

# Lab Batch 221

## SRS

### STAGE-1 SUBMISSION

### REPORT(Modified)

#### Project: Graph Plotting

##### Introduction:

The objective of our program would be to create a command language for the user for giving input, which when used efficiently, can plot graphs and histogram.

We decided to opt for histogram just to extend the project domain, incase our main part finished earlier

Note- The Histogram team has not yet done the finishing work of the histogram functions.

Objectives: Our program will take input from a text file that is to be written by user using commands and syntax specified on following pages.

- A user can plot **multiple curves**. (upto 3 curves) in different windows.
- He can also use the commands to plot **tangents** and **normals** of any given curve at a specified point
- Calculation of **slope** of a given curve at a point
- Calculation of the **definite integral** with respect to the independent variable in the given range can also be done using commands.
- To ensure to keep the syntax of commands easy to maintain its user-friendliness
- Give flexibility to user to give information in text file about ,say function f1, in any order.However it is expected of user to give certain information that is mandatory to plot graph of that function.

##### **Functional Requirements :**

1. A tar package containing EzWindows and fparser 4.4.03(slightly modified by input and computation teams to suit purpose).The latest EZWindows folder (ezcs101 folder), should be kept in the home folder of the user.
2. Linux environment (Ubuntu)
3. GCC compiler

## OUR TEAM :

1. Ashish Sonone
2. Ashish Agrawal
3. Anshul Avasthi
4. Anshul Purohit
5. Anurag Gautam
6. Anwar Ahmed Ansari
7. Ashutosh Upadhyay

Work Assignment : Currently we are writing program to plot  $y=f(x)$  type functions .Our module 4 is working on extending it to histogram.

- Module 1: INPUT from Text file and using fparser to interpret.

Working members : Ashish Sonone, Ashutosh Upadhyay

- Module 2: Write Utility functions to calculate derivative,Area ,find maxima,minima,points of Inflection,etc.

Working members : Anshul Avasthi, Anshul Purohit, Ashish Sonone

- Module 3: Working on Ezwindows to plot graphs.

Working Members : Anshul Purohit, Ashish Sonone, Anwar Ahmad Ansari

- Module 4: Working on extending the functionality of our project to histogram.

Working Members : Anurag Gautam, Ashish Agrawal , Anshul Avasthi

### **Input:**

User can give required input through a text file called 'command.txt'. The following is the syntax:

# sign maybe used for comments that the user wishes to insert. It should be the first character on a line in order for it to be treated as a comment. Only the given line will be

treated as a comment.

1. **G** User will have to type this keyword in the first line of text file in order to plot a function of a single variable.
2. **fn=** \_\_\_\_\_ represents the  $n^{\text{th}}$  function. Here the blank may be filled by composites of the functions defined in the function list.(note no spaces b/w f , n & = characters .However function expression may include spaces but must be in same single line)

Now following commands describe the function and must be used b/w two "fn=" commands

3. **x(a,b)** user gives the portion of the x-axis he wishes to see be plotted ie, graph shall be shown from a to b where  $a < b$ . (note :again no spaces b/w characters in this command)

OPTIONAL COMMANDS.

5. **c** *colorname* :Here colorname is to be replaced by one of the color options given below .It is the colour of the given curve. If not specified, colour shall be assumed to be Black. following color options are available(NOTE: **c** is lowercase)

White,Red ,Green ,Yellow ,Blue, Magenta, Cyan ,Black.

6. **T(x1 , x2, ....)** This command plots the tangent at  $x_1, x_2, \dots$
7. **N((x1 , x2, ....)** This plots the normal to the curve at  $x_1, x_2, \dots$
8. **A (a,b)** The area and integral of the curve covered between,  $x=a, x=b, y=0$  &  $y=f(x)$  will be calculated and displayed in the terminal.
9. **m** view minima points.(if any)
10. **M** View Maxima points (if any)
12. **~** This must be the last line of text file which indicates end of text file.

For example sample input text file may contain something like this:

```
G
f1= x^2+4*x^3+5
x(-5,5)
M
T (0.3, -0.4, ) # my tangent points
I
A (-3,3)
```

# *my first function ends here*

$f3 = \sin(x + \tan(\arccos(x)))$

$x(-1,1)$

*c Blue*

*m*

*l*

$T(0.0, .14, -0.64, -0.45)$

$N(2.5)$

### If possible...

**histogram** allows user to plot histograms, it will have its own subset of commands. STILL UNDER CONTRUCTION it is an open ended project.

### Output:

A graph will be displayed on the EzWindows screen.

**Our program Supports following functions and their composites:**

**NOTE: Function commands are CASE SENSITIVE.**

- **A^B**            **A raised to the power B**
- **-A**            **unary minus**
- **A\*B** **A/B** **A%B**        **multiplication, division and modulo**
- **A+B** **A-B**        **addition and subtraction**
- **abs(A)**        **Absolute value (magnitude) of A**
  
- **cos(A)**
- **sin(A)**
- **tan(A)**

- **cot(A)**
- **csc(A)**
- **sec(A)**
  
- **cbrt(A)**            **Cube root of A.**
- **sqrt(A)**            **Square root of A**
  
- **exp(A)**            **Exponential of A. Returns the value of e raised to the**  
                          **power A**
- **exp2(A)**            **Base 2 exponential of A**
  
- **floor(A)**            **Floor of A. Returns the largest integer not greater**  
                          **than A.**
  
- **log(A)**            **Natural (base e) logarithm of A.**
- **log2(A)**            **Base 2 logarithm of A.**
- **log10(A)**            **Base 10 logarithm of A**

**Note: Composites can be defined using simple parenthesis ( )**

e.g  $\sin(x^2 + 2*\cos(x^3))+ \sin(\cos(\tan(\log(\operatorname{acot}(x^{3.75} - x^{7.3}))))))$

**Following are the constants that can be used in function expressions:**

1. pi
2. e

Eg.  $\sin(\pi*x^2 + e)$

**For the completion of the plotter, we had to use [Fparser v4.4.3](#) for C++.**

This C++ library offers a class which can be used to parse and evaluate a mathematical function from a string (which might be eg. requested from the user). The syntax of the function string is similar to mathematical expressions written in C/C++ (the exact syntax

is specified in the documentation below). The function can then be evaluated with different values of variables.

For example, a function like `"sin(sqrt(x*x))` can be parsed from a string (either `std::string` or a C-style string) and then evaluated with different values of  $x$  and  $y$ . This library can be useful for evaluating user-inputted functions, or in some cases interpreting mathematical expressions in a scripting language.

This library aims for maximum speed in both parsing and evaluation, while keeping maximum portability. The library should compile and work with any standard-conforming C++ compiler.

### **Assumptions:**

- One assumption that we had to make to plot smooth curves was to plot discrete points at a small enough distance from each other so that they appear smooth due to the resolution limit.
- The program will compute only an approximate integral as it uses the property of approximate Riemann sums. It would have less error if the partition between the two extremes of  $x$  is fine enough.
- The derivative obtained would also be approximate as slope is computed between two discrete points.

### **Limitations:**

- Only explicit functions can be plotted.
- The graph of the function becomes faint as the function jumps up sharply to infinity or jumps down sharply to  $-\infty$ .
- The user can only view that portion of the graph in the range of the  $x$ -axis and  $y$ -axis (if range of  $y$  axis is given by user or set by defaults) at a time. To view the other portion of the graph user will have to modify the input file, save it and recompile the program.
- It is not printer-compatible.
- Input can only be given through a `.txt` file, `.doc` and equivalent files are not supported.
- Intersection of curves cannot be computed but only approximated.

**Work Input (Excluding lab timings)**

Anshul Avasthi - 4 hrs

Anshul Purohit - 5 hrs

Ashish Sohane - 8 hrs

Anurag Gautam - 5 hrs

Ashish Agrawal - 1 hrs

Ashutosh Upadhyay - 1 hrs

Anwar Ahmad - 1 hrs