Task 9
Compute the LL^T factorization of A, where $A \in \mathbb{R}^{nxn}$ is a tridiagonal and symmetric positive definite. Use this decomposition to solve a system of equations $A^T Ax = b$

Maciej Sobczynski

April 26, 2018

1 Method description

In this task, we consider only matrices that are tridiagonal and symmetric positive definite. This means that the matrices have non-zero elements only on the main diagonal and the first diagonals below and above the main one. Because the matrices considered are symmetric values on both diagonals next to the main diagonal have to be equal. Therefore the matrices considered in this task take the following form

$$A = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ 0 & \cdots & 0 & \beta_{n-1} & \alpha_n \end{bmatrix}$$

The task is to perform LL^T (Cholesky) factorization on the matrix A and use the factorization to solve a system of equations $A^T Ax = b$ where

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

The factorization is obtained in the following way

Let D be nxn diagonal matrix with diagonal elements δ_j for $j = 1, ..., n$ and

$$L = \begin{bmatrix} 1 & & & & \\ l_1 & 1 & & & \\ & \ddots & \ddots & & \\ & & l_{n-2} & 1 & \\ & & & l_{n-1} & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} \delta_1 & & & & \\ & \delta_2 & & & \\ & & \ddots & & \\ & & & \delta_{n-1} & \\ & & & & \delta_n \end{bmatrix}$$

$$\delta_1 = \alpha_1,\ l_1 = \frac{\beta_1}{\delta_1}$$

$$\delta = \alpha_j - \frac{\beta_{j-1}^2}{\delta_{j-1}},\ j = 2, ..., n\ ,\ l_j = \frac{\beta_j}{\delta_j},\ j = 2, ..., n-1$$

The factorization cannot be completed if any diagonal element δ is zero. This situation does not occur when A is positive definite.

The factorization can be obtained from D in the following fashion

$$A = L^C (L^C)^T$$

Where $L^C = L + D^{\frac{1}{2}}$

$$L^C = \begin{bmatrix} \sqrt{\delta_1} & & & & \\ \frac{\beta_1}{\sqrt{\delta_1}} & \sqrt{\delta_2} & & & \\ & \ddots & \ddots & & \\ & & \frac{\beta_{n-2}}{\sqrt{\delta_{n-2}}} & \sqrt{\delta_{n-1}} & \\ & & & \frac{\beta_{n-1}}{\sqrt{\delta_{n-1}}} & \sqrt{\delta_n} \end{bmatrix}$$

Once the decomposition is obtained, the system of equations can be solved in a similar fashion to regular $Ax = b$ system by simply using three forward /

back substitutions instead of the usual two, noting that L^2 is also triangular:

$$A^T Ax = L^T L^2 L^T x = b$$

Forward substitute to get y:

$$L^T y = b,\ y = L^2 L^T x$$

Backward substitute to get z:

$$L^2 z = y,\ z = L^T x$$

Forward substitute to get x:

$$L^T x = z$$

2 Program description

The program consists of following files,

1. *LLT.m* - The main function of the program. Can be run with the *LLT(A,b,ex)* command. The function performs LL^T factorization with the help of *cholesky.m* and uses the result of the factorization to solve the system of equations. The function returns the vector of solutions to the system. If an exact solution is not provided, the result of *linsolve(A*A,b)* will be used instead of it

2. *cholesky.m* - A function that performs LL^T decomposition on a tridiagonal and symmetric positive definite matrix. The code is optimised to only calculate factorization for matrices of this type and is not suitable for other matrices.

3. *examples.m* - A simple script to perform the calculations for provided examples. It calls the LLT function on predefined matrices and vectors

4. *triPosDef.m* - A simple function to generate a random tridiagonal symmetric positive definite n by n matrix;

5. *tridiag.m* - A modified version of the code provided during the laboratories. Creates a tridiagonal matrix where diagonals above and below the main one are identical.

The *LLT.m* function will stop calculations and display an error message if the provided matrix or vector does not fit the conditions. That is, when the vector is not a column vector or it's height is lower than the height of the matrix, when the matrix is not square, tridiagonal, symmetric or positive definite.

3 Numerical examples

3.1 Example 1

$$A = \begin{pmatrix} 3 & 1 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 \\ 0 & 1 & 3 & 1 & 0 \\ 0 & 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & 1 & 3 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Condition number of A: 3.7321
Condition number of A^2: 13.9282
Decomposition error: 9.3847×10^{-17}
Relative error: 0.1816
Forward stability error: 0.0130
Backward stability error: 0.0405

3.2 Example 2

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{pmatrix} \quad b = \begin{pmatrix} 3 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 3 \end{pmatrix}$$

Condition number of A: 48.3742
Condition number of A^2: 2.3401×10^3
Decomposition error: 1.1332×10^{-16}

Relative error: 49.0085
Forward stability error: 0.0209
Backward stability error: 0.0053

3.3 Example 3

$$A = \begin{pmatrix} 5 & 1 & 0 & 0 & 0 & 0 \\ 1 & 5 & 2 & 0 & 0 & 0 \\ 0 & 2 & 5 & 3 & 0 & 0 \\ 0 & 0 & 3 & 5 & 2 & 0 \\ 0 & 0 & 0 & 2 & 5 & 1 \\ 0 & 0 & 0 & 0 & 1 & 5 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 5 \\ 2 \\ 0 \end{pmatrix}$$

Condition number of A: 9.5416
Condition number of A^2: 91.0415
Decomposition error: 9.8126×10^{-17}
Relative error: 0.2944
Forward stability error: 0.0032
Backward stability error: 0.0157

3.4 Example 4

A is a random $10x10$ tridiagonal and symmetric positive definite matrix and

$$b = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ 10 \end{pmatrix}$$

Condition number of A: 16.5097
Condition number of A^2: 272.5689
Decomposition error: 3.5351×10^{-17}
Relative error: 0.2770
Forward stability error: 0.0025
Backward stability error: 0.0172

3.5 Example 5

A is a random $100x100$ tridiagonal and symmetric positive definite matrix and

$$b = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ 100 \end{pmatrix}$$

Condition number of A: 39.5692
Condition number of A^2: 1.5657×10^3
Decomposition error: 1.1071×10^{-16}
Relative error: 0.4910
Forward stability error: 2.7992×10^{-04}
Backward stability error: 0.0055

3.6 Example 6

A is a random $1000x1000$ tridiagonal and symmetric positive definite matrix and

$$b = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ 1000 \end{pmatrix}$$

Condition number of A: 69.4416
Condition number ofA^2: 4.8221×10^3
Decomposition error: 9.4986×10^{-17}
Relative error: 0.4134
Forward stability error: $3.6735 \times 10^{(}$-05)
Backward stability error: 0.0018

4 Analysis of results

The program overall returns pretty accurate results (in example 2 the results are very much inaccurate). The decomposition error for the Cholesky decomposition implemented in this program is usually lower than in the built in function *chol*. Other than that all tested errors are reasonable.

5 Source code

5.1 LTT.m

```
function [x] = LLT(A,b,ex)
% The function solves a system of equations LL^Tx=b
% Input variables:
% A The tridiagonal and symmetric positive definite matrix.
% b A column vector of constants.
% ex An exact solution.
% Output variable:
% A colum vector with the solution.
% If an exact solution is not provided, the result of linsolve(A*A,b)
% will be used instead of it

if (~exist('ex', 'var'))
        ex = linsolve(A*A,b);
end
B=A*A*ex;

%Check conditions
%Dimensions
[m,n]=size(A);
[o,p]=size(b);
if(m~=n || p ~=1 || o ~=n)
    disp("Dimensions incorrect.");
    return;
end

%Tridiagonal
test=tril(A,-2) + triu(A,2);
if(~isequal(test, zeros(n,n)))
    disp("A is not tridiagonal.")
    return;
end

%Symmetric
if(~issymmetric(A))
```

```
    disp("A is not symmetric.")
    return;
end

%Positive definite
[~,p] = chol(A);
if(p>0)
    disp("A is not positive definite.")
    return;
end

L = cholesky(A);

%Solve equations
y = L'\b;
z = L*L\y;
x = L'\z;

disp("Condition number of A: ");
disp(cond(A));


disp("Condition number of A^2: ");
disp(cond(A)^2);

disp('Decomposition error:');
error_dec=norm(A-L*L')/norm(A);
disp(error_dec);

disp('Relative error:');
error1=norm(x-ex)/norm(ex);
disp(error1);

disp('Forward stability error:');
error2=error1/cond(A)^2;
disp(error2);

disp('Backward stability error:');
```

```
error3=norm(B-A*A*x)/(norm(A*A)*norm(x));
disp(error3);

end
```

5.2 cholesky.m

```
function [LC] = cholesky(M)
% The function finds the Cholesky factiorization of an tridiagonal po
% definite matrix
% Input variables:
% M The tridiagonal and symmetric positive definite matrix.
% Output variable:
% An lower triangular matrix such that L*L^T = M.

n = length( M );
D = zeros(n);
LC = zeros(n);
D(1,1)= M(1,1);

for i=2:n
    D(i,i) = M(i,i)-(M(i-1,i))^2/D(i-1,i-1);
end
LC=sqrt(LC+D);
for i=1:n-1
    LC(i+1,i) = M(i+1,i)/sqrt(D(i,i));
end
end
```

5.3 examples.m

```
disp('-------------------------Example 1-------------------------');
A = [3 1 0 0 0;1 3 1 0 0;0 1 3 1 0; 0 0 1 3 1;0 0 0 1 3];
b=transpose([1 1 1 1 1]);
x=LLT(A,b)

disp('-------------------------Example 2-------------------------');
A = tridiag1([2 2 2 2 2 2 2 2 2 2],[1 1 1 1 1 1 1 1 1]);
b=transpose([3 4 4 4 4 4 4 4 4 3]);
```

```
x=LLT(A,b)

disp('-------------------------Example 3-------------------------');
A = tridiag1([5 5 5 5 5 5],[1 2 3 2 1]);
b=transpose([1 3 4 5 2 0]);
x=LLT(A,b)

disp('-------------------------Example 4-------------------------');
A=triPosDef(10);
x=LLT(A,transpose(1:10))

disp('-------------------------Example 5-------------------------');
A=triPosDef(100);
x=LLT(A,transpose(1:100));

disp('-------------------------Example 6-------------------------');
A=triPosDef(1000);
x=LLT(A,transpose(1:1000));
```

5.4 triPosDef.m

```
function [A] = triPosDef(n)
%The function generates a random tridiagonal symmetric
%positive definite n by n matrix;
b = randn(1,n-1);
a = [abs(b), 0] +[0, abs(b)] +abs(randn(1,n));
A=diag(b,-1)+diag(b,1)+diag(a);
end
```

5.5 tridiag1.m

```
function [A] =tridiag1(a,b)
% [A] =tridiag1(a,b)
% A is a tridiagonal matrix with diagonal entries a(1),...,a(n)
% b(1),...,b(n-1) (below and under a diagonal)
c=b;
a=a(:); b=b(:); c=c(:);
na=max(size(a));
nb=max(size(b));
```

```
nc=max(size(c));
A=zeros(na);

if   nb~=(na-1)  ||  nc~=(na-1)
     disp('Dimensions of a, b and c   are not correct!');
     return;
end

A=diag(a)+  diag(b,-1)+diag(c,1);
end
```