Objective of the Lab

- Since the course's main objective is to find the performance of a piece of code, this lab focuses on how to profile a simple piece of code.
- Learn how to initialize memory dynamically
- Learn how to plot the performance graphs using GNU plot `https://pranabdas.github.io/linux/gnuplot/`

Experiment 1 - Estimate the value of π

Read the following link on estimating the value of π using a randomised algorithm technique.
`https://risk-engineering.org/notebook/monte-carlo-pi.html`

Performance Graphs

- Graph 1 - Time taken by the code vs Number of samples
- Graph 2 - Error vs Number of samples

Experiment 2

Write a C/C++ program to perform matrix-vector multiplication. Given a matrix A and a vector bx, finding the vector b is called the Matrix-Vector multiplication. The size of the matrix involved is $N \times N$. Matrix-vector multiplication can be done in two ways.

$$b := Ax$$

In the above, $b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$, $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & & & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}$ and $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$

Method 1

$$b_i := \sum_{j=1}^{N} a_{ij} x_j, \forall i \in \{1, 2, \cdots, N\}$$

Method 2

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = x_1 \times \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{N1} \end{bmatrix} + x_2 \times \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{N2} \end{bmatrix} + \cdots + x_N \times \begin{bmatrix} a_{1N} \\ a_{2N} \\ \vdots \\ a_{NN} \end{bmatrix}$$

Performance Graphs

For N values of $[128, 512, 1024, 2048, 4096, 8192]$

- Graph 1 - Time taken by Method 1 and Method 2 vs N
- Graph 2 - FLOPs vs N (Check how to find the estimate of FLOPs)