Numerical Methods

Report

Date of the exercise: **21/03/2019**

Exercise: **Solving systems of linear equations**

Group: 2, Team:

Subsection (names):

1. Mateusz Nowotnik

2. Dawid Tomala

Task 1 – Implementation of the relaxation method

```
#include <iostream>
#include <math.h>
using namespace std;

double fabsMax(double a, double b, double c)
{
    a = fabs(a); b = fabs(b); c = fabs(c);
    double maxim = fmax(a, b);
    return fmax(maxim, c);
}

void convTest(double a[3][4])
{
    if (a[0][0] < a[0][1] + a[0][2]) {throw "Convergence Test Failed";}
    if (a[1][1] < a[1][0] + a[1][2]) {throw "Convergence Test Failed";}
    if (a[2][2] < a[2][0] + a[2][1]) {throw "Convergence Test Failed";}
}

int main()
{
    int k, n, xIndex[3];
    double e, a[3][4], maxVal[3], temp;
    double x[3], prevX[3], r[3], maxVal2, tempNum;
    bool stopAcc = false;

    cout << "This program works for system of 3 linear equations of 3 variables [in the form: a1*x1 + a2*x2 + a3*x3 = d]" << endl;
    cout << "Form of the input: a1 a2 a3 d" << endl;

    cout << "Type in coefficients and result of every equation: " << endl;
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 4; j++)
            cin >> a[i][j];

    cout << "Give the accuracy: ";
    cin >> e;

    for (int i = 0; i < 3; i++) {
        maxVal[i] = fabsMax(a[0][i], a[1][i], a[2][i]); // find max value of i-th column and store it in the maxVal array
        for (int j = 0; j < 3; j++) {
            if (maxVal[i] == fabs(a[j][i])) {
                if (i != j) { // swap rows if the max value is not on the diagonal
                    for (int k = 0; k < 4; k++) {
                        temp = a[i][k];
                        a[i][k] = a[j][k];
                        a[j][k] = temp;
                    }
                }
            }
        }
    }

    convTest(a); // check the convergence


    for (int i = 0; i < 3; i++) {
        xIndex[i] = 0;
        x[i] = 0;
        prevX[i] = 0;
```

```
59.
60.         r[i] = a[i][3];
61.     }
62.     n = 0;
63.
64.     while (stopAcc == false) {
65.         n++;
66.         maxVal2 = fabsMax(r[0], r[1], r[2]);
67.         for (int i = 0; i < 3; i++) {
68.             if (maxVal2 == fabs(r[i])) {
69.                 tempNum = r[i]/maxVal[i]; // calculate the approximate x[i]
70.                 x[i] += tempNum;
71.                 if (fabs(tempNum - prevX[i]) < e) {
72.                     // stop if the epsilon is bigger than the assumed accuracy
73.                     stopAcc = true;
74.                     break;
75.                 }
76.                 prevX[i] = tempNum;
77.                 for (int j = 0; j < 3; j++)
78.                     r[j] += -(a[j][i]*prevX[i]); // calculate the residuals
79.             }
80.         }
81.     }
82.
83.     cout << "\nSolution:" << endl;
84.     cout << "x1" << " = " << x[0] << endl;
85.     cout << "x2" << " = " << x[1] << endl;
86.     cout << "x3" << " = " << x[2] << endl;
87.     cout << "# of iterations: " << n << endl;
88.
89.     return 0;
90. }
```

To check the convergence of the system, we had to check if the absolute value of the diagonal coefficient in each of the equations is larger than the sum of the absolute values of the other coefficients in the equation.

Solving the given system of linear equations:

$$9x_1 - 2x_2 + x_3 = 50$$
$$-2x_1 + 2x_2 + 7x_3 = 19$$
$$x_1 + 5x_2 - 3x_3 = 18$$

Input to the program:

```
9 -2 1 50
-2 2 7 19
1 5 -3 18
   0.01
```

The input/output from the console:

```
This program works for system of 3 linear equations of 3 variables
Form of the input: a1 a2 a3 d
Type in coefficients and result of every equation:
9 -2 1 50
-2 2 7 19
1 5 -3 18
Give the accuracy: 0.01

Solution:
x1 = 6.15451
x2 = 4.31497
x3 = 3.23937
# of iterations: 23
```

Obtained solution:

$$x_1 = 6.15451$$
$$x_2 = 4.31497$$
$$x_3 = 3.23937$$

Real solution:

$$x_1 = 2357/383 \cong 6.15405$$
$$x_2 = 1652/383 \cong 4.31332$$
$$x_3 = 1241/383 \cong 3.24021$$

Task 2 – Implementation of the Gauss method

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int determinant(double mat[3][4]) {
5.     int det = 0;
6.     det=mat[0][0]*mat[1][1]*mat[2][2]+mat[0][1]*mat[1][2]*mat[2][0]+mat[0][2]*mat[1][
    0]*mat[2][1]-
    (mat[2][0]*mat[1][1]*mat[0][2]+mat[2][1]*mat[1][2]*mat[0][0]+mat[2][2]*mat[1][0]*mat
    [0][1]);
7.     return det;
8.  }
9.  int main ()
10. {
11.     double a[3][4], x[3], s;
12.     cout << "This program works for system of 3 linear equations of 3 variables" <<
    endl;
13.     cout << "Form of the input: a1 a2 a3 d" << endl;
14.
15.     cout << "Type in coefficients and result of every equation: " << endl;
16.     for (int i = 0; i < 3; i++)
17.         for (int j = 0; j < 4; j++)
18.             cin >> a[i][j];
19.
20.     x[0] = 0;
21.     x[1] = 0;
22.     x[2] = 0;
23.
```

```
    if ((determinant(a))==0) {
        cout << "The matrix is singular ==> can't be solved by Gaussian Elimination"
;
        return 0;
    }

    for (int k = 0; k < 2; k++) {
        for (int i=k+1; i < 3; i++) {
            for (int j=k+1; j < 3; j++) {
                a[i][j] -= ((a[i][k]*a[k][j])/a[k][k]);
            }
        a[i][3] -= ((a[i][k]*a[k][3])/a[k][k]);
        }
    }


    x[2] = a[2][3]/a[2][2];

    for (int i = 2; i > -1; i--) {
        s = 0;
        for (int j=i+1; j < 3; j++) {
            s += a[i][j]*x[j];
        }
        x[i] = (a[i][3]-s)/a[i][i];
    }

    cout << endl;
    for (int m = 0; m < 3; m++) {
        cout << "x" << m+1 << " = " << x[m] << endl;
    }

    return 0;
}
```

We are choosing to solve the same system of linear equations.

The input/output from the console:

```
This program works for system of 3 linear equations of 3 variables
Form of the input: a1 a2 a3 d
Type in coefficients and result of every equation:
9 -3 2 50
-2 2 7 19
1 5 -3 18

x1 = 6.26515
x2 = 4.31061
x3 = 3.27273
```

Obtained solution:

$$x_1 = 6.26515$$
$$x_2 = 4.31061$$
$$x_3 = 3.27273$$