

# **ECE 7650 ASSIGNMENT REPORT**

## **(Advance Matrix Algorithm)**

**Author:** Jamiu Babatunde Mojolagbe  
**Department:** Electrical and Computer Engineering  
**Student ID:** #7804719  
**Email:** [mojolagm@myumanitoba.ca](mailto:mojolagm@myumanitoba.ca)  
**Course:** ECE 7650  
**Homework:** 2  
**Sub. Date:** November 14, 2016

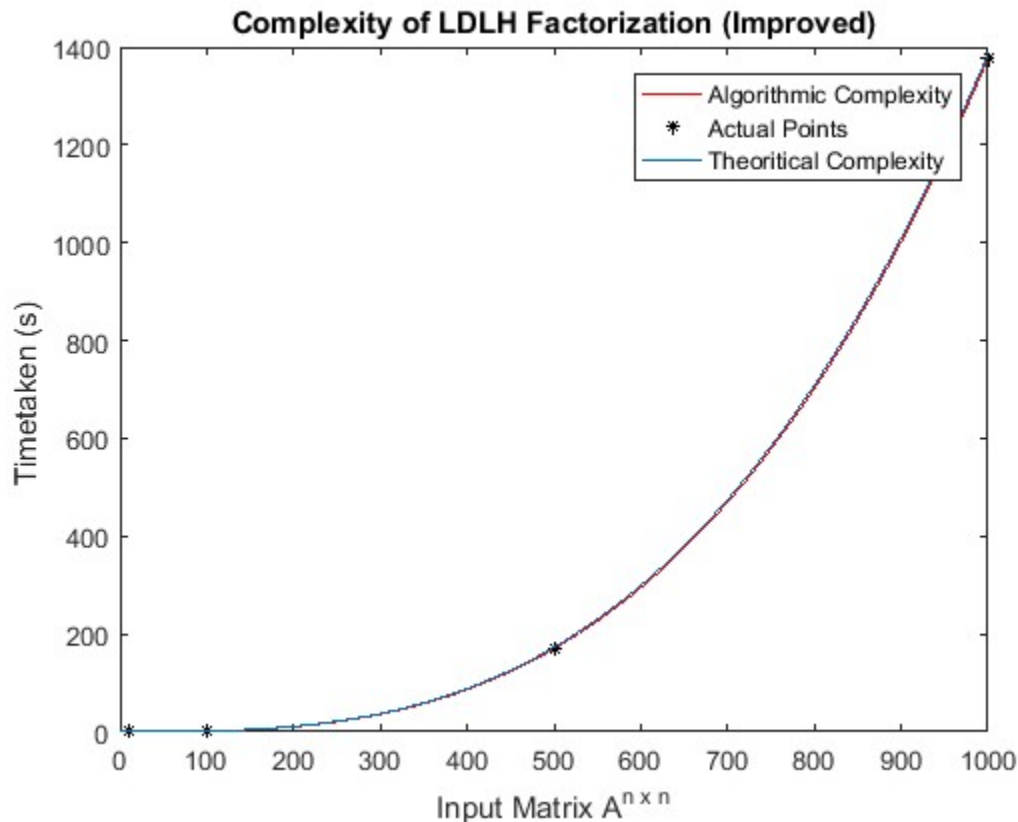
## QUESTION ONE (1)

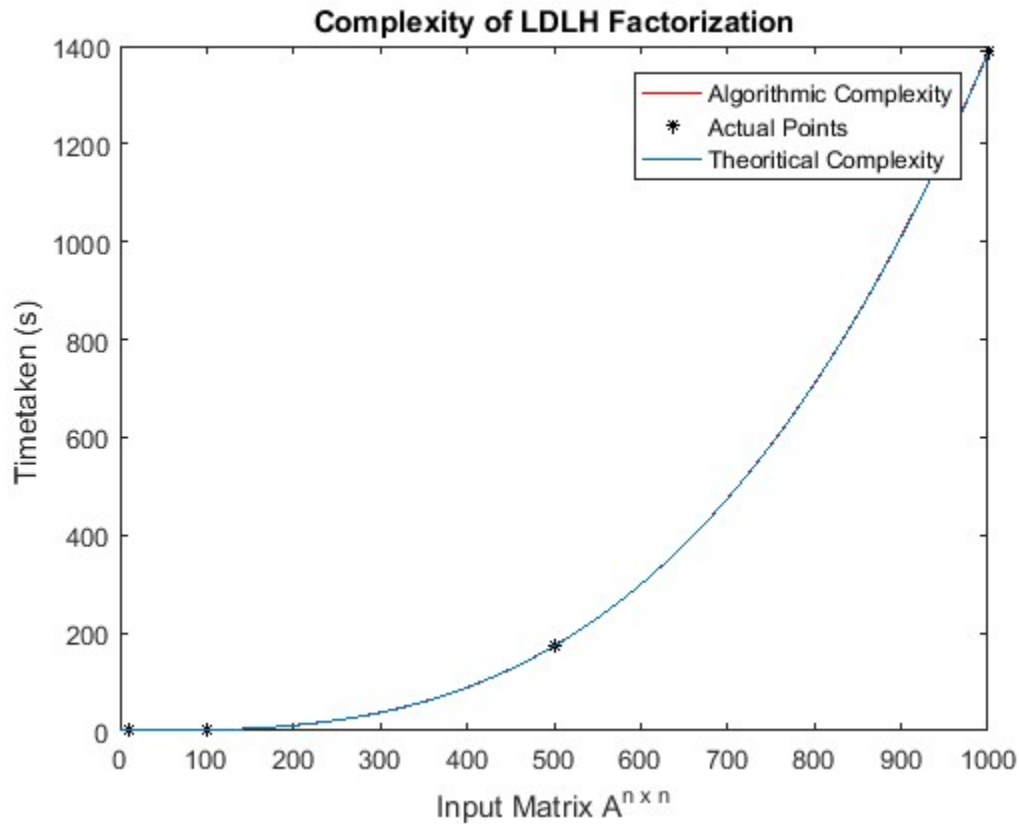
LDLH factorization was implemented in the MATLAB files named `ldlh.m` and `ldlhImproved.m`. The main difference between the two implementations is that, diagonal entries,  $D$ , is used as a diagonal matrix in `ldlh.m` but it was used just as a column vector and return as the same in `ldlhImproved.m`.

The driver programs for the two variants of LDLH is named with the function names as shown above with suffix 'Driver'. So, for `ldlh.m`, the driver/test program is `ldlhDriver.m`; and for `ldlhImproved.m`, the driver/test program is `ldlhImprovedDriver.m`. A function named `rherm.m` was used for generating random Hermitian indefinite matrices and it is found at <https://www.mathworks.com/matlabcentral/fileexchange/25912-random-hermitian-matrix-generator/content/rherm.m><sup>4</sup>. The following results were obtained:

Matrix Dimension (nxn)	Time (seconds) – <code>ldlh.m</code>	Time (seconds) – <code>ldlhImproved.m</code>
10x10	0.0035685425106160	0.00579037299873811
100x100	1.56746074271622	1.39526137488197
500x500	173.413785821126	169.959223738156
1000x1000	1388.00895329618	1376.18212587115

The graphs below can be obtained from the driver programs; however, the above data plotting is also contained in files named `reportDataPlottingQ1.m` and `reportDataPlottingQ11.m` for `ldlhImprovedDriver.m` and `ldlhDriver.m` respectively.





### Observation:

It can be observed from the table above that the time taken to complete the factorization is roughly equal to the  $n^3$  of the number of input matrix dimension as the obtained points and algorithmic complexity curve of polynomial of that order roughly fit together. Though the result for the two variants are approximately the same, the improved version tends to be a little bit better.

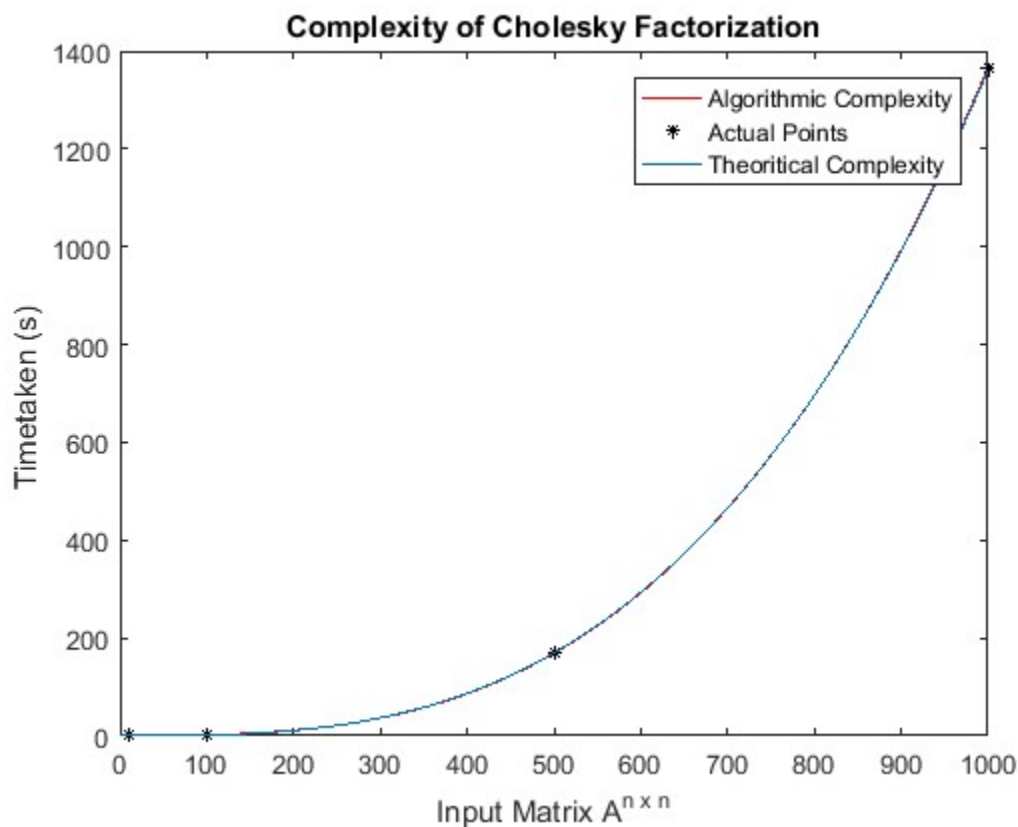
Two proves were conducted in the program (ldlhImprovedDriver.m),  $\text{norm}(A - \text{LDLH})$  and determinant of A is compared with the product of the diagonal entries of D. The norms yield approximately zero values in all presented cases and the determinants of A were all the same as the product of diagonal entries D; these proofs show that the factorization works.

## QUESTION TWO (2)

Cholesky factorization of a matrix was implemented in the file named `cholesky.m`. The driver program is named `choleskyDriver.m`. A function named `spd.m` was created and used for generating random Hermitian positive definite matrices. The following results were obtained:

Matrix Dimension (nxn)	Time Taken (seconds)
10x10	0.0146152114229664
100x100	1.84716424266648
500x500	168.710732680456
1000x1000	1365.32622999174

The graph below can be obtained from the driver program; however, the above data plotting is also contained in a file named `reportDataPlottingQ2.m`



### Observation:

It can be observed from the table above that the time taken to complete the factorization is roughly equal to the  $n^3$  of the number of input matrix dimension as the obtained points and algorithmic complexity curve of polynomial of that order roughly fit together. However, it can also be observed from the table above when compared to the one under LDLH factorization that, Cholesky factorization is slightly faster than LDLH factorization even for the two variants presented above.

One prove was conducted which is  $\text{norm}(A - LLH)$ , which in each case yields approximately zero value; this in fact, confirm that the factorization worked.

### QUESTION THREE (3)

#### MATLAB BUILT IN FUNCTIONS FOR LDLH AND CHOLESKY

##### LDLH DECOMPOSITION

***What is the name of the function?***

It is implemented in MATLAB with name **ldl**

***What parameters does it take?***

It takes maximum of four (4) parameters. They are as listed below in order of their priority/occurrence:

Order	Type	Value	Description
1	Matrix	[Hermitian matrix]	Hermitian/symmetric matrix
2	Double	0 to 0.05	It uses this parameter as the pivot tolerance. It must be a double scalar lying in the interval [0, 0.5]. The default value is 0.01
2 or 3	String	'upper'	When this parameter is supplied, it references only the diagonal and upper triangle of input matrix and assumes that the lower triangle is the complex conjugate transpose of the upper triangle.
2 or 3 or 4	String	'vector'	It uses this parameter to determine the return type of permutation matrix P. A vector is returned if this argument is supplied or matrix if it is absent.

***What is/are the return values?***

The return values and their order of occurrence actually depend on the supplied argument(s)/parameter(s) to the function. Below is the possible combination of parameters as shown in the table above and their corresponding order of return value(s).

$L = \text{ldl}(A)$

$[L,D] = \text{ldl}(A)$

$[L,D,P] = \text{ldl}(A)$

$[L,D,p] = \text{ldl}(A, 'vector')$

$[U,D,P] = \text{ldl}(A, 'upper')$

```
[U,D,p] = ldl(A,'upper','vector')
[L,D,P,S] = ldl(A)
[L,D,P,S] = LDL(A,THRESH)
[U,D,p,S] = LDL(A,THRESH,'upper','vector')
```

### Description of return values

L – Lower triangular matrix

D – Diagonal matrix

P – Permutation matrix

p – Permutation matrix in form of a vector (vectored permutation matrix)

S – Scaling matrix

U – Upper triangular matrix

## CHOLESKY DECOMPOSITION

### *What is the name of the function?*

It is implemented in MATLAB with name **chol**. The **chol** function assumes that input matrix is (complex Hermitian) symmetric. Input matrix must be positive definite.

### *What parameters does it take?*

It takes maximum of three (3) parameters. They are as listed below in order of their priority or occurrence:

Order	Type	Value	Description
1	Matrix	[Complex H. matrix]	Complex Hermitian/symmetric positive definite matrix. If this parameter is not complex Hermitian, <b>chol</b> uses the (complex conjugate) transpose of the upper triangle as the lower triangle
2	String	'lower'	It uses this parameter to produce a lower triangular matrix L from the diagonal and lower triangle of an input matrix A, satisfying the equation $L*L'=A$ .
2	String	'upper'	When this parameter is supplied, it produces an upper triangular matrix R from the diagonal and upper triangle of matrix A, satisfying the equation $R'*R=A$ .

<b>2 or 3</b>	String	'vector'	It uses this parameter to determine the return type of permutation matrix P. A vector is returned if this argument is supplied or matrix if it is absent.
---------------	--------	----------	---

### ***What is/are the return values?***

The return values and their order of occurrence actually depend on the supplied argument(s)/parameter(s) to the function. Below is the possible combination of parameters as shown in the table above and their corresponding order of return value(s).

$R = \text{chol}(A)$   
 $L = \text{chol}(A, \text{'lower'})$   
 $R = \text{chol}(A, \text{'upper'})$   
 $[R, p] = \text{chol}(A)$   
 $[L, p] = \text{chol}(A, \text{'lower'})$   
 $[R, p] = \text{chol}(A, \text{'upper'})$   
 $[R, p, S] = \text{chol}(A)$   
 $[R, p, s] = \text{chol}(A, \text{'vector'})$   
 $[L, p, s] = \text{chol}(A, \text{'lower'}, \text{'vector'})$   
 $[R, p, s] = \text{chol}(A, \text{'upper'}, \text{'vector'})$

### **Description of return values**

L – Lower triangular matrix

R – Upper triangular matrix

S – Permutation matrix

s – Permutation matrix in form of a vector (vectored permutation matrix)

p – Positive integer

**Note:** The explanation and details presented above for the two functions are based on MATLAB documentation found in <sup>1,2</sup>.

## QUESTION FOUR (4)

### PIVOTING STRATEGIES USED DURING LU DECOMPOSITION

There are generally varieties of pivoting strategies used during the LU decomposition of a dense matrix. These includes partial pivoting, complete/full pivoting and rook pivoting among others.

#### ROLE OF PIVOTING

The main role of pivoting in LU decomposition is to prevent instability that is inherent in Gaussian Elimination method - a precursor of LU decomposition. Mainly, this is done with help of permutation and identity matrices since pivoting involves swapping of rows and/or columns of a given matrix.

#### PARTIAL PIVOTING

##### **What is the goal?**

Here as expected, the goal is to use the permutation matrix to place the largest entry of the first column of the matrix at the top of that first column which has smaller value compared to others.

##### **How is it done and what is the complexity?**

Given a matrix  $\mathbf{A}^{n \times n}$ , the following steps are required for partial pivoting:

**Step 1:** Scan through the rows of  $\mathbf{A}$  for the largest entry at column  $\mathbf{k}$ ; this being correspond to elimination  $\mathbf{k}$ .

**Step 2:** Once this entry is found, it is then moved into the pivot position  $\mathbf{A}_{kk}$  on the diagonal of the matrix  $\mathbf{A}$ . This is achieved by row interchange via the help of permutation matrix,  $\mathbf{P}$ ; such that matrix  $\mathbf{A}$  is multiplied by  $\mathbf{P}$ .

**Step 3:** The LU decomposition is then continued and the new pivot is used to clear out the entries below it in its column in order to obtain the upper triangular matrix  $\mathbf{U}$ .

**Complexity:** The process requires complexity of  $\mathbf{O(n^2)}$  comparisons for a  $\mathbf{nxn}$  matrix.

Following equations depicts the process:

$$M'_k = (P_{n-1} \cdots P_{k+1})M_k(P_{k+1} \cdots P_{n-1})$$

$$(M_{n-1}M_{n-2} \cdots M_2M_1)^{-1} = L$$

$$M_{n-1}P_{n-1}M_{n-2}P_{n-2} \cdots M_2P_2M_1P_1A = U''^3$$

##### **Practice Example:**

$$\mathbf{A} = \begin{pmatrix} 5 & 6 & 4 \\ 2 & 3 & 5 \\ 9 & 2 & 7 \end{pmatrix}$$

$$\text{Pivot } \mathbf{A} \text{ using permutation matrix, } \mathbf{P}_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\text{Finally, } \mathbf{P}_1\mathbf{A} = \begin{pmatrix} 9 & 2 & 7 \\ 2 & 3 & 5 \\ 5 & 6 & 4 \end{pmatrix}$$



## **References**

1. <https://www.mathworks.com/help/matlab/ref/ldl.html?searchHighlight=ldl>
2. <https://www.mathworks.com/help/matlab/ref/chol.html>
3. <http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-reid-LU-pivoting.pdf>
4. <https://www.mathworks.com/matlabcentral/fileexchange/25912-random-hermitian-matrix-generator/content/rherm.m>