MATH466/462 Project 2. *Due in class on Monday, Feb 10, 2020*

Instruction: your project report should include necessary mathematical justification, description, and details of your algorithms/conclusions. Your MATLAB codes and generated outputs may be attached in the end of the report. Make sure you addressed all the questions in each problem. Both the report and codes will be graded. Please submit a printed hard-copy.

Problem A (10 pts): Thomas algorithm for tridiagonal matrix systems

The Gaussian elimination or LU factorization of dense $n \times n$ matrices in general costs $O(n^3)$ operations, but we can do much faster (in $O(n)$ operations) if the matrix A is a sparse tridiagonal matrix (arising in ODEs).

1. Implement the Thomas algorithm for tridiagonal matrix in MATLAB as the following function :

```
function x=Thomas(d1,d2,d3,b)
%Input: d1,d2,d3 are the vectors of the 3 diagonals (lower,main,upper) of A
%       b is the right hand side vector
%Output: x is solution of Ax=b

```

2. Test your code with the following script. What is the approximate slope of each line (last straight segment)?

```
dim=2.^(5:13); %test dimensions, reduce the dimensions if too slow
cpu=zeros(length(dim),3); %for store CPU times of 3 methods
for k=1:length(dim)
  n=dim(k); b=rand(n,1);
  d1=-ones(n,1);d2=2*ones(n,1);d3=-ones(n,1); % A=tridiag(-1,2,-1)
  tic;x1=Thomas(d1,d2,d3,b);cpu(k,1)=toc; %call your own Thomas algorithm

  A=spdiags([d1,d2,d3],-1:1,n,n);%construct the same sparse matrix A
  tic; x2=A\b;cpu(k,2)=toc; %solve by MATLAB's Sparse direct solver
  err=norm(x1-x2,inf)% check max err close to 0

  tic; [L,U]=lu(full(A)); x3=U\(L\b); cpu(k,3)=toc; %solve by LU factorization
end
loglog(dim,cpu(:,1),'-o',dim,cpu(:,2),'-x',dim,cpu(:,3),'-*') %plot CPU time growth
legend('Thomas solver','Sparse solver','LU solver','Location','best')

```

3. (Bonus) Speed up your Thomas codes using Vectorization, can you do faster than MATLAB's direct solver? https://www.mathworks.com/help/matlab/matlab_prog/vectorization.html

Problem B (10 pts): Cholesky factorization

For any symmetric positive-definite matrix A, we have the Cholesky factorization: $A = R^T R$, where R is an upper triangular matrix. Due to symmetry, such a special LU factorization is faster than the general LU factorization.

1. Ready this: `https://www.maths.manchester.ac.uk/%7Ehigham/papers/high09c.pdf`

2. Implement the Cholesky factorization algorithm in MATLAB as:

```
function R=Cholesky(A)
%Input: a symmetric positive-definite matrix A
%Output: a upper triangular matrix R such that A=R'*R

```

3. Test your code with the following script. What you observe and why?

```
dim=2.^(5:12); %test dimensions, reduce the dimensions if too slow
cpu=zeros(length(dim),3); %for store CPU times of 3 methods
for k=1:length(dim)
n=dim(k);b=rand(n,1);
B=randn(n,n); A=B'*B; %construct random dense SPD matrix A
tic;R=Cholesky(A);x1=R\(R'\b); cpu(k,1)=toc; %call your own Cholesky algorithm

tic; x2=A\b;cpu(k,2)=toc; %solve by MATLAB's Sparse direct solver
err=norm(x1-x2,inf) % check max err close to 0

tic; [L,U]=lu(A); x3=U\(L\b); cpu(k,3)=toc; %solve by LU based Gaussian elimination
end
loglog(dim,cpu(:,1),'-o',dim,cpu(:,2),'-x',dim,cpu(:,3),'-*') %plot CPU time growth
legend('Cholesky solver','Backslash solver','LU solver','Location','best')

```

4. (Bonus) If A is symmetric indefinite, a modified Cholesky factorization is the LDL factorization:

$$A = LDL^T,$$

where L is a lower unit triangular matrix (i.e., has unit diagonal), and D is a diagonal matrix. Based on your above Cholesky factorization of $A = R^T R$, write MATLAB codes to find the LDL factorization: $A = LDL^T$.

```
function [L,D]=LDLt(A)
%Input: a symmetric matrix A
%Output: L is a lower unit triangular matrix,
%        D is a digonal matrix such that A=L*D*L'

```

You may take a look at: `https://en.wikipedia.org/wiki/Cholesky_decomposition`