Final Project for M522 Numerical Analysis (Parallel Computing)

Pavel Buklemishev

March 18, 2025

1 You are required to build upon a template code that solves the model of heat diffusion, you are required to use domain decomposition with MPI to enhance your solver. you may assume there are only two MPI processors, and you are asked to partition the whole domain into two equal sub-domain. The MPI communication to allow domain decomposition should only be as large as the size of one array (i.e. massive communication for half of the grid is not allowed)

The code should output the infinity norm of residual clearly for the user. You may use block Gauss-Seidel but **a Red-Black Gauss-Seidel** is favourable. You should test your code with one and two MPI processors, and evaluate the efficiency, it should be included in the code as well. You may use a large grid resolution for this purpose, e.g. **512 x 512.**

1. We realize the Red-Black Gauss-Seidel scheme.
2. We skip this part because we generalize the problem to several processors.

3. Grid resolution would be 512 x 512 and 1024 x 1024 because my PC doesn't support more.

4. We realize the inf norm by the attached *calculate_diff.sh* which computes takes the matrices from the two files: *grid_parallel.txt* and *grid_nonparallel.txt*. The first is my parallel code results, the second one is provided modified *template.c* results.

   Actually, we also avoid the boundary conditions in the script due to noncontinuous BC in the corners.

2 This question extends from Question 1, that you cannot assume before-hand how many MPI processors will be, so your code should be able to adapt with all sensible number of MPI processors (you can safely assume the user knows what he/she is doing and has good knowledge in this field). You are only required to partition the domain by rows (row- blocking as demonstrated throughout the lectures).

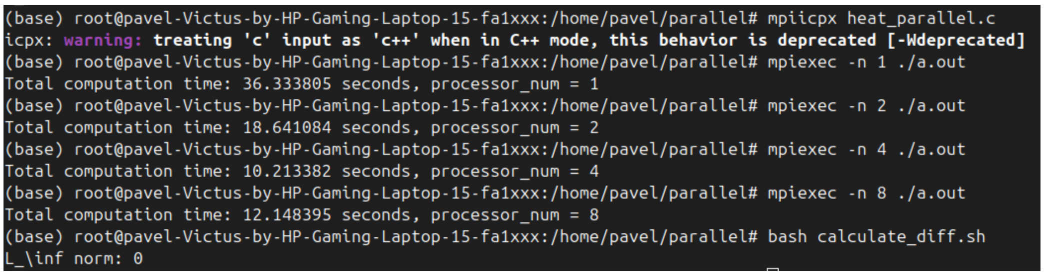
Again, you should not **do massive communication**. Test your code with **one, two and four MPI processors** and compute the efficiency. You may use a large grid resolution for this purpose, e.g. **1024 x 1024 or even 2048 x 2048**. *Parallel testing can be affected by many issues (e.g. whether other people are also using the same computational nodes), you may test your code several times and report the best results.

1. We send-request only the processor's boundary lines between processors

2. We can't use more than 1024 x 1024 due to the fact that the segmentation fault error appears on my PC.

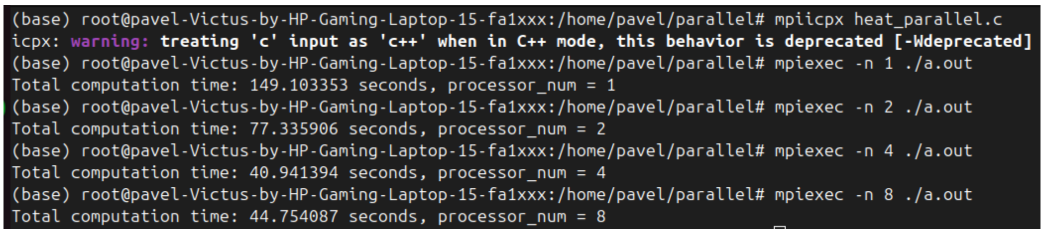
```
#!/bin/bash


```

```
MATRIX1="grid_parallel.txt"
MATRIX2="grid_nonparallel.txt"

MAX_NORM=$(paste "$MATRIX1" "$MATRIX2" | awk '{
    max = 0;
    for (i=2; i<=NF/2-1; i++) {
        diff = ($i - $(i + NF/2));
        if (diff < 0) diff = -diff;
        if (diff > max) max = diff;
    }
} END { print max }')

echo "L_\inf norm: $MAX_NORM"
```



```
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# mpiicpx heat_parallel.c
icpx: warning: treating 'c' input as 'c++' when in C++ mode, this behavior is deprecated [-Wdeprecated]
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# mpiexec -n 1 ./a.out
Total computation time: 36.333805 seconds, processor_num = 1
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# mpiexec -n 2 ./a.out
Total computation time: 18.641084 seconds, processor_num = 2
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# mpiexec -n 4 ./a.out
Total computation time: 10.213382 seconds, processor_num = 4
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# mpiexec -n 8 ./a.out
Total computation time: 12.148395 seconds, processor_num = 8
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# bash calculate_diff.sh
L_\inf norm: 0
```

Figure 1: Parallel computations results 512x512



```
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# mpiicpx heat_parallel.c
icpx: warning: treating 'c' input as 'c++' when in C++ mode, this behavior is deprecated [-Wdeprecated]
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# mpiexec -n 1 ./a.out
Total computation time: 149.103353 seconds, processor_num = 1
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# mpiexec -n 2 ./a.out
Total computation time: 77.335906 seconds, processor_num = 2
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# mpiexec -n 4 ./a.out
Total computation time: 40.941394 seconds, processor_num = 4
(base) root@pavel-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/pavel/parallel# mpiexec -n 8 ./a.out
Total computation time: 44.754087 seconds, processor_num = 8
```

Figure 2: Parallel computations results 1024x1024

Both computations exhibit an approximately linear scaling with respect to a small number of processors.
The code will be attached to the email.