Project 1.2 Report

Curve Fitting

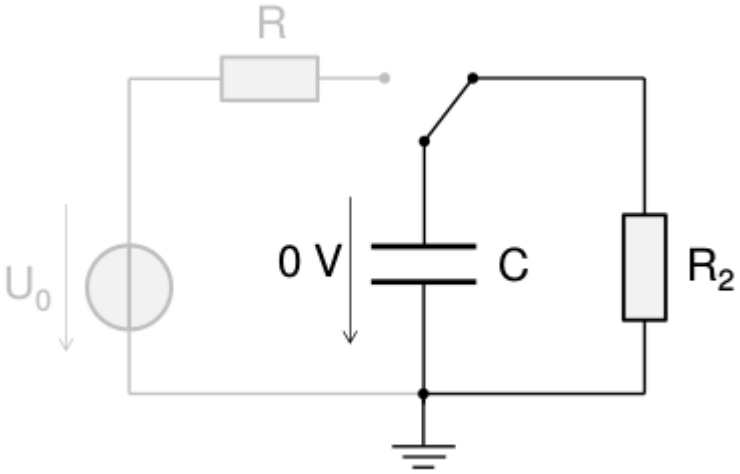
Introduction

In this project, task is to solve a curve fitting problem by writing a C program which solves it in reasonable time.

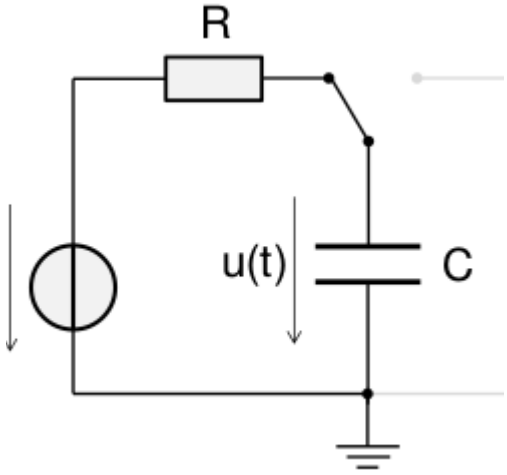
Curve Fitting is the process in which we have some given points and we try to find equation, by finding their unknown constants, which best fits that given points i.e. error is minimum.

Objective

Consider the circuit:



Initailly or at time < 0, the capacitor is not connected to voltage source and fully discharged.



Now, at time(t) = 0 switch is turned the voltage source get connected to capacitor and start charging. For time(t) > 0, voltage across is u(t) which is

$$u(t) = U_0 \cdot (1 - e^{\frac{-t}{R \cdot C}})$$

It is given that capacitor is 10mF, R(resistance) is unknown (have value between 10 and 100 ohm) and U0 is also unknown (have value between 12V and 20V).

Now, we have given some points we have to find best combination of U0 and R for which the error is minimum. In other words, we have to find the curve which best fits the given points.

$$e = \sqrt{{e_0}^2 + {e_1}^2 + {e_2}^2 + \cdots}$$

where e_i is

$$e_i = (u_i - u(t_i))$$

Understanding the Program

Including necessary standard header files in program.

1. Stdio.h for taking or giving input and output to standard terminal or to file.

2. Math.h for using exp function and power function in our program.

3. time.h for caluculating time elapsed by program.

```
#include<stdio.h>
#include<math.h>
#include <time.h>
```

Structure:

We define the structure Vector which stores two values of type double x and y. We will use it to handle values of t and u which we will read in next section.

```
// Structure to handle the all points
typedef struct{
    double x;
    double y;
} Vector;
```

Funtions:

Funtion which implement the equation which is written in comment. It takes value of U0, R, C and t and return the u(t) at that t.

```
// Equation u(t) = U0*(1 − e^(−t/RC))
 double equation(double U0,double R, double C, double t){
    return U0*(1 - exp(-t/(R*C)));
 }
```

Initialized Some Variables:

Define array of type Vector size of 100 to store points (value of u and t) which can be read from the file.

n : stored the current points which have been read from input file.

C: capacitor value

R: resistance(unknown)

U0: voltage of voltage source(unknown)

```
Vector points[100];

// number of points readed from the file
int n = 0;
// Capacitor value 1mF
double C = 0.01;
// let intial resistance and U0 are 0
double R;
double U0;
```

File Handling:

Open file data.bin in read and binary mode using fopen function in C.

If condition for checking that file is opened currently. If file does not file correctly program will exit and show error that “Error in reading file data.bin”.

```
// open data.bin file in read and binary mode
FILE *input = fopen("data.bin","rb");

// exit the program if file not found
if(!input){
   perror("Error in reading file data.bin");
   return 1;
}
```

Time to read data from the file. While loop will go through the complete file till it doesn’t ends and read and stores data to points struct line by line.

Here n++ increases the n value by one that tells how much points we have read from the file.

```
// read points from file one by one and add  it to points array of Vector
while(!feof(input)){
   fread(&points[n],sizeof(Vector),1,input);
   n++;
}
```

Finding minimum point:

This is heart of our program which finds the best combination of U0 and R for which we have minimum error.

Assuming initially minErro is Infinity.

Two for loops of u and r go through all possible values of U0 and R and each time it finds error that is less than the minError(minimum error) is saves that value of U0,R and changes the minError to that current error.

The inner for loop of k is for calculating the error which we already have discussed in Objective task.

```
int k;
float u,r;
double err, minError = INFINITY;
for(u=12.0;u <= 20.0;u += 0.1){
   for(r=10.0;r<=100.0;r += 0.1){
      err = 0.0;
      for(k=0;k<n;k++){
         double y = equation(u,r,C,points[k].x);
         err += pow(y-points[k].y,2.0);
      }
      if(err < minError){
         minError = err; U0 = u; R = r;
      }
   }
}
```

More accurate minimum point:

Don’t think these loops are same as and copied from the above point. These for loops are for finding more accurate value of U0 and R.

We already have find the value of U0 and R with accuracy of 0.1 but for more accuracy we have to go with more smaller points. So we go through values wthe small change(0.001) in u and r in between U0 – 0.5 and U0 + 0.5, R – 0.5 and R + 0.5.

```
for(u = U0 - 0.5;  u <= U0 + 0.5; u += 0.001){
   for(r = R - 0.5; r <= R + 0.5; r += 0.001){
      err = 0.0;
      for(k=0;k<n;k++){
         double y = equation(u,r,C,points[k].x);
         err += pow(y-points[k].y,2.0);
      }
      if(err < minError){
         minError = err;
         U0 = u;
         R = r;
      }
   }
}
```

Two print statement here prints the value of R and U0.

```
printf("R: %lf \t\t",R);
printf("U0: %lf\n",U0);
```

End:

fclose close our input file.

cpu_time_used calculate the time that has been taken by program and printf print that.

Return 0; ends our program and close.

```
  fclose(input);
  // end time when program works end
  end = clock();
  // calculating elapsed time
  cpu_time_used = ((double) (end - start)) /
CLOCKS_PER_SEC;

  printf("Time: %lfs\n",cpu_time_used);
  return 0;
```