HPC4M assignment 4

Filippo Zanetti

Exercise 3

I implemented a parallel code that adapts the number of subdivisions of the domain $[0, 1]$ ($\texttt{nx}$) according to the number of processors being used ($\texttt{nproc}$). In detail, I used a starting number $\texttt{nx} = 5000$ and changed it as

$$\texttt{nx} = \texttt{nx} - \mathrm{mod}(\texttt{nx} - 1, \texttt{nproc}).$$

In this way, I'm sure that $\texttt{nx}$ satisfies $(\texttt{nx} - 1)/\texttt{nproc} \in \mathbb{N}$. This means that the number of subdivisions changes slightly for every instance of the results. I used $\Delta t = 1/2(\Delta x)^2$, i.e. the largest time step that produces stable results, with final time of 0.1s. I also computed the 2-norm of the error, using the expression of the exact solution; for the last instance, with $\texttt{nproc} = 36$, the error is $8.58 \cdot 10^{-7}$.

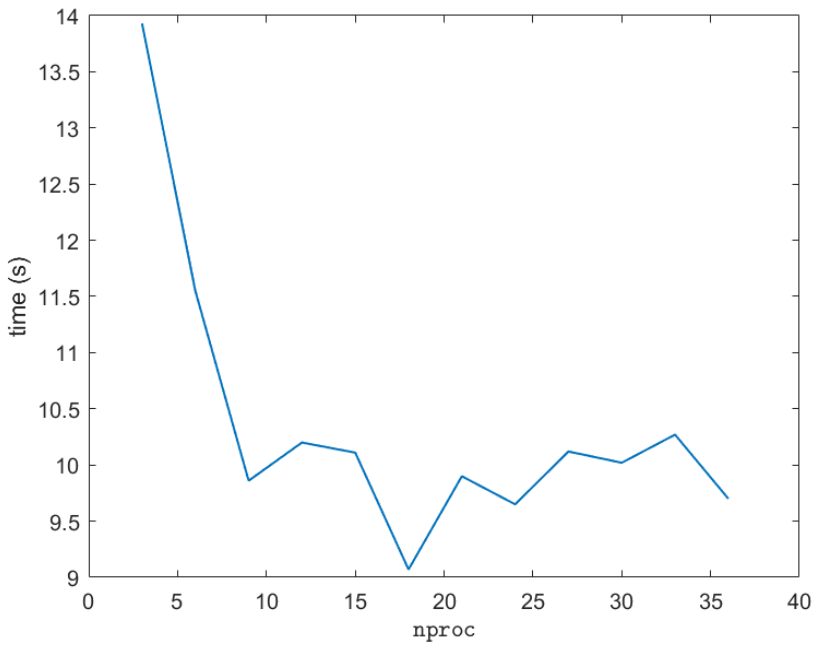
The next table shows the results in terms of $\texttt{nx}$, number of subdivions assigned to each processor ($\texttt{J}$) and computational time in seconds, for $\texttt{nproc}$ going from 3 to 36. The next figure shows a plot of these results.

Table 1: Results with $\texttt{nx} \approx 5000$ and various number of processors.

nproc	nx	J	time
3	4999	1668	13.93
6	4999	835	11.55
9	4996	557	9.86
12	4993	418	10.20
15	4996	335	10.11
18	4987	279	9.07
21	4999	240	9.90
24	4993	210	9.65
27	4996	187	10.12
30	4981	168	10.02
33	4984	153	10.27
36	4969	140	9.70

The computational time decreases sharply until the value $\texttt{nproc} = 9$ is reached and then it stagnates. This may be because the computational time

Figure 1: Computational time against number of processors.



gained using more processors is compensated by the larger number of messages to be shared between the processors.