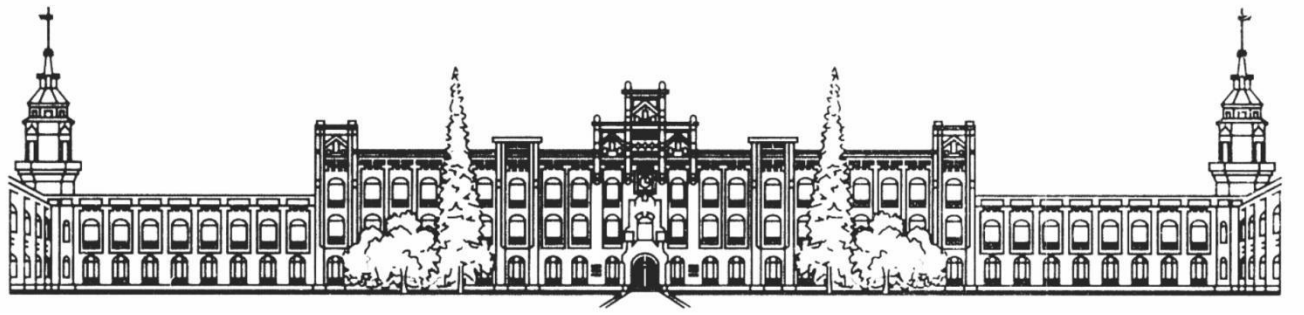

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського» Інститут прикладного системного аналізу

**Лабораторна робота №2**

курсу «Чисельні методи 1»

з теми «Прямі методи розв’язання СЛАР»

Варіант №9

Виконав студент 2 курсу групи КА-91
Косицький Вадим Вікторович
перевірила старший викладач
Хоменко Ольга Володимирівна

Київ-2021

## Завдання 1

1. СЛАР завдання 1 розв'язати методом Гаусса з вибором головного елементу. Для цього реалізувати обраний метод для довільної СЛАР розмірності n×n. Розв'язати СЛАР з точністю $\varepsilon = 10^{-5}$ (кількість знаків після коми 5).
2. Обчислити **det A**.

## Завдання 2

1. СЛАР завдання 2 розв'язати методом прогонки, для чого написати відповідну програму для довільної СЛАР розмірності n×n. Розв'язати СЛАР з точністю $\varepsilon = 10^{-5}$ (кількість знаків після коми 5).

**№ 9.**

$$\begin{cases} 1{,}7x_1 - 1{,}8x_2 + 1{,}9x_3 - 57{,}4x_4 = 10, \\ 1{,}1x_1 - 4{,}3x_2 + 1{,}5x_3 - 1{,}7x_4 = 19, \\ 1{,}2x_1 + 1{,}4x_2 + 1{,}6x_3 + 1{,}8x_4 = 20, \\ 7{,}1x_1 - 1{,}3x_2 - 4{,}1x_3 + 5{,}2x_4 = 10. \end{cases}$$

**A =**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 172 | -82 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 232 | -912 | -234 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 22 | -115 | -16 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | -50 | 473 | 87 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | -23 | 408 | -196 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | -26 | -338 | -77 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 195 | -606 | -42 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | -84 | 776 | 209 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 64 | -207 | 53 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -139 | -245 |

**b =**

| |
|---|
| 581 |
| 899 |
| -345 |
| 343 |
| -123 |
| 667 |
| 538 |
| -666 |
| 724 |
| 980 |

## Текст програми:

### Файл main.py (Метод Гауса з вибором головного елемента)

```
from func import *

eps = 0.0001

#main
n, data = input_data()

print("Gauss")
s1 = Gauss_main_element(data.copy(),n, eps )
s1.pop()
verification(data.copy(), s1,n)
```

### Файл main.py (Метод Гауса з вибором головного елемента)

```
from func import *

eps = 0.0001

#main
n, data = input_data()

print("Progonka")
s2 = progonka(data.copy(),n, eps)
s2.pop()
verification(data.copy(), s2,n)
```

### Файл func.py (Тут реалізовано усі потрібні алгоритми)

```
from numpy import *
eps1 = 0.0001


def input_data():
    print("Enter number of equations: ")
    n = int(input())
    print("Enter koef of  equations: ")
    data = []
    for j in range(n):
        data.append([float(i) for i in input().split()])
        pass
    return n, data

def Gauss(data, n, eps):
    # Forward
    det = 1
    solution1 = []
    key1 = 1
    problem_equations = []
    output(data, n)
    for k in range(n - 1):
        for i in range(k + 1, n):
            multiplier = data[i][k] / data[k][k]
            data[i][k] = 0
            for j in range(k + 1, n + 1):
                data[i][j] -= data[k][j] * multiplier
            if (abs(data[i][k + 1]) < eps):
                problem_equations.append(i)
```

```
                pass
            pass
        # solve problem of 0
        if (len(problem_equations) == n - k - 1):
            print("Infinity solution")
            key1 = 0
            break
        for i in problem_equations:
            data.append(data[i])
            data.pop(i)
            pass
        output(data, n)
        pass
    for i in range(n):
        det *= data[i][i]
    # Back
    if (key1):
        solution1.append(data[n - 1][n] / data[n - 1][n - 1])
        for k in range(1, n + 1):
            sum = data[n - k - 1][n]
            for i in range(k):
                sum -= data[n - k - 1][n - 1 - i] * solution1[i]
            solution1.append(sum / data[n - 1 - k][n - 1 - k])

        solution1.pop(n)
        solution1.reverse()
        solution1.append(det)
        print("          ", end="")
        for i in range(n):
            print("x", i + 1, end="      ")
            pass
        print("det")
        print("Solution: ", end="")
        for i in solution1:
            print("%.5f" % i, end=" ")
            pass
        print("")

        pass
    return solution1


def Gauss_main_element(data, n, eps):
    det = 1
    permutation = 0
    solution2 = []
    key2 = 1
    problem_equations2 = []
    output(data, n)
    # Forward
    for k in range(n - 1):
        max = k
        for i in range(k + 1, n):
            if (abs(data[max][k]) < abs(data[i][k])): max = i
        if (max != k):
            data.append(data[k])
            data[k] = data[max]
            data[max] = data[-1]
            data.pop()
            permutation += 1
            pass
        output(data, n)
```

```
        for i in range(k + 1, n):
            multiplier = data[i][k] / data[k][k]
            data[i][k] = 0
            for j in range(k + 1, n + 1):
                data[i][j] -= data[k][j] * multiplier
            if (abs(data[i][k + 1]) < eps):
                problem_equations2.append(i)
                pass
            pass
        # solve problem of 0
        if (len(problem_equations2) == n - k - 1):
            print("Infinity solution")
            key2 = 0
            break
        for i in problem_equations2:
            data.append(data[i])
            data.pop(i)
            pass
        output(data, n)

        pass
    for i in range(n):
        det *= data[i][i]
    if(permutation%2 != 0): det *= -1
    # Back
    if (key2):
        solution2.append(data[n - 1][n] / data[n - 1][n - 1])
        for k in range(1, n + 1):
            sum = data[n - k - 1][n]
            for i in range(k):
                sum -= data[n - k - 1][n - 1 - i] * solution2[i]
            solution2.append(sum / data[n - 1 - k][n - 1 - k])

        solution2.pop(n)
        solution2.reverse()
        solution2.append(det)
        print("          ", end="")
        for i in range(n):
            print("x",i+1,end="      ")
            pass
        print("det")
        print("Solution: ", end="")
        for i in solution2:
            print("%.5f" % i, end=" ")
        print("")
        pass
    return solution2


def progonka (data, n, eps):
    det = 1
    #verification of sufficient conditions
    key = 1
    if(abs(data[0][0]) <= abs(data[0][1]) or abs(data[n-1][n-1]) <= abs(data[n-1][n-2])):
key = 0
    for i in range(1,n-1):
        if((abs(data[i][i]) <= abs(data[i][i-1]) + abs(data[i][i+1])) or (data[i][i-1] ==
0) or (data[i][i+1] == 0)): key = 0

    if (key):
        #forward propagation
        alfa = []
        beta = []
```

```
        alfa.append(data[0][n]/data[0][0])
        beta.append(-data[0][1]/data[0][0])
        for i in range(1,n-1):
            beta.append(-data[i][i+1]/(data[i][i]+data[i][i-1]*beta[i-1]))
            alfa.append((data[i][n]-data[i][i-1]*alfa[i-1])/(data[i][i]+data[i][i-
1]*beta[i-1]))
        beta.append(0)
        alfa.append((data[n-1][n]-data[n-1][n-2]*alfa[n-2])/(data[n - 1][n - 1]+data[n-
1][n-2]*beta[n-2]))

        #back propagation
        solution = []
        solution.append(alfa[n-1])
        for i in range(1,n+1):
            solution.append(beta[n-i]*solution[i-1]+alfa[n-i])
            pass
        solution.reverse()
        solution.pop()
        print("Solution: ", end="")
        for i in solution:
            print("%.5f" % i, end=" ")
    else:
        print("\n verification of sufficient conditions is false")
    return solution

def output(data, n):
    Data = data.copy()
    for i in range(n):
        print("%+.7F  %+.7F  %+.7F  %+.7F  %+.7F  " % tuple(data[i]))
    print("\n")



def verification (data, solution, n):
    b = []
    for i in data:
        b.append(i[n])
        i.pop()
        pass
    d = array(data, float)
    s = array(solution, float)
    s.reshape(n,1)
    b = array(b)
    b.reshape(n,1)
    if (all((dot(d,s) - b < eps1))):
        print("\nVerification is done")
    else:
        print("Verification is not successfull")

    pass
```

**Результат роботи програми (Метод Гауса з вибором головного елементу)**

C:\Users\kosva\Desktop\ЧМ\venv\Scripts\python.exe C:/Users/kosva/Desktop/ЧМ/main.py

Enter number of equations:

4

Enter koef of  equations:

1.7 -1.8 1.9 -57.4 10

1.1 -4.3 1.5 -1.7 19

1.2 1.4 1.6 1.8 20

7.1 -1.3 -4.1 5.2 10

Gauss

+1.7000000  -1.8000000  +1.9000000  -57.4000000  +10.0000000

+1.1000000  -4.3000000  +1.5000000  -1.7000000  +19.0000000

+1.2000000  +1.4000000  +1.6000000  +1.8000000  +20.0000000

+7.1000000  -1.3000000  -4.1000000  +5.2000000  +10.0000000

+7.1000000  -1.3000000  -4.1000000  +5.2000000  +10.0000000

+1.1000000  -4.3000000  +1.5000000  -1.7000000  +19.0000000

+1.2000000  +1.4000000  +1.6000000  +1.8000000  +20.0000000

+1.7000000  -1.8000000  +1.9000000  -57.4000000  +10.0000000

+7.1000000  -1.3000000  -4.1000000  +5.2000000  +10.0000000

+0.0000000  -4.0985915  +2.1352113  -2.5056338  +17.4507042

+0.0000000  +1.6197183  +2.2929577  +0.9211268  +18.3098592

+0.0000000  -1.4887324  +2.8816901  -58.6450704  +7.6056338

+7.1000000 -1.3000000 -4.1000000 +5.2000000 +10.0000000
+0.0000000 -4.0985915 +2.1352113 -2.5056338 +17.4507042
+0.0000000 +1.6197183 +2.2929577 +0.9211268 +18.3098592
+0.0000000 -1.4887324 +2.8816901 -58.6450704 +7.6056338

+7.1000000 -1.3000000 -4.1000000 +5.2000000 +10.0000000
+0.0000000 -4.0985915 +2.1352113 -2.5056338 +17.4507042
+0.0000000 +0.0000000 +3.1367698 -0.0690722 +25.2061856
+0.0000000 +0.0000000 +2.1061168 -57.7349485 +1.2670103

+7.1000000 -1.3000000 -4.1000000 +5.2000000 +10.0000000
+0.0000000 -4.0985915 +2.1352113 -2.5056338 +17.4507042
+0.0000000 +0.0000000 +3.1367698 -0.0690722 +25.2061856
+0.0000000 +0.0000000 +2.1061168 -57.7349485 +1.2670103

+7.1000000 -1.3000000 -4.1000000 +5.2000000 +10.0000000
+0.0000000 -4.0985915 +2.1352113 -2.5056338 +17.4507042
+0.0000000 +0.0000000 +3.1367698 -0.0690722 +25.2061856
+0.0000000 +0.0000000 +0.0000000 -57.6885714 -15.6571429

x 1 x 2 x 3 x 4 det

Solution: 5.81058 -0.23424 8.04169 0.27141 -5265.81280

Verification is done

Process finished with exit code 0

**Результат роботи програми (Метод прогонки)**

C:\Users\kosva\Desktop\ЧМ\venv\Scripts\python.exe C:/Users/kosva/Desktop/ЧМ/main.py

Enter number of equations:

10

Enter koef of  equations:

172 -82 0 0 0 0 0 0 0 0 581

232 -912 -234 0 0 0 0 0 0 0 899

0 22 -115 -16 0 0 0 0 0 0 -345

0 0 -50 473 87 0 0 0 0 0 343

0 0 0 -23 408 -196 0 0 0 0 -123

0 0 0 0 -26 -338 -77 0 0 0 667

0 0 0 0 0 195 -606 -42 0 0 538

0 0 0 0 0 0 -84 776 209 0 -666

0 0 0 0 0 0 0 64 -207 53 724

0 0 0 0 0 0 0 0 -139 -245 980

Progonka

Solution: 2.93921 -0.92018 2.65857 1.18877 -0.99265 -1.57828 -1.39912 0.05003 -3.93470 -1.76766

Verification is done

## Висновок:

Виконуючи дану лабораторну роботу я навчився застосовувати прямі чисельні методи розв’язання СЛАР. Я реалізував 3 алгоритми: метод Гауса, метод Гауса з постовпчиковим вибором головного елементу і метод прогонки.

Усі алгоритми я реалізував у файлі func.py, задля більшої читабельності коду та ненагромадження головного файлу main.py.

Під час виконання методу Гауса з вибором головного елемента, після кожної перестановки рядків та зміни рядка (додавання домноженого на коефіцієнт головного рядка) я виводив поточний стан матриці коефіцієнтів. У кінці вивів рішення системи разом із порахованим детермінантом матриці. Після чого підставив розв’язки у систему та перевірив, щоб усі елементи отриманого вектору були меншими за епсилон)

Результати: 5.81058,  -0.23424,  8.04169, 0.27141

Det =  -5265.81280

Під час виконання методу прогонки я здійснив аналогічну перевірку як із методом Гауса.

Результати:  2.93921,  -0.92018, 2.65857, 1.18877, -0.99265, -1.57828, -1.39912, 0.05003,  -3.93470, -1.76766