I. EXERCISES - SESSION 4

A. Comments on Sessions:

The goals of this session are the following:

1. Using structures and pointers
2. Read and write from FILES
3. Fitting with gnuplot and plot files
4. Numerical Integration

NOTE: The beginners will download a file called session04b.tar.gz, while the advanced programmers will download a file called session04a.tar.gz. The problems are marked "beginners", "advanced" and "All". Therefore:

- Qs.1 is for beginners who have not attempted problem 6 in exercise 2, which was summing $1/n$ series.
- Qs. 2 is for all beginners.
- Qs. 3 is only for advanced
- Qs. 4 for everyone!

B. Preparations:

Preparations: Download the tar file from

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

under the section DCF sessions. Untar the file and move into the correct directory.

C. Warm-up

1. (Beginners who have not attempted prob 6 from session 2): Summing Series: Write a code to sum the following series in two ways:

$$S_{\rm up} = \sum_{n=1}^{N} \frac{1}{n}$$

and

$$S_{\rm down} = \sum_{n=N}^{n=1} \frac{1}{n}$$

(a) Calculate the relative error between the two ways of summing, defining the error as a function of N, which is the number of terms retained.

$$\epsilon_{\rm rel} = 2 \left| \frac{S_{\rm up} - S_{\rm down}}{S_{\rm up} + S_{\rm down}} \right|$$

You will need to work with single precision and go as large as $N = 10^{10}$ in order to see the results. This means you could increment the number of terms retained in powers of 10, starting from some minimum value and going all the way until the maximum value of 10^{10}. Store the relative erros as a function of N into a file.

(b) Which way of summing will work well and why?

2. (Beginners): Open the codes deriv_test.c and deriv.c and summarize what you understand:

3. (Advanced): Make the following changes to the codes deriv_test.c and deriv.c so that you calculate the first derivative of the function $\exp(\alpha x)$:

   (a) Pass the extra parameter α as a command-line argument.

   (b) Change the function my_func so that is accepts two arguments: one being x and the other the address of α passed through a void pointer, that is:

```
double
my_func(double x, void *params)
```

   and make suitable changes in the code so that you now pass the address of the parameter α through a void pointer. The use of a void pointer makes the function more generic and this is preferred over passing the

address of α, which will be of type double *. List the changes you needed to make in the codes:

(c) Compile and run the code for some value of x and α.

D. Problems

4. (All): Open the code integ_test.c. This code integrates the function $\exp(\beta x/\alpha)$ between the limits $(-1, 1)$ using the trapezoid rule. Set up a function that uses Simpson's rule and a higher-order rule called the three-eights rule where the elementary weights are given in the table I

Name	Degree of polynomial	Elementary Weights	# of points
Trapezoid	1	$\{h/2, h/2\}$	2
Simpson	2	$\{h/3, 4h/3, h/3\}$	3
3/8	3	$\{3h/8, 9h/8, 9h/8, 3h/8\}$	4

TABLE I: Table of Elementary Weights for Newton-Cotes methods

Elementary weights can be combined to get the integration weights by overlapping the first and the last points in each interval. For example, we get Simpson rule weights from the elementary weights as follows:

$$w_i = (h/3, 4h/3, 2h/3, 4h/3, 2h/3, \cdots 4h/3, h/3) \tag{1}$$

where we have overlapped the third point of the first elementary interval with the first point of the second elementary interval where the weights for the intervals are given by the second entry in the table I. What are

the weights for 3/8 rule? List out the steps you will need to write a code that implements the simpson's rule.

What will be steps for the three-eights algorithm?

Note that the Simpson's rule needs odd number of points, while this is not a restriction for the trapezoid rule. Similarly pay attention to the three-eights rule too! Note that the code passes the parameters α and β through structures and pointers to structures. Use a similar prototype for the codes you will be setting up. You will have to create a makefile for this problem. You could make a copy of the make_deriv_test which can be modified.

5. Make a log-log plot of the file obtained using gnuplot and obtain the slope of the linear region for all the three rules.