HPC4M assignment 3

Filippo Zanetti

Exercise 2

I implemented the code such that:

- If nproc $= N$, every processor computes exactly one row of the result.
- If nproc $> N$, the last nproc $- N$ processors are left without doing anything.
- If nproc $< N$, then each processor computes $\text{int}(N/\texttt{nproc}) + 1$ rows of matrix A; the result is then collected and only the first N rows of the result are considered.

The result, for $N = 5$ and nproc $= 3$ is the following

```
Proc   0  .  Rows:
        350          360          330          260          150
        900          920          840          660          380
Proc   2  .  Rows:
       3750         3800         3450         2700         1550
       5100         5160         4680         3660         2100
Proc   1  .  Rows:
       1650         1680         1530         1200          690
       2600         2640         2400         1880         1080
Time =      1.762600000000000E-002
A * B =
        350          360          330          260          150
        900          920          840          660          380
       1650         1680         1530         1200          690
       2600         2640         2400         1880         1080
       3750         3800         3450         2700         1550
```

Exercise 3

I implemented three codes for this exercise:

- A serial code, which does not use parallelization
- A parallel code that splits the domain into vertical strips; each strip considers $\text{int}(N/\texttt{nproc})$ columns of the discretization, where N is the number of grid points per dimension.
- A parallel code that splits the domain into squares; I find the largest square smaller than `nproc`, i.e. $\texttt{nsq}^2 = \text{int}(\sqrt{\texttt{nproc}})$. The last $\texttt{nproc} - \texttt{nsq}^2$ processors are left idle, while the first ones actively compute the integral, each one working on a square with $\text{int}(N/\texttt{nsq})$ discretization points for each dimension.

I used the simple 2D trapezoidal rule: on each rectangular element of the discretization, the integral is approzimated as $\Delta x \Delta y/4 \cdot (f(a)+f(b)+f(c)+f(d))$, where a, b, c, d represent the four vertexes.

I used as parameters $a = b = 20$. Using the serial code, the results are the following

```
     2 subdivisions , error: 0.546E+04
     4 subdivisions , error: 0.567E+04
     8 subdivisions , error: 0.575E+04
    16 subdivisions , error: 0.126E+04
    32 subdivisions , error: 0.659E+03
    64 subdivisions , error: 0.445E+03
   128 subdivisions , error: 0.623E+03
   256 subdivisions , error: 0.104E+02
   512 subdivisions , error: 0.224E+01
  1024 subdivisions , error: 0.541E+00
  2048 subdivisions , error: 0.134E+00
  4096 subdivisions , error: 0.335E-01
  8192 subdivisions , error: 0.837E-02
 16384 subdivisions , error: 0.209E-02
 32768 subdivisions , error: 0.523E-03
 65536 subdivisions , error: 0.130E-03
131072 subdivisions , error: 0.331E-04
```

I chose to use $N^* = 100000$ subdivisions per side, which ensures a sufficiently small error. Using the serial code and N^*, the computational time is 136.38 seconds.

Now, we can compare the computational times using N^* subdivisions and a variable number of processors. We also compare the approach using strips and squares.

Table 1: Computational time using a variable number of processors, in the case with strips and squares parallelization

`nproc`	Time(s)	
	strips	squares
2	71.47	-
4	36.06	35.04
8	17.98	-
9	15.97	15.79
12	11.99	-
16	9.27	8.92
25	6.75	6.74
36	3.99	3.94

Figure 1: Computational time against number of processors for the parallel code using strips