CONSERVATION LAWS ON GPUS
EXERCISES

André R. Brodtkorb
Researcher, Department of Mathematics and Cybernetics, SINTEF Digital
Associate Professor, OsloMet – Oslo Metropolitan University

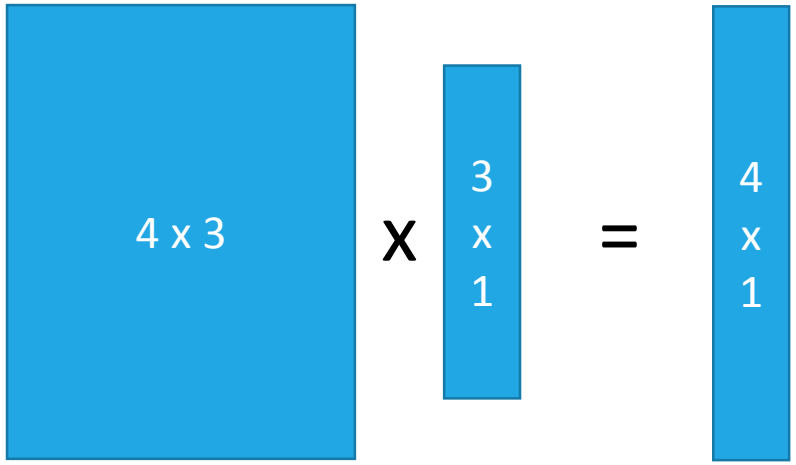
Exercise 1: Matrix addition

1. Make a directory that is your own, call it <your-username>
2. Make a copy of the Jupyter notebook "01 HelloPyCuda.ipynb"
3. Move the copy to <your-username>/MatrixAddition.ipynb
4. Change the code to add two matrices instead of vectors
   - Hint: CUDA uses blockIdx.x*blockDim.x + threadIdx.x; to get the global x coordinate. How do you think we get the y coordinate?
   - Hint: You can calculate the linear address of a 2d element using j*cols+i
5. Repeat the exercise with PyOpenCL
   - Hint: OpenCL uses get_global_id(0); to get the x coordinate.

Exercise 2: Matrix-vector product

1. Make a copy of the Jupyter notebook "01 HelloPyCuda.ipynb"
2. Move the copy to <your-username>/MatrixVector.ipynb

- Implement matrix-vector product to multiply an m x n matrix by a n x 1 vector
  - Hint: Create one thread per output (4x1), and let each tread calculate its own result/sum



Exercise 3: Computing Pi

1. Make a copy of the Jupyter notebook "ComputePi.ipynb"
2. Move the copy to <your-username>/ComputePi.ipynb
3. Start implementing the CUDA kernel for computing Pi.
   - Hint: Parts where you need to change and implement things are hilighted
4. Create a new function
   def computePi2GPU(n_points)
   which implements version 2 of the code.
   Hint: Do you also perhaps need to create a new kernel?
5. What is the performance difference?
   Hint: How many points can you sample with the two versions?
   Hint: You can get the current time using
   import time
   tic = time.time()
   <timer her>
   elapsed = time.time() – tic
6. Continue with version 3 etc.

SINTEF