HPC for mathematicians Assignment 3

Yasmin Hengster

February 2021

Exercise 2

This exercise aimed to solve the $2D$ wave equation:

$$u_{tt} = u_{xx} + u_{yy} \tag{1}$$

with the initial conditions(Figure 1):

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

for $x, y \in (-1, 1)$ and $t \in (0, T)$ and the boundary conditions:

$$u(x = \pm 1, y, t) = u(x, y = \pm 1, t) = 0$$

The used numerical scheme for the wave equation is central differencing in

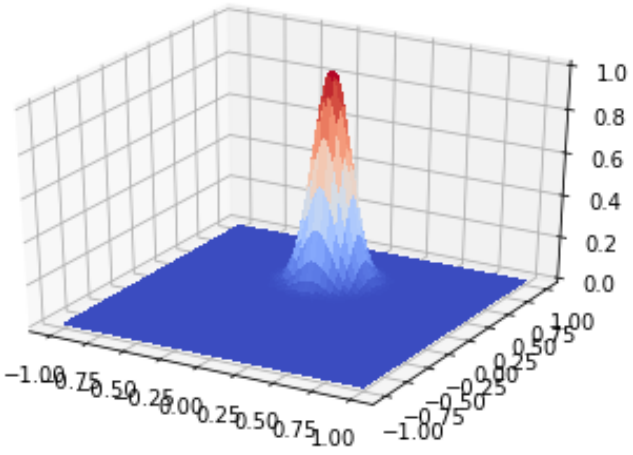


Figure 1: Initial condition.

space and leap-frog for time:

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

where i and j represent the spacial spacing for $M+1$ steps and n is the time spacing with $N+1$ time steps and $\Delta x = \Delta y = 2/M$ and $\Delta t = 0.2/M$.

To compute the new time step the last two time steps and the new time step were stored in an array of the size $[3 \times M+1 \times M+1$. For the parallelisation there are different ways to spilt the matrix for the different processes:

- Horizontal stripes
- Vertical stripes
- Squares

0.1 Horizontal stripes

In the first step I used horizontal stripes. Each process computed the new time step for a number J of rows such that:

$$J = (M-1)/size + 2$$

is an integer where size is the number of processes. The results for $t = 0.33$, $t = 0.67$ and $t = 1$ are shown in Figure 2. I used $M = 2305$ and 4 processes for these results. The time to compute $u(t = 1)$ was around 5 min and 30 sec. Saving the output took about 2.3 sec for each output file (4). Therefore the time for saving was 9.2 sec and the actual time for computing $u(t = 1)$ was around 5 min.

0.2 Performance

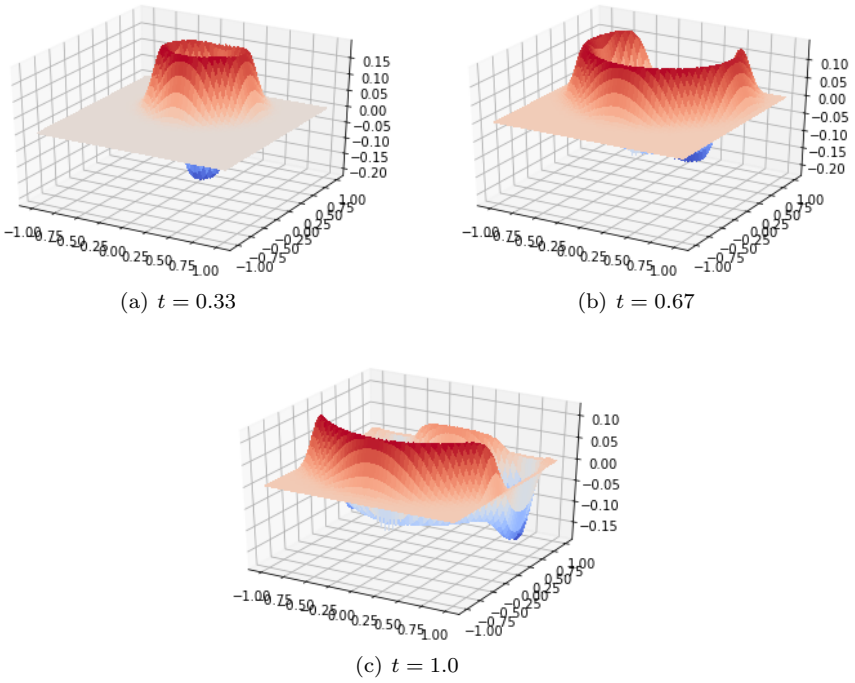
To compare the behaviour for different numbers of processes, I computed the *speed up* for different numbers of processes and a fixed M(strong scaling) with:

$$\text{speed up} = \frac{t(p)}{t_{ser}}$$

where $t(p)$ is the time for the computation with p processes without saving the output and $t_{ser} = 1152$ sec is the time for the serial code. The efficiency is computed by:

$$\text{eff} = \frac{t(p)}{p \cdot t_{ser}}$$

The weak scaling is just the run time where M is chosen such that the amount of work for each process is the same ($J = const$) and Δt is fixed. The strong scaling shows how the speed up increases with an increasing number of processes. The ideal speed up would be a linear line but due to communication



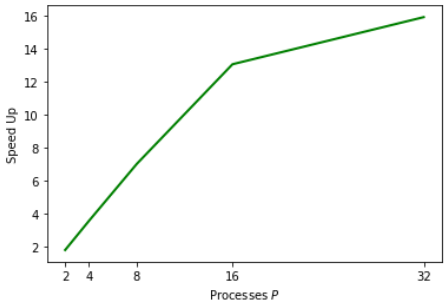
(a) $t = 0.33$

(b) $t = 0.67$

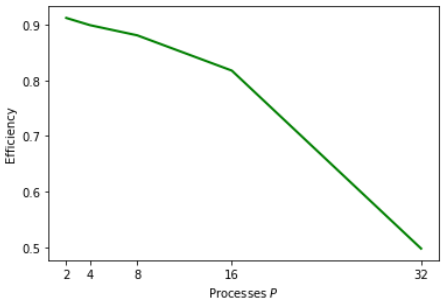
(c) $t = 1.0$

Figure 2: Solutions for different times.

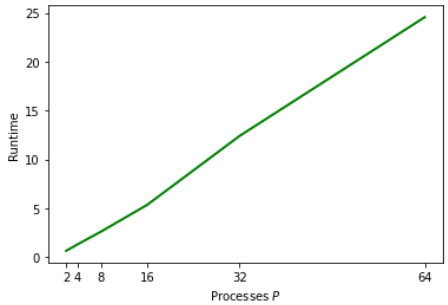
between the different processes the speed up isn't as good as expected. We can see the same for the efficiency that a higher number of processes doesn't balance with the time saving. The expectation for the weak scaling would be that the run time remains constant but again the the time for the communication between the different processes slows down the computation such that the savings due to more processes are less than the additional time for more communication.



(a) Strong scaling



(b) Efficiency



(c) Weak scaling

Figure 3: Performance of the parallel code