DEPARTMENT OF PHYSICS
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS

PH5720 Numerical Methods and Programming **Session 07** **3 March 2020**
Time: 2:00 pm - 5:00 pm

Goal of this session:

1. Write a program for Lagrange interpolation.
2. Write a program for numerical methods of integration.
3. Implementation of numerical integration methods and error Assessment.
4. Please download the data file "sample_data.dat" from the moodle website.
5. You may use the codes uploaded on moodle for examples/syntax etc.
6. Please upload your plots (.pdf file) on moodle and submit this lab sheet by **Tuesday 17 March 2020, 5:00 pm**.

---

1 Problems:

1. Interpolation: You have seen in class that the simple Lagrange interpolation that fits a $N-1$ degree polynomial using N points results in the following formula:

$$f(x) = \lambda_0(x)y_0 + \lambda_1(x)y_1 + \cdots + \lambda_{N-1}(x)y_{N-1}$$

where,

$$\lambda_i(x) = \prod_{j=0\, j\neq i}^{N-1} \frac{(x - x_i)}{(x_i - x_j)}$$

where N stands for the number of points you use at a time, x_i and y_i are the points and the function evaluated at the points respectively. Your task is to interpolate the data in the file sample_data.dat using Lagrange interpolation. Note that the original data has energy measured at 25 MeV intervals. Define a new set of points call it x_new such that x_new is at 2 MeV interval. Interpolate the data in the file for the following values of N

(a) $N = 9$, that is taking all the 9 points, which means you interpolate the data with a degree 8 polynomial

(b) $N = 3$, that is taking 3 points at a time, which means you now interpolate with a degree 2 polynomial.

In order to read the data points from the file, use the following piece of code which is one way of reading from a file. (Remember to declare a file pointer called in_ptr at the beginning of the your main code).

```
in_ptr = fopen(``sample_data.dat", ``r"); /*this opens the file in the
read-only mode, therefore the file should exist!*/

fgets(line, sizeof(line), in_ptr);      /*this gets the first line from the
file, which is a comment in this case and hence we do nothing with it*/
```

```
 n = 0;      /*this is a counter that we use to read from the file and write
into the two arrays x_old and f_old. Use double x_old[N] and double f_old[N] to
declare arrays*/

 while(fgets(line, sizeof(line), in_ptr) != NULL)
 {
  sscanf(line,``%lf %lf", &x_old[n], &f_old[n]);

  n ++;
 }

 N_old = n; /*this is the number of points in the datafile*/
```

The above loop is executed until the end of the file is reached. So in short, we skip the first line since we do nothing with the first line we read using the fgets command, but then use the data from the second line onwards until the file ends. Remember to close the file after you have read the data from it.

Write the value of the function evaluated at the points x_new into a file for both the cases. Plot the data file as well as the two different interpolated functions. The correct curve is a Lorentzian. Which interpolation works well and what do you learn?

2. Integrate the following function

$$I = \int_0^3 dx \frac{1}{2+x^2} \tag{1}$$

using trapezoid rule. The exact result is:

$$I = \frac{1}{\sqrt{2}} \tan^{-1}\left(\frac{3}{\sqrt{2}}\right)$$

Compare the numerical result for the fixed step-size trapezoid against the exact and plot the relative errors for

the simple trapezoid rule on a log-log plot and comment on your results (i.e. convergence with increased N).

3. Write a double-precision program to integrate an arbitrary function numerically using the trapezoid rule, the Simpson rule, and Gaussian quadrature. For our assumed problem there is an analytic answer:

$$\frac{dN(t)}{dt} = e^{-t} \quad \Rightarrow \; N(1) = \int_0^1 e^{-t} dt = 1 - e^{-1}$$

Compute the relative error $\epsilon = |(numerical - exact)/exact|$ in each case. Present your data in the tabular form. Try N values of $2, 10, 20, 40, 80, 160$. (Hint: Even numbers may not be the assumption of every rule.)

4. Design your own C++ Class to compute the integral

$$I = \int_0^1 exp(x)\ cos(x)\ dx \tag{2}$$

which implements both the Trapezoidal rule and Simpson's rule.