Numerical Methods 1

Report for Project 1 – Solving a System of Linear Equations AX=B using Cholesky's LDL^T factorization (given that A is Symmetric and Positive-Definite).

By Noman Noor (id number: 302343)

Date: 02/06/2022 (2nd of June 2022)

How to use and test the function:

The function call "**SolveLinEqUsingCholeskyLDLT(A,b)**" returns the required vector x.

To run the test referenced within this report, use the provided script via typing into the console: "**Test_SolveLinEqUsingCholeskyLDLT**".

My approach to implementing the required function:

First, we calculate the L and D Matrices such that A=LDL^T (L^T is simply calculated by L' in MATLAB) using the following formulae:

$$D_{ii} = A_{ii} - \sum_{k=1}^{i-1} L_{ik}{}^2 D_{kk} \quad \text{and} \quad L_{ij} = \frac{1}{D_{jj}} \cdot \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk} D_{kk}\right).$$

[For more details, refer to Cholesky decomposition - Wikipedia and/or Total Internal Reflection : Musings on Machine Learning, Technology, and Art (avishek.net)].

Since we know A=LDL^T, we solve LDL^{T}X=B instead of AX=B. To do this, we solve LZ=B first. Then, we solve DY=Z. Finally, we solve L^{T}X=Y to finally find the required X (containing values of all the x$_i$ of the solution). Refer to the comments in the source code for more detailed information.

Some useful facts used for getting non-Computer derived known solutions:

I use some obvious facts in order to avoid having to derive the solutions by hand for every Matrix tested:

1. If Ax=b and b where b=[0,0,….0] then x=[0,0,….0].
2. If Ax=b and b is a vector whose i-th element is the sum of the i-th row of A ("b=sum(A,2)" in MATLAB), then x=[1,1,…1].
   a. Similarly, if each i-th element of b is n times the sum of i-th row of A, then x=[n,n,….n].

If you think about these, they are very trivial to comprehend and prove. As such, they won't be explicitly proved in this report.

Test 1: Using my function to solve the Tutorial List C problem 3 – (A is a 3x3 S.P.D. Matrix):

We know the "X" vector beforehand from Tutorials and I also verified it again using Wolfram Alpha (and MATLAB linsolve function). Also, we know this solution is unique for the given A and B because A is invertible and as such it should have a unique solution (X) for any given B. Though I say that, I do also solve it for some other obvious values of the vector b.

A	**B**	**X we know beforehand**	**X calculated by my function**	**X calculated by linsolve**
$\begin{bmatrix}1 & 1 & 0\\ 1 & 5 & 2\\ 0 & 2 & 10\end{bmatrix}$	[1, -7, -4] (from tutorials)	[3, -2, 0] (from tutorials)	3, -2, 0	3, -2, 0
	[2, 8, 12] (sum of elements of each row of A)	[1, 1, 1]	[1, 1, 1]	[1, 1, 1]
	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]

Test 2: Testing on a S.P.D. Matrix “A” that I derived by hand:

The matrix A used here was derived by hand for a Numerical Methods 2 project of mine (and designed to have the Eigenvalues [2,6] though that part is irrelevant to this task).

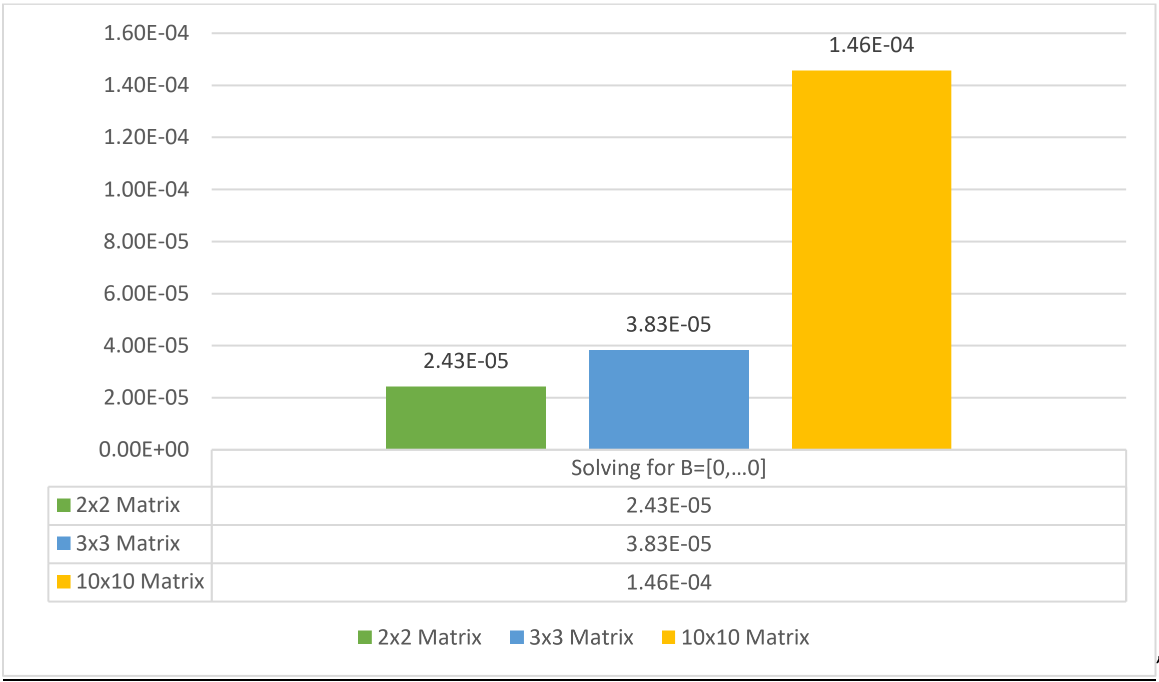
A	B	X we know beforehand	X calculated by my function	X calculated by linsolve
$\begin{bmatrix} 5.2 & -1.6 \\ -1.6 & 2.8 \end{bmatrix}$	[3.6, 1.2] (sum of elements of each row of A)	[1, 1]	[1, 1]	[1, 1]
	[0, 0]	[0, 0]	[0, 0]	[0, 0]

Test 3: Testing on a 10x10 S.P.D. Matrix “A” Generated in MATLAB:

The matrix Symmetric Positive Definite Matrix A was generated/derived by the MATLAB function sprandsym(n,density,rc) which returns a matrix with a reciprocal condition number equal to rc. The distribution of entries is nonuniform; it is roughly symmetric about 0; all are in [−1,1]. If rc is a vector of length n, then R has eigenvalues rc. Thus, if rc is a positive (nonnegative) vector then R is a positive (nonnegative) definite matrix. Due to the limited accuracy of computers (floating-point precision, etc.), this might not be perfect but it’s accurate enough for our tests. This test is mostly about the performance difference when the datasets get bigger. Upon running, we notice that it’s fully correct.

A *(Approximate values, for full precision, refer to the test function’s console output)*	B	X we know beforehand	X calculated by my function	X calculated by linsolve
$\begin{vmatrix} 2.5926 & 2.3611 & 2.2532 & 2.2277 & 2.9389 & 1.7335 & 2.5129 & 2.1777 & 2.2820 & 2.1020 \\ 2.3611 & 3.5079 & 1.9345 & 2.9141 & 3.6353 & 2.5202 & 2.9145 & 3.0109 & 2.8824 & 2.4276 \\ 2.2532 & 1.9345 & 3.0196 & 2.2851 & 2.6306 & 1.7190 & 2.2361 & 2.1751 & 2.4021 & 1.9228 \\ 2.2277 & 2.9141 & 2.2851 & 3.4238 & 3.2294 & 2.3708 & 2.8969 & 2.9972 & 2.7653 & 2.2158 \\ 2.9389 & 3.6353 & 2.6306 & 3.2294 & 4.2120 & 2.6583 & 3.3765 & 3.2278 & 3.1987 & 2.9368 \\ 1.7335 & 2.5202 & 1.7190 & 2.3708 & 2.6583 & 2.3980 & 1.8069 & 2.0564 & 2.2348 & 1.8306 \\ 2.5129 & 2.9145 & 2.2361 & 2.8969 & 3.3765 & 1.8069 & 3.9979 & 3.3248 & 2.8972 & 2.2030 \\ 2.1777 & 3.0109 & 2.1751 & 2.9972 & 3.2278 & 2.0564 & 3.3248 & 3.8290 & 3.2284 & 2.1168 \\ 2.2820 & 2.8824 & 2.4021 & 2.7653 & 3.1987 & 2.2348 & 2.8972 & 3.2284 & 3.0995 & 2.2310 \\ 2.1020 & 2.4276 & 1.9228 & 2.2158 & 2.9368 & 1.8306 & 2.2030 & 2.1168 & 2.2310 & 2.1629 \end{vmatrix}$	[0,0, …, 0]	[0,0, …, 0]	[0,0, …, 0]	[0,0, …, 0]
	4.5*sum(A,2) (vector containing 4.5 times the sum of each row of A)	[4.5, 4.5, …, 4.5]	[4.5, 4.5, …, 4.5]	[4.5, 4.5, …, 4.5]

TIMING:



	Solving for B=[0,…0]
🟩 2x2 Matrix	2.43E-05
🟦 3x3 Matrix	3.83E-05
🟨 10x10 Matrix	1.46E-04

These are statistics we get from using **timeit** function in MATLAB. While performance will differ from device to device and other circumstances, there are some things we can generally take away from the data.
It may seem like there is a relatively decent increase in times, it should be noted that it actually isn't when looking at the larger perspective. Many algorithms would have been worse. We can see that the times still remain extremely smaller than a single second.

CONCLUSION:

From the testing and reading through the function source code (combined with the description of my approach as described in the beginning parts of this report and the comments on the code), we can see that the approach works as intended and is implemented as intended.

Furthermore, upon running the tests or using the function, it can be noticed that the runtime of the function is very small as the result is returned practically instantly even for a 10x10 matrix (or bigger).