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

Maciej Sobczyski

April 25, 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

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

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 by solving $L^C y = b$ for y by forward substitution, and $(L^C)^T x = y$ for x by back substitution to get the final result.

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)* 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.

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. *ForwardSub.m* - A simple function to solve a system of linear equations $Ax = b$ where A is lower triangular by forward substitution

5. *BackwardSub* - A simple function to solve a system of linear equations $Ax = b$ where A is upper triangular by using backward substitution.

The *cholesky.m* and *LLT.m* functions 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

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

$$L = \begin{pmatrix} \sqrt{3} & 0 & 0 & 0 & 0 \\ \frac{\sqrt{3}}{3} & \frac{2\sqrt{2}\sqrt{3}}{3} & 0 & 0 & 0 \\ 0 & \frac{\sqrt{2}\sqrt{3}}{4} & \frac{\sqrt{2}\sqrt{21}}{4} & 0 & 0 \\ 0 & 0 & \frac{2\sqrt{2}\sqrt{21}}{21} & \frac{\sqrt{21}\sqrt{55}}{21} & 0 \\ 0 & 0 & 0 & \frac{\sqrt{21}\sqrt{55}}{55} & \frac{12\sqrt{55}}{55} \end{pmatrix} \quad x = \begin{pmatrix} \frac{5}{18} \\ \frac{1}{6} \\ \frac{2}{9} \\ \frac{1}{6} \\ \frac{5}{18} \end{pmatrix}$$

3.2 Example 2

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

$$L = \begin{pmatrix} \sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}\sqrt{3}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{2}\sqrt{3}}{3} & \frac{2\sqrt{3}}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{3}}{2} & \frac{\sqrt{5}}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2\sqrt{5}}{5} & \frac{\sqrt{5}\sqrt{6}}{5} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{5}\sqrt{6}}{6} & \frac{\sqrt{6}\sqrt{7}}{6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{6}\sqrt{7}}{7} & \frac{2\sqrt{2}\sqrt{7}}{7} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}\sqrt{7}}{4} & \frac{3\sqrt{2}}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2\sqrt{2}}{3} & \frac{\sqrt{10}}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{3\sqrt{10}}{10} & \frac{\sqrt{10}\sqrt{11}}{10} \end{pmatrix} \quad x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

3.3 Example 3

4 Analysis of results

We worked hard, and achieved very little.

5 Source code

5.1 LTT.m

```
function [x] = LLT(A,b)
% 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.
% Output variable:
% A colum vector with the solution.

%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(~isequal(diag(A,1),diag(A,-1)))
    disp("A is not symmetric.")
    return;
end

%Positive definite
if(prod(diag(A))==0||any(diag(A)<0))
    disp("A is not positive definite.")
    return;
```

```
end

%Print condition number of A, as required in appendix
disp("Condition number of A: ")
disp(cond(A))

L = cholesky(A);

disp('Error:');
error_dec=norm(A-L*L')/norm(A);
disp(error_dec);
%Solve equations
y = ForwardSub(L,b);
x = BackwardSub(transpose(L),y);
```

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]);
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],[1 1 1 1 1 1 1
b=transpose([3 4 4 4 4 4 4 4 4 3]);
LLT(A,b)
```

5.4 ForwardSub.m

```
function x = ForwardSub(a,b)
% The function solves a system of linear equations ax=b
% where a is lower triangular by using forward substitution.
n = length(b);
x(1,1) = b(1)/a(1,1);
for i = 2:n
    x(i,1)=(b(i)-a(i,1:i-1)*x(1:i-1,1))./a(i,i);
end
```

5.5 BackwardSub.m

```
function y = BackwardSub(a,b)
% The function solves a system of linear equations ax=b
% where a is upper triangular by using backward substitution.
n = length(b);
y(n,1) = b(n)/a(n,n);
for i = n-1:-1:1
    y(i,1)=(b(i)-a(i,i+1:n)*y(i+1:n,1))./a(i,i);
end
```