Parallel and Distributed Algorithms and Programs
Project - Distributing the vibrating string equation

Hadrien Croubois
hadrien.croubois@ens-lyon.fr

Aurélien Cavelan
aurelien.cavelan@ens-lyon.fr

All documents are available on my website : `http://hadriencroubois.com/#Teaching`

This poject is due for december 4, 2015, 23h59, Lyon's time (UTC+1). Send your archive to both our emails.

Part 1

Physics background

Vibrating string equation

The vibrating string equation models the propagation of a wave on string or membrane. This differential equation links the temporal variation of the local position (u) to the current distribution.

$$\frac{\mu}{T}\frac{\partial^2 u}{\partial t^2} = \Delta u$$

With T the tension and μ the linear mass of the medium. This equation can be reformulated as :

$$\frac{1}{v^2}\frac{\partial^2 u}{\partial t^2} = \Delta u, \tag{1}$$

with $v = \sqrt{\frac{T}{\mu}}$ the speed of sound in the medium. This project's objective is to simulate this equation on a 2D surface using a dedicated cellular automaton that behaves similarly.

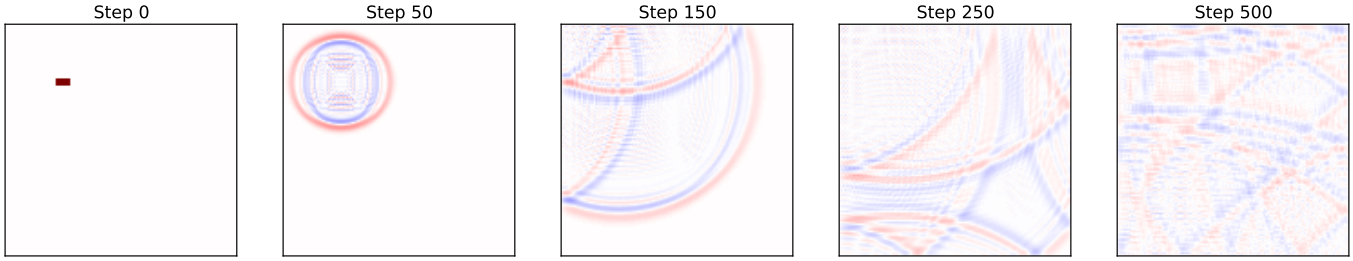

FIGURE 1 – Spreading of the shock-wave induced by an initial perturbation of the medium on a non toric grid (with walls)

Euler's approximation of differential equations

Euler's method provides a way, given a simple differential equation $f'(t) = g(f(t))$, to very simply simulate the behavior of f step by step starting from our initial condition.

$$\begin{aligned} f(t+\mathrm{d}t) &= f(t) + f'(t) \times \mathrm{d}t \\ &= f(t) + g(f(t)) \times \mathrm{d}t \end{aligned}$$

In our case we have a second order differential equation, meaning that we will have to apply this schema twice.

$$\begin{aligned} u(t+\mathrm{d}t) &= u(t) + u'(t) \times \mathrm{d}t \\ u'(t+\mathrm{d}t) &= u'(t) + u''(t) \times \mathrm{d}t \\ &= u'(t) + v^2 \times \Delta u(t) \times \mathrm{d}t \end{aligned}$$

Be carefull with the value of dt ! This method is computationnaly unstable and using a large value for dt will produce very bad results.

Question n°1

a) Give an example of differential equation and dt where applying this algorithm would lead to unacceptable results.

b) For a differential equation of order n, i.e. a differential equation involving the n-th derivative, what values should be stored to compute the next step ?

Part 2

Cellular automata

Model

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

— $d \in \mathbb{N}$ is the dimension,

— Q is the set of possible states,

— $r \in \mathbb{N}$ is the radius,

— $\delta : Q^{[\![-r,r]\!]^d} \to Q$ is the local transition function.

The cellular automaton is defined on a grid of cells of $\mathbb{Z}^d$. Each cell has a state in Q. The transition function δ is used to compute the next state of a cell, given the current state of cells in the local neighborhood. The global transition function $\bar{\delta}$ extrapolate the local transition in that it compute the state of the whole grid.

Formally, a configuration for $\mathcal{A}$ is an element $X \in Q^{\mathbb{Z}^d}$ and

$$\bar{\delta} : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$$

is the transition function that is invariant under translation in each of all the d dimensions and that coincides with δ at the origin : $\bar{\delta}(X)_{0,0,\dots,0} = \delta(X|_{[\![-r,r]\!]^d})$. The evolution of the world, as modeled by a cellular automaton, is the sequence $(X^t)_{t\in\mathbb{N}}$ where X^t is the world after t time-steps : $X^{t+1} = \bar{\delta}(X^t)$. This sequence is depends only on the local transition function δ and on the initial state of the world X^0.

In this assignment, we consider a 2D periodic grid of dimension $N \times M$, which can be seen as a finite grid $\mathcal{G} = [\![0, N-1]\!] \times [\![0, M-1]\!]$. We also restrict ourselves to a radius of 1, meaning that for every cell $(i, j) \in \mathcal{G}$ we consider a neighborhood that includes :

$$\begin{array}{ccc} (i-1,j-1) & (i,j-1) & (i+1,j-1) \\ (i-1,j) & (i,j) & (i+1,j) \\ (i-1,j+1) & (i,j+1) & (i+1,j+1) \end{array}$$

and we handle the edges by considering $\mathcal{G}$ as a torus. Meaning that, given $X^t = (x^t_{i,j})_{(i,j)\in\mathcal{G}}$ at step t, the next step $X^{t+1} = \bar{\delta}(X^t)$ is defined as follows

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

Question n°2

a) How many applications of the function δ are necessary to compute X^t on $[\![0, N-1]\!] \times [\![0, M-1]\!]$?

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

c) What are the time and communication costs of your algorithm ?

d) Can you adapt it to a non toric grid (finite bounded grid) ? What would the complexity be ?

Laplace operator and convolution

The Laplace operator, which is involved in the vibrating string equation, is a second order derivative operator.

$$\Delta f = \nabla^2 f = \nabla.\nabla f$$

In the nth dimension euclidean place it can very simply be computed as :

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$$

A basic approximation of this operator on a 2D grid of values consist in computing the convolution of this grid with a 3×3 kernel.

0	1	0
1	-4	1
0	1	0

Question n°3

a) Given this method for computing the Laplace operator, and the equation 1, describe a cellular automaton to simulate the evolution of a vibrating environment.

b) (STEP 0) Implement a non distributed version of this algorithm.

c) (STEP 1) Implement a distributed version of this algorithm using MPI on a grid of processors.

Performance evaluation

Question n°4

a) How would you evaluate the performances of your code ? Describe a protocol for evaluating the scalability of your application.

b) Run this protocol, produce at least one figure and comment your results.

Part 3

New features

Walls

Walls are cells where the value is fixed and cannot change throughout time. This value is usually set to 0 but any value could technically be used that would represent a local equilibrium.

Question n°5

a) How can you modify you cellular automaton (Q and δ) to add walls to the model ?

b) (STEP 2) Modify your implementation of the cellular automaton to include this feature.

Sensors

Sensors are cells that, in addition to applying the physics equation, maintain a record of past events. Each sensor has a counter which is increased as each step. If cell $(i, j) \in \mathcal{G}$ is a sensor, we would like it to record :

$$record_{i,j}^{t} = \sum_{s=0}^{t} (x_{i,j}^{s})^2$$

Question n°6

a) How can you modify you cellular automaton (Q and δ) to add sensors to the model ?

b) (STEP 3) Modify your implementation of the cellular automaton to include this feature.

Local environment characteristics

So far we considered our environment to be homogeneous, meaning that v is the same throughout the whole grid. We may want to model the interface between different phase by having this value being defined locally. Each cell would therefore have it's own v describing the local characteristics of the environment.

Question n°7

a) How can you modify you cellular automaton (Q and δ) to add local definition of v to the model ?

b) (STEP 4) Modify your implementation of the cellular automaton to include this feature.

c) Can you reconsider past choices ?

Science !

Question n°8

a) How would you consider testing your automaton ? What physical phenomenon would you consider reproducing ? Fell free to provide input files describing those experiments ?

Part 4

Implementation details

All code are expected to be written in C using MPI. Your programs must compile (with `mpicc`) and run (with `mpirun`) on the machines of the ENS (`slsuX-YY`).

All real number should be represented as double.

Steps

You executable should progressively implement the different steps (step 0 to step 4). For you executable to pass a step, it will have to implement the requested features. You should be able to run the program for step 2 through the command

```
./myprogram -step 2 ...
```

Even if your implementation of step 3 would answers to both step 1 and step 2, your program should still handle `./myprogram -step 1 ...` as a valid command. You are free to have multiple arguments reuse the same piece of code if it answers multiple questions.

Environment description

The environment is a grid of size $N \times M$. The environment is described in binary files with the following specifications :

Type 1 (Step 0 to 3) :

The content of the environment starts at position `0x25` and consists of $N \times M$ type 1 blocks. Each block is 9 bits long. Therefore, the total size of a type 1 file is $25 + 9 \times N \times M$ bits.

	Position		Size	Type	Description
header	0x00	0	1	char	Type of file (value = 1)
	0x01	1	8	size_t	Number of columns
	0x09	9	8	size_t	Number of rows
	0x11	17	8	double	Velocity (v)
body	0x19+09*i	$25 + 9i$	1	char	Block type (0x00 = vibrating, 0x01 = wall)
	0x1A+09*i	$26 + 9i$	8	double	Value

Table 1 – Type 1 file structure

Type 2 (Step 4) :

The content of the environment starts at position `0x17` and consists of $N \times M$ type 2 blocks. Each block is 17 bits long. Therefore, the total size of a type 2 file is $17 + 17 \times N \times M$ bits.

	Position		Size	Type	Description
header	`0x00`	0	1	`char`	Type of file (value = 2)
	`0x01`	1	8	`size_t`	Number of columns
	`0x09`	9	8	`size_t`	Number of rows
body	`0x11+0x11*i`	$17 + 17i$	1	`char`	Block type (`0x00` = vibrating, `0x01` = wall, `0x02` = sensor)
	`0x12+0x11*i`	$18 + 17i$	8	`double`	Value
	`0x1A+0x11*i`	$26 + 17i$	8	`double`	Local velocity

TABLE 2 – Type 2 file structure

Input parameters

Input parameters are as follows :

Required arguments

1. `-step <nb>`
2. `-i <input file>`. Path to the input file describing the environment.
3. `-iteration <number>`. Number of iteration to compute.
4. `-dt <number>`. Value for dt (size of a time-step).
5. `-grid <x> <y>`. Dimensions of the processor grid. You program is only expected to run on $x * y$ processors. If another number of processors is given, you should print and error message.

Optional arguments

1. `-lastdump <output path>`. If this parameter is enabled, the matrix of values at the last step must be written to the specified file.
2. `-alldump <output path>`. If this parameter is enabled, all matrices (for each iteration), must be written to the specified file. Each file name (one for each iteration) must include the iteration step using `sprintf`. Path must not be longer than 256 characters.
3. `-sensor <output path>`. If this parameter is enabled, sensor values must be exported to the specified file.

If no optional argument if given, no output is expected

Exemple of valid input line

```
./myprogram -step 3 -i basicconfig.env -iterations 1000 -dt 1.0 -grid 4 8 -alldump
                    output_%03d.dump -sensor output_sensor.log
```

Those parameter should run the simulation asked at step 2 (including walls and sensors) on the grid described in the file `basicconfig.env`, for 1000 iterations that are all 1.0 long, on a grid of 4×8 processors. At each step the grid should be exported to `output_000.dump`, `output_001.dump` ... etc. At the end of the simulation, the sensor data should be exported to `output_sensor.log`

Output format

Files produced by `-lastdump` and `-alldump` must be binary files containing the matrix of values in row-major format. For more information about writting binary files in C, go read `man fwrite`. The expected size of the file is

$$8 * nbcols * nbrows \quad \text{(bytes)}$$

Files produced by `-sensor` must be ascii files with one line per sensor. Each line must contain the x and y coordinates of the sensor as well as the measured value.

```
...
0 125 46.24691
0 126 44.26151
0 127 43.27426
...
```

Part 5

Handover details

Don't worry if some features are missing, instead, explain in your report what you have achieved and where you got stuck.

All theoretical answers must be written in your `.pdf` report. I'd recommend this report to be written using LaTeX, but you are free to use applications like Microsoft Word or Open Office (as long as you export it as a PDF). This report, and all your code must be provided put in a `.tar.gz` archive named `name.tar.gz` in which you must have a folder `name` (replace `name` by your name!). You must also provide a makefile, such that running make at the root of the folder `name` will compile everything and create all executables at the root of `name`.

Your code is your property, protect it with a license! Also, when sending us your archive don't forget to provide us with a `md5` checksum so we can verify its integrity.