Numerical Methods 1 - Report

Mikołaj Jędrzejewski

May 2024

1 Problem statement

Solving a matrix equation $AX = B$, where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ and A has the form

$$A = \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix},$$

where $A_{ij} \in \mathbb{R}^{p \times p}$ and $n = 2p$, A_{11} is lower triangular and A_{22} is upper triangular. Calculate $det(A)$.

2 Used notation

For a matrix A,

1. $A(i, j)$ refers to an element of A in the i-th row and the j-th column
2. $A(i, :)$ refers to the i-th row of A
3. $A(:, j)$ refers to the j-th column of A

For Laplace expansion for a matrix A,

$$C(i, j) = (-1)^{i+j} M(i, j)$$

$$det(A) = \sum_{j=1}^{n} A(i, j) \cdot C(i, j)$$

where $C(i, j)$ denotes a co-factor and $M(i, j)$ a minor of A.

3 Method description

3.1 Solving system to linear equations

Let us divide the solution into two part. We can define X and B to be of the form

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \qquad\qquad B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$$

where $X_1, X_2, B_1, B_2 \in \mathbb{R}^{p \times m}$. We will solve first for X_1 and then for X_2. Thus, we can rewrite the equation $AX = B$ in a block form

$$\begin{pmatrix} A_{11} & 0 \\ A_{12} & A_{22} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \tag{1}$$

After applying matrix multiplication we arrive at formulas for B_1 and B_2,

$$A_{11}X_1 = B_1 \tag{2}$$

$$A_{22}X_2 = B_2 - A_{11}X_1 \tag{3}$$

Notice that since A_{11} is lower triangular we can easily solve for X_1 with forward substitution. Thus,

$$X_1(1,k) = \frac{B_1(1,k)}{A_{11}(1,1)} \tag{4}$$

$$X_1(i,k) = \frac{B_1(i,k) - \sum_{j=1}^{k-1} A_{11}(i,j) \cdot X_1(j,k)}{A_{11}(i,i)} \tag{5}$$

for each system k. In order to compute X_2 we will use a similar approach since A_{22} is upper triangular,however, using backward substitution instead.

3.2 Calculating $det(A)$

Notice that $det(A)$ can be computed as a product of the entries on the main diagonal. We can prove that observation by applying Laplace expansion multiple times. For the first p elements of matrix A we apply Laplace expansion along i-th row which has only one non-zero element $A(i,i)$. Thus,

$$det(A) = det(A_{22}) \cdot \prod_{i=1}^{p} A(i,i) \tag{6}$$

Analogously, we can calculate $det(A_{22})$ by applying Laplace expansion p times along j-th column for which only $A(j,j)$ in non-zero. Thus,

$$det(A) = \prod_{i=p+1}^{n} A(j,j) \cdot \prod_{i=1}^{p} A(i,i) = \prod_{i=1}^{n} A(i,i) \tag{7}$$

4 MATLAB implementation

Project is implemented in multiple files, the guide describes what each of them contains. For more information run ***help filename.m*** in MATLAB command window.

4.1 Guide through the files

- ***solve.m*** solves the problem using methods described in Section 3
- *solve_matlab.m* solves the problem using MATLAB built-in functions
- ***substitute.m*** implements forward and backward substitution
- *generate_case.m* generates a single test case
- *test_method.m* provides method limitations by calculating errors from Section 4.3
- *test_program.m* runs both solution and provides error defined in Section 4.2
- *plot_time.m* plots dependency of n and runtime for both solutions
- *plot_error.m* plots dependency of n and errors defined in Section 4.2 and Section 4.3
- *plot_method_error.m* plots dependency of n and errors defined in Section 4.3

4.2 Errors representing correctness

Tests for a solution are generated by choosing matrices A and X which allow to obtain matrix B as $B = AX$. We determine the correctness of a solution for computed $\hat{X}$ and true solution X as

$$error_X = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \left| X(i,j) - \hat{X}(i,j) \right|}{n^2} \tag{8}$$

The true value of $det(A)$ is not know, the error is computed as the absolute difference between solution defined in Section 3.2 and the solution using MATLAB built-in ***det*** function.

4.3 Limitations of a solution

To analyze the source and behaviour of error of a solution to solving systems of linear equations $AX = B$, where $A \in \mathbb{R}^{n \times n}$, $\in \mathbb{R}^{n \times m}$ we compute following metrics

- Condition number of A: $cond(A) = ||A^{-1}|| \cdot ||A||$.
- Relative error: if X is the exact solution to the system $AX = B$ and $\hat{X}$ is the computed solution by a method then:

$$error_1 = \frac{||\hat{X} - X||}{||X||}$$

- Forward stability error:

$$error_2 = \frac{||\hat{X} - X||}{||X|| \cdot cond(A)}$$

- Backward stability error (right residual error):

$$r_R = \frac{||B - A\hat{X}||}{||A|| \cdot ||\hat{X}||}$$

5 Simple examples

Example 1

Using function *generate_case(4, 3, 'natural')* the following matrices where generated (4 unknowns and 3 systems).

$$A = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 7 & 5 & 0 & 0 \\ 6 & 6 & 3 & 4 \\ 4 & 8 & 0 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 10 & 15 & 50 \\ 64 & 51 & 125 \\ 118 & 120 & 164 \\ 109 & 87 & 143 \end{pmatrix}$$

Error for the custom method is 0 and about 10^{-15} for MATLAB's function for calculating X. Computed determinants differ by about 10^{-14}.

Example 2

Using function *generate_case(4, 1, 'normal')* the following matrices where generated (4 unknowns and 1 system, entries from standard normal distribution).

$$A = \begin{pmatrix} 0.5134 & 0 & 0 & 0 \\ -0.2898 & -0.2223 & 0 & 0 \\ 2.1261 & -1.1517 & 0.3727 & -0.3525 \\ -0.3852 & 0.4556 & 0 & 1.2731 \end{pmatrix} \qquad B = \begin{pmatrix} -0.2758 \\ 0.1164 \\ -1.3386 \\ 0.3065 \end{pmatrix}$$

Error for the custom method is about 10^{-15} and about 10^{-16} for MATLAB's function for calculating X. Computed determinants differ by about 10^{-15}.

6 Testing method correctness

The function *generate_case* might not always generate a well-conditioned matrix but it allows for creating a diagonally dominant matrix, which can make the condition number smaller.

6.1 Diagonally dominant matrix

Function *plot_error* is hard-coded to work with diagonally dominant matrices. When *plot_error(4, 500)* was run for hard-coded $m = 1$ it created the plots visible on Figures 1, 2 and 3.

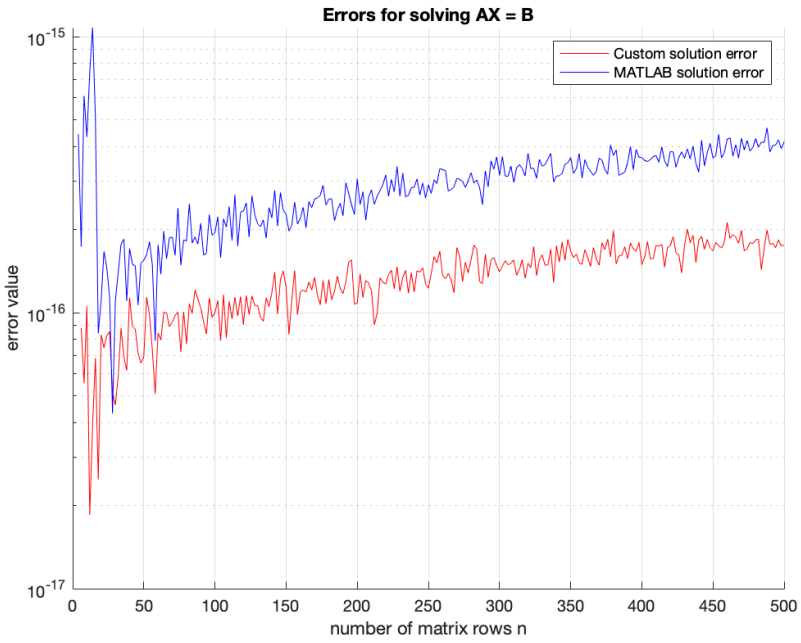

Figure 1: Plot of correctness of solutions for solving $AX = B$.

Custom solution seems to be slightly more correct. However for diagonally dominant matrices the determinant error grows exponentially. As can be seen on Figure 2. The value is too big to store and not all cases can be seen on the graph. For some cases the error cannot be plotted on a log scale.

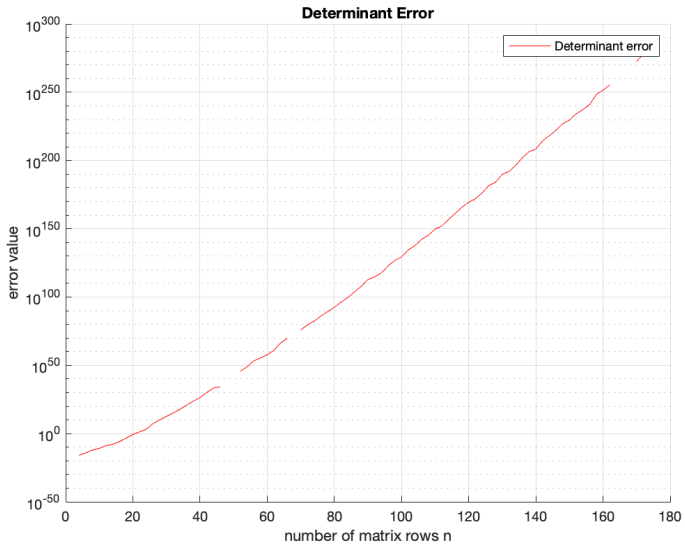

Figure 2: Determinant error plot

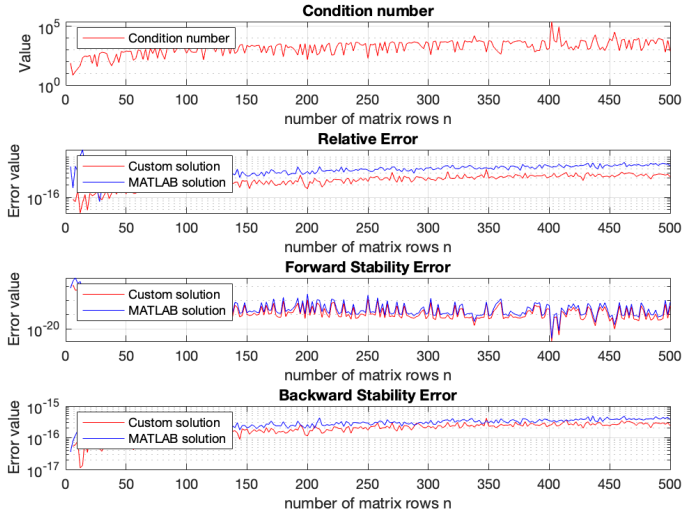

Figure 3: Correctness of solutions for solving $AX = B$.

Custom solution seems to have slightly smaller error values. Condition number is still high despite A being diagonally dominant.

6.2 Not diagonally dominant matrix

Changing code of *plot_error* by setting option for not enforcing A to be diagonally dominant shows that the error grows exponentially for both solutions. Error for solving $AX = B$ is presented on Figure 1 determinant difference on Figure 2.

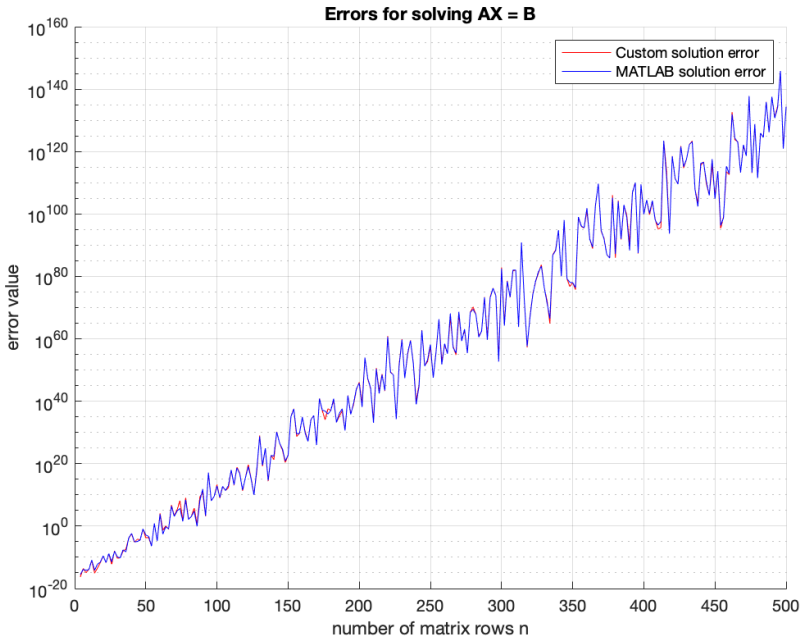

Figure 4: Plot of correctness of solutions for solving $AX = B$.

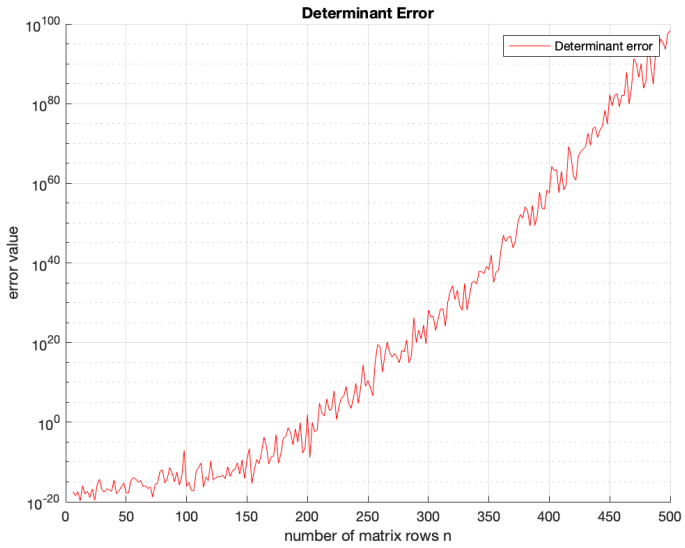

Figure 5: Determinant error plot

Generated matrices are mostly ill-conditioned with condition number quickly increasing to 10^{20}. Relative and forward stability errors reach 10^{200} for both methods upon closer inspection.

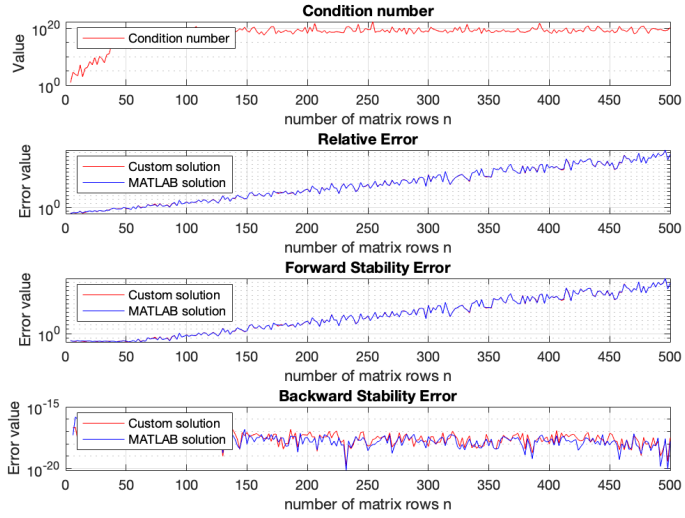

Figure 6: Correctness of solutions for solving $AX = B$.

7 Runtime analysis

In this section runtime dependency on the number of systems m is compared between the two solutions.

7.1 Single system ($m = 1$)

We can run multiple tests increasing the number n and keeping the number of systems $m = 1$. Figure 4 shows average times per one system for solving the problem for both solutions.

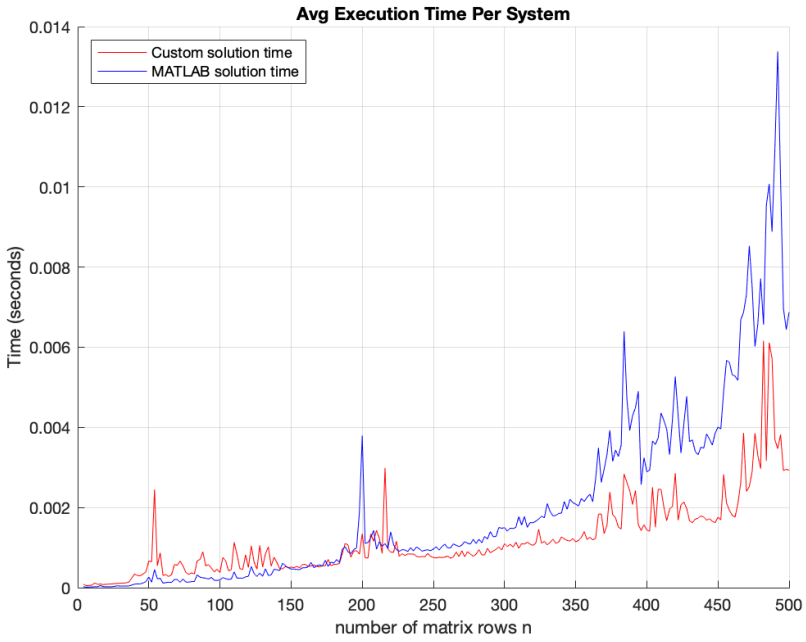

Figure 7: Average run times ($m = 1$)

Custom solution follows a similar trend as the solution using MATLAB's functions, however, is slightly faster.

7.2 Multiple systems ($m = O(n)$)

When the number of systems m is increased to $O(n)$ then the custom solution is slower than the solution using MATLAB's functions. The major spike for the custom solution happened on other runs, however, it was also present for the other solution as well. The results are shown on Figure 8 for $m = n$ and Figure 9 for $m = 2n$.

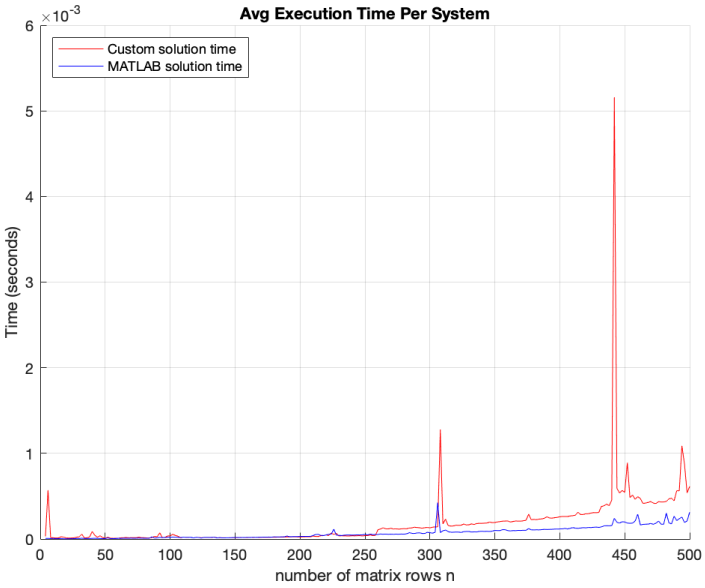

Figure 8: Average run times ($m = n$)

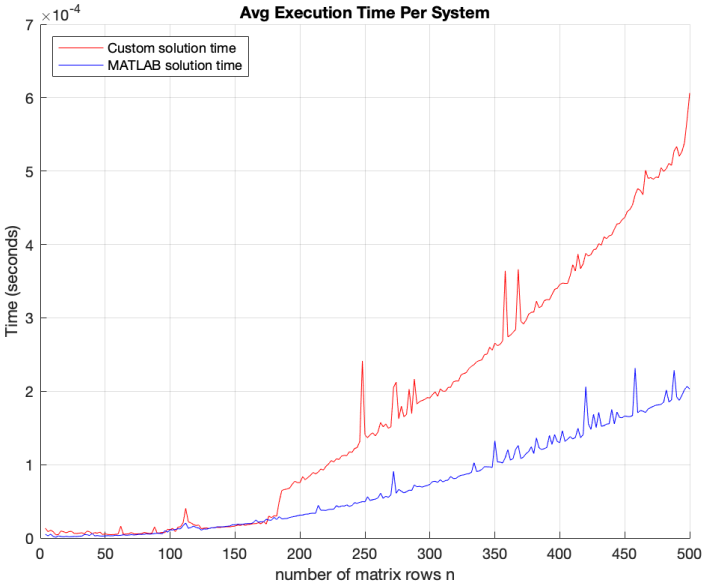

Figure 9: Average run times ($m = 2n$)

8 Conclusions

Custom solution is more accurate and has slightly better numerical properties when solving $AX = B$. For more systems the solution using MATLAB's built-in functions is faster and does not feature as big spikes for run times as the custom solution. The difference between computed determinants by MATLAB's ***det*** function and the custom one grows to absurd values and should not be used for large matrices.

References

[1] W. Cheney D. Kincaid. *Numerical Analysis: Mathematics of Scientific Computing, 3rd Edition.* Providence, RI: American Mathematical Society, 2002.