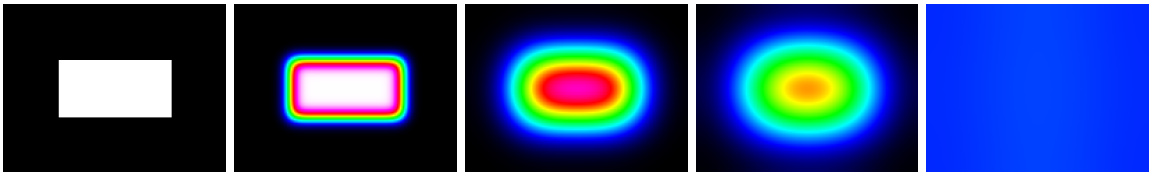
Distributing the Heat Equation

Jean-Marie Madiot, Julien Herrmann
jeanmarie.madiot@ens-lyon.fr
julien.herrmann@ens-lyon.fr

This homework is due Sunday, December 07, 23h59, Lyon time. Send your archive to both email addresses.

1 Heat equation

The heat equation (for information, $\frac{\partial x}{\partial t} = \nabla^2 x$) is a local rule allowing one to compute the evolution of temperature of a system over time. In this homework, we will implement *average cellular automata* that behave similarly:



The first picture corresponds to a plate with a rectangular distribution of temperature, the following ones show how the temperature evolves over time, to finally converge to a uniform distribution.

2 Cellular automata

A cellular automaton is a quadruplet $\mathcal{A} = (d, Q, r, \delta)$ where

- $d \in \mathbb{N}$ is the dimension,
- Q is a set, the states,
- $r \in \mathbb{N}$ is the radius,
- $\delta : Q^{[\![-r,r]\!]^d} \to Q$ is the local rule[1].

The cellular automaton is defined on a grid of cells of $\mathbb{Z}^d$. Each cell has a state in Q. The local function δ is the rule to apply to get the next state of a cell, depending on its neighborhood. The global function $\delta^\dagger$ computes the next step for all the states of the whole grid, according to the rule δ.

To formalize this, a configuration for $\mathcal{A}$ is an element $X \in Q^{\mathbb{Z}^d}$ and δ lifts to a global function on the set of configurations: $\delta^\dagger : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$, such that $\delta^\dagger$ is invariant under translation each in all the d directions and coincides with δ at the origin (i.e. $\delta^\dagger(X)_{0,0,..,0} = \delta(X|_{[\![-r,r]\!]^d})$).

We focus on the sequence $(X^t)_{t\in\mathbb{N}}$ where X^t is the state of the grid at time t: $X^{t+1} = \delta^\dagger(X^t)$. The sequence depends on its initial state X^0.

[1] When a and b are integers, $[\![a, b]\!]$ represents the set $\{a, a+1, \ldots, b\}$.

That being said, we will consider in this homework 2-dimensional periodic grids of period N in each direction, which can be thought as finite grids $\mathcal{G} = [\![0, N-1]\!]^2$, and the radius of the automata will be 1, meaning that for every cell coordinates $(i, j) \in \mathcal{G}$ we will consider as its neighborhood the set of cells of coordinates $(i-1, j-1)$, $(i, j-1)$, $(i+1, j-1)$, $(i, j-1)$, (i, j), $(i, j+1)$, $(i+1, j-1)$, $(i+1, j)$, $(i+1, j+1)$ and we handle the edges by considering $\mathcal{G}$ as a torus. This means that, given the values $X^t = (x^t_{i,j})_{(i,j)\in\mathcal{G}}$ at step t, the next step $X^{t+1} = \delta^\dagger(X^t)$ is defined as follows:

$$\boxed{(\delta^\dagger(X^t))_{i,j}} \triangleq \delta\left(\begin{array}{|c|c|c|}\hline x^t_{i-1,j+1} & x^t_{i,j+1} & x^t_{i+1,j+1} \\ \hline x^t_{i-1,j} & x^t_{i,j} & x^t_{i+1,j} \\ \hline x^t_{i-1,j-1} & x^t_{i,j-1} & x^t_{i+1,j-1} \\ \hline\end{array}\right)$$

Question 1. How many applications of the function δ are necessary to compute X^t on $[\![0, N-1]\!]^2$?

Question 2. How would you implement this 2D cellular automaton on a network with a 2D toric grid topology? Be careful to explain where each cell of data is stored, and where the computation of its next state is performed.

Question 3. Write a distributed algorithm that computes X^{t+1}, given X^t distributed on a toric 2D grid the way you decided in the previous question. What are the time and communication costs of this algorithm? Can you adapt it to a non-toric grid? What would be the complexity on a ring topology?

3 Average automata

Definition 1. *If $p \neq 0$, the* p-average automaton *is the cellular automaton where $Q = \mathbb{R}$ and δ is:*

$$\delta\left(\begin{array}{|c|c|c|}\hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline\end{array}\right) \triangleq (1-p)\cdot e + p\cdot\frac{b+d+f+h}{4} \quad .$$

In other terms, the new value of a cell is the average of the old values of cells inside its immediate neighborhood, weighed by p, and of the old value of the cell itself, weighed by $1-p$.

Question 4. In the file `average.c`, implement on a toric grid the algorithm of Question 3 for any p-average automata. See Section 6 for implementation details.

When X and Y are two configurations of the grid $\mathcal{G}$, we write $X+Y$ the state of $\mathcal{G}$ defined by the point-wise sum of the two states: $(X+Y)_{i,j} = X_{i,j} + Y_{i,j}$. Similarly, for every $k \in \mathbb{R}$ the state $k \cdot X$ is defined as: $(k \cdot X)_{i,j} = kX_{i,j}$. A function f is linear if $\forall X, Y, k$, $f(X+Y) = f(X) + f(Y)$ and $f(k \cdot X) = k \cdot f(X)$.

When computing complexity bounds, we assume that basic operations on real numbers (such as sum, product, ...) cost 1 unit of time. For the space complexity, storing one real number takes one unit of memory. Finally, we assume that sending one real number from one processor to another takes c units of time (as usual, independent communications can be performed in parallel).

Question 5. In the case of a *p-average automaton*, prove that $\delta^\dagger$ is linear. Use that property to derive an algorithm that, given X^0, computes X^t in time $O(\log t)$ for a fixed N. What is the time and space complexity, in terms of both t and N? Compare to the complexity in the general case.

Question 6. Describe a distributed variant of the above algorithm on a $p \times p$ processors fully connected network. What are the new time, space, and communication costs?

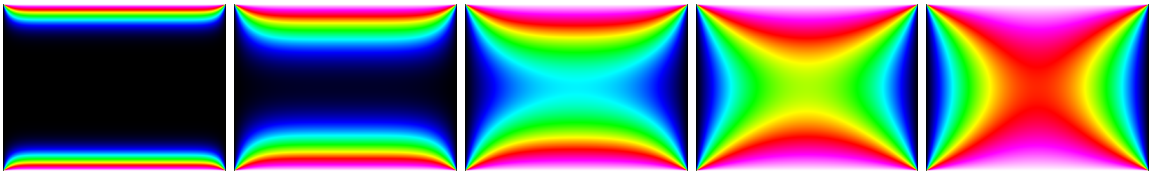
Question 7 (Sparse initial conditions)**.** Let Z be the state of the grid such that $Z_{i,j}$ is 1 when $i = j = 0$ and 0 otherwise, and let assume that Z^t is *already computed* and distributed on a $p \times p$ grid. Given an arbitrary *sparse* state X^0 of the grid (i.e., $|X^0|$, the number of non-zero cells in X^0, is small), how can you compute X^t on the $p \times p$ fully connected grid? What are, the time/space/communication complexities in terms of N and $|X^0|$?

Question 8. In the file `sparse.c`, implement Question 7 on a fully connected grid: first, given t and the dimensions of Z, precompute Z^t using Question 4. Then, provide the user with an interface that lets her update X^0 by entering $(i, j, x^0_{i,j})$ one by one, starting from $X^0 = 0$ and keep X^t updated at each new entry.

Question 9. Give some fixed points of $\delta^\dagger$ in $\mathbb{R}^{\mathbb{Z}^d}$ and in $\mathbb{R}^{[\![0,N-1]\!]^d}$ (note that the latter is a torus).

4 Thermal reservoirs

A thermal reservoir is a thermodynamic system with constant, unalterable temperature, influencing in turn the temperature of the environment. The pictures below describe the evolution of a plate with warm thermal reservoirs on the horizontal edges and cold thermal reservoirs on the vertical edges.



We suppose now $0 < p < 1$. We augment Q with special states: $Q = \mathbb{R} \uplus \mathcal{C}$ where $\mathcal{C}$ a set of special states called *constants* written as C_k where k is a real number. Now the notion of average automaton is refined, so that a cell in state C_k stays in the same state forever, and is considered of temperature k by its neighbors.

Formally, we define the temperature $\theta(x)$ of a state as $\theta(x) = k$ when $x = C_k$ and $\theta(x) = x$ otherwise. The local rule becomes the following function:

$$\delta\left(\begin{array}{|c|c|c|}\hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline\end{array}\right) \triangleq \begin{cases} C_k & \text{if } e = C_k \ , \\ (1-p)\cdot\theta(e) + p\cdot\frac{\theta(b)+\theta(d)+\theta(f)+\theta(h)}{4} & \text{otherwise .} \end{cases}$$

Question 10. Give an example of a fixed point with one constant, and another with two different constants. Give the limit of (X^t), if it exists, given that X^0 has at most one constant.

Question 11. For a p-average automaton with constants, is $\delta^\dagger$ linear?

Question 12 (Bonus)**.** Prove that the sequence (X^t) converges when X^0 has at least one constant. (We consider real-valued cells. It does not work in a discrete setting: give an counterexample of X^0 yielding a cycle with `double`-valued cells.)

Question 13. In the file `constants.c`, adapt Question 4 to the setting of Section 4.

5 Graphical implementation

Question 14. Use the display of the first processor to provide the user with a simple graphic interface, featuring a $[0, 1]$-valued heat map:

- for Question 8: where the interface is a black, cold map and the user can click to set some $X^0_{i,j}$ to 1.
- for Question 13: given a map of C_0 and C_1 constants, compute and display the fixed point in a distributed setting.

You can use Douglas Thain's simple X11 graphics library.
`http://www3.nd.edu/~dthain/courses/cse20211/fall2013/gfx/`

6 Implementation details

Use the C programming language. Using any other language including C++ is likely to halve your mark.

Your programs must compile (with `mpicc`) and run (with `mpirun`) on the machines of the ENS (`slsu0-01`, `slsu0-02` etc.). Provide a `name.tar.gz` archive in which you have a folder `name` where `name` is your name. Provide a makefile, such that running `make` at the root of `name` will compile everything and create all executable binaries at the root of `name`. Use `double` to represent real numbers.

Input Input data is given via standard input. Note that you can use standard input with `./myprogram` and then type things manually, or with `./myprogram < file` to use the file `file` a standard input. The format is the following:

- width of the grid on first line of standard input
- height on second line
- p on third line
- t the number of iterations on the fourth
- then, the initial grid, described as a sparse matrix on each line with its coordinates:
  - `0 i j x` is **input data** for a value `x`$\in [0,1]$ with coordinates $(\texttt{i}, \texttt{j})$
  - `1 i j k` is **input data** of a constant `k`$\in [0,1]$ with coordinates $(\texttt{i}, \texttt{j})$
  - `2 i j 0` is a **request** to give the value at position $(\texttt{i}, \texttt{j})$ after t iterations.

Output Answer to each of the `2 i j 0` requests by printing the temperature of the corresponding cell. Note that:

- in the sparse case, requests and input data can be interleaved (and constants will be ignored).
- in the other cases, the first request triggers the computation (subsequent input data will be ignored).