# Exercise 3

# Parallel & Distributed Computer Systems

# Dec 8, 2019

Implement in `CUDA` the evolution of an Ising model in two dimensions for a given number of steps $k$.

The Ising model (/ˈaɪsɪŋ/; German: [ˈiːzɪŋ]), named after the physicist Ernst Ising, is a mathematical model of ferromagnetism in statistical mechanics. The model consists of discrete magnetic dipole moments of atomic "spins" that can be in one of two states (+1 or −1). The spins are arranged in a square lattice with periodic boundary conditions, allowing each spin to interact with its neighbors. The dipole moments update in discrete time steps according to the majority of the spins within the $3 \times 3$ window centered to each lattice point. Windows centered on the edge lattice points wrap around to the other side (known as toroidal or periodic boundary conditions).

In this homework, we modify the standard model in the following two ways.

1. The neighborhood is a $5 \times 5$ window centered to each lattice point.
2. The influence of the neighbors is a weight matrix $w$ that is inversely analogous to the distance and it holds that
   - $w_{i,j} > 0,\ \forall i,\ j \in \{-2, -1, 0, 1, 2\}^2$,
   - the weight $w_{0,0} is undefined, and
   - $\sum_{i,j} w_{i,j} = 1$.
3. The magnetic moment gets the value of the sign of the weighted influence of its neighbors, or remains the same in the case that the weighted influence is zero. *Make sure you take into account possible floating point errors, when checking if the value is zero.*

Follow the specifications shown in the online tester

## V0. Sequential

Write the sequential version, in `C` to simulate an Ising model of size `nxn` for `k` iterations, staring from a uniform random initial state. The size `n` and number of iterations `k` are given. Make sure to avoid `if` statements for the boundaries. Use two arrays, read from one and write to the other, then swap the pointers for the next iteration. Terminate if no changes are made. Persuade yourself and us that your code works correctly!

## V1. GPU with one thread per moment

Modify the sequential code to make a kernel function and call the kernel with a grid that matches the Ising model and one thread per moment. Confirm that works correctly by matching it to the result of the sequential run.

## V2. GPU with one thread computing a block of moments

Assign more work per thread, but in preparation for the next request, make each thread to compute a block of moments. Confirm that works correctly by matching it to the result of the sequential run.

## V3. GPU with multiple thread sharing common input moments

Use shared memory so that threads do not read multiple times from the main memory but from the shared memory. Confirm that works correctly by matching it to the result of the sequential run.

## What to submit

- A 3-page report in PDF format (any pages after the 3rd one will not be taken into account). Report execution times of your implementations with respect to size $n$.
- Upload the source code on GitHub, BitBucket, Dropbox, Google Drive, etc. and add a link in your report.
- Check the validation of your code using the automated tester on e-learning.