I. EXERCISES - SESSION 3

A. Comments on Sessions:

The goals of this session are the following:

1. Pass functions as pointers to subroutines
2. Read and write from FILES
3. Fitting with gnuplot and plot files
4. Numerical Differentiation

NOTE: Beginners will do questions in the warm-up section and the first problem. The advanced programmers will answer both the questions in the problem section.

B. Preparations:

1. Preparations: Download the session03.tar.gz file from

   ```
   www.physics.iitm.ac.in/~suna/nummethods.html
   ```

   under the section DCF sessions. The file usually downloads into the Desktop directory or the Downloads directory. Open the terminal (click on Applications → System tools → Konsole for Ubuntu running GNOME or search for Terminal on Ubuntu running Unity). Move the .tar.gz file to the home directory using the following command (remember to hit enter after each of the following commands):

   ```
   mv Desktop/session03.tar.gz ./
   ```

   Untar the files using the following commands:

   ```
   tar xfvz session03.tar.gz
   ```

   This will automatically open a directory called session03. Move into that directory by typing the following command on the terminal:

   ```
   cd session03/
   ```

   You can check the files in that directory by typing: (lists)

   ```
   ls
   ```

C. Warm-up

1. Basic Programing (for beginners): In this section, you will learn about passing addresses of functions. We will once again work with area.c code. Open the code area.c in the editor gedit. You will see that this code calls a subroutine area_calc.c which is a separate C code to calculate area. In the code area_calc.c, a function "my_func" is called. This function exists in the main code area.c and it returns the area of various geometries depending on the value of the integer "opt", which is provided by the user through command line input. Since the function my_func is in the main code and another code area_calc.c calls it, you need to declare the function prototype in the code area_calc.c. Try the following:

   (a) Compile the code using the makefile provided with the following command:

   ```
   make -f make_area
   ```

   and run the code using the following command:

```
./area 1
```

where area is the executable and 1 refers to the value of the option. This is passed to the main program through the array argv[ ] as a string and is converted to an integer through the function "atoi". Try running for different values of the opt and see the result. Make sure you understand the code, which also illustrates the use of conditional statements ("if"-"else if"-"else").

(b) Let us now modify the codes in the following way: In the code area.c, replace the line

```
area = area_calc(opt, r, l, b, h);
```

by the following:

```
area = area_calc(opt, r, l, b, h, &my_func);
```

You are now passing the address of the function my_func. In the code area_calc.c, remove the function prototype for my_func and replace the function

```
area_calc(int opt, double r, double l, double b, double h)
```

by

```
area_calc(int opt, double r, double l, double b, double h, double (*func)(int, double, double, double, double))
```

This means that the function area_calc now gets six inputs, the first one is an integer, the next 4 are real numbers in double precision and the last is a pointer to a function, where the function now called func, takes five inputs, where the first one is an integer and the next four are real numbers in double precision. Next replace the line

```
area = my_func(opt, r, l, b, h);
```

by

```
area = (*func)(opt, r, l, b, h);
```

This means that you are accessing the function through its address. Compile the code once more using the makefile and run it. Remember to enter a valid option number when you run the executable!

D. Problems

2. Numerical Differentiation: Open the code deriv_test.c. This code calls a subroutine called deriv.c that calculates the first derivative using either forward or central difference (via the value of option flag - see the subroutine deriv.c). Here we pass the address of the function to the subroutine and use the address to call the function in the subroutine. Make sure you understand this step. In class we analyzed the forward difference algorithm. Now repeat the exercise for the central difference, by first compiling the code using the makefile provided, and running the executable. Note that if you run the code, the first derivative using central difference is already calculated and the results are written into the two files. You will need to plot $\log_{10}$ of the error versus $\log_{10}$ of the step-size h. We had seen in class that the relative error for the central difference algorithm roughly scales as:

$$\epsilon_{\rm rel} = \frac{\epsilon_{\rm m}}{h} + Ch^2$$

where C is a constant and the first term is due to the round-offs that will make a big contribution for small values of h, while the second term is the approximation error that is large for larger values of h. Check that you see these trends in your error plots. Extract the slopes for the two linear region. Make sure you label the axes.

Use the gnuplot tutorial for help with fitting and labeling. Please write the values of the slope that you get in the space provided below:

3. More on derivatives: Let us now determine the second derivative of the function $\exp x$.

   (a) Write a function, similar to deriv.c (that is a separate C file) that will calculate the second derivative of a function that is passed to it. This means you will pass the address of the function whose second derivative is needed, similar to the deriv.c code. Note that you will need to call the function in the subroutine using (again refer deriv.c):

   ```
   (*func) (x)
   ```

   instead of

   ```
   func(x)
   ```

   Make sure you understand the difference before you proceed further! Use the following expression for the second derivative:

   $$f''(x) = \frac{f'(x+h) - f'(x-h)}{h}$$

   Using the expression for the first derivative using the two-point formulas (forward and central difference), we get:

   $$f''(x) = \frac{f(x+h) + f(x-h) - 2f(x)}{2h^2}$$

   Calculate the relative error between the exact and the calculated second derivatives and store the result in a file. Remember to modify the makefile so that it also compiles and links the code that calculates the second derivative as well.

   (b) Obtain an expression for the round-off error given using the equation above for the second derivative.

   (c) Plot the output file using gnuplot (plot $\log_{10}$ of the error versus $\log_{10}$ of the step-size h) and extract the slope of the two linear regions. What do you understand from this behavior, i.e. how does the round-off error and the approximation error behave with respect to h?