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

1. #include <iostream>
2. #include <math.h>
3. **using** **namespace** std;
5. **double** fabsMax(**double** a, **double** b, **double** c)
6. {
7. a = fabs(a); b = fabs(b); c = fabs(c);
8. **double** maxim = fmax(a, b);
9. **return** fmax(maxim, c);
10. }
12. **void** convTest(**double** a[3][4])
13. {
14. **if** (a[0][0] < a[0][1] + a[0][2]) {**throw** "Convergence Test Failed";}
15. **if** (a[1][1] < a[1][0] + a[1][2]) {**throw** "Convergence Test Failed";}
16. **if** (a[2][2] < a[2][0] + a[2][1]) {**throw** "Convergence Test Failed";}
17. }
19. **int** main()
20. {
21. **int** k, n, xIndex[3];
22. **double** e, a[3][4], maxVal[3], temp;
23. **double** x[3], prevX[3], r[3], maxVal2, tempNum;
24. **bool** stopAcc = **false**;
26. cout << "This program works for system of 3 linear equations of 3 variables [in the form: a1\*x1 + a2\*x2 + a3\*x3 = d]" << endl;
27. cout << "Form of the input: a1 a2 a3 d" << endl;
29. cout << "Type in coefficients and result of every equation: " << endl;
30. **for** (**int** i = 0; i < 3; i++)
31. **for** (**int** j = 0; j < 4; j++)
32. cin >> a[i][j];
34. cout << "Give the accuracy: ";
35. cin >> e;
37. **for** (**int** i = 0; i < 3; i++) {
38. 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
39. **for** (**int** j = 0; j < 3; j++) {
40. **if** (maxVal[i] == fabs(a[j][i])) {
41. **if** (i != j) { // swap rows if the max value is not on the diagonal
42. **for** (**int** k = 0; k < 4; k++) {
43. temp = a[i][k];
44. a[i][k] = a[j][k];
45. a[j][k] = temp;
46. }
47. }
48. }
49. }
50. }
52. convTest(a); // check the convergence
54. **for** (**int** i = 0; i < 3; i++) {
55. xIndex[i] = 0;
56. x[i] = 0;
57. prevX[i] = 0;
59. r[i] = a[i][3];
60. }
61. n = 0;
63. **while** (stopAcc == **false**) {
64. n++;
65. maxVal2 = fabsMax(r[0], r[1], r[2]);
66. **for** (**int** i = 0; i < 3; i++) {
67. **if** (maxVal2 == fabs(r[i])) {
68. tempNum = r[i]/maxVal[i]; // calculate the approximate x[i]
69. x[i] += tempNum;
70. **if** (fabs(tempNum - prevX[i]) < e) {
71. // stop if the epsilon is bigger than the assumed accuracy
72. stopAcc = **true**;
73. **break**;
74. }
75. prevX[i] = tempNum;
76. **for** (**int** j = 0; j < 3; j++)
77. r[j] += -(a[j][i]\*prevX[i]); // calculate the residuals
78. }
79. }
80. }
82. cout << "\nSolution:" << endl;
83. cout << "x1" << " = " << x[0] << endl;
84. cout << "x2" << " = " << x[1] << endl;
85. cout << "x3" << " = " << x[2] << endl;
86. cout << "# of iterations: " << n << endl;
88. **return** 0;
89. }

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:

Input to the program:

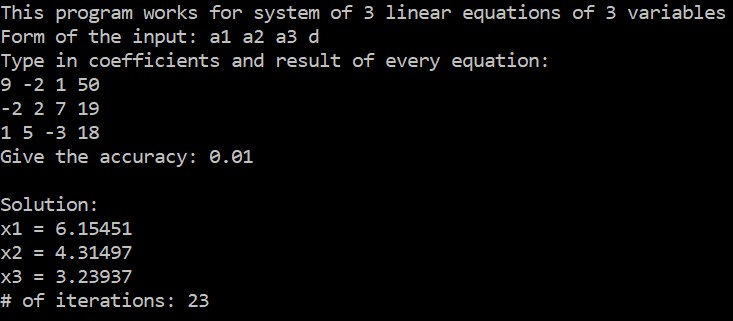
9 -2 1 50

-2 2 7 19

1 5 -3 18

0.01

The input/output from the console:



Obtained solution:

Real solution:

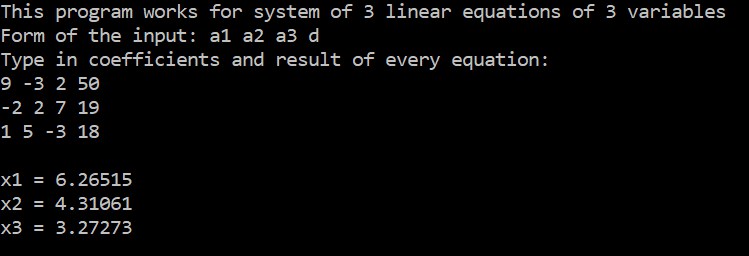
**Task 2 –** Implementation of the Gauss method

1. #include <iostream>
2. **using** **namespace** std;
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;
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];
20. x[0] = 0;
21. x[1] = 0;
22. x[2] = 0;
24. **if** ((determinant(a))==0) {
25. cout << "The matrix is singular ==> can't be solved by Gaussian Elimination";
26. **return** 0;
27. }
29. **for** (**int** k = 0; k < 2; k++) {
30. **for** (**int** i=k+1; i < 3; i++) {
31. **for** (**int** j=k+1; j < 3; j++) {
32. a[i][j] -= ((a[i][k]\*a[k][j])/a[k][k]);
33. }
34. a[i][3] -= ((a[i][k]\*a[k][3])/a[k][k]);
35. }
36. }

39. x[2] = a[2][3]/a[2][2];
41. **for** (**int** i = 2; i > -1; i--) {
42. s = 0;
43. **for** (**int** j=i+1; j < 3; j++) {
44. s += a[i][j]\*x[j];
45. }
46. x[i] = (a[i][3]-s)/a[i][i];
47. }
49. cout << endl;
50. **for** (**int** m = 0; m < 3; m++) {
51. cout << "x" << m+1 << " = " << x[m] << endl;
52. }
54. **return** 0;
55. }

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

The input/output from the console:



Obtained solution: