# I. EXERCISES - SESSION 8

## A. Comments on Sessions:

The goals of this session are the following:

1. Using gsl libraries
2. Handling Matrices
3. Linear Algebra routines

Everyone attempts all the problems!

## B. Preparations:

Preparations: You have to write all the codes from scratch, so make a directory called session08 using the following command:

```
mkdir session08
```

Move into the correct directory and follow the rest of the work sheet

## C. Problems:

1. Let us solve the Poisson's Equation in 1D using a tridiagonal solver. We solved the following equation last week using LU decomposition:

$$\frac{d^2\phi}{dx^2} = -\frac{\rho(x)}{\epsilon_0}$$

where we can re-cast it in the following dimensionless form:

$$-u''(x) = f(x)$$

Using $f(x) = (3x + x^2)e^x$, we get the following analytic solution

$$u(x) = x(1-x)e^x$$

Using the discretized version from last week: $AX = B$, where

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 & 0\cdots \\ -1 & 2 & -1 & 0 & 0\cdots \\ 0 & -1 & 2 & -1 & 0\cdots \\ \vdots & & & & \end{pmatrix} \qquad B = \begin{pmatrix} h^2 f(x_1) \\ h^2 f(x_2) \\ \vdots \end{pmatrix}$$

solve for $X$ using a tridiagonal solver from the gsl library. Vary the dimension of the matrix $N$ from 10 to $10^5$. Once again use the clock function to compute the complexity time. Please make a log-log plot of the time taken by the tridiagonal solver as a function of $N$, which is the size of the matrix. Find the slope and comment on your results, comparing with what you had from the last session. Note you do not need to plot the error as a function of $x$. Only one plot for the scaling of time with the dimension of the matrix is required.

2. Adaptive Integration: Most of the integration routines you have seen use a fixed number of points. While this is very easy to implement, it may not be the most efficient in some case. For example, if a function varies a lot in one region, we would like to add more points here and decrease the number of points elsewhere where the function is smooth. Of course we could plot the integrand and decide how to split it. This is of course not convenient. What we would like to do is to write an adaptive code that does this automatically. Follow the instructions below and try an adaptive code for the trapezoid rule.

   (a) First integrate the entire range: i.e. if the integral is defined as:

   $$I = \int_a^b f(x)dx \tag{1}$$

   then we evaluate the above integral as usual by the following function call:

```
I0 = trapezoidal_rule(npts, a, b, &func);   /*You already know this!*/
```

   We call this $I0$, which is the first pass at evaluating the function.

   (b) Now we divide the range $[a, b]$ into two parts by defining $c = \dfrac{a+b}{2}$. Then we split the integral into two parts:

   $$\int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx \tag{2}$$

   and call the trapezoid rule to evaluate each of the two pieces. i.e:

```
I1L = trapezoidal_rule(npts, a, c, &func);   /*integrating between a and c*/
I1R = trapezoidal_rule(npts, c, b, &func);   /*integrating between c and b*/
I1 = I1L + I1R;    /*the integral is a sum of the two pieces*/
```

   Next we check for signs of improvement: i.e.. compare the results $I0$ and $I1$: $|I0 - I1| < \epsilon$. If the error is less than the tolerance $\epsilon$, we stop the process, otherwise continue by splitting $I1L$ and $I1R$ further into two pieces so that we have divided the original integration into 4 pieces. This is continued until we get the desired absolute or relative errors (this is set by the user).

   (c) Let us integrate the following function

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

   using trapezoid rule with without the adaptive procedure.The exact result is:

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

   Compare the numerical result against the exact and plot the relative errors for the simple trapezoid rule and the adaptive version on a log-log plot and comment on your results.