HPC projects

Hrachya Astsatryan, Hélène Hénon

June 20, 2024

1 Project 1 : Euler scheme for the advection equation

The advection equation in 1D is :

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0 \tag{1}$$

where $u(x,t)$, $x \in [0, L_x]$ is a scalar (wave), advected during time t.
We consider the initial condition :

$$u_0(x) = u(x,0) = \exp(-\frac{x}{2}) \tag{2}$$

The true solution is :

$$u_t(x,t) = u_0(x-t) \tag{3}$$

We consider a periodic boundary condition, meaning $u(0,t) = u(10,t)$.

But we want to solve it by using finite differences (https://en.wikipedia.org/wiki/Finite_difference_method) and the explicit Euler scheme (https://en.wikipedia.org/wiki/Euler_method). Here is the according scheme, using a upwind scheme :

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{u_i^n - u_{i-1}^n}{\Delta x} = 0 \tag{4}$$

where:

- Δx is the spatial step size, defined as $\Delta x = \frac{L_x}{N_x}$, with N_x being the number of spatial grid points.
- Δt is the time step size, defined based on the Courant-Friedrichs-Lewy (CFL) condition for stability: $\Delta t \leq \frac{\Delta x}{c_{\max}}$, where $c_{\max}$ is the maximum wave speed (which is 1 in this case, since the advection speed is 1).

By appropriately choosing Δx and Δt, we can ensure numerical stability and accuracy of the finite difference scheme.

When we rearrange it :

$$u_i^{n+1} = (1 - \frac{\Delta t}{\Delta x})u_i^n + \frac{\Delta t}{\Delta x}u_{i-1}^n \tag{5}$$

We can write this equation in matrix form :

$$U^{n+1} = AU^n \tag{6}$$

with, by posing $a = \frac{\Delta t}{\Delta x}$

$$U^n = \begin{pmatrix} u_0^n \\ u_1^n \\ \vdots \\ u_{N_x-1}^n \end{pmatrix} \qquad U^{n+1} = \begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ \vdots \\ u_{N_x-1}^{n+1} \end{pmatrix} \qquad A = \begin{pmatrix} 1-a & 0 & \cdots & 0 & a \\ a & 1-a & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & a & 1-a & 0 \\ 0 & \cdots & 0 & a & 1-a \end{pmatrix}$$

1. Implement the Euler scheme in sequential
2. Parallelize using OpenMP
3. Parallelize using MPI by sending as few messages as possible!
4. Trace the speedup (https://en.wikipedia.org/wiki/Speedup)

2 Project 2 : Solving Laplace's equation with Jacobi method

Laplace's equation in 1D is :

$$\frac{\partial^2 T}{\partial x^2} = 0 \tag{7}$$

where $T(x), x \in \mathbb{R}$ is a twice-differentiable real-valued function, representing the temperature.

This function is defined on a domain $[0, L]$, with boundary condition such that $T(0) = 15°C = c_0$ and $T(L) = 35°C == c_L$.

By using a finite differences scheme of second order, we can have the following scheme :

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} = 0 \tag{8}$$

where $\Delta x = \frac{L}{N_x}$ with N_x being the number of spatial grid points.

This can be written in matrix form such that $AU = b$ with, by posing $a = -\frac{2}{\Delta x^2}$ and $b = \frac{1}{\Delta x^2}$:

$$A = \begin{pmatrix} a & b & 0 & \cdots & 0 \\ b & a & b & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & b & a & b \\ 0 & \cdots & 0 & b & a \end{pmatrix} \qquad U = \begin{pmatrix} T_0 \\ T_1 \\ \vdots \\ T_{N_x-1} \end{pmatrix} \qquad b = \begin{pmatrix} c_0 \\ 0 \\ \vdots \\ c_L \end{pmatrix}$$

We want to solve this linear system using the Jacobi method, element based formula (https://en.wikipedia.org/wiki/Jacobi_method).

1. Implement this in sequential
2. Parallelize using OpenMP
3. Parallelize using MPI by sending as few messages as possible!
4. Trace the speedup (https://en.wikipedia.org/wiki/Speedup)