Final project

Miguel De Le Court

Methods in High performance computing
DD2356

Stockholm, June 2021

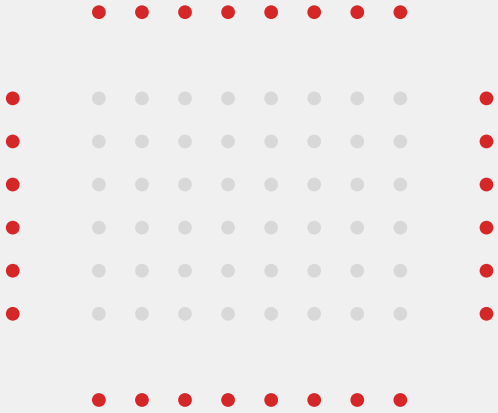
Outline

The last version of these slides and the code presented here are available at
`https://github.com/MiguelDLC/DD2356-Project`

Structure of the code

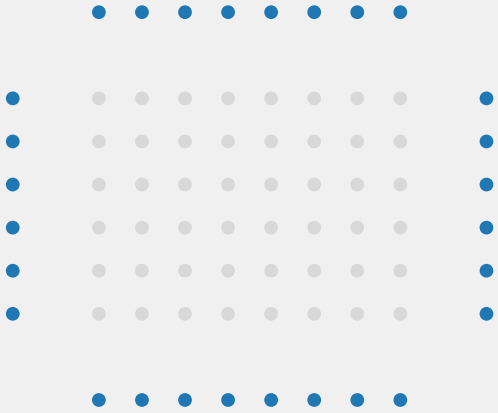
```
for(int iter = 0; iter < maxiter; iter++){
    wait_recieve(neighbours, recv_requests);
    compute_boundary(Told, Tnew, boundary_data, M, N, f, dx, dy, jmin, imin, kdt);
    i_recieve(boundary_data, neighbours, M, N, recv_requests, ComArray);

    internal_bounds(Told, Tnew, dx, dy, kdt, M, N);
    i_send(Tnew, neighbours, M, N, send_requests, ColDtype, ComArray);

    compute_inside(Told, Tnew, dx, dy, kdt, M, N);

    wait_send(neighbours, send_requests);

    double * temp = Tnew;
    Tnew = Told;
    Told = temp;
    //at the end : Tnew is garbage, Told is good
}
```

Structure of the code



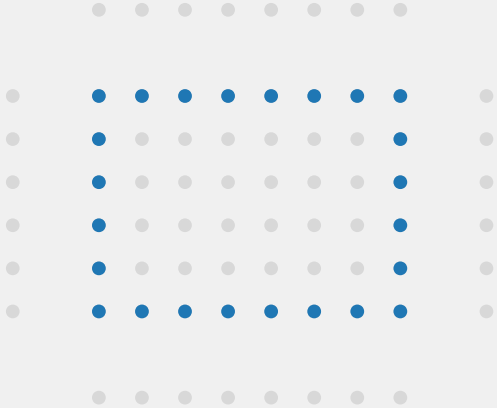
```
2     wait_recieve(neighbours, recv_requests);
```

Structure of the code



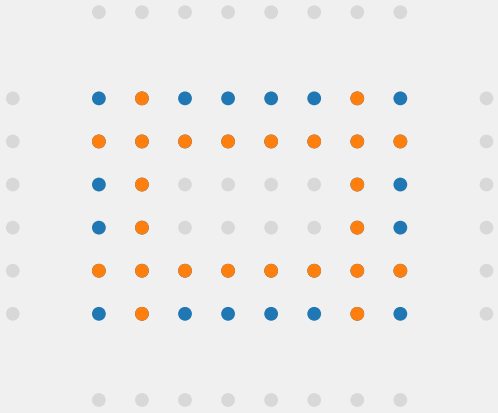
```
3    compute_boundary(Told, Tnew, boundary_data, M, N, f, dx, dy, jmin, imin, kdt);
```

Structure of the code



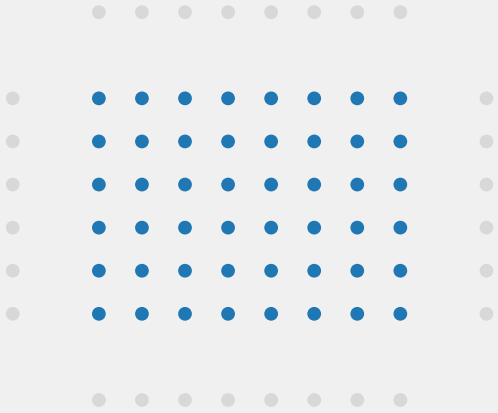
```
6       internal_bounds(Told, Tnew, dx, dy, kdt, M, N);
7       i_send(Tnew, neighbours, M, N, send_requests, ColDtype, ComArray);
```

Structure of the code



```
9       compute_inside(Told, Tnew, dx, dy, kdt, M, N);
```

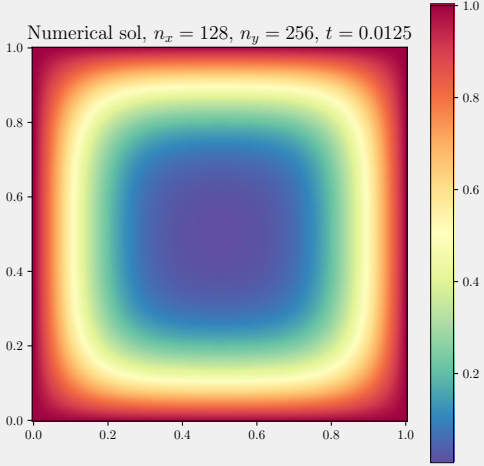
Structure of the code



```
11      wait_send(neighbours, send_requests);
```

Validation

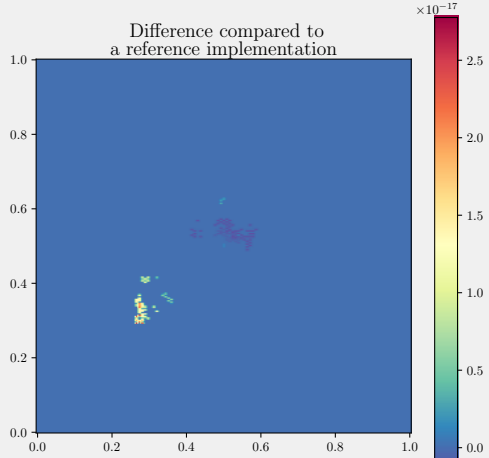
Result computed with the MPI solver



Validation

Comparison with a reference implementation

First validation test : check against a reference implementation



Validation

Comparison with an analytical solution

Second validation test : check against a reference implementation and ensure that the error decreases as expected

Expected error :

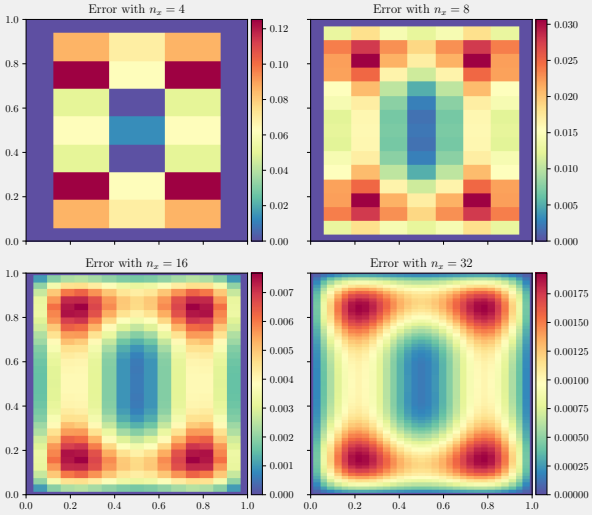
$$\|\boldsymbol{T} - \boldsymbol{T}_{\text{ref}}\| = \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t) = \mathcal{O}(\Delta x^2)$$

Analytical solution :

$$T_{\text{ref}}(x, y, t) = 1 - \frac{16}{\pi^2} \sum_{k \text{ odd}} \sum_{l \text{ odd}} \sin(k\pi x) \sin(l\pi y) e^{-\pi^2 (k^2 + l^2) t}$$

Validation

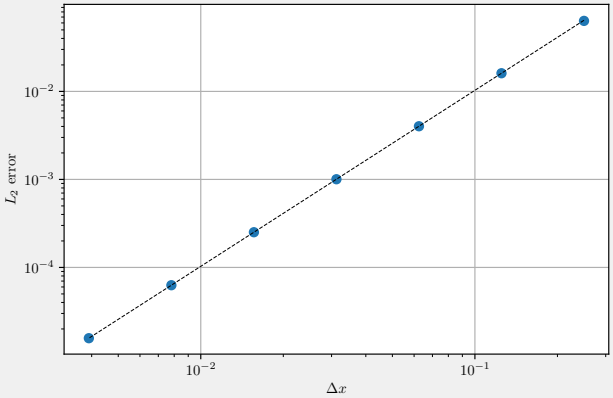
Comparison with an analytical solution



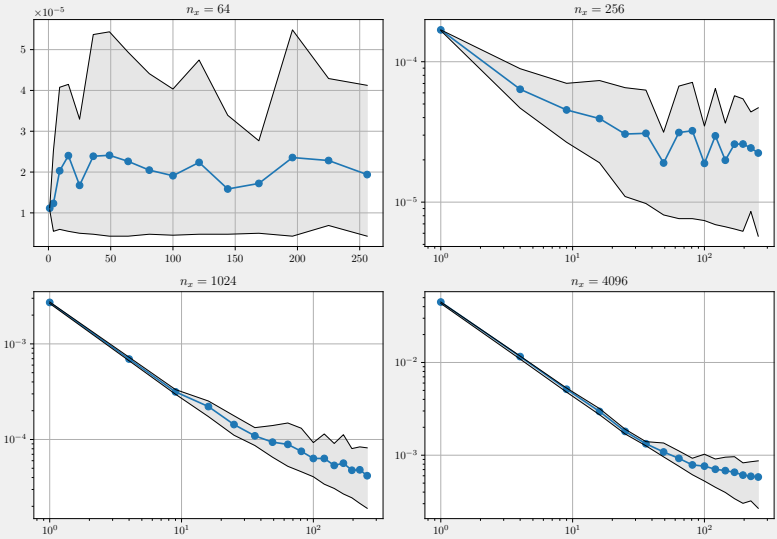
Validation

Comparison with an analytical solution

Second validation test : check against a reference implementation and ensure that the error decreases as expected



Scaling

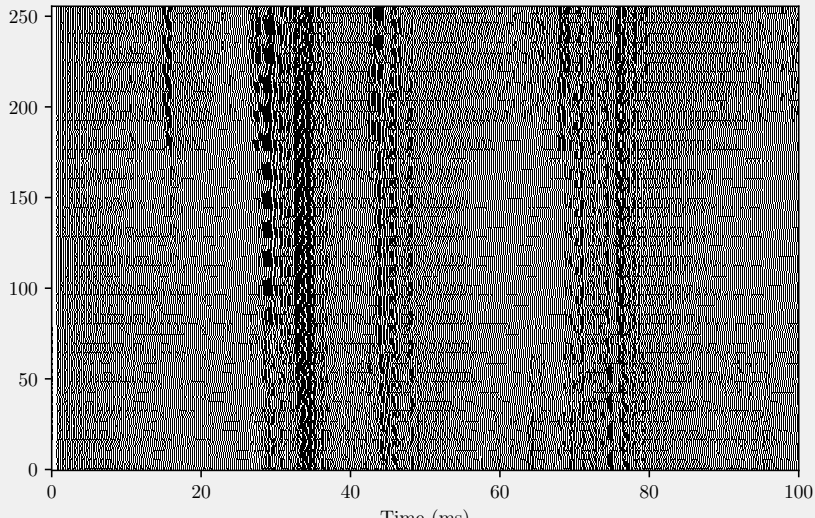


Idle propagation

```
1  for(int iter = 0; iter < maxiter; iter++){
2      wait_recieve(neighbours, recv_requests);
3      tsamples[ti++] = MPI_Wtime();
4      compute_boundary(Told, Tnew, boundary_data, M, N, f, dx, dy, jmin, imin, kdt);
5      tsamples[ti++] = MPI_Wtime();
6      i_recieve(boundary_data, neighbours, M, N, recv_requests, ComArray);
7      tsamples[ti++] = MPI_Wtime();
8      internal_bounds(Told, Tnew, dx, dy, kdt, M, N);
9      tsamples[ti++] = MPI_Wtime();
10     i_send(Tnew, neighbours, M, N, send_requests, ColDtype, ComArray);
11     tsamples[ti++] = MPI_Wtime();
12     compute_inside(Told, Tnew, dx, dy, kdt, M, N);
13     tsamples[ti++] = MPI_Wtime();
14     wait_send(neighbours, send_requests);
15
16     double * temp = Tnew;
17     Tnew = Told;
18     Told = temp;
19     //at the end : Tnew is garbage, Told is good
20 }
```

Idle propagation

Comparison with an analytical solution



Idle propagation

Comparison with an analytical solution