# 104283 – Introduction to Numerical Analysis

**Spring Semester, 2023**

# Assignment 4

## Numerical Integration | Runge-Kutta Method

### Instructions:

1. Assignment is done in singles only and submitted through Moodle.
2. Submit the following two files ONLY:
   2.1. Report – including your code, results, derivations, explanations (if required), etc. Saved as a single *.pdf file: **report.pdf**
   2.2. Single .ZIP file with your Python codes.
3. Python files should be stored as a *.py files inside the zip file. Pay attention - your code should be included in the report as well (as image or text).
4. The zip file name format is as follows:
   4.1. HW
   4.2. Assignment number
   4.3. Underscore
   4.4. Student name (first and last name with no spaces, last name in CAPITAL letters)
   4.5. Underscore
   4.6. Student number

Example:
"HW4_harryPOTTER_999333666.zip"

5. The submission should include only the required submission files and nothing else. No subfolders or any unnecessary files should appear in the zip file. Do not use rar file or anything other than zip.
6. Submissions not following this format will not be accepted.
7. Late submissions policy – given $n$ days late submission, $2n$ points reduction penalty.
8. Make sure you adhere to all the principles learned in class. Using specialized external Python libraries is not allowed (unless otherwise specified).

## Chaos and The Lorenz Attractor

In this assignment you will explore the phenomenon of chaos through the Lorenz equations, a system of ordinary differential equations that exhibits chaotic dynamics. Chaos is a phenomenon characterized by the sensitivity of a system to initial conditions, resulting in highly unpredictable and complex behavior.

### 1. Runge-Kutta Integrator

Write the following function in Python:

**RK4_step(func, dt, tk, wk)**

The function gets the following parameters as input:

- Some function - **func** (Python function).
- Time step - **dt**.
- Time value – **tk**.
- Function state – **wk.**

The function calculates a single RK4 step and returns: $(t_{k+1}, w_{k+1})$. Meaning, the next time step and its corresponding function state evaluation at that time. You may use Algorithm 5.2 (page 288) for reference.

### 2. Lorenz Equations

The Lorenz equations are a system of three coupled non-linear ordinary differential equations developed by Edward Lorenz in 1963 to study the dynamics of weather patterns:

$$\begin{cases} \frac{dx}{dt} = \sigma(y - x) \\ \frac{dy}{dt} = x(\rho - z) - y \\ \frac{dz}{dt} = xy - \beta z \end{cases}$$

The constants $\sigma$, $\rho$, and $\beta$ are parameters that determine the behavior of the system. To observe chaotic behavior, a common choice is: $\sigma = 10,\ \beta = \frac{8}{3},\ \rho = 28.$

Write the following function in Python:

**lorenz(t,w)**

The function gets the time $t$ and state $w = x, y, z$ as input arguments. The function returns the set of values $\left(\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt}\right)$. Use the suggested values for $\sigma$, $\rho$, and $\beta$.

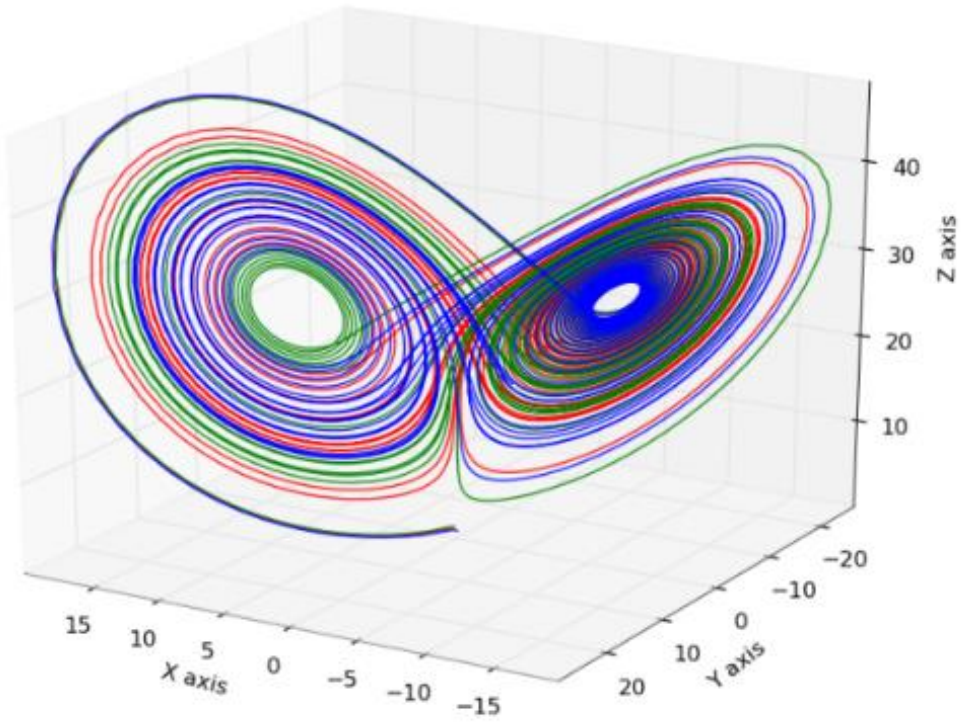
### 3. 3D Plot

In a chaotic system, a small change in initial conditions will result in different behaviors which are magnified over time.
Solve the equations for two different initial conditions using the functions you wrote. Create a 3D plot of the two results on the same figure, allowing you to observe the divergence of trajectories over time. Choose time step $dt = 0.01$ and any pair of initial states $(x_0, y_0, z_0)$. Modify the total solution time so that the results are clear.

For 3D plotting, use the included code snippet (3dplot.py). Add your code and results to the report.

### Deliverables:

1. PDF report including your code and plot. Attach LARGE images.
2. ZIP file with your Python code.




*Lorenz Attractor*