Warsaw, 07.04.2019.

Assignment A: Solving linear algebraic equations #14

Final Report

Course: ENUME

Author: Kacper Kamieniarz

Index number: 293065

Advisor: Andrzej Miękina

Table of Contents

Task 1

Overview

The assignment consists of 5 tasks, first of which is a design of a procedure generating an $N \times N$ matrix following a given pattern. Input arguments of the function are N (for size of the matrix) and x which is some value used for filling the cells of the matrix.

$$\mathbf{A}_{N,x} = \begin{bmatrix} x^2 & \frac{3x}{2} & -\frac{3x}{2} & \frac{3x}{2} & \cdots & \frac{3x}{(-1)^{N-1}\cdot 2} & \frac{3x}{(-1)^{N}\cdot 2} \\ \frac{3x}{2} & \frac{18}{4} & -\frac{18}{4} & \frac{18}{4} & \cdots & \frac{18}{(-1)^{N-1}\cdot 4} & \frac{18}{(-1)^{N}\cdot 4} \\ -\frac{3x}{2} & -\frac{18}{4} & \frac{27}{4} & -\frac{27}{4} & \cdots & \frac{27}{(-1)^{N-4}\cdot 4} & \frac{27}{(-1)^{N-3}\cdot 4} \\ \frac{3x}{2} & \frac{18}{4} & -\frac{27}{4} & \frac{36}{4} & \cdots & \frac{36}{(-1)^{N-5}\cdot 4} & \frac{36}{(-1)^{N-4}\cdot 4} \\ -\frac{3x}{2} & -\frac{18}{4} & \frac{27}{4} & -\frac{36}{4} & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & & \frac{(N-1)\cdot 9}{4} & -\frac{(N-1)\cdot 9}{4} \\ \frac{3x}{(-1)^{N}\cdot 2} & \frac{18}{(-1)^{N}\cdot 4} & \frac{27}{(-1)^{N-3}\cdot 4} & \frac{36}{(-1)^{N-4}\cdot 4} & \cdots & -\frac{(N-1)\cdot 9}{4} & \frac{N\cdot 9}{4} \end{bmatrix}$$

Figure 1

Results

Results of generating 2 matrices using parameters $N = 5, x = 1$ and $N = 4, x = 3$ are shown below

```
>> makeMatrix(5, 1)

ans =

    1.0000    1.5000   -1.5000    1.5000   -1.5000
    1.5000    4.5000   -4.5000    4.5000   -4.5000
   -1.5000   -4.5000    6.7500   -6.7500    6.7500
    1.5000    4.5000   -6.7500    9.0000   -9.0000
   -1.5000   -4.5000    6.7500   -9.0000   11.2500
```

Figure 2

```
>> makeMatrix(4, 3)

ans =

    9.0000    4.5000   -4.5000    4.5000
    4.5000    4.5000   -4.5000    4.5000
   -4.5000   -4.5000    6.7500   -6.7500
    4.5000    4.5000   -6.7500    9.0000
```

Figure 3

Comparing obtained matrices with the assignment criteria shows that function $makeMatrix(\dots)$ follows the given pattern and matrices generated for different N and x values are in accordance with criteria.

Task 2

Overview

Using procedure designed in Task 1, we generate matrices $A_{N,x}$ with $N = \{3, 10, 20\}$ and $x = \log(\alpha)$, where

α – the value for which $\det(A_{N,x}) = 0$;

The only value dependent on α is x which only appears in the first row as well as the first column of the generated matrix. Hence it is obvious that the determinant of the matrix is equal to 0 whenever x is equal to 0. Therefore this follows

$$x = 0 \rightarrow \log(\alpha) = 0 \rightarrow \alpha = 1$$

So the value for which $\det(A_{N,x}) = 0$; is $\alpha = 1$.

For such matrices, we plot the dependence of $\det(A_{N,x})$ and $cond(A_{N,x})$ on $\alpha_N \in [\alpha_N - 0{,}01, \alpha_N + 0{,}01]$ using $semilogy(\dots)$ Matlab function, where

$cond(\dots)$ - Condition number of a matrix - the ratio of the largest singular value of that matrix to the smallest singular value.

$\det(\dots)$ – determinant of a matrix

Results

The graphs obtained, were plot using semilogy(…) function.

The figure of dependence shows, that the bigger the N value, the higher the determinant values, also as alfa approaches 1, the determinant rapidly decreases.

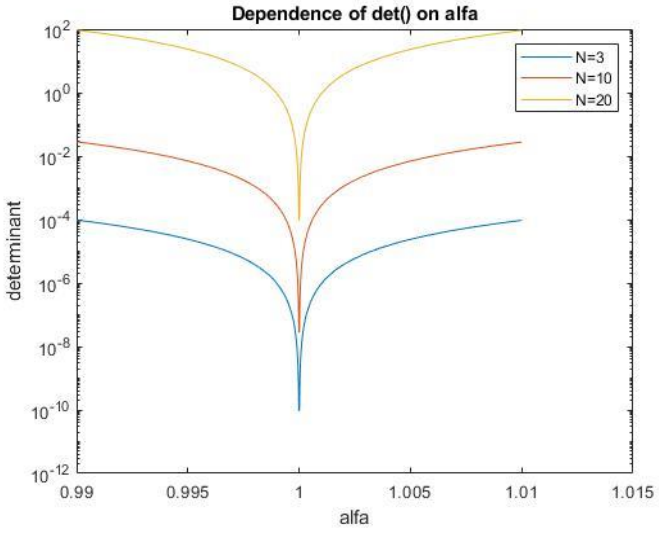

Figure 4

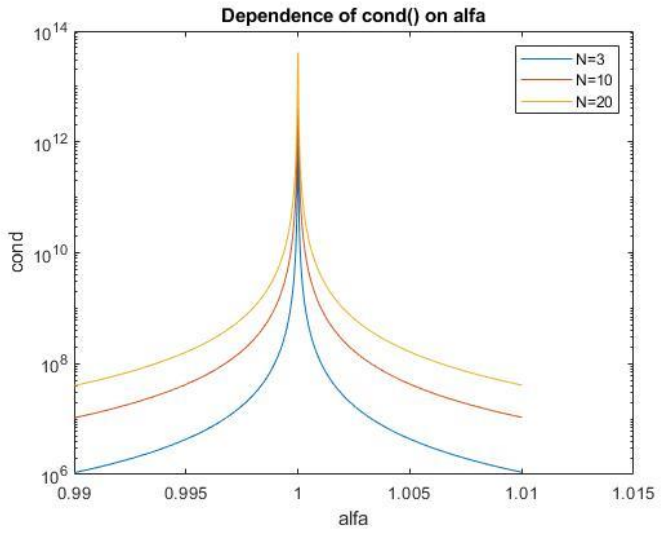

Figure 5

Same as in case of the determinant, the bigger the N, the greater the values that condition number takes on. As alfa approaches 1, the values of condition number are rapidly increasing unlike the determinant values in the previous case.

Task 3

Overview

LU decomposition of matrix A is the factorization of a given square matrix into two triangular matrices, one upper triangular matrix and one lower triangular matrix, such that the product of these two matrices gives the original matrix.

$L \cdot U = A$

The Cholesky decomposition of a Hermitian positive-definite matrix A is a decomposition of the form

$A = LL^T$, where

L is a lower triangular matrix with real and positive diagonal entries,

L^T denotes the conjugate transpose of L.

Using built in Matlab functions $chol(...)$ and $lu(...)$ along with scheme presented on lecture slide #3-16 we design functions:

- $inverseLU(...)$ which takes a matrix as argument and returns its inverse computed using LU decomposition,
- $inverseLLT(...)$ which also takes matrix and returns its inverse computed using Cholesky decomposition.

Algorithm

To calculate the inverse matrix A^{-1} we use the equality $A \cdot A^{-1} = I$, which we can rewrite as

$$\mathbf{A} \cdot \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,N} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N,1} & y_{N,2} & \cdots & y_{N,N} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$
$$\mathbf{y}_1 \quad \mathbf{y}_2 \quad \cdots \quad \mathbf{y}_N \qquad \mathbf{e}_1 \quad \mathbf{e}_2 \quad \cdots \quad \mathbf{e}_N$$

After which we end up with N systems of linear equations, all of which contain the same matrix A, which we decompose using LU or LLT factorization. After the decomposition we are left with $L \cdot U \cdot A^{-1} = I_N$, where

$Y = U \cdot A^{-1}$, therefore

$Y_N = \frac{I_N}{L}$, which we plug into $Y = U \cdot A^{-1}$ to obtain

$$A^{-1} = \frac{Y}{U}$$

In the LLT case we proceed in exactly the same way, except the upper triangle matrix U is now the transpose of L.

Results

To validate the algorithm of inverseLU(…) the script was used,

```
>> ALU = makeMatrix(7, 20);
ALUi = inverseLU(ALU);
ALU*ALUi
```

That is the result

```
ans =

    1.0000   -0.0000         0         0         0         0         0
         0    1.0000         0         0         0         0         0
         0         0    1.0000         0         0         0         0
         0         0         0    1.0000         0         0         0
         0         0         0         0    1.0000         0         0
         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0    1.0000
```

Figure 6

Similarly in case of inverseLLT(…)

```
>> ALLT = makeMatrix(7, 20);
ALLTi = inverseLU(ALLT);
ALLT*ALLTi
```

With result being

```
ans =

    1.0000   -0.0000         0         0         0         0         0
         0    1.0000         0         0         0         0         0
         0         0    1.0000         0         0         0         0
         0         0         0    1.0000         0         0         0
         0         0         0         0    1.0000         0         0
         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0    1.0000
```

Figure 7

Tasks 4 and 5

Overview

Using function from task 1 for generating matrices and procedures from task 3 for computing their inverses we generate the estimates $A_{N,x}^{-1}$ of $A_{N,x}$ for $N \in \{3, 10, 20\}$ and $x = \frac{2^k}{300}$ with $k \in \{0, 1, 2, \dots, 21\}$

For calculating the norms with root-mean-square error as well as the maximum error, the following formulas were used

$$\|\mathbf{A}\|_2 = \sup\left\{\sqrt{\lambda} \mid \lambda \in \mathrm{Spect}\left(\mathbf{A}^T\mathbf{A}\right)\right\} \quad \text{– induced by } \|\mathbf{x}\|_2$$

$$\|\mathbf{A}\|_\infty = \sup\left\{\sum_{\nu=1}^{N}\left|a_{n,\nu}\right| \mid n = 1, \dots, N\right\} \quad \text{– induced by } \|\mathbf{x}\|_\infty$$

```
function [B] = RMS(A)
X = A*A.';
B = max(sqrt(eig(X)));
end
```

```
function [B] = maximumError(A)
B = max((sum(abs(A), 2)));
end
```

In this script 22 matrices are generated for each N, along with their inverses obtained with LU and LLT. Also both RMS and maximumError matrices are generated δ_2 and δ_∞ for their comparison with Matlab built in norm(…) function.

Results

The figures for the dependence of δ_2 and δ_∞ on x were plot using semilogy(…) function.

Comparing functions maximumError(A) and norm(…) Matlab function showed that in fact there is no difference at all in these values, however RMS(A) compared with norm(…) gives some particularly small discrepancies of around 10^{-14} value.

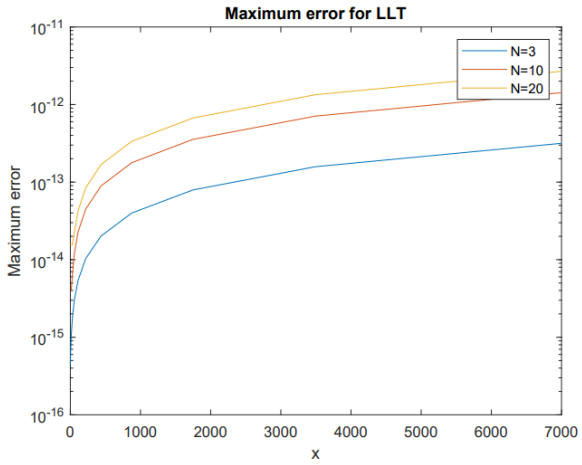

Figure 8

δ_∞ on x dependence in case of LLT inverse shows that the greatest values of error were obtained for matrices with N = 20, with N = 3 giving the smallest value of error, so in general the bigger the N, the greater the error.

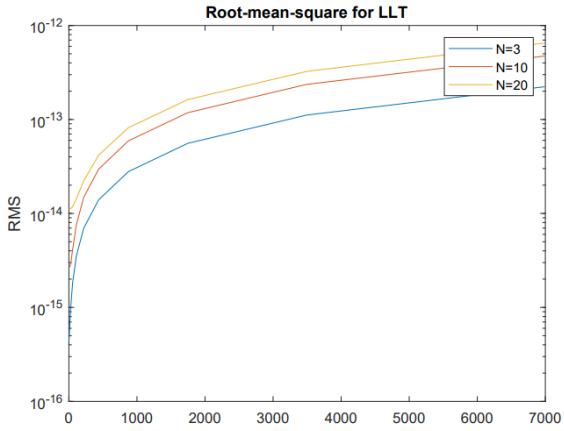

Figure 9

δ_2 follows the same pattern as δ_∞ meaning that the greatest values of error were obtained with 20x20 matrices with the smallest errors for 3x3 matrices.

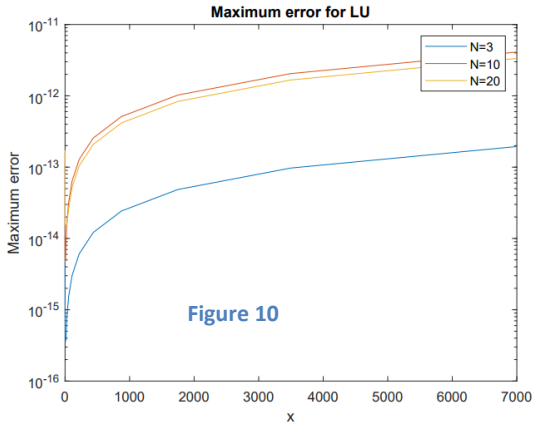

Figure 10

The situation looks slightly different with inverses using LU decomposition, because the values of $\delta_\infty$are maximal for 10x10 matrices with smallest for 3x3.

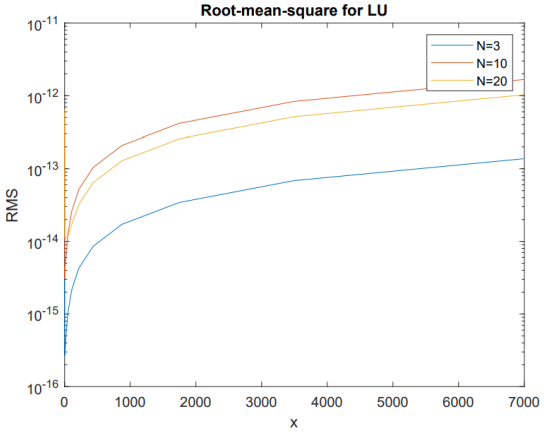

Figure 11

Similarly δ_2 error values are greatest for 10x10 matrices with smallest values for 3x3 matrices.

Conclusions

We observed that in case of LLT inverse, the greater values of N meant less accuracy, therefore smaller matrices are inverted more accurately. Comparing both methods of inverting a matrix shows that for values of N < 20 error values are very similar, however for N = 20 the inverses obtained with LLT seem more accurate, as the error values are smaller.