Parallelizing Laplace with MPI

Author: Ruchita Nagare

1) Python code can be found here.

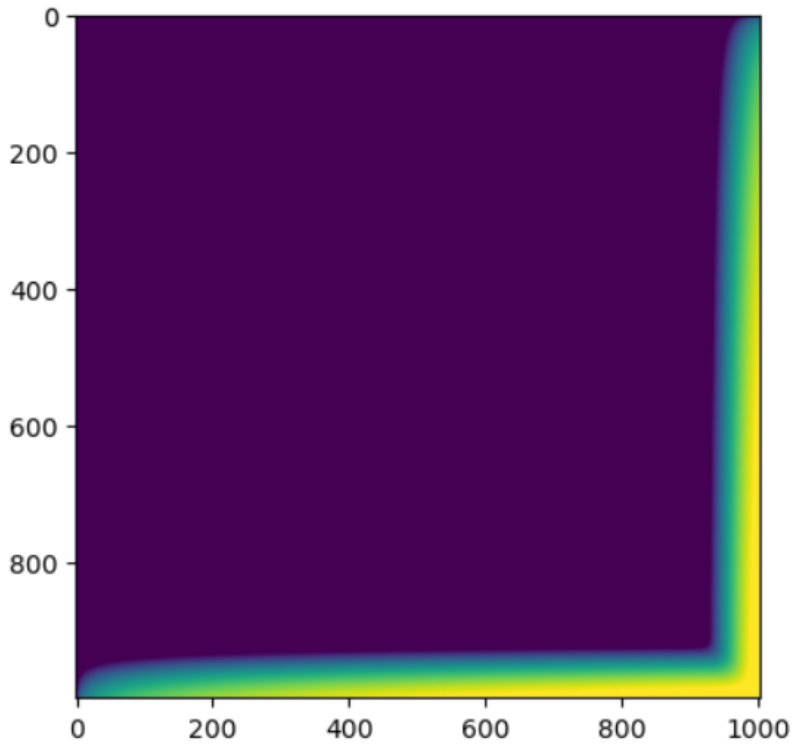
2) Results Visualizations

Following plot shows the output for 1000 as maximum iterations.

```
: import numpy as np
  import matplotlib.pyplot

  plate = np.fromfile("plate_full_4_1000.out", dtype=float).reshape((1000,1002))
  #matplotlib.pyplot.imshow(plate)
  matplotlib.pyplot.imshow(plate, norm=matplotlib.colors.LogNorm(0.1,50,clip=True))
```

```
: <matplotlib.image.AxesImage at 0x2506bcbe9d0>
```

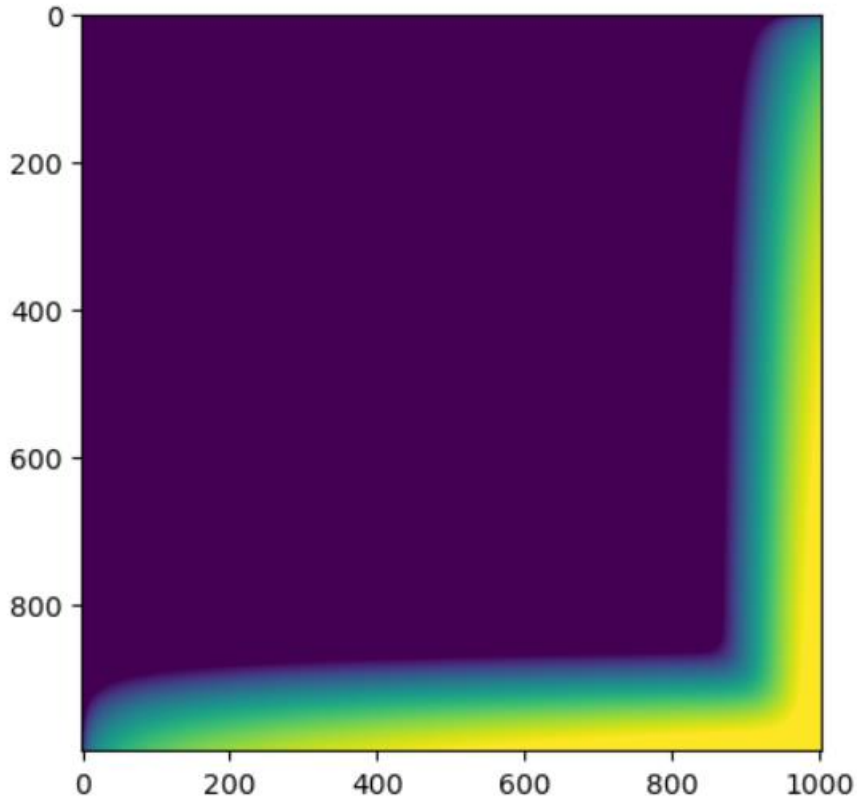


Following plot shows the output for 3500 as maximum iterations.

```
: import numpy as np
  import matplotlib.pyplot

  plate = np.fromfile("plate_full_4_final.out", dtype=float).reshape((1000,1002))
  #matplotlib.pyplot.imshow(plate)
  matplotlib.pyplot.imshow(plate, norm=matplotlib.colors.LogNorm(0.1,50,clip=True))
```

```
: <matplotlib.image.AxesImage at 0x1f4dbb1cac0>
```



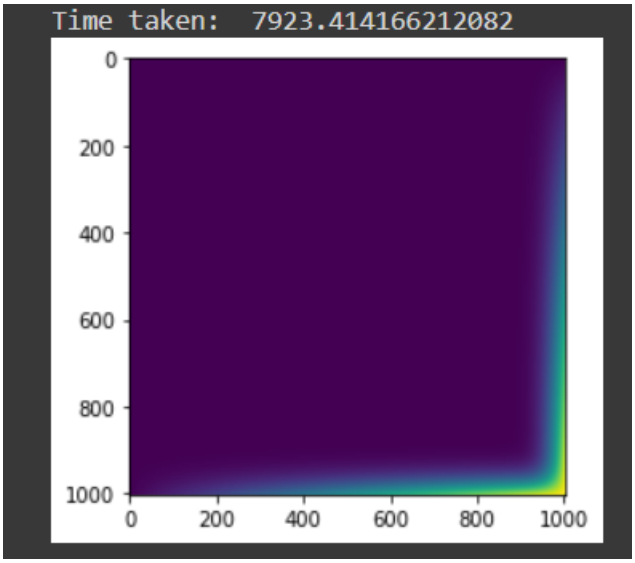
The convergence is at 3372.

```
PE 0 completete iteration: 3371
Starting exchange and calculations
PE 1 completete iteration: 3371
Starting exchange and calculations
PE 2 completete iteration: 3371
Starting exchange and calculations
PE 3 completete iteration: 3371
Starting exchange and calculations
dt_list: [0.00179401049296146 2, 0.0035880209859300294, 0.0053820318242081555, 0.00999533087259
1524]
reached here
reached here
reached here
reached here
(1000, 1002)
(1000, 1002)
```

The above figure shows the convergence of each PE, and *dt_list* holds the dt of each PE at convergence. The last line is the shape of the result array which contains final temperatures of all PEs. The top and bottom ghost cells have been removed. Whereas the right and left padding have been retained.

3) Timing of parallel code vs serial code

Serial code output and time taken:
It took about 7923 seconds ~ 2.2 hours ~ 132 minutes to completion.



Parallel code time taken:
It took about 26 mins to completion.

```
real    26m55.435s
user    0m0.019s
sys     0m0.191s
```

Evidently the parallel code was faster and is a little more than 4 times faster than serial code!