FYSS5120 Efficient Numerical Programming - Demo 2

Drop solutions before the demo session to the Nexcloud box (link)
Please indicate clearly your name in the file name.
Commented Python scripts (.py or .ipy) and Jupyter notebooks (.ipynb) are fine.

1. **Find a root of the function**

   $$f(x) = x^3 \sin(x) \cos(x) \exp(-x)$$

   somewhere near $x = 1.0$ using iteration.
   A plot would be nice to see.
   Hint: $f(x) = 0 \Leftrightarrow x = x + f(x)$, iterate this starting from $x = 1.0$.

2. **Time NumPy `sum()` method vs. Python `sum()` function.**

   ```
   import numpy as np
   N = 10000000
   A = np.random.random(N)
   ```

   and compare the execution times of `A.sum()` and `sum(A)`.

3. Write a Python code that computes the distances of 1000 particles in three-dimensional space. The particle coordinates are in the NumPy `NxD` array `x`. The sample code `potential_simple.py`, in Python examples directory (link) shows how it could be done using NumPy broadcasting, but it's doens't perform well.

   Time the distance computations using, for example, `time.perf_counter`:

   (a) Compute the distance array (`r` in the sample code) using `numpy.linalg.norm()`.
   Link to documentation: `numpy.linalg.norm`

   (b) Compute `r` using `numpy.einsum()`

   ```
   r = np.sqrt(np.einsum('ijk,ijk->ij',d,d))
   ```

   (c) A version that uses `scipy.spatial.distance.pdist`.
   Link to documentation: `scipy.spatial.distance.pdist`

   (d) A basic `for`-loop, but with Numba `@njit` decorator,

   ```
   from numba import njit
   @njit
   def distance(x):
       for i in ...
           ...
       return r
   ```