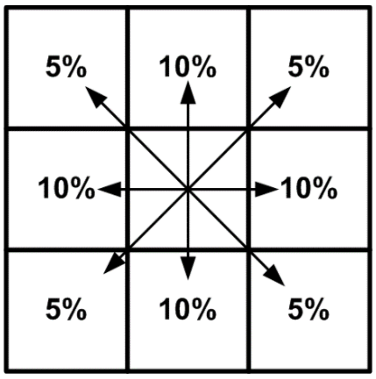
Project topic
Iterative 1D convolution

Heat flow (for instance in chips) can be simulated with the **HotSpot algorithm** [W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron,and M.R. Stan. HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design. IEEE Trans. VLSI Syst., 14(5), 2006].

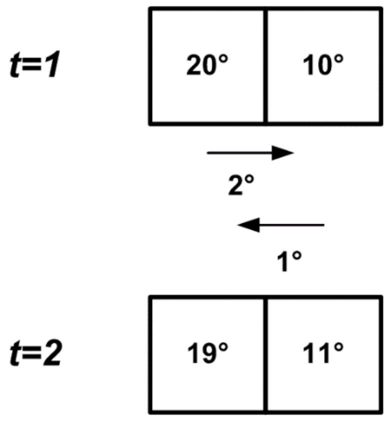
A 2D surface is partitioned into cells where each cell has a certain temperature. We simulate the heat flow in a discrete way. Each time step, part of the heat will flow to the neighbouring cells:



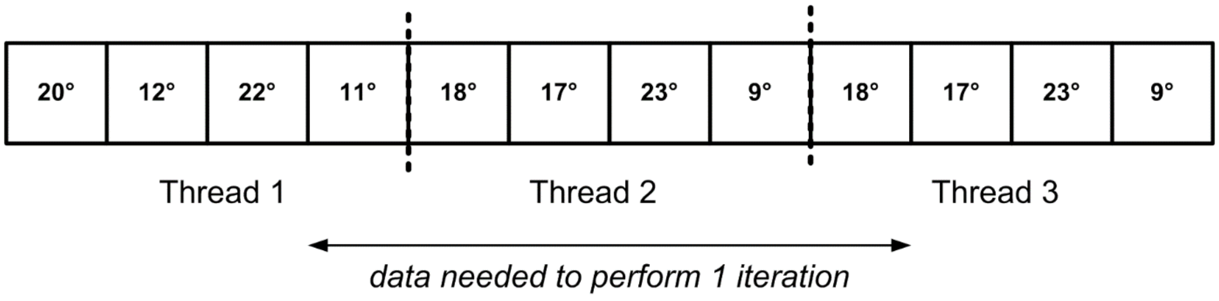
Calculating the dispersion of heat for each cell is similar to a convolution with the following fitler:

0.05	0.1	0.05
0.1	0.4	0.1
0.05	0.1	0.05

For simplicity, let’s do it in 1D. Two neighboring cells exchange heat, the resulting temperature is:

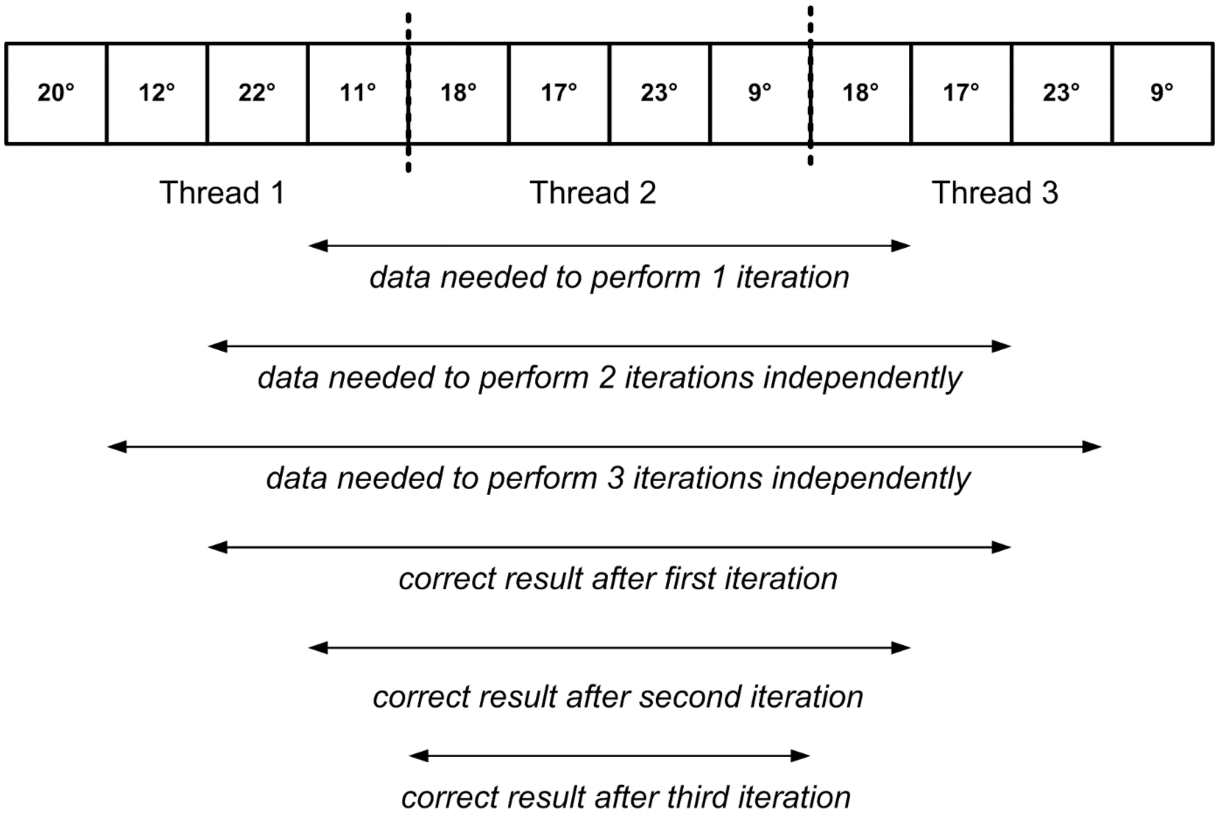


To execute the simulation in parallel, the cell array is partitioned, and each thread is responsible for the correct simulation of one subarray. However, it needs to know the temperature of border cells:

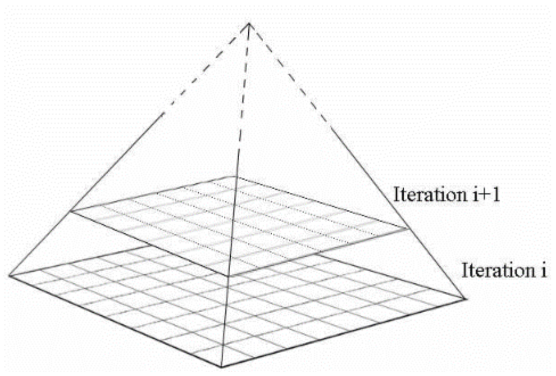


In this way, 1 time step can be calculated in parallel, after which the threads should synchronize their data before the next iteration can start.

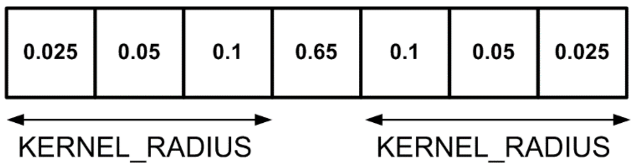
The hotspot algorithm enables more iterations to be executed independently by taking more border cells as starting data:


20°
12°
22°
11°
18°
17°
23°
9°
18°
17°
23°
9°
Thread 1
Thread 2
Thread 3
data needed to perform 1 iteration
data needed to perform 2 iterations independently
data needed to perform 3 iterations independently
correct result after first iteration
correct result after second iteration
correct result after third iteration

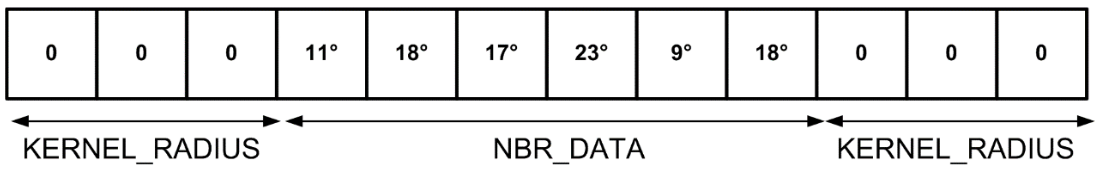
As shown in the figure, if three border cells (at each side) are added to the thread's subarray, the thread can calculate three iterations independently. Starting with n cells, after each iteration, the 2 outer cells should no longer be considered since not correct. So, after 3 iterations, the initial subarray is simulated correctly. In 2D this creates a "pyramidal effect":



The given source code is an iterative 1D convolution with variable kernel size (more than 1 border cell is considered). The convolution kernels can be larger than 3. Here a kernel with radius of 3:



To alleviate border effects (kernel going outside the data on the left and right), we *pad* the data array on the left and right with zeroes:



The data is randomly generated. A moving heat source is added which inserts heat into the system.

Optimize and parallelize the sequential algorithm in various ways. The output is a performance report. Indicate which optimization strategies work.
What is the effect of parameters NBR_DATA and KERNEL_RADIUS on the speedup?

Implement at least 3 versions:

1. Synchronize all threads globally after each time step.
2. Loosely synchronize after each time step: a thread waits for another thread if it needs its data (here: neighbors). As soon as the data is up-to-date, the thread continues. It does not wait for other threads.
3. Implement the hotspot idea so that several (= parameter) iterations can be performed independently.

Check for each version whether the result is the same as the sequential version!