1. Calculation of Fourier transformation in CUDA using Numba package

Write the CUDA kernel for calculating the Discrete Fourier transformation (DFT), where DFT calculates the contribution of k-th frequency as follows

$$\overline{S_k} = \sum_{n=0}^{N-1} s_n e^{-i\frac{2\pi}{N}nk}$$

Evaluate the following matrix equation

$$\begin{pmatrix} \overline{S_0} \\ \overline{S_1} \\ \overline{S_2} \\ \vdots \\ \overline{S_{N-1}} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^1 & \dots & W^{N-1} \\ 1 & W^2 & W^3 & \dots & W^{N-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & W^{N-1} & W^{N-2} & \dots & W^1 \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ \\ s_{N-1} \end{pmatrix}$$

where $W = e^{-i\frac{2\pi}{N}}$ using DFT or FFT variants.

- In case you choose DFT, implement any standard method for evaluation the matrix equation and construct the *W* matrix by yourself.
- In case you choose FFT, implement Cooley Tukey algorithm for fast evaluation of matrix equation.
- Compare the results with one of the options listed below
  - Numpy *fft* implementation
  - *cuFFT* CUDA function
  - *cupy.fft* from CuPy package

Deliver these graphical outputs:

- Original function and its spectra obtained from DFT, FFT and from Numpy fft.
- Comparison of scaling behaviour (Numpy, DFT, FFT) with increasing *N*.