16 April 2019

Maciej Zalewski
Group 2

Decomposition of a tridiagonal matrix using Cholesky factorization in solving system of equations $A^T Ax = b$.

Project 5

1 Method description

In linear algebra and numerical analysis the Cholesky factorization is being used to decompose a symmetric positive definite matrix into a product of a lower triangular matrix and its transposition.

$$A = LL^T \tag{1}$$

Cholesky decomposition efficiency in solving systems of linear equations is nearly twice as much as the normal LU decomposition.

For factorization of tridiagonal matrices it is possible to use slightly modificated algorithm which uses special properties of such matrices.

$$A = \begin{pmatrix} a_{1,1} & a_{2,1} & 0 & \dots & 0 \\ a_{2,1} & a_{2,2} & a_{3,2} & \dots & 0 \\ 0 & a_{3,2} & a_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{n,n} \end{pmatrix} \tag{2}$$

$$L = \begin{pmatrix} l_{1,1} & 0 & 0 & \dots & 0 \\ l_{2,1} & l_{2,2} & 0 & \dots & 0 \\ 0 & l_{3,2} & l_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & l_{n,n} \end{pmatrix} \tag{3}$$

For $n \times n$ matrix A we form matrices L (lower triangular) and L^T for an equation (1). Multiplying matrices on the right side and comparing outcome with A we recive formulas for the matrix L.

$$l_{1,1} = \sqrt{a_{1,1}}$$

$$\forall k \in \{2, ..., n\}$$

$$l_{k,k} = \sqrt{a_{k,k} - l_{k,k-1}^2}$$

Formula for main diagonal in matrix L. $a_{i,j}$ stands for corresponding elements in A.

$$l_{k,k-1} = \frac{a_{k,k-1}}{l_{k-1,k-1}}$$

Matrix A is tridiagonal, so matrix L have only 2 non-zero diagonals (). This fact is being used in the algorithm to skip 0 elements, thus increase efficiency.

Using the matrix L created in such way we can solve $A^T Ax = b$ by substitution (1).

$$LL^T LL^T x = b$$

Now it is possible to simplify calculations by next substitutions. Created equations are solved in given order:

1. $LQ = b$
2. $L^T Z = Q$
3. $LY = Z$
4. $L^T x = Y$

2 Program description

After you ran the program, you will the menu:

Cholesky tridiagonal factorization

Input tridiagonal matrix A

Input vector b

Display A and b

Compute factorization of A

Compute $A^T Ax = b$

Calculate Errors

FINISH

- Pressing 'Input tridiagonal matrix A' allows to input new matrix A by giving vector of size n for main diagonal and vector of size n-1 for other diagonals. New matrix is checked whether it is positive definite. Otherwise predefined matrix is used.
- 'Input vector b' allows to input new vecotr b which should be of size n (A $n \times n$). If not, vector b is predefined.
- 'Display A and b' shows current matrix A and vector b.
- Button 'Compute factorization of A' uses Cholesky decomposition on the matrix A, shows the result and stores it in matrix L.
- Option 'Compute $A^T Ax = b$' solves the given system of linear equations for the current matrix A and vector b using Cholesky decomposition.
- 'Calculate Errors' computes the condition number for A, the decomposition error and the relative error.
- Last option 'EXIT' closes the program.

Functions used:

1. MenuCholeskyTrid.m - script for GUI, implements other functions.

2. CholeskyTrid.m - function calculating matrix L for tridiagonal matrix A using Cholesky decomposition.
3. CholeskyTrid1.m - function uses CholeskyTrid.m to calculate L, then solves $A^T Ax = b$ with Lower_triangular.m and Upper_triangular.m .
4. Lower_triangular.m - function used to compute solution for $Ax = b$ where A is lower triangular matrix.
5. Upper_triangular.m - function used to compute solution for $Ax = b$ where A is upper triangular matrix.
6. tridiag1.m - function used to create tridiagonal $n \times n$ matrix from two vectors.
7. calculate_errors.m - for current A, b and L calculates the condition number of A, the decomposition error and the relative error.

   (**Note.** *All source codes can be found in section 5.*)

3 Numerical examples

Numerical tests are done for the predefined data in the program and some other cases to check how the algorithm behaves. Solution to the system of linear equations and decomposition of A is done with functions CholeskyTrid1 and CholeskyTrid. Then condition number for A and errors are computed with calculate_errors. Data for different exaples is set manually in the menu.

- Decomposition error

$$error\_dec = \frac{norm(A - L \cdot L^T)}{norm(A)}$$

- Relative error

$$error\_rel = \frac{norm(x - z)}{norm(x)}$$

  Where z is the solution calculated with MATLAB builtin functions.

Tests: x is the result obtained from using implemented Cholesky function on the given matrix A and vector b.

1. Predefined data: $A = hess(hilb(6))$ and $b = [1, 2, 3, 4, 5, 6]$

$$A = \begin{pmatrix} 0.0000 & -0.0000 & 0 & 0 & 0 & 0 \\ -0.0000 & 0.0003 & 0.0006 & 0 & 0 & 0 \\ 0 & 0.0006 & 0.0138 & 0.0224 & 0 & 0 \\ 0 & 0 & 0.0224 & 0.3198 & -0.3764 & 0 \\ 0 & 0 & 0 & -0.3764 & 1.4534 & -0.2935 \\ 0 & 0 & 0 & 0 & -0.2935 & 0.0909 \end{pmatrix}, \; b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}$$

Result:

$$x = \begin{pmatrix} 2.5487 \cdot 10^{14} \\ 1.0908 \cdot 10^{14} \\ -0.5945 \cdot 10^{14} \\ 0.3367 \cdot 10^{14} \\ 0.2506 \cdot 10^{14} \\ 0.8090 \cdot 10^{14} \end{pmatrix}$$

Tablica 1: 1st example *cond*(A) and errors

cond(A)	Decomposition error	Relative error
$1.4951 \cdot 10^{7}$	$1.0715 \cdot 10^{-18}$	$1.3830 \cdot 10^{-5}$

2. A and b from input

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

Result:

$$x = \begin{pmatrix} 0.0926 \\ -0.0000 \\ 0.0741 \\ -0.0000 \\ 0.0926 \end{pmatrix}$$

Tablica 2: 2nd example *cond*(A) and errors

cond(A)	Decomposition error	Relative error
3.7321	$9.3847 \cdot 10^{-17}$	$3.6610 \cdot 10^{-16}$

3. A and b from input

$$A = \begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 0 \\ -1 & 5 & -2 & 0 & 0 & 0 \\ 0 & -2 & 6 & -3 & 0 & 0 \\ 0 & 0 & -3 & 7 & -3 & 0 \\ 0 & 0 & 0 & -3 & 8 & -2 \\ 0 & 0 & 0 & 0 & -2 & 9 \end{pmatrix}, b = \begin{pmatrix} 6 \\ 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

Result:

$$x = \begin{pmatrix} 0.8197 \\ 1.1818 \\ 1.3503 \\ 1.1053 \\ 0.5888 \\ 0.1693 \end{pmatrix}$$

Tablica 3: 3rd example $cond(A)$ and errors

$cond(A)$	Decomposition error	Relative error
6.3402	$7.3944 \cdot 10^{-17}$	$6.1616 \cdot 10^{-16}$

4 Analysis of the results

In conclusion, solving equations $A^TAx = b$, where $A \in \mathbb{R}^{n\times n}$ is tridiagonal and symmetric positive definite with the Cholesky factorization method ($A = LL^T$) is accurate. When special properties of such tridiagonal matrices are taken into account, this method is even more efficient than the classical one.

As expected the method does not produce big errors when $cond(A)$ is small. The biggest relative error $error\_rel = 1.3830 \cdot 10^{-5}$ occured for default program data $A = hess(hilb(6))$ where $cond(A) = 1.4951 \cdot 10^7$, which is significantly more then $cond(A)$ in other examples (10^{-16}).

Decomposition errors in every case are negligibly small (10^{-17}).

In summary, this method is an accurate and quick way of solving $A^TAx = b$ for specific matrices A, because it uses special properties of tridiagonal symmetric matrices. One must have in mind to apply it for A with small condition number to lower the relative error.

5 Source Codes

MenuCholeskyTrid.m:

```
% MENU
clear
clc
finish=7;
control=1;

%default data
A = hess(hilb(6));
b = [1,2,3,4,5,6];

while control~=finish

    control=menu('Cholesky tridiagonal factorization', 'Input tridiagonal
                 'Display A and b','Compute factorization of A',...
                 'Compute A^{T}Ax=b', 'Calculate Errors', 'EXIT');

    switch control

        case 1
            a = input('main diagonal = ');
            c = input('upper and lower diagonals = ');
            A = tridiag1(a,c,c);
            chol(A);

        case 2
            b = input('b = ');

        case 3
            disp('A = '); disp(A)
            disp('b = '); disp(b)

        case 4
            L = CholeskyTrid(A);
            disp('L = '); disp(L)

        case 5
            X = CholeskyTrid1(A,b);
            disp('X = '); disp(X);

        case 6
```

```
            disp('cond A = '); disp(cond(A))
            calculate_errors(A,L,X,b);

        case 7
            disp('EXIT')
    end
end
```

CholeskyTrid.m:

```
function [L]=CholeskyTrid (A)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

[m,n] = size(A);
if m~=n
    disp('m should be equal to n');
    return;
end

L = eye(n);
L(1,1) = sqrt(A(1,1));
for i=2:n
    L(i,i-1) = A(i,i-1)/L(i-1,i-1);
    L(i,i) = sqrt(A(i,i) - L(i,i-1)^2);
end

end
```

CholeskyTrid1.m:

```
function [X]=CholeskyTrid1(A,b)

[m,n] = size(A);
if m~=n
    disp('m should be equal to n');
    return;
end

L = CholeskyTrid(A);
Q = Lower_triangular(L,b);
Z = Upper_triangular(L',Q);
Y = Lower_triangular(L,Z);
X = Upper_triangular(L',Y);

end
```

Lower_triangular.m:

```
function [x]=Lower_triangular(A,b)
% [x]=Lower_triangular1(A,b)
% x is the solution of Ax=b, where A is lower triangular

[m,n]= size(A);
x=zeros(n,1);

if m~=n,
    disp('m should be equal to n');
    return;
end

if norm(A-tril(A),'fro')>0
    disp('A is not lower triangular!');
    return;
end

d=diag(A);

if ~all(d),
    disp('Diagonal element of A  equals  0');
    return;
end

x(1)=b(1)/A(1,1);

for i=2:n,
    s=b(i);
    j = i-1;
    s=s-A(i,j)*x(j);
    x(i)=s/A(i,i);
end
end
```

Upper_triangular.m:

```
function [x]=Upper_triangular(A,b)
% [x]=Upper_triangular1(A,b)
% x is the solution of Ax=b, where A is upper triangular

[m,n]= size(A);
x=zeros(n,1);

if m~=n,
    disp('m should be equal to n');
    return;
```

```
end

if norm(A-triu(A),'fro')>0
    disp('A is not upper triangular!');
    return;
end

d=diag(A);

if ~all(d),
    disp('Diagonal element of A  equals  0');
    return;
end

x(n)=b(n)/A(n,n);

for i=n-1: -1:1,
    s=b(i);
    j=i+1;
    s=s-A(i,j)*x(j);
    x(i)=s/A(i,i);
end
end
```

tridiag1.m:

```
function [A] =tridiag1(a,b,c)
% [A] =tridiag1(a,b,c)
% A is a tridiagonal matrix with diagonal entries a(1),...,a(n)
% b(1),...,b(n-1) (upper diagonal) and c(1),...,c(n-1) (lower diagonal)

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
```

calculate_errors.m:

```
function [] = calculate_errors(A,L,X,b)

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

    M = A'*A;
    b=b(:);
    Z = M\b;
    disp('Z =');disp(Z)
    error_rel = norm(X - Z)/norm(X);
    disp('error_rel = '); disp(error_rel)

end
```