# PH3205-Computational Physics

SPRING 2022

Bipradeep Saha (19MS135)

*Indian Institute of Science Education and Research, Kolkata,*
*Mohanpur, West Bengal, 741246, India.*

*January 28, 2022*

# Worksheet 4

A notebook is provided containing solution to both the problems: *WS4_notebook.ipynb*. Also individual *.py* files are provided separately for this.

## Problem 1: Generalized RK4 method and its implementation on Lorenz system.

For this problem a generic RK4 solver is defined as $RK4(q, derivs, t, h)$ which accepts current position and derivative function and returns the next position.
Python File: *4_1.py*. Output image: *Problem1.jpg*

## Problem 2: Solving PDE using Euler and RK4 method

The heat equation is :

$$\frac{dP(x,t)}{dt} = D\frac{d^2P(x,t)}{x^2}$$

On discretizing the spatial variable, we have the following PDE to solve:

$$\frac{dP_i(t)}{dt} = D \times \frac{P_{i+1}(t) + P_{i-1}(t) - 2P_i(t)}{\Delta x^2} \quad [i = 1, 2, 3, \ldots, N-1]$$

For the implicit Euler method, we also discretize the time, thus we have:

$$P_n(t_{j+1}) = P_n(t_j) + \frac{D\Delta t}{\Delta x^2}[P_{n-1}(t_j) - 2P_n(t_j) + P_{n+1}(t_j)]$$

If you select $D$ such that $D\Delta t/\Delta x^2 = 1/2$, then the implicit Euler method reduces to:

$$\boxed{P_n(t_{j+1}) = \frac{1}{2}[P_{n-1}(t_j) + P_{n+1}(t_j)]}$$

For solving using RK4 method, we first set $D/\delta x^2 = 1/2\Delta t$(from the given condition of the problem). Then we have N-2 set of 1st order ODE to solve given by:

$$\boxed{\frac{dP_i(t)}{dt} = \frac{1}{2\Delta t} \times [P_{i+1}(t) + P_{i-1}(t) - 2P_i(t)]} \quad [i = 1, 2, 3, \ldots, N-1]$$

We use the generic RK4 solver defined above to solve. Also the boundary conditions are set as $P(x = 0, t) = 0 = P(x = x_N, t)$.

In the python file we defined 3 functions:

- $euler\_heat(q, derivs, t, h)$ : This function defines the update step using Euler Method.
- $heat\_derivs(t, q)$ : This function is the derivative function for euler method above.
- $heat\_derivs\_rk4(t, q)$ : This function is the derivative function to be passed to generic RK4 solver from problem 1.

Python File: *4_2.py*. Output image: *Problem2.jpg*

**Explaining the figure**:

- The problem specifies the spatial step size $\Delta x = 0.01$, however we are free to select the temporal size $\Delta t$. The constant $D$ is specified by both these discretization.
- For larger $\Delta t$, the initial Gaussian function is more narrowly peaked.
- Thus we run the code to evolve the solution for 3 different step size.
- Further the initial solution is normalize to 1.
- We observe that using both the method, the solution is almost similar. We have also plotted the difference between the methods.