**EEE 206 - PROGRAMMING**

**Project 3 Report**

**Linear Equations**

Submitted by:

*Batuhan Yaman (EEE)*

*Elif Hürcan (EEE)*

*Özge Yılmaz (EEE)*

Project Mentors:

Dr. Doğan Çörüş

Faculty of Engineering and Natural Sciences

Kadir Has University

Spring 2022

**TABLE OF CONTENTS**

[1 INTRODUCTION 3](#_Toc104717596)

[1.1 Project Description of Level 1 3](#_Toc104717597)

[1.2 Project Description of Level 2 3](#_Toc104717598)

[1.3 Project Description of Bonus Part 3](#_Toc104717599)

[2 CREATING THE PROGRAM 4](#_Toc104717600)

[3 SIMULATIONS AND VERIFICATION 8](#_Toc104717601)

[3.1 Running the Program with Different Equations 8](#_Toc104717602)

[3.2 Solving the System and Comparing the Results 9](#_Toc104717603)

[4 CONCLUSION 13](#_Toc104717604)

[REFERENCES 14](#_Toc104717605)

[APPENDIX A: THE C SCRIPT 15](#_Toc104717606)

# **1 INTRODUCTION**

Systems of linear equations are used in almost all applied sciences, such as mathematics, physics, and circuit analysis [1]. Systems of linear equations have many solutions, such as matrix, and elimination methods. Just as we can solve a simple mathematical operation with the help of coding, linear equation systems are solved according to the determined solution method.

## **1.1 Project Description of Level 1**

In this project, we aimed to create a program for solving a system of linear equations for the level 1 step. We are free to choose how the equations are entered into the system by the user. Our source code needs to have hard-coded equations to solve, and the source code must be easily modified to handle any system of linear equations. Our program should report the value of each variable.

## **1.2 Project Description of Level 2**

The system of equations to be solved at this level will be provided and included in a separate file. Each line of the file is expected to contain a single equation. In this step, what is required of us is that the program we have written reads the file, decodes the relevant system, and reports the values ​​of each variable.

## **1.3 Project Description of Bonus Part**

Unlike the previous steps, in this step, our program is expected to be able to handle equations with and without multiple solutions in linear solution, and it should use a recursive function to solve linearly. Although our algorithm can fulfill the requirements that are given in steps 1 and step 2, it is not able to implement the bonus part, but only understanding the system has no solutions.

# **2 CREATING THE PROGRAM**

We created our program that can calculate the 2 or 3 unknown variables of the linear equation systems by using the Dev-C++ program. We should mention that even though the name of the program includes C++, the program is also available for C compiling. In this chapter, we are going to cover how we designed our program.

First of all, we need to decide how we are going to solve the systems of linear equations with. There are several ways to do it, however, we decided to use the variable elimination method. In this method for 3 unknown variables, we are going to eliminate variables by multiplying the constants of them and reducing the unknown variable number to 2. When we obtain two-piece of equations with 2 unknown variables, we will use the same method to reduce it to 1, and basically find the values of the variables. For the equations that have 2 unknown variables, we are going to follow the same process.

At the beginning of our script, we are just introducing the variables that we are going to use in the program. The constants of the variables of each equation are stored in the arrays, which are f1, f2, and f3. Since the constants can be any number including decimal numbers, the type of the variables is “float”. Also, we introduced result1, result2, and result 3 variables to be able to simulate the program whether it is working properly. There are also other variables that you can see as k, l, and m. “k” and “l” variables are representing the constants of the new equations which are including 2 unknown variables. “m” is for the results of these equations. You can see the introduction of the script in Figure 2.1.

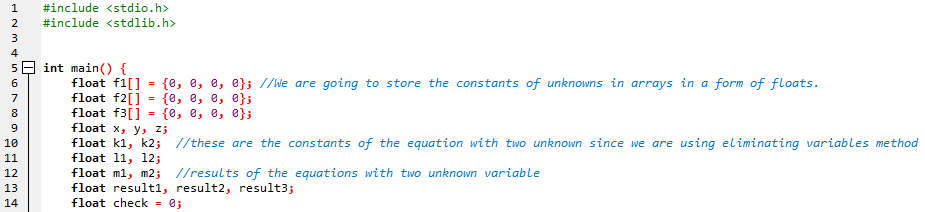


Figure 2.1: Introduction part of the script

We need to take the equations from the user. Our program can read the equations from a text file. First, we should open the file into the program. To be able to move the cursor in the file, we are utilizing the fseek() function. Thanks to this function, we can easily go to the next column or row in the file. We are using the fscanf() function to read data from the file and write it to the appropriate location. Basically, we are writing the constants the equations, and the results into an array to be able to use them in further operations. In Figure 2.2, you can see the script of the process of reading the data from the file.

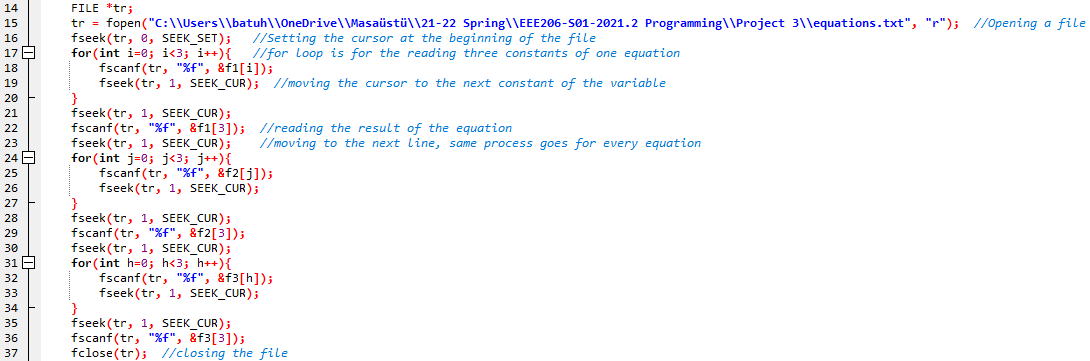


Figure 2.2: Reading the data from the file

After that point, we are printing the read data to the screen to be able to let the user know whether they are read correctly. To do that, we are using the printf() function as you can see in Figure 2.3.

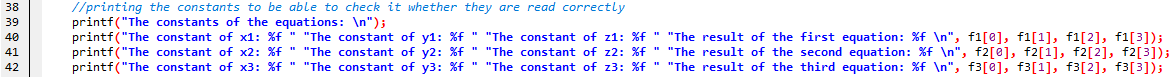


Figure 2.3: Printing the read data to the screen to allow users to make sure they are read correctly

At this point, we need to know if the input data are valid or not. That means, if the constants of the equations are the same as each other since it will not be a valid input to calculate the variables, the program should terminate the program with a warning. We are using an if condition to compare the constants with each other. You can see this implementation in Figure 2.4.



Figure 2.4: Comparing the constants of the equations between themselves to make sure that they are not equal to other functions' constants

Now, by using the variable elimination method, we can calculate the x, y, and z values. However, we need to know if the system includes two unknown or three unknown variables first. To achieve that, we used if cause to check the constants are equal to 0. In Figure 2.5, you can see the mathematical operations that we did to calculate the values of x and y for the equations that have two unknown variables.

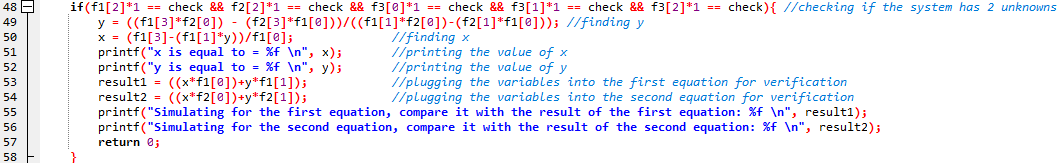


Figure 2.5: Mathematical operations to calculate the values of x, y, and z

If the system does not fulfill the if clause, the program will assume the system has 3 unknown variables. “else” will solve the function for the x, y, and z. In Figure 2.6, you can see the mathematical operations for the solution of the system with three unknown variables

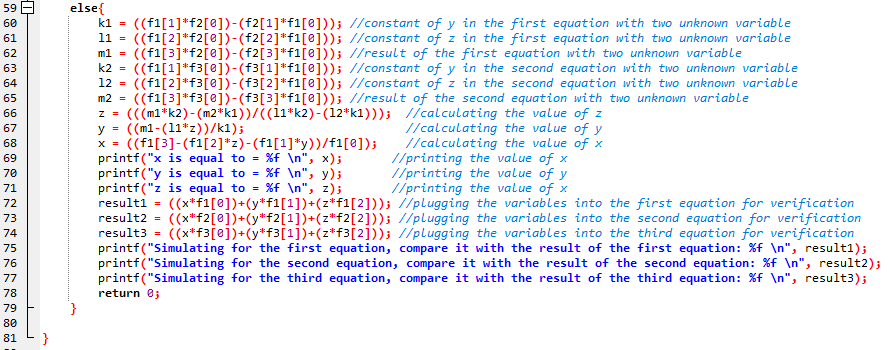


Figure 2.6: Printing the values of the variables and the results of the equations with calculated values of the variables

After the calculations, we plug the variables into the functions to verify that the results are equal to the given functions’ results. In the oncoming chapter, we are going to make simulations to verify that our program works properly.

# **3 SIMULATIONS AND VERIFICATION**

In this section, we are going to make some simulations of our program. We will run our program first to observe the outcomes, and we are going to solve these equations by hand or calculator and compare the results with the results of the program.

## **3.1 Running the Program with Different Equations**

In this section, we are going to run our program to see the results of the equations. In Figure 3.1, you can see our system of linear equations with 2 unknown variables.

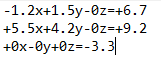


Figure 3.1: The system of linear equations with 2 unknown variables in the text file

In Figure 3.2, you can see the simulation of the program for the system.

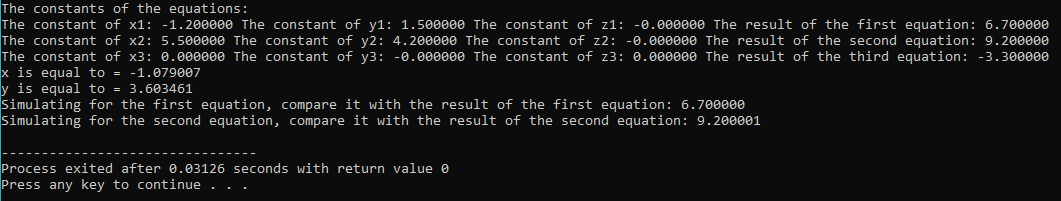


Figure 3.2: The results of the simulation for two unknown

As we can see, our program understood that there are two unknown variables and solved them for x and y. After it found the values, the program plugged the values for both equations, and as we can see, the results of the simulations are equal to the results of the input functions.

In Figure 3.3, you can see the system of linear equations with 3 unknown variables.



Figure 3.3: The system of linear equations with 3 unknown variables in the text file

Figure 3.4 demonstrates the simulation of the program for the new system.

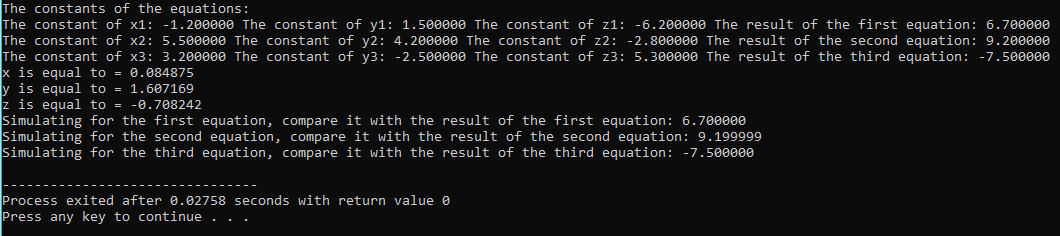


Figure 3.4: The results of the simulation for three unknown

As we can see in Figure 3.4, we successfully read the data from the text file and find the values of each variable. It can also be observable that the results of each equation are the same as the simulations.

In the oncoming section, we are going to solve these equations by hand by utilizing a calculator.

## **3.2 Solving the System and Comparing the Results**

In this section, we will solve the equations that we solved with the help of the code that we wrote in the previous section without any code and check if our code is working correctly. To do this, we will use the method of elimination that we use in our program. Let us first consider the equations with two unknowns, which are found in figure 3.1.

First, we eliminated the x's and found the y's. For this, we multiplied the equations found in 3.1 and 3.2 by +5.5 and +1.2, respectively.

Thus, the coefficients of the x's were equalized, and when we add up 3.5 and 3.7, the x's were eliminated, and we have only the y's left.

When we looked at Figure 3.2, we found that the y value found by our program is the same as the y value found in our own solution. Now we can find the value of x by replacing the value of y that we found in 3.1. We could also find the x value by eliminating the y values.

When we looked at Figure 3.2, we found that the x value found by our program is the same as the x value found in our own solution. Thus, we have proved that our program works correctly.

Now let's try to solve the equations with three unknowns, which are found in figure 3.3.

First, we found the common multiples of the y and z values and eliminated them, and only x values remain. When the equations were summed, the x value was found as the same as our program.

Now we can find the values of y and z by replacing the value of x that we found in 3.14 and 3.15.

We eliminated the y's and found the z's. For this, we multiplied the equations found in 3.20 and 3.21 by +4,2 and -1.5, respectively.

Thus, the coefficients of the y's were equalized, and when we add up 3.22 and 3.23, the y's were eliminated, and we have only the z's left.

When we looked at Figure 3.4, we found that the z value found by our program is the same as the z value found in our own solution. Now we can find the value of y by replacing the value of z that we found in 3.22.

When we looked at Figure 3.4, we found that the y value found by our program is the same as the y value found in our own solution. Thus, we have proved that our program works correctly.

# **4 CONCLUSION**

Using the Dev-C++ program, we designed a program that can calculate 2 or 3 unknown variables of systems of linear equations. We did some simulations of our program and to observe the results, we first ran our program and checked it. We tried solving these equations manually or with a calculator and compared the results with the results of the program. According to the results of the simulation, we can say that we achieved to create a program that solves systems of linear equations with 2 or 3 unknown variables correctly by reading the data from the text file.

# **REFERENCES**

[1] Anton, H. and Rorres, C. (1994). Elementary linear algebra: Application version. (7th ed.). New York: John Wiley

# **APPENDIX A: THE C SCRIPT**

#include <stdio.h>

#include <stdlib.h>

int main() {

float f1[] = {0, 0, 0, 0}; //We are going to store the constants of unknowns in arrays in a form of floats.

float f2[] = {0, 0, 0, 0};

float f3[] = {0, 0, 0, 0};

float x, y, z;

float k1, k2; //these are the constants of the equation with two unknown since we are using eliminating variables method

float l1, l2;

float m1, m2; //results of the equations with two unknown variable

float result1, result2, result3;

float check = 0;

FILE \*tr;

tr = fopen("C:\\Users\\batuh\\OneDrive\\Masaüstü\\21-22 Spring\\EEE206-S01-2021.2 Programming\\Project 3\\equations.txt", "r"); //Opening a file

fseek(tr, 0, SEEK\_SET); //Setting the cursor at the beginning of the file

for(int i=0; i<3; i++){ //for loop is for the reading three constants of one equation

fscanf(tr, "%f", &f1[i]);

fseek(tr, 1, SEEK\_CUR); //moving the cursor to the next constant of the variable

}

fseek(tr, 1, SEEK\_CUR);

fscanf(tr, "%f", &f1[3]); //reading the result of the equation

fseek(tr, 1, SEEK\_CUR); //moving to the next line, same process goes for every equation

for(int j=0; j<3; j++){

fscanf(tr, "%f", &f2[j]);

fseek(tr, 1, SEEK\_CUR);

}

fseek(tr, 1, SEEK\_CUR);

fscanf(tr, "%f", &f2[3]);

fseek(tr, 1, SEEK\_CUR);

for(int h=0; h<3; h++){

fscanf(tr, "%f", &f3[h]);

fseek(tr, 1, SEEK\_CUR);

}

fseek(tr, 1, SEEK\_CUR);

fscanf(tr, "%f", &f3[3]);

fclose(tr); //closing the file

//printing the constants to be able to check it whether they are read correctly

printf("The constants of the equations: \n");

printf("The constant of x1: %f " "The constant of y1: %f " "The constant of z1: %f " "The result of the first equation: %f \n", f1[0], f1[1], f1[2], f1[3]);

printf("The constant of x2: %f " "The constant of y2: %f " "The constant of z2: %f " "The result of the second equation: %f \n", f2[0], f2[1], f2[2], f2[3]);

printf("The constant of x3: %f " "The constant of y3: %f " "The constant of z3: %f " "The result of the third equation: %f \n", f3[0], f3[1], f3[2], f3[3]);

if(((f1[0] == f2[0] && f1[1] == f2[1] && f1[2] == f2[2])||(f1[0] == f3[0] && f1[1] == f3[1] && f1[2] == f3[2])|| (f2[0] == f3[0] && f2[1] == f3[1] && f2[2] == f3[2]))||(f1[0]==f2[0]==f3[0] && f1[1]==f2[1]==f3[1] && f1[2]==f2[2]==f3[2]) ){

return(printf("All equations must be different from each other.")); //the error of two same equation

}

if(f1[2]\*1 == check && f2[2]\*1 == check && f3[0]\*1 == check && f3[1]\*1 == check && f3[2]\*1 == check){ //checking if the system has 2 unknowns

y = ((f1[3]\*f2[0]) - (f2[3]\*f1[0]))/((f1[1]\*f2[0])-(f2[1]\*f1[0])); //finding y

x = (f1[3]-(f1[1]\*y))/f1[0]; //finding x

printf("x is equal to = %f \n", x); //printing the value of x

printf("y is equal to = %f \n", y); //printing the value of y

result1 = ((x\*f1[0])+y\*f1[1]); //plugging the variables into the first equation for verification

result2 = ((x\*f2[0])+y\*f2[1]); //plugging the variables into the second equation for verification

printf("Simulating for the first equation, compare it with the result of the first equation: %f \n", result1);

printf("Simulating for the second equation, compare it with the result of the second equation: %f \n", result2);

return 0;

}

else{

k1 = ((f1[1]\*f2[0])-(f2[1]\*f1[0])); //constant of y in the first equation with two unknown variable

l1 = ((f1[2]\*f2[0])-(f2[2]\*f1[0])); //constant of z in the first equation with two unknown variable

m1 = ((f1[3]\*f2[0])-(f2[3]\*f1[0])); //result of the first equation with two unknown variable

k2 = ((f1[1]\*f3[0])-(f3[1]\*f1[0])); //constant of y in the second equation with two unknown variable

l2 = ((f1[2]\*f3[0])-(f3[2]\*f1[0])); //constant of z in the second equation with two unknown variable

m2 = ((f1[3]\*f3[0])-(f3[3]\*f1[0])); //result of the second equation with two unknown variable

z = (((m1\*k2)-(m2\*k1))/((l1\*k2)-(l2\*k1))); //calculating the value of z

y = ((m1-(l1\*z))/k1); //calculating the value of y

x = ((f1[3]-(f1[2]\*z)-(f1[1]\*y))/f1[0]); //calculating the value of x

printf("x is equal to = %f \n", x); //printing the value of x

printf("y is equal to = %f \n", y); //printing the value of y

printf("z is equal to = %f \n", z); //printing the value of x

result1 = ((x\*f1[0])+(y\*f1[1])+(z\*f1[2])); //plugging the variables into the first equation for verification

result2 = ((x\*f2[0])+(y\*f2[1])+(z\*f2[2])); //plugging the variables into the second equation for verification

result3 = ((x\*f3[0])+(y\*f3[1])+(z\*f3[2])); //plugging the variables into the third equation for verification

printf("Simulating for the first equation, compare it with the result of the first equation: %f \n", result1);

printf("Simulating for the second equation, compare it with the result of the second equation: %f \n", result2);

printf("Simulating for the third equation, compare it with the result of the third equation: %f \n", result3);

return 0;

}

}