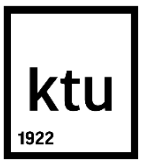


Kaunas University of Technology
Faculty of Informatics

Numerical Methods and Algorithms

Engeneering Project 2

Student name, surname, academical group

Student **Zahi El Helou**, IFU-1

Position

Instructor **KRIŠČIŪNAS Andrius**

Kaunas, 2023

Task:

1. **Linear equation systems:** Write program which solves system of linear equations (Gaussian algorithm for even task numbers / Reflection for odd). The program must work without crashes (errors or exceptions) with any equation provided in table 1. If it has infinite number of solutions, at least one of them should be provided. If there are no solutions, program must identify this case and output the message about situation. In the report provide algorithm description and at least 5 random systems of linear equations from table 1 solved including obtained results and its validation. (2 points)

2. **Non-linear equation system:** Graphically solve system of non-linear equations provided in table 2. In different graphics represent the surfaces $Z1$ ($x1$, $x2$) and $Z2$ ($x1$, $x2$). Solve the system of non-linear equations graphically and using Newtons (for odd task numbers) or Quasi-Newton (for even task numbers). Validate obtained results using external sources. (2 points)

3. **Optimization**: According to the task provided in Table 3, create the objective function which represents the goodness of the solution and solve it using gradient descend (even task numbers) or steepest gradient descend method (odd task numbers). Graphically visualize (animate point movements) optimization process of given and added points represented by different colors. In the report provide initial and final point configurations, analytical expression of objective function and graph how objective function depends on iteration. (~6 points)

1. **Linear equation systems:**

Task Number	Eq. system	Task number	Eq. system
1	$\begin{cases} x_1 - 2x_2 + 3\,x_3 + 4\,x_4 = 11 \\ x_1 - x_3 + x_4 = -4 \\ 2x_1 - 2x_2 + 2\,x_3 + 5x_4 = 7 \\ -7x_2 + 3\,x_3 + x_4 = 2 \end{cases}$	2	$\begin{cases} 3x_1 + 7x_2 + x_3 + 3\,x_4 = 37 \\ x_1 - 6x_2 + 6x_3 + 9x_4 = 11 \\ 4x_1 + 4x_2 - 7\,x_3 + x_4 = 38 \\ -x_1 + 3x_2 + 8x_3 + 2x_4 = -1 \end{cases}$
3	$\begin{cases} x_2 + 2x_3 + x_4 = 2 \\ 6x_1 - 2x_2 + 3x_3 + 4x_4 = -15 \\ 3x_2 + 4\,x_3 - 3\,x_4 = 10 \\ -4x_2 + 3\,x_3 + x_4 = -2 \end{cases}$	4	$\begin{cases} 3x_1 + 7x_2 + x_3 + 3\,x_4 = 40 \\ x_1 - 6x_2 + 6x_3 + 8\,x_4 = 19 \\ 4x_1 + 4x_2 - 7\,x_3 + x_4 = 36 \\ 4x_1 + 16x_2 + 2x_3 = 48 \end{cases}$
5	$\begin{cases} x_1 + 2x_2 + x_3 = -4 \\ 2x_1 + 5x_2 + 4\,x_4 = 3 \\ 14x_1 - 8x_2 + 4x_3 + x_4 = 7 \\ 4x_1 + 10x_2 + 8\,x_4 = 2 \end{cases}$	6	$\begin{cases} 3x_1 + x_2 - x_3 + 5x_4 = 20 \\ -3x_1 + 4x_2 - 8\,x_3 - x_4 = -36 \\ x_1 - 3x_2 + 7\,x_3 + 6x_4 = 41 \\ 5x_2 - 9\,x_3 + 4x_4 = -16 \end{cases}$

In that we will write a program which solves system of linear equations using Gaussian algorithm. We will provide the algorithm and at least 5 random systems of linear equations from the solved equations including obtained results and its validations.

The algorithm is explained:

```
eps = 1e-10
A1=np.hstack((A,b))  # expanding matrix
# Forward elimination step of Gaussian elimination
for i in range(0, n - 1):  # Loop over each row starting from the first to the penultimate
    for j in range(i + 1, n):  # Loop over each row below the current row i
        if np.abs(A1[i, i]) > eps:  # Check if the pivot element is not too small
            # Perform row operations to make elements below the pivot in the current column zero
            A1[j, i:n + nb] = A1[j, i:n + nb] - A1[i, i:n + nb] * A1[j, i] / A1[i, i]
            A1[j, i] = 0  # Set the element below the pivot to zero after the operation
    print(A1); print(" ");


print(A1)

# Backward substitution step of Gaussian elimination
x = np.zeros(shape=(n, nb))  # Initialize the solution vector x with zeros
for i in range(n - 1, -1, -1):  # Loop from the last row to the first row
    if np.abs(A1[i, i]) <= eps:  # Check if the diagonal element is effectively zero
        if np.abs(A1[i, n - 1]) <= eps:  # Check if the corresponding b element is effectively zero
            if np.abs(A1[i, n]) > eps:  # Inconsistent system check
                print("No solution")
            else:  # System has many solutions
                print("Many solutions")
            continue
    if np.abs(A1[i, i]) > eps:  # If the diagonal element is not too small
        # Calculate the solution for the current variable
        x[i, :] = (A1[i, n:n + nb] - A1[i, i + 1:n] * x[i + 1:n, :]) / A1[i, i]
        print(x); print(" ");

# Print the solution vector x
print(x)
```

The code snippet provided is an implementation of the Gaussian elimination algorithm with partial pivoting in Python, using the NumPy library. Gaussian elimination is a method for solving systems of linear equations. It consists of two main steps: forward elimination and backward substitution.

In the forward elimination phase, the algorithm transforms the matrix of coefficients into an upper triangular form by subtracting multiples of the pivot row from the rows below it. This is done to create zeros below the diagonal, effectively simplifying the system into one that is easier to solve.

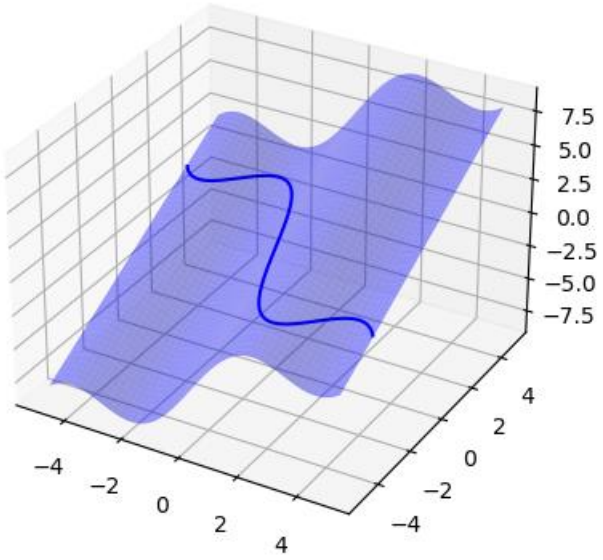
The backward substitution phase then solves for the unknowns by starting from the last row and substituting the known values back into the previous rows. The code handles cases where the matrix may have no solution or many solutions by checking if the diagonal elements are zero after the forward elimination phase. It also includes a check for numerical stability by introducing a small threshold ‘`eps`’ to avoid division by numbers that are effectively zero. The result is a vector of solutions for the system of equations, which is then validated by multiplying the original coefficient matrix by the solution vector and comparing it to the original constants vector.

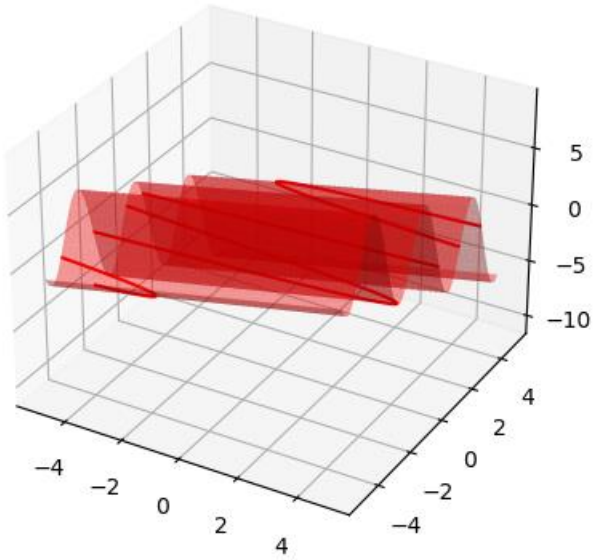
1st eq. system	2nd eq. system	3rd eq. system	4th eq. system	5th eq. system
[[1. -2. 3. 4. 11.] [1. 0. -1. 1. -4.] [2. -2. 2. 5. 7.] [0. -7. 3. 1. 2.]] [[1. -2. 3. 4. 11.] [0. 2. -4. -3. -15.] [0. 2. -4. -3. -15.] [0. -7. 3. 1. 2.]] [[1. -2. 3. 4. 11.] [0. 2. -4. -3. -15.] [0. 0. 0. 0. 0.] [0. 0. -11. -9.5 -50.5]] [[1. -2. 3. 4. 11.] [0. 2. -4. -3. -15.] [0. 0. 0. 0. 0.] [0. 0. -11. -9.5 -50.5]] [[1. -2. 3. 4. 11.] [0. 2. -4. -3. -15.] [0. 0. 0. 0. 0.] [0. 0. -11. -9.5 -50.5]] [[0.] [0.] [0.] [5.31578947]] Many solutions [[0.] [0.47368421] [0.] [5.31578947]] [[-9.31578947] [0.47368421] [0.] [5.31578947]] [[-9.31578947] [0.47368421] [0.] [5.31578947]] Solution: [[-9.31578947] [0.47368421] [0.] [5.31578947]] Ax-b: Validation [[11.] [-4.] [7.] [2.]]	[[3 7 1 3 37] [1 -6 6 9 11] [4 4 -7 1 38] [-1 3 8 2 -1]] [[3 7 1 3 37] [0 -8 5 8 -1] [0 -5 -8 -3 -11] [0 5 8 3 11]] [[3 7 1 3 37] [0 -8 5 8 -1] [0 0 -11 -8 -10] [0 0 11 8 10]] [[3 7 1 3 37] [0 -8 5 8 -1] [0 0 -11 -8 -10] [0 0 0 0 0]] [[3 7 1 3 37] [0 -8 5 8 -1] [0 0 -11 -8 -10] [0 0 0 0 0]] Many solutions [[0.] [0.] [0.90909091] [0.]] [[0.] [0.69318182] [0.90909091] [0.]] [[10.41287879] [0.69318182] [0.90909091] [0.]] [[10.41287879] [0.69318182] [0.90909091] [0.]] Solution: [[10.41287879] [0.69318182] [0.90909091] [0.]] Ax-b: Validation [[37.] [11.70833333] [38.06060606] [-1.06060606]]	[[0 1 2 1 2] [6 -2 3 4 -15] [0 3 4 -3 10] [0 -4 3 1 -2]] [[0 1 2 1 2] [6 -2 3 4 -15] [0 3 4 -3 10] [0 -4 3 1 -2]] [[0 1 2 1 2] [6 -2 3 4 -15] [0 0 8 3 -12] [0 0 -3 -7 28]] [[0 1 2 1 2] [6 -2 3 4 -15] [0 0 8 3 -12] [0 0 0 -5 23]] [[0 1 2 1 2] [6 -2 3 4 -15] [0 0 8 3 -12] [0 0 0 -5 23]] [[0.] [0.] [0.] [-4.6]] [[0.] [0.] [0.225] [-4.6]] [[0.] [-1.3625] [0.225] [-4.6]] [[0.] [-1.3625] [0.225] [-4.6]] Solution: [[0.] [-1.3625] [0.225] [-4.6]] Ax-b: Validation [[-5.5125] [-15.] [10.6125] [1.525]]	[[3 7 1 3 40] [1 -6 6 8 19] [4 4 -7 1 36] [4 16 2 0 48]] [[3 7 1 3 40] [0 -8 5 7 5] [0 -5 -8 -3 -17] [0 6 0 -4 -5]] [[3 7 1 3 40] [0 -8 5 7 5] [0 0 -11 -7 -20] [0 0 3 1 -1]] [[3 7 1 3 40] [0 -8 5 7 5] [0 0 -11 -7 -20] [0 0 0 0 -6]] [[3 7 1 3 40] [0 -8 5 7 5] [0 0 -11 -7 -20] [0 0 0 0 -6]] No solution [[0.] [0.] [1.81818182] [0.]] [[0.] [0.51136364] [1.81818182] [0.]] [[11.53409091] [0.51136364] [1.81818182] [0.]] [[11.53409091] [0.51136364] [1.81818182] [0.]] Solution: [[11.53409091] [0.51136364] [1.81818182] [0.]] Ax-b: Validation [[40.] [19.375] [35.45454545] [57.95454545]]	[[1 2 1 0 -4] [2 5 0 4 3] [14 -8 4 1 7] [4 10 0 8 2]] [[1 2 1 0 -4] [0 1 -2 4 11] [0 -36 -10 1 63] [0 2 -4 8 18]] [[1 2 1 0 -4] [0 1 -2 4 11] [0 0 -82 145 459] [0 0 0 0 -4]] [[1 2 1 0 -4] [0 1 -2 4 11] [0 0 -82 145 459] [0 0 0 0 -4]] [[1 2 1 0 -4] [0 1 -2 4 11] [0 0 -82 145 459] [0 0 0 0 -4]] No solution [[0.] [0.] [-5.59756098] [0.]] [[0.] [-0.19512195] [-5.59756098] [0.]] [[1.98780488] [-0.19512195] [-5.59756098] [0.]] [[1.98780488] [-0.19512195] [-5.59756098] [0.]] Solution: [[1.98780488] [-0.19512195] [-5.59756098] [0.]] Ax-b: Validation [[-4.] [3.] [7.] [6.]]

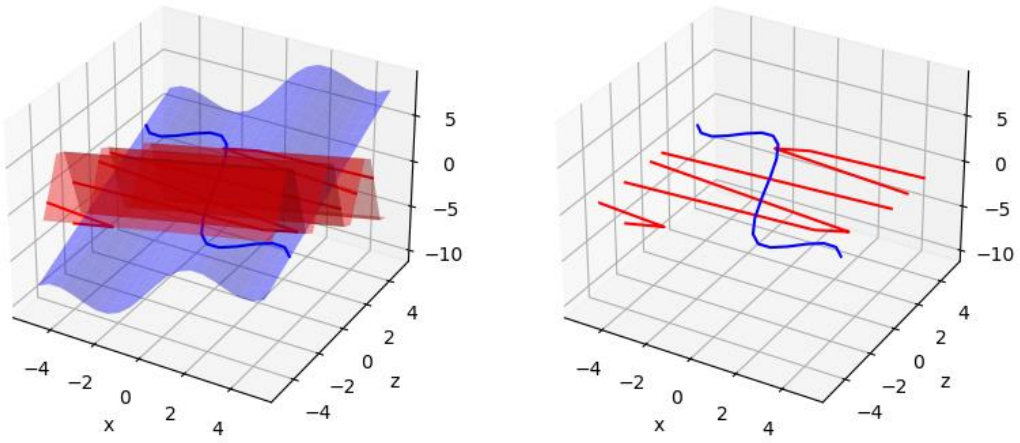
2. **Non-linear equation system**

6	$\begin{cases} 2\sin(x_1) + x_1 + x_2 = 0 \\ 4\cos(2x_2) - x_2 + 0.5x_1 = 0 \end{cases}$

Now, we will graphically solve the system of non-linear equations:







In the 2nd part of Part 2, we have to do the quasi newton for a non-linear equation

Input interpretation

solve

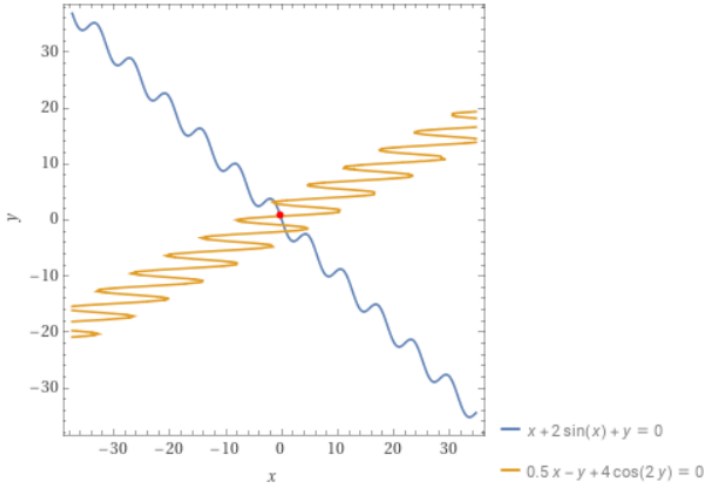
$x + y + 2\sin(x) = 0$

$0.5\,x - y + 4\cos(2\,y) = 0$

Result

$x = -0.229599$ and $y = 0.684774$

Implicit plot



Download Page

POWERED BY THE **WOLFRAM LANGUAGE**