24 April 2018

Wojciech Ciok
EA1

3. Computing Inverse and cond_1(A) by LDLT method

Report 1

Table of Contents

1 Description of the methods

The input matrix A for the methods is assumed to be tridiagonal, symmetric positive definite.

1.1 Function cond$_1$

The function is defined in the following way:

$$cond_1 = ||A||_1||A^{-1}||_1$$

As we can see I will need two functions. One which will calculate 1-norm of given matrix A, and second which will produce the inverse of matrix A.

1.2 Function FirstNorm

The function sums absolute values of entries in each column and returns the biggest sum.

$$||A||_1 = \max_{1\leq j\leq n} \sum_{j=1}^{n} |a_{i,j}|$$

1.3 Function Invert

Matrix A^{-1}is an inverse of A if A*A^{-1} = I, where I is an identity matrix. For a matrix 2x2:

$$A^{-1} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} = \frac{1}{|A|} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}.$$

Computing an inverse can be pretty hard. Luckily, we can use our assumptions about A and use the LDLT decomposition.

$$\text{If } A = LDL^T \text{ then } A^{-1} = L^{T-1}D^{-1}L^{-1}$$

where D is diagonal, L is lower triangular. Since A is tridiagonal

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

As we can see we need additional functions for LDLT decomposition, transposing a matrix, inverting a diagonal matrix and inverting a lower triangular matrix as in (1).

1.4 Function MyTranspose

The transpose of a matrix is an operator which flips a matrix over its diagonal, that is it switches the row and column indices of the matrix by producing another matrix Formally, the i'th row, j'th column element of AT is the j'th row, i'th column element of A:

$$[A^T]_{i,j}=[A]_{j,i}$$

If A is an m x n matrix then AT is an n x m matrix.

1.5 Function InvertDiagonal

For a diagonal matrix:

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

the inverse matrix is

$$D^{-1} = \begin{pmatrix} 1/d_{1,1} & 0 & 0 & \dots & 0 \\ 0 & 1/d_{2,2} & 0 & \dots & 0 \\ 0 & 0 & 1/d_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1/d_{n,n} \end{pmatrix} \tag{3}$$

1.6 Function InvertLower

The input matrix is of a following form:

$$L = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{2,1} & 1 & 0 & \dots & 0 \\ 0 & l_{3,2} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}. \tag{4}$$

I'll denote the output matrix as:

$$LI = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ i_{2,1} & 1 & 0 & \dots & 0 \\ i_{3,1} & i_{3,2} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ i_{n,1} & i_{n,2} & i_{n,3} & \dots & 1 \end{pmatrix}. \tag{5}$$

For the diagonal entries just under the main diagonal

$$i_{k+1,k} = -1 * l_{k+1,k}$$

For the entries on the diagonal below that:

$$i_{k+2,k} = l_{k+1,k} * l_{k+2,k+1} - l_{k+2,k}$$

The rest of the entries are calculated in the following way:

$$i_{k,i} = (-L(k, 1:(k-1)) * LI(1:(k-1), i))/l_{k,k}$$

1.7 Function LDLFact

In order to decompose our tridiagonal matrix

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & 0 & \dots & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & 0 \\ 0 & a_{3,2} & a_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{n,n} \end{pmatrix} \tag{6}$$

we will find lower triangular matrix:

$$L = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{2,1} & 1 & 0 & \dots & 0 \\ 0 & l_{3,2} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \tag{7}$$

and a diagonal matrix

$$D = \begin{pmatrix} d_{1,1} & 0 & 0 & \dots & 0 \\ 0 & d_{2,2} & 0 & \dots & 0 \\ 0 & 0 & d_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & d_{n,n} \end{pmatrix} \tag{8}$$

and then we will find L^T by transposing L.

To determine L, D:
$a_{1,1} = d_{1,1} \implies d_{1,1} = a_{1,1}$
$a_{k,k-1} = l_{k,k-1} * d_{k-1,k-1} \implies l_{k,k-1} =_{k,k-1} /d_{k-1,k-1}$
$a_{k,k} = l_{k,k-1} * a_{k-1,k} + d_{k,k} \implies d_{k,k} = a_{k,k} - l_{k,k-1} * a_{k-1,k}, k = 2, ..., n$

To transpose L one can use MyTranspose.

2 Code

2.1 Function cond_1

```
function [answer] = cond1(A)
%cond1 returns product of 1-norm of matrix A and 1-norm of its
%inverse
%The argument matrix A is assumed to be a tridiagonal, symmetric positive
%definite.
%The inverse of the matrix is calculated using LDLT decomposition method.

AI = Invert(A);
answer = FirstNorm(A)*FirstNorm(AI);
end
```

2.2 Function Invert

```
function [AI] = Invert(A)
    [L,D] = LDLFact(A);
    LI = InvertLower(L);
    LIT = MyTranspose(LI);
    DI = InvertDiagonal(D);

    AI = LIT*DI*LI;
end
```

2.3 Function MyTranspose

```
function[T] = MyTranspose(A)
%my implementation of transposition
    [row, col] = size(A);
    T = zeros(col, row);
    iX = 1;
    for iCol = 1:col
        iY = iCol;
        for iRow = 1:row
            T(iY) = A(iX);
            iY = iY + col;
            iX = iX + 1;
        end
    end
end
```

2.4 Function InvertDiagonal

```
function[DI] = InvertDiagonal(D)
%inverts a diagonal matrix
    [row, col] = size(D);
    DI = zeros(col, row);
    for i = 1:col
        DI(i,i) = 1/D(i,i);
    end
end
```

2.5 Function InvertLower

```
function [LI] = InvertLower(L)
    n = length(L);
    LI = eye(n);
    for i = 1:n-1
        LI(i+1,i) = -L(i+1,i);
    end
    for i = 1:n-2
        LI(i+2,i) = L(i+1,i)*L(i+2,i+1)-L(i+2,i);
    end
    for k = 4:n
        for i = 1:k-2
            LI(k,i) = (-L(k,1:(k-1))*LI(1:(k-1),i))/L(k,k);
        end
    end
end
```

2.6 Function LDLFact

```
function[L,D,LT] = LDLFact(A)
   % A is a symmetric, tridiagonal matrix
   % factor A such that A=LDL^T
   % L a lower triangular matrix with 1 for all its diagonal
      entries
   [n,m]=size(A);
   if n~=m
       error("matrix is not square");
   end
   L = eye(n);
   Diag = zeros(n,1);
   Diag(1)=A(1,1);
   L(2,1)=A(2,1)/Diag(1);
   for i=2:n
       Diag(i)=A(i,i)-L(i,i-1).^2*Diag(i-1);
```

```
            if i~=n
                L(i+1,i)=(A(i+1,i)-L(i+1,i-1).*L(i,i-1)*Diag(i-1))/Diag(i);
            end
    end
    D = zeros(n,n);
    for i = 1:n
        D(i,i)=Diag(i);
    end
    LT = MyTranspose(L);
end
```

3 Description of the program

There is a possibility to work with a program by running a menu script implemented in the following way:

```
finish=4;
kontrol=1;
OK=0;
while kontrol~=finish ,
    kontrol=menu('Testing program','Your matrix A','LDL decomposition','Inverse','FINISH');
    switch kontrol
        case 1
            OK=1;
            A=input('Your matrix A = ')
        case 2
            if OK
                [L,D,LT] = LDLFact(A);
                disp('L = '), disp(L)
                disp('D = '), disp(D)
                disp('LT = '), disp(LT)
            else disp('A is unknown')
            end
        case 3
             if OK
                 disp('Inverse of A:  '),disp(inv(A))
             else disp('   A is unknown ')
            end
        case 4
            disp('FINISH')
    end
end
```

Options in the program:

1. Your matrix A
   Define your input Matrix A in the command window. A should be tridiagonal, symmetric positive definite.

2. LDL decomposition
   Displays L, D and L^T obtained from decomposition of the given matrix A.

3. Inverse
   Displays the inverse of the given matrix A.

4. FINISH
   End the program.

Testing Program
Your matrix A
LDL decomposition
Inverse
FINISH

4 Analysis

4.1 Example Results

a)

$$A = \begin{pmatrix} 11 & 3 & 0 & 0 \\ 3 & 13 & 1 & 0 \\ 0 & 1 & 7 & 5 \\ 0 & 0 & 5 & 21 \end{pmatrix} \tag{9}$$

$$Invert(A) = A^{-1} = \begin{pmatrix} 0.0971 & -0.0227 & 0.0039 & -0.0009 \\ -0.0227 & 0.0833 & -0.0143 & 0.0034 \\ 0.0039 & -0.0143 & 0.1746 & -0.0416 \\ -0.0009 & 0.0034 & -0.0416 & 0.0575 \end{pmatrix} \tag{10}$$

$cond_1(A) = 6.0947$

b)
I'm using full(gallery('tridiag',n,a,b,a)) to generate a square, symmetric, tridiagonal matrix and then check the matrix using chol() to find out if the matrix is positive

definite.
full(gallery('tridiag',10,2,13,2)):

$$A = \begin{pmatrix} 13 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 13 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 13 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 13 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 13 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 13 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 13 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 13 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 13 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 13 \end{pmatrix} \tag{11}$$

$Invert(A) = A^{-1} =$

$$\begin{pmatrix} 0.0788 & -0.0124 & 0.0020 & -0.0003 & 0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 & -0.0000 \\ -0.0124 & 0.0808 & -0.0127 & 0.0020 & -0.0003 & 0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 \\ 0.0020 & -0.0127 & 0.0808 & -0.0127 & 0.0020 & -0.0003 & 0.0000 & -0.0000 & 0.0000 & -0.0000 \\ -0.0003 & 0.0020 & -0.0127 & 0.0808 & -0.0127 & 0.0020 & -0.0003 & 0.0000 & -0.0000 & 0.0000 \\ 0.0000 & -0.0003 & 0.0020 & -0.0127 & 0.0808 & -0.0127 & 0.0020 & -0.0003 & 0.0000 & -0.0000 \\ -0.0000 & 0.0000 & -0.0003 & 0.0020 & -0.0127 & 0.0808 & -0.0127 & 0.0020 & -0.0003 & 0.0000 \\ 0.0000 & -0.0000 & 0.0000 & -0.0003 & 0.0020 & -0.0127 & 0.0808 & -0.0127 & 0.0020 & -0.0003 \\ -0.0000 & 0.0000 & -0.0000 & 0.0000 & -0.0003 & 0.0020 & -0.0127 & 0.0808 & -0.0127 & 0.0020 \\ 0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 & -0.0003 & 0.0020 & -0.0127 & 0.0808 & -0.0124 \\ -0.0000 & 0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 & -0.0003 & 0.0020 & -0.0124 & 0.0788 \end{pmatrix} \tag{12}$$

$\text{cond}_1(A) = 1.8887$

c)
A = full(gallery('tridiag',1000,-1,3.5,-1));
$cond_1(A) = 3.6667$

4.2 Errors

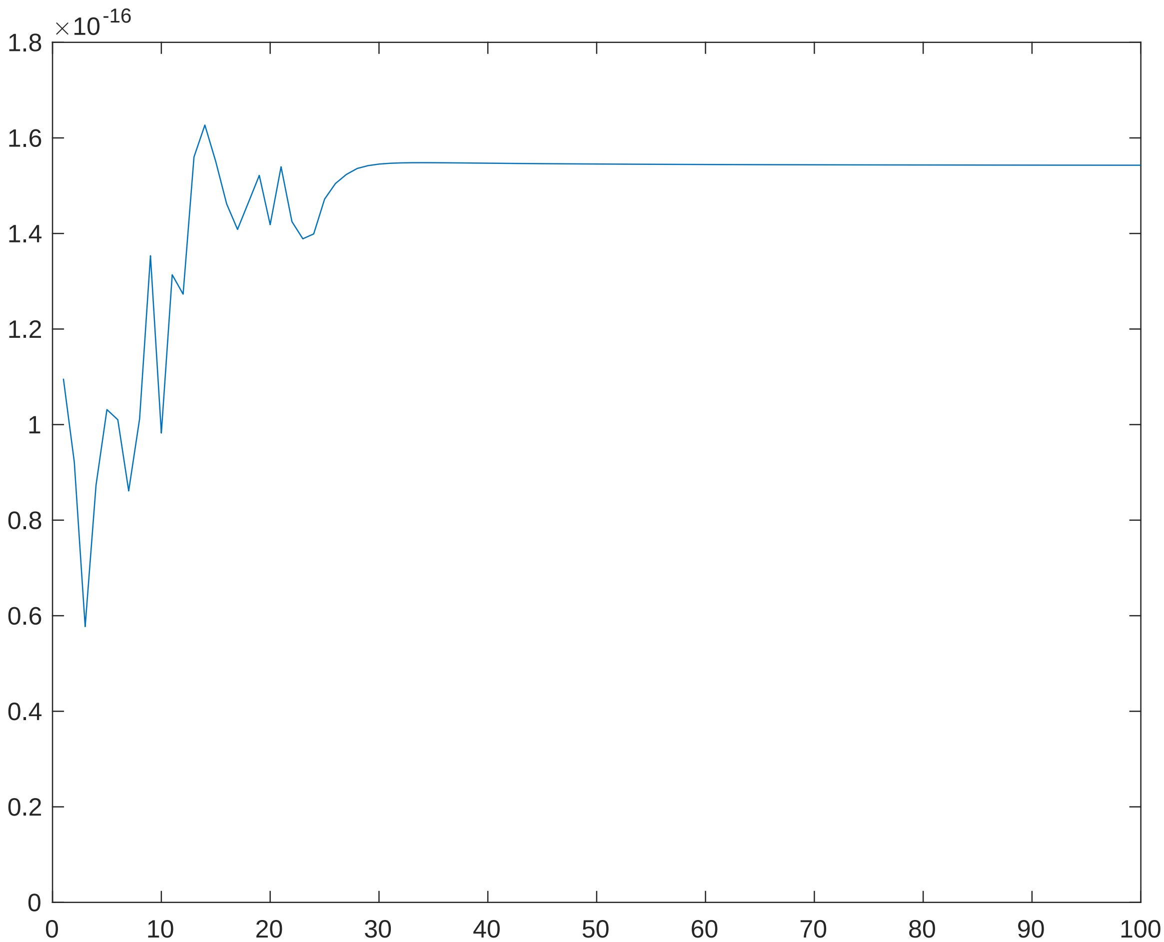
4.2.1 Error checking function

```
function [] = Errors(A)
    I = eye(size(A));
    X = Invert(A);
    AA = inv(X);
    cond_A = cond(A);
    RRE = norm(A*X-I)/(norm(A)*norm(X));
    LRE = norm(X*A-I)/(norm(A)*norm(X));
    error = norm(AA-A)/norm(A);
    disp('cond:')
    cond_A
    disp('right residual error:')
    RRE
    disp('left residual error:')
```

```
    LRE
    disp('error:')
    error
end
```

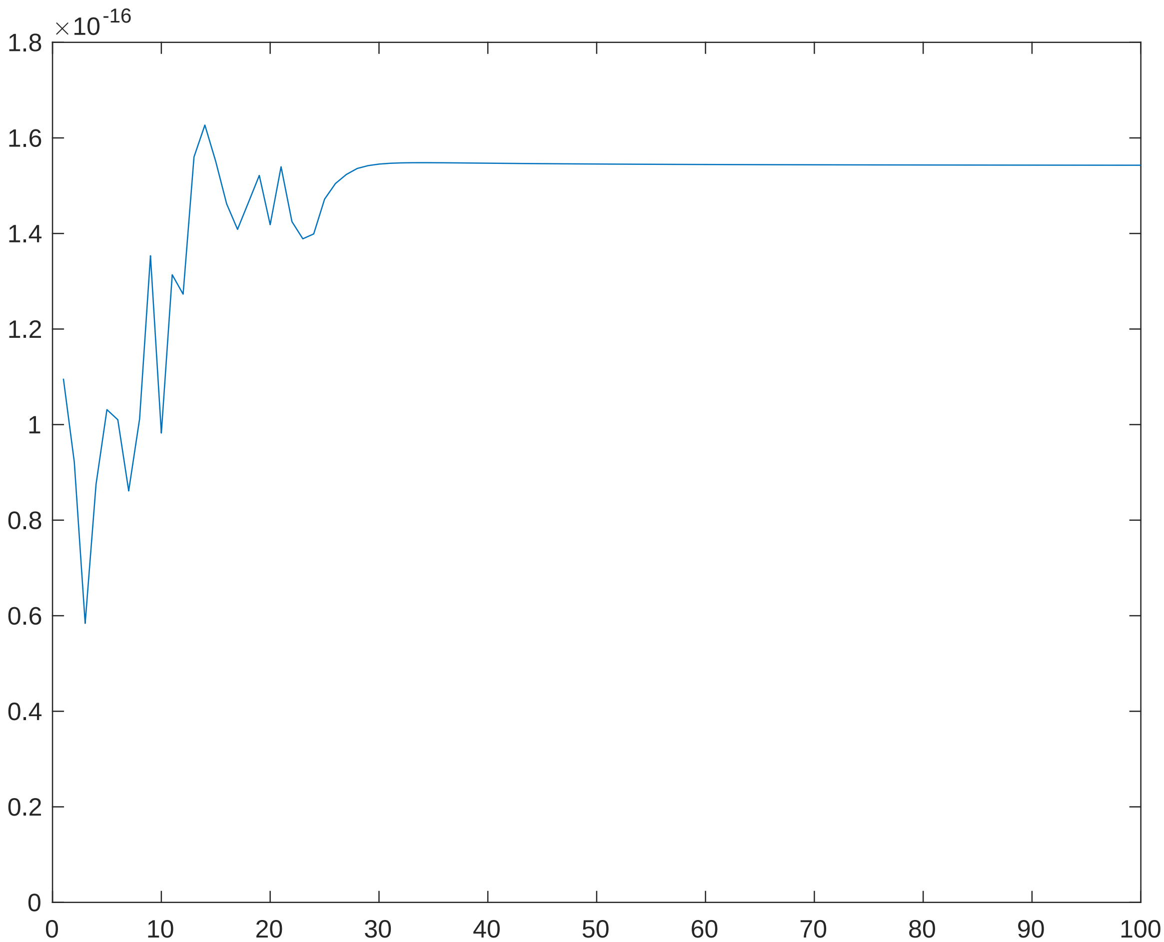
4.2.2 Right residual error

Error calculated for matricies of size from 3 to 103 and with 5, 20 and 5 on diagonals


×10^-16
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50
60
70
80
90
100

4.2.3 Left residual error

Error calculated for matricies of size from 3 to 103 and with 5, 20 and 5 on diagonals

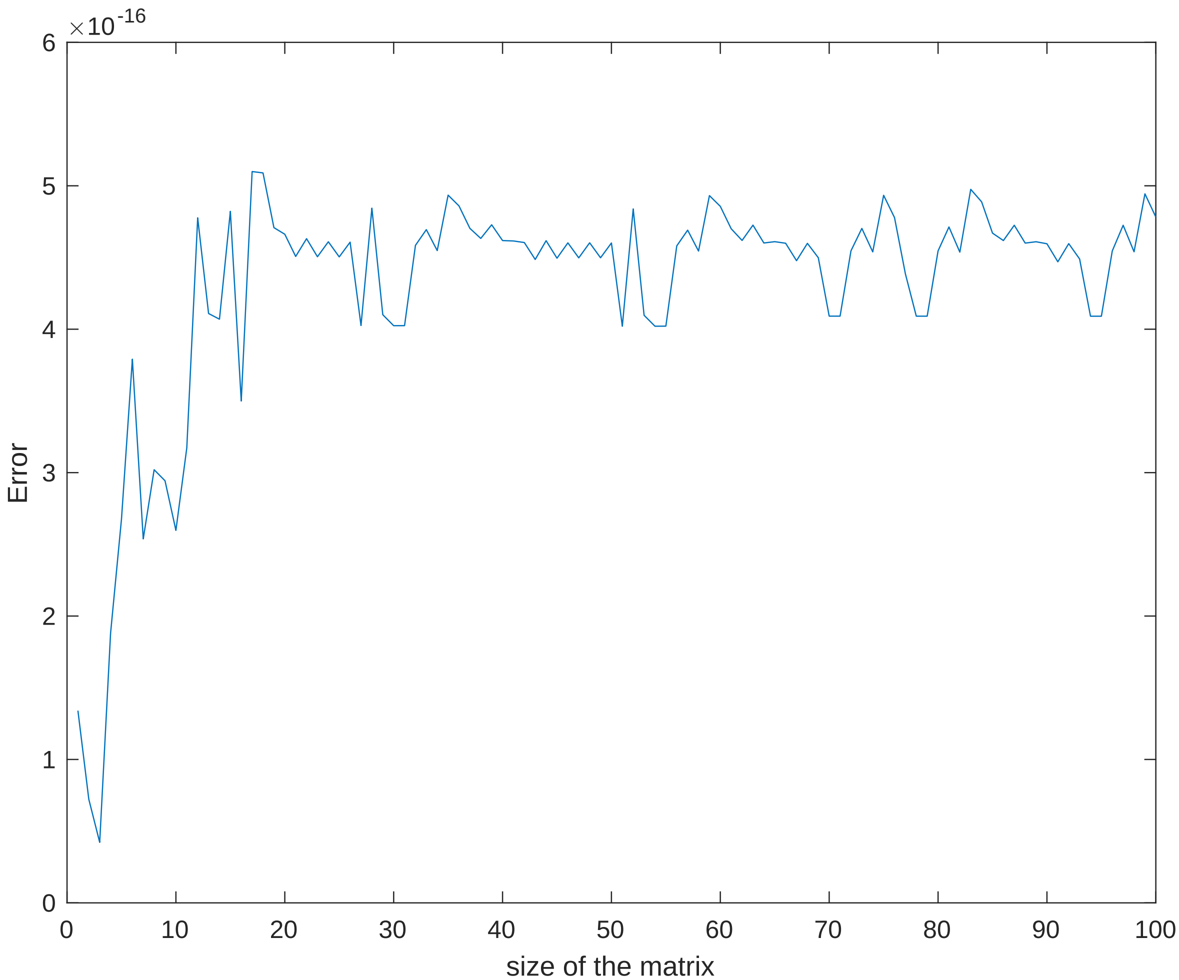


×10^-16
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50
60
70
80
90
100

4.2.4 Error

Error calculated for matricies of size from 3 to 103 and with 5, 20 and 5 on diagonals by formula:

$$error = norm(inv(Invert(A))/norm(A) \tag{13}$$


×10^-16
6
5
4
3
2
1
0
Error
0
10
20
30
40
50
60
70
80
90
100
size of the matrix

5 Conclusions

Generally the errors are very small, as small as 10^{-16}. The errors grow to about size of the given matrix equal to 30 and then hovers between $4*10^{-16}$ and $5*10^{-16}$. The errors may differ for much larger matrices. The methods have been optimalized taking into account tridiagonality of the input matrix. When calculating errors, when inverting an inverse, the matlab function inv() is used because implemented by me method Invert() works on tridiagonal matrices and the inverse of such a matrix is not tridiagonal.