Assignment 3

Tianming Bai

February 24, 2021

1 Predict result

In this assignment, we consider 2D wave equation:

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

subject to the initial conditions

$$\begin{aligned} u(x, y, t = 0) &= e^{-40\left((x-0.4)^2+y^2\right)} \\ u_t(x, y, t = 0) &= 0 \end{aligned}$$

and homogeneous Dirichlet boundary conditions

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

Using central differencing for space and leap-frog for time, we have

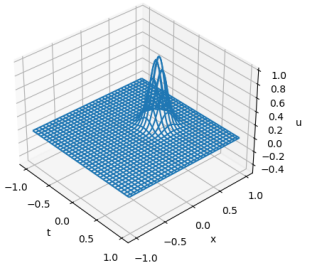
$$U_{i,j}^{n+1} = \frac{(\Delta t)^2}{(\Delta x)^2}\left(U_{i+1,j}^n + U_{i-1,j}^n + U_{i,j+1}^n + U_{i,j-1}^n - 4U_{i,j}^n\right) + 2U_{i,j}^n - U_{i,j}^{n-1}$$

1.1 Plot the results

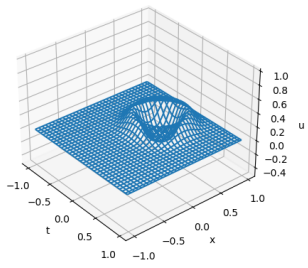
In Figure 1 we plot our results for different time T. For different ways of decomposition of domain, the results are the same.

1.2 Comparison of performance

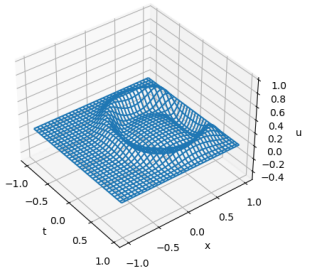
In this section, we compare the performance of different ways of decomposition of domain. I first test these three methods on my own laptop. Due to the limitation of cpu cache, I can only set M up to around 500. I find that vertical version is the slowest and equally sized square is the fastest. When $M = 501$, horizontal version takes around $5.3s$, equally sized square version takes around $4s$ and vertical version takes around $5.5s$. The reason that equally sized square version is faster is because the number of points passing between two processes at certain time node is smaller. For other two methods, the number of points passing between two processes is fixed, which is $M - 2$. For equally sized square



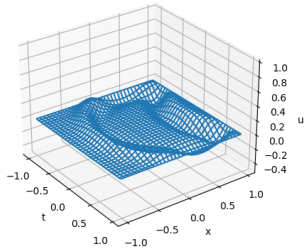
(a) $T = 0$



(b) $T = \frac{1}{3}$

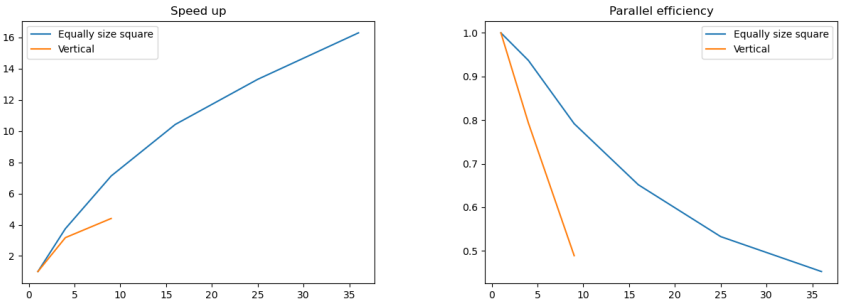


(c) $T = \frac{2}{3}$



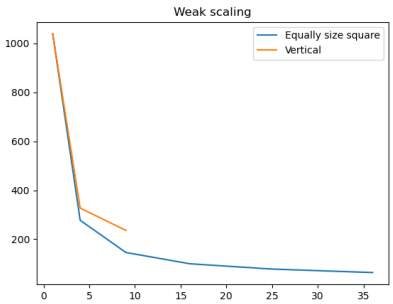
(d) $T = 1$

Figure 1: Results at different time T



(a) Speed up (Strong scaling)

(b) Efficiency



(c) Weak scaling

Figure 2: Comparing two methods of decomposition

version, it is $\frac{M-2}{\sqrt{nproc}}+2$. The reason why vertical version is slower than horizontal version might be c++ is row-major.

Then I use Cirrus to do computation for $M = 2305$. I choose the number of processes to be $1, 4, 9, 16, 25, 36$. I compute weak and strong scaling, and also parallel efficiency. (Threre might be some bugs in my vertical version code, when the number of process larger than 16, program takes longer time than with less processes, so I don't put full data for vertical version here.)