**GitHub：**https://github.com/DayBeha/HPC\_assignment

### Exercise 1 - Cythonize the STREAM Benchmark

**Task 1.1:** Develop a Cython version of the STREAM benchmark. Make sure to define the ctypes for obtaining full performance.

**Task 1.2:** Plot the bandwidth results varying the arrays' size. Answer the question: how does the bandwidth measured with Cython code compare to bandwidth obtained in Assignment II.

**bandwidth before Cython:**

STREAM\_ARRAY\_SIZE = 5

Copy time use:6.00 us data amount:280 bandwith:466666666.65324736 /s

add time use:217.00 us data amount:280 bandwith:12903225.80644366 /s

scale time use:42.00 us data amount:420 bandwith:99999999.99976783 /s

triad time use:33.00 us data amount:420 bandwith:127272727.27549024 /s

STREAM\_ARRAY\_SIZE = 10

Copy time use:2.00 us data amount:560 bandwith:2799999999.919484 /s

add time use:22.00 us data amount:560 bandwith:254545454.53813493 /s

scale time use:7.00 us data amount:840 bandwith:1200000000.060655 /s

triad time use:19.00 us data amount:840 bandwith:442105263.1580984 /s

STREAM\_ARRAY\_SIZE = 50

Copy time use:2.00 us data amount:2800 bandwith:13999999999.597422 /s

add time use:16.00 us data amount:2800 bandwith:1749999999.9496777 /s

scale time use:6.00 us data amount:4200 bandwith:6999999999.798711 /s

triad time use:20.00 us data amount:4200 bandwith:2100000000.0561864 /s

STREAM\_ARRAY\_SIZE = 100

Copy time use:2.00 us data amount:5600 bandwith:27999999999.194843 /s

add time use:18.00 us data amount:5600 bandwith:3111111111.021649 /s

scale time use:6.00 us data amount:8400 bandwith:13999999999.597422 /s

triad time use:22.00 us data amount:8400 bandwith:3818181818.264707 /s

STREAM\_ARRAY\_SIZE = 500

Copy time use:4.00 us data amount:28000 bandwith:69999999997.9871 /s

add time use:20.00 us data amount:28000 bandwith:14000000000.374577 /s

scale time use:8.00 us data amount:42000 bandwith:52499999998.490326 /s

triad time use:25.00 us data amount:42000 bandwith:16799999999.88994 /s

STREAM\_ARRAY\_SIZE = 1000

Copy time use:4.00 us data amount:56000 bandwith:139999999995.9742 /s

add time use:24.00 us data amount:56000 bandwith:23333333333.741753 /s

scale time use:10.00 us data amount:84000 bandwith:83999999997.58452 /s

triad time use:28.00 us data amount:84000 bandwith:30000000000.326855 /s

STREAM\_ARRAY\_SIZE = 5000

Copy time use:4.00 us data amount:280000 bandwith:700000000174.16 /s

add time use:41.00 us data amount:280000 bandwith:68292682925.7901 /s

scale time use:17.00 us data amount:420000 bandwith:247058823530.3748 /s

triad time use:36.00 us data amount:420000 bandwith:116666666666.90979 /s

STREAM\_ARRAY\_SIZE = 10000

Copy time use:3.00 us data amount:560000 bandwith:1866666666267.5867 /s

add time use:48.00 us data amount:560000 bandwith:116666666668.70876 /s

scale time use:25.00 us data amount:840000 bandwith:335999999997.79877 /s

triad time use:47.00 us data amount:840000 bandwith:178723404256.51248 /s

STREAM\_ARRAY\_SIZE = 50000

Copy time use:6.00 us data amount:2800000 bandwith:4666666666532.474 /s

add time use:116.00 us data amount:2800000 bandwith:241379310344.8172 /s

scale time use:97.00 us data amount:4200000 bandwith:432989690721.58813 /s

triad time use:138.00 us data amount:4200000 bandwith:304347826085.5503 /s

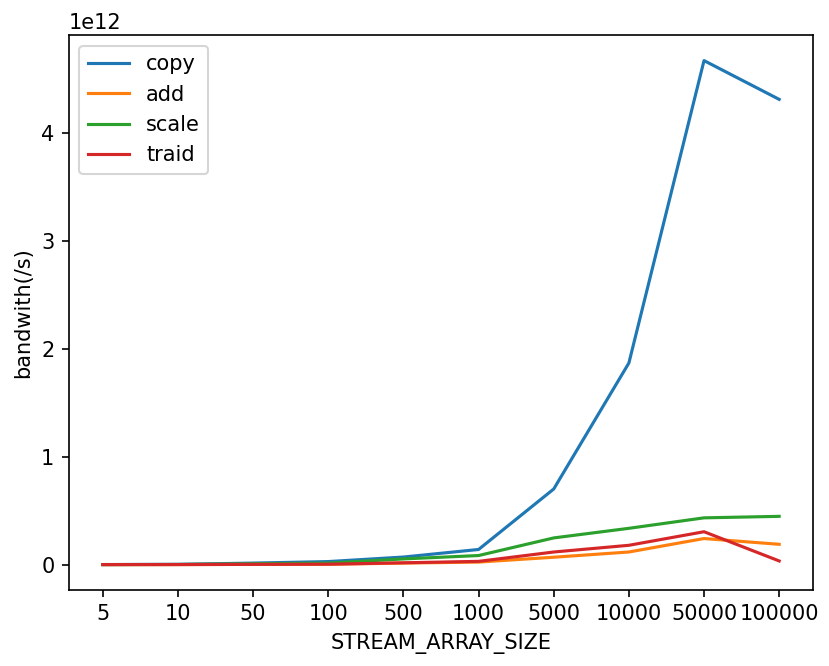
STREAM\_ARRAY\_SIZE = 100000

Copy time use:13.00 us data amount:5600000 bandwith:4307692307384.4946 /s

add time use:298.00 us data amount:5600000 bandwith:187919463087.44543 /s

scale time use:188.00 us data amount:8400000 bandwith:446808510638.6426 /s

triad time use:2489.00 us data amount:8400000 bandwith:33748493370.832153 /s



**bandwidth after Cython:**

STREAM\_ARRAY\_SIZE = 5

Copy time use:111.00 us data amount:280 bandwith:25225225.225382917 /s

add time use:14.00 us data amount:280 bandwith:199999999.99424887 /s

scale time use:22.00 us data amount:420 bandwith:190909090.9036012 /s

triad time use:14.00 us data amount:420 bandwith:300000000.0151638 /s

STREAM\_ARRAY\_SIZE = 10

Copy time use:13.00 us data amount:560 bandwith:430769230.7384495 /s

add time use:8.00 us data amount:560 bandwith:699999999.979871 /s

scale time use:10.00 us data amount:840 bandwith:840000000.069104 /s

triad time use:10.00 us data amount:840 bandwith:839999999.9758452 /s

STREAM\_ARRAY\_SIZE = 50

Copy time use:10.00 us data amount:2800 bandwith:2799999999.919484 /s

add time use:8.00 us data amount:2800 bandwith:3500000000.385078 /s

scale time use:10.00 us data amount:4200 bandwith:4199999999.879226 /s

triad time use:10.00 us data amount:4200 bandwith:4200000000.34552 /s

STREAM\_ARRAY\_SIZE = 100

Copy time use:11.00 us data amount:5600 bandwith:5090909091.019609 /s

add time use:8.00 us data amount:5600 bandwith:6999999999.798711 /s

scale time use:10.00 us data amount:8400 bandwith:8399999999.758452 /s

triad time use:10.00 us data amount:8400 bandwith:8399999999.758452 /s

STREAM\_ARRAY\_SIZE = 500

Copy time use:13.00 us data amount:28000 bandwith:21538461538.761898 /s

add time use:10.00 us data amount:28000 bandwith:27999999999.194843 /s

scale time use:15.00 us data amount:42000 bandwith:28000000000.23105 /s

triad time use:15.00 us data amount:42000 bandwith:28000000000.23105 /s

STREAM\_ARRAY\_SIZE = 1000

Copy time use:17.00 us data amount:56000 bandwith:32941176470.716637 /s

add time use:14.00 us data amount:56000 bandwith:40000000002.02184 /s

scale time use:22.00 us data amount:84000 bandwith:38181818182.64707 /s

triad time use:22.00 us data amount:84000 bandwith:38181818182.64707 /s

STREAM\_ARRAY\_SIZE = 5000

Copy time use:49.00 us data amount:280000 bandwith:57142857143.156044 /s

add time use:43.00 us data amount:280000 bandwith:65116279068.7356 /s

scale time use:74.00 us data amount:420000 bandwith:56756756756.82772 /s

triad time use:74.00 us data amount:420000 bandwith:56756756756.82772 /s

STREAM\_ARRAY\_SIZE = 10000

Copy time use:86.00 us data amount:560000 bandwith:65116279069.576225 /s

add time use:81.00 us data amount:560000 bandwith:69135802469.51677 /s

scale time use:142.00 us data amount:840000 bandwith:59154929577.613754 /s

triad time use:142.00 us data amount:840000 bandwith:59154929577.613754 /s

STREAM\_ARRAY\_SIZE = 50000

Copy time use:685.00 us data amount:2800000 bandwith:40875912408.74309 /s

add time use:1407.00 us data amount:2800000 bandwith:19900497512.438717 /s

scale time use:673.00 us data amount:4200000 bandwith:62407132243.69207 /s

triad time use:671.00 us data amount:4200000 bandwith:62593144560.37016 /s

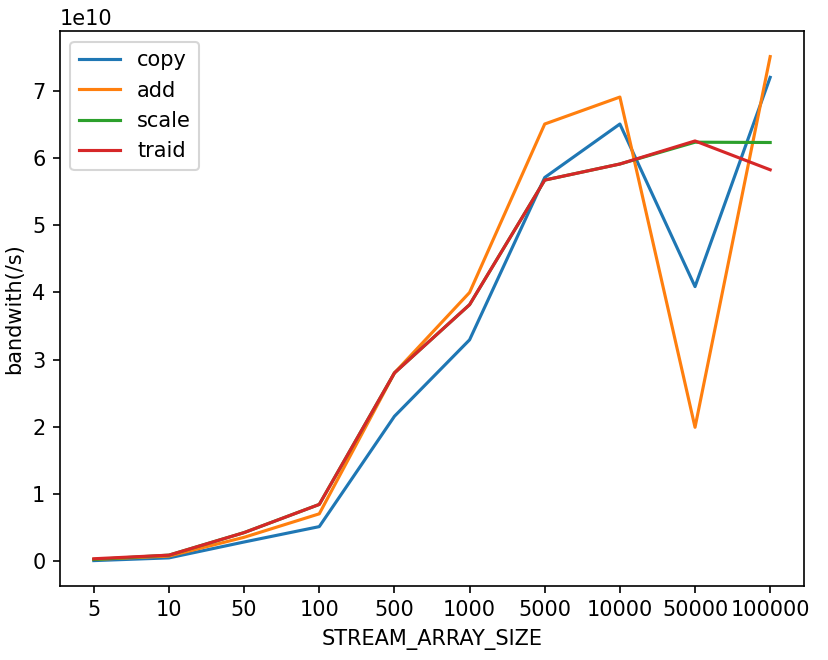
STREAM\_ARRAY\_SIZE = 100000

Copy time use:777.00 us data amount:5600000 bandwith:72072072072.1107 /s

add time use:745.00 us data amount:5600000 bandwith:75167785234.92216 /s

scale time use:1347.00 us data amount:8400000 bandwith:62360801781.71723 /s

triad time use:1441.00 us data amount:8400000 bandwith:58292852185.98991 /s



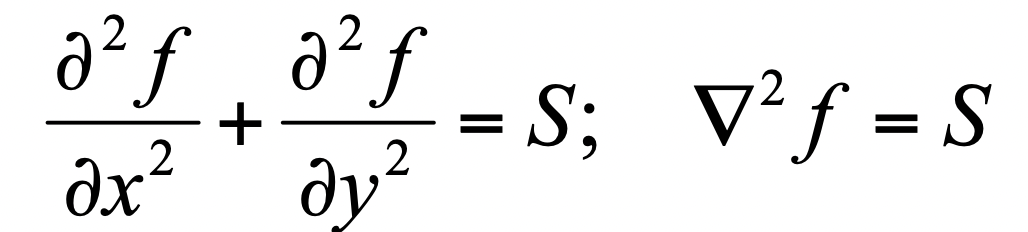
**Code:**

#cython: language\_level=3  
  
import numpy as np  
cimport numpy as np  
  
def copy(float[:] a, float[:] b):  
 # cdef float[:] c = np.empty(len(a), dtype=np.float32)  
 for i in range(len(a)):  
 b[i] = a[i]  
 # return c  
  
  
def scale(float scalar, float[:] a, float[:] b):  
 # cdef float[:] b = np.empty(len(a), dtype=np.float32)  
 for i in range(len(a)):  
 b[i] = scalar \* a[i]  
 # return b  
  
def add(float[:] a, float[:] b, float[:] c):  
 # cdef float[:] c = np.empty(len(a), dtype=np.float32)  
 for i in range(len(a)):  
 c[i] = a[i] + b[i]  
 # return c  
  
def triad(float[:] a, float scalar, float[:] b, float[:] c):  
 # cdef float[:] c = np.empty(len(a), dtype=np.float32)  
 for i in range(len(a)):  
 c[i] = a[i] + scalar \* b[i]  
 # return c

Conclusion: As the plots shows, the performance after Cython is no better than before which not as expectation. I both tried pass variable to functions in \*.pyx and define a new array. No better performance appear.

### Exercise 2 - Gauss-Seidel for Poisson Solver

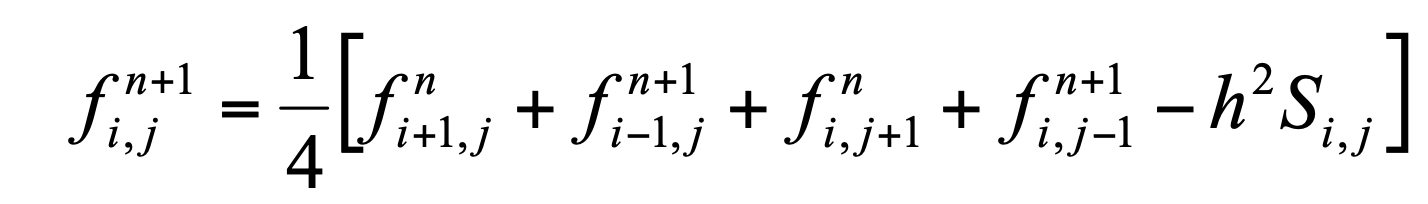
In this exercise, we develop and optimize the Gauss-Seidel solver for solving the 2D Poisson Equation:



We solve Poisson's equation by it to convergence by a forward time-centered space differencing (FTCS) on a grid using the Gauss-Seidel Method:

* We discretize the function f on a square box of size 1.
* We use a uniform grid with N grid points in the x-direction and N grid points in the y-direction.
* We impose the values at the boundary equal to zero.
* We have no source S
* We initialize the simulation with random numbers

The Gauss-seidel iteration for the Poisson Equation in 2D is:



In a Python code a Gauss-Seidel iteration can be written as follow:

def gauss\_seidel(f):

newf = f.copy()

for i in range(1,newf.shape[0]-1):

for j in range(1,newf.shape[1]-1):

newf[i,j] = 0.25 \* (newf[i,j+1] + newf[i,j-1] +

newf[i+1,j] + newf[i-1,j])

return newgrid

, where the grid values at the boundaries are fixed to zero, and no source is included.

Then for running the 1,000 iterations:

for i in range(1000):

x = gauss\_seidel(x)

**Task 2.1:** Develop the Gauss-Seidel solver with Python List, array, or NumPy. Plot the performance varying the grid size.

**Task 2.2:** Profile the code to identify the part of the code to optimize. You can use the tool of your choice.

**Task 2.3:** Use the Cython Annotation tool to identify the parts to use Cython

**Task 2.4:** Use Cython to optimize the part you identified as the most computationally expensive. Compare the performance with the results obtained in Task 2.1.

**Task 2.5:** Use PyTorch to port your code to Nvidia GPUs. For this you will need to express the two nested loops operations as numpy roll operations in 2D as we did for the diffusion code. If you don't have an Nvidia GPU, you can use the Google Colab, as in [3.9.1 Tutorial: Use Google Colab GPUs for PyTorch+GPU3.9.1 Tutorial: Use Google Colab GPUs for PyTorch+GPU](https://canvas.kth.se/courses/37558/pages/3-dot-9-1-tutorial-use-google-colab-gpus-for-pytorch+gpu?wrap=1))

**Task 2.6:** Use CuPy to port your code to Nvidia GPUs. See [C.3 Tutorial - Introduction to CuPyC.3 Tutorial - Introduction to CuPy](https://canvas.kth.se/courses/37558/pages/c-dot-3-tutorial-introduction-to-cupy?wrap=1)

**Task 2.7:**Measure the performance (execution time) with GPU (PyTorch and cupy) and make a plot of the execution time varying the size the of the grid. Compare and comment of the performance difference with and without GPU.

**Task 2.8:**Save the newgrid matrix as an hdf5 file using h5py  (see the tutorial [C.4 Tutorial - The HDF5 Library/ModuleC.4 Tutorial - The HDF5 Library/Module](https://canvas.kth.se/courses/37558/pages/c-dot-4-tutorial-the-hdf5-library-slash-module?wrap=1))

**Bonus Task B.1:** Develop the function you want to optimize in C/C++ or Fortran, make it a library, and couple C and Python. You can use the approach of your choice (see the lecture on foreign function interfaces). This is part of the bonus exercise.

### Bonus Exercise - Get familiar with VTK and Paraview

In this exercise, we ask you to experiment with one of the most used tools to visualize datasets coming from HPC simulations, Paraview. One of the most common and most used formats for visualization is called VTK.

As part of the exercise, you will need to repeat all the different steps presented in the [C.6 Tutorial - VTK &amp; Visualization with ParaviewC.6 Tutorial - VTK &amp; Visualization with Paraview](https://canvas.kth.se/courses/37558/pages/c-dot-6-tutorial-vtk-and-visualization-with-paraview?wrap=1)

To install sputniPIC follow the instructions in the tutorial for paraview:

1. Install the **mpich and hdf5 libraries using spack**
2. Install the SputniPIC code
3. When you run sputniPIC, the code will create a number of VTK files in the data folder (be sure to create a directory called data otherwise, you will get an error and you can't run sputniPIC)
4. Download and install paraview on your system
5. Use paraview for the visualization

**Task 3.1:** Make a volume plot and a slice of the rhoe at the last step recorded in the vtk file as shown in the tutorial. Add the snapshots you took to your report.