HPC for Mathematicians - Week 5 Assignment

Aikaterini Karoni

February 24, 2021

Workshop 5 - Exercise 2

In this exercise we consider the following 2D wave equation:

$$u_{tt} = u_{xx} + u_{yy} \quad \text{for } x, y \in (-1, 1), t \in (0, T) \tag{0.1}$$

subject to the initial conditions

$$u(x, y, t = 0) = e^{-40((x-0.4)^2+y^2)}$$
$$u_t(x, y, t = 0) = 0$$

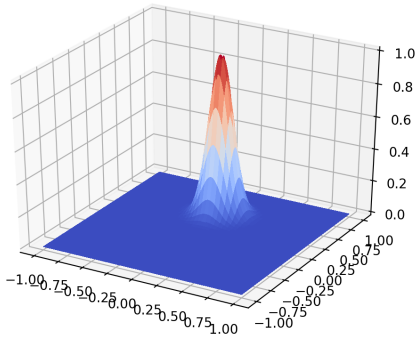
and homogeneous Dirichlet boundary conditions

$$u(x = \pm 1, y, t) = u(x, y = \pm 1, t) = 0 \tag{0.2}$$

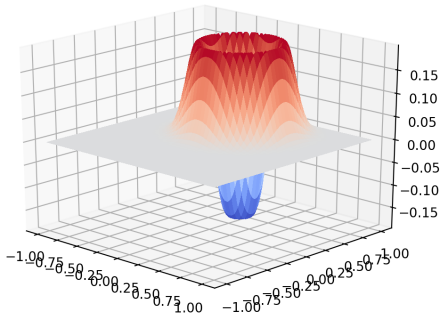
Using central differences for space and leap-frog for time, eq.0.1 can be written in a discretized form as:

$$U_{ij}^{n+1} = 2U_{ij}^{n} - U_{ij}^{n-1} + \frac{\Delta t^2}{\Delta x^2}\left(U_{i+1,j}^{n} + U_{i-1,j}^{n} + U_{i,j+1}^{n} + U_{i,j-1}^{n} - 4U_{ij}^{n}\right) \tag{0.3}$$

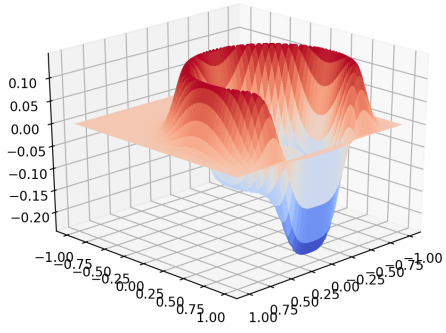
Our results for $t = 0$, $t = 0.3333$, $t = 0.6667$ and $t = 1$ are presented in the graphs below for the case of horizontal domain splitting for MPI computation.



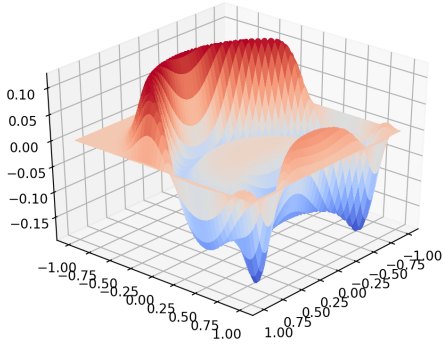
(a) t = 0



(b) t = 0.3333



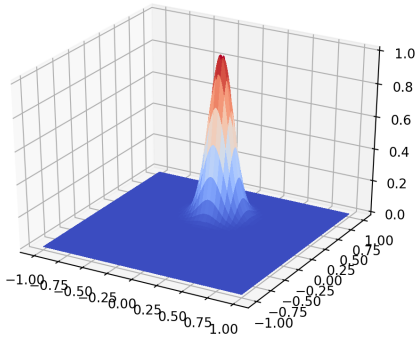
(c) t = 0.6667



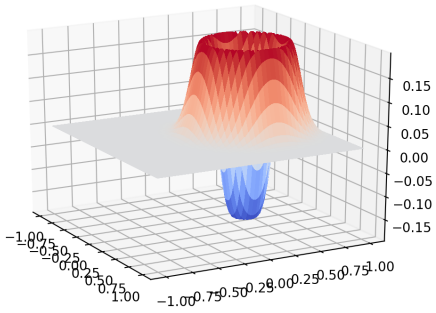
(d) t = 1

Figure 1: Horizontal splitting of the computational domain.

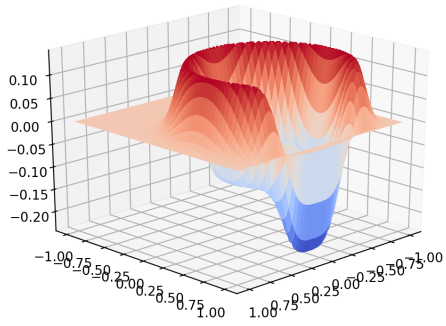
If we instead split the computational domain vertically, the solutions we obtain are, as expected exactly the same as for horizontal splitting and are presented in the graphs below:



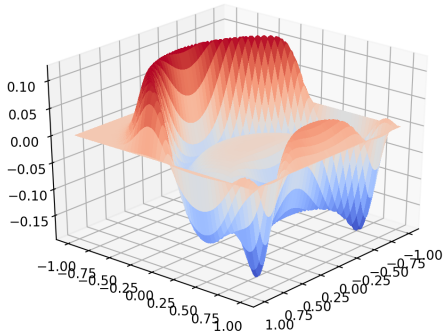
(a) t = 0



(b) t = 0.3333



(c) t = 0.6667



(d) t = 1

Figure 2: Vertical splitting of the computational domain.

Lets us now measure the run time for different numbers of processors for horizontal and vertical splitting:

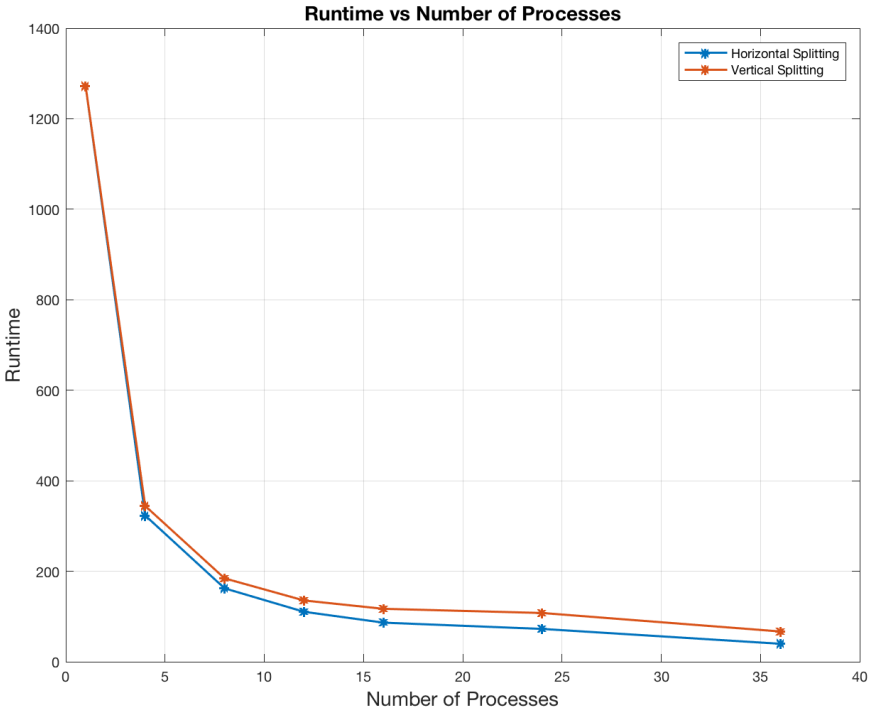


Figure 3: Comparing runtime vs number of processors for horizontal and vertical domain splitting

We can see that horizontal splitting results in faster runtimes, which makes sense since C++ is row major, i.e. array elements on the same on the same row are memory contiguous. During halo swapping between two processes we need to send a row (horizontal splitting) or column (vertical splitting) from one process to the other and vice versa. Sending a row is faster in C++ since it is row major, thus horizontal splitting is faster. (Note: the number of length intervals used for these graphs is 2305).

Below we plot the speed up and parallel efficiency versus the number of processors. As can be seen, the parallel efficiency drops with the number of processors with the exception of an "elbow" on the graph at 24 processes for which I was not able to think of an explanation. The speed up increases as the number of processors grow, as expected.

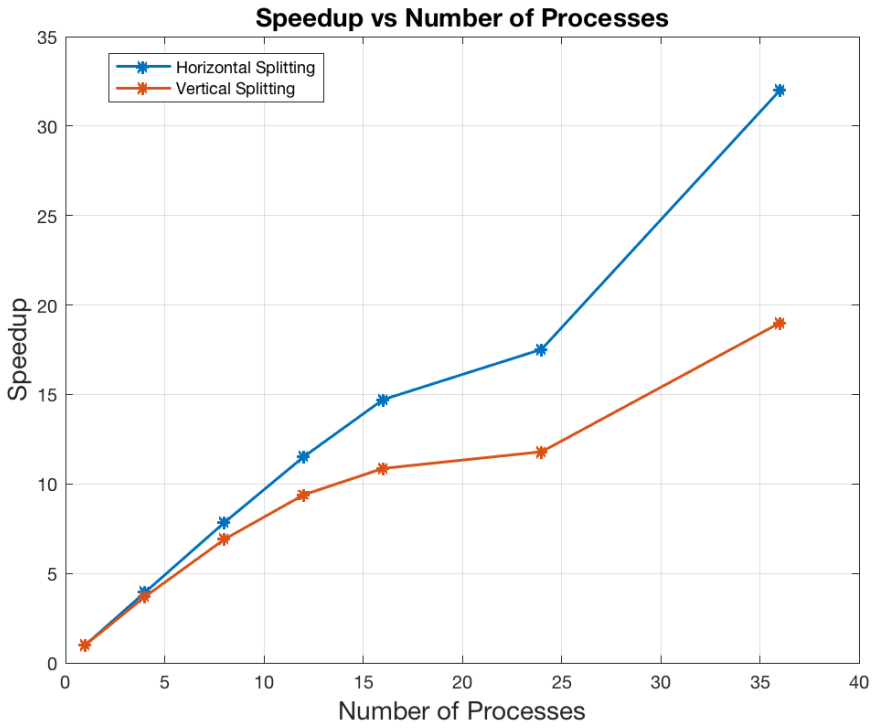


Figure 4: Comparing speedup vs number of processors for horizontal and vertical domain splitting

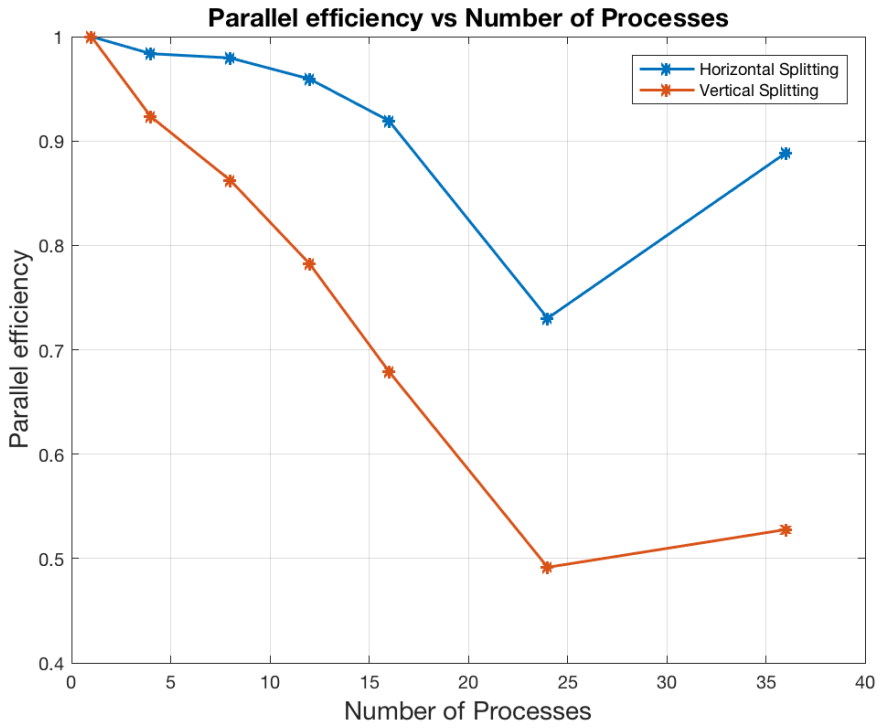


Figure 5: Comparing parallel efficiency vs number of processors for horizontal and vertical domain splitting