_{j,j}}$$

Which in 2 by n matrix L will be represented as:

$$X_{ij} = \frac{y_{ij} - X_{i,j+1} * L_{2,j}}{L_{1,j}}$$

This was all the calculations that were made before starting the actual program. As we can notice, having tridiagonal positive definite matrix simplified our calculations, and made our program be memory and time efficient.

3. Program Description

The whole project was done in one matlab file, which contains two functions, one being for Cholesky decomposition and second one being for solving the given equation. The file starts with initialization of the matrix A taken from one of the numerical methods homework exercises and arbitrary taken matrix B. Those two matrices were taken for error checking at the code writing stage.
After, randomly created matrices were taken for more complex examples, where user can specify the number of test cases, size of the matrices and interval for matrix entries. Moreover, at this part the try catch statement was used for determining the positive definiteness of the randomly created matrix A.
All of the above was performed on 3 by n matrix A, which after would be represented in n by n matrix to see difference between my function and build in functions.

Cholesky(A, n)

Our main calls Cholesky function which receives 3 by n matrix A and its biggest dimension being n. This function returns a tuple of lower matrices L and L^T which are results of Cholesky decomposition. The function works according to the following formula that was described above:

$$L_{1,j} = \sqrt{A_{2,j} - L_{j,j-1}^2} \qquad L_{2,j} = \frac{A_{3,j-1}}{L_{j-1,j-1}}$$

Then this function reverts our L matrix to obtain L^T, which is swapping two rows. Function contains two loops, which are not nested, so this function performs operations in O(n) time complexity.

Solve(MN, MT, B, m, n)

After our previous function has returned the Cholesky decomposition, our main calls Solve function which takes 5 inputs: MN, MT, B, m, n. Here, MN and MT represent L and L^T matrices respectively, matrix B given from equation and m and n, which are dimensions of matrix B. This function returns the matrix X which should have been calculated. Function works according to the following formulas, that were described above:

$$y_{ij} = \frac{B_{ij} - y_{i,j-1} * L_{1,j}^T}{L_{2,j}} \qquad X_{ij} = \frac{y_{ij} - X_{i,j+1} * L_{2,j}}{L_{1,j}}$$

This function contains two nested loops, which run in dimensions of matrix B. Hence, the time complexity of this function is O(n*m).
First nested loop iterates through top to bottom of matrix B, it is needed because in our algorithm matrix entries of y are dependent on previous entries of y.
Second nested loop iterates through bottom to top of matrix y, it is needed because in our algorithm matrix entries of X are dependent on next entries of X.

4. Numerical Tests

Functions were tested with build in matlab functions as chol, for finding Cholesky decomposition and division for finding matrix X. They were also tested with different sized matrices and different entries of matrices. In all cases functions show high performance and were as fast as build in functions.
Errors were checked by subtracting the build in result with result of my functions.

$$e = |BuildInResultX - MyResultX|$$

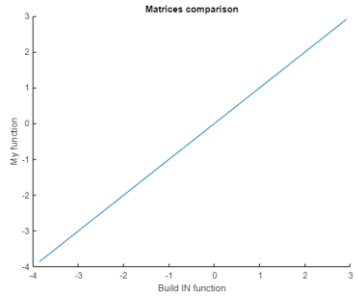
Following is table of results for different types of X:

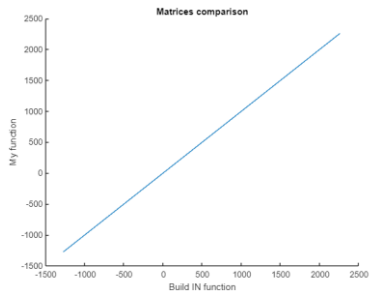
i	Size of Matrix A (n)	Size of Matrix B (m)	Error
1	10	1	1.0e-14
2	100	1	1.0e-14
3	1000	1	1.0e-12
4	10	10	1.0e-13
5	100	10	1.0e-12
6	1000	10	1.0e-11
7	10	50	1.0e-12
8	100	50	1.0e-10
9	1000	50	1.0e-10

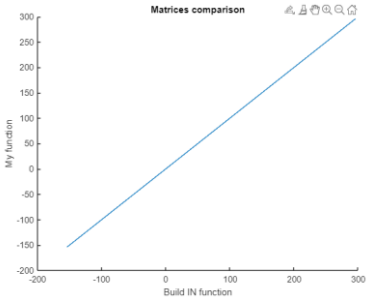
Let's plot both result in one graph, y axis being result of our function and x axis being result of build in function. We would like to get a straight line at the end:

Graphs:

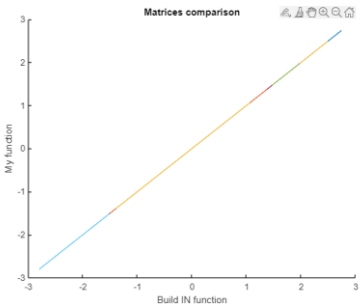
1) For b = 10 and for n = 10, 100, 1000 respectively

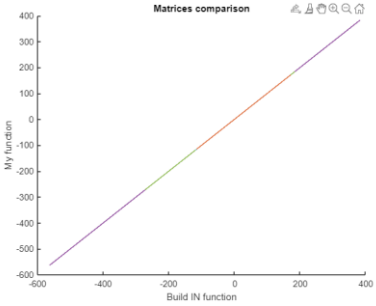


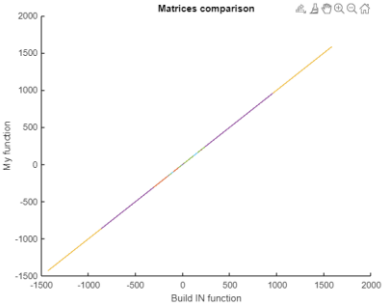




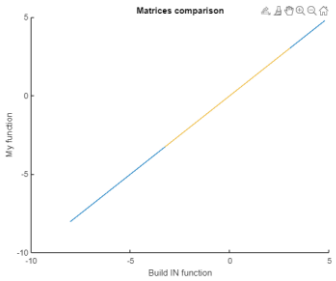
2) For b = 10 and for n = 10, 100, 1000 respectively

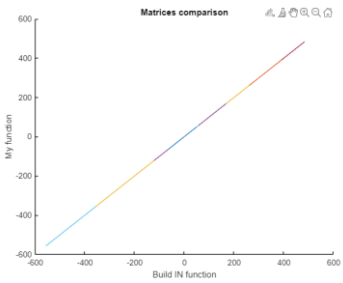


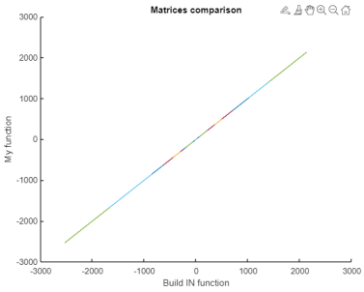




3) For b = 50 and for n = 10, 100, 1000 respectively







As we can see the difference between those 2 functions are very small. It can be seen from the table and also from the the straight line graphs.

5. Conclusion

To conclude, Cholesky decomposition is very useful algorithm for finding matrix X in an equation. It simplifies the calculations and makes the algorithm work faster if the provided matrix is well defined. Moreover, as we could see the algorithm can get even faster and efficient if it comes to tridiagonal matrices. Results of using Cholesky decomposition is as accurate as with any other methods, therefore, in certain cases as in ours, it is best to use this method.