PHYS52015 Coursework:
stencils and collectives : OpenMP

1 Introduction

In this report, implementations of a parallel algorithm for the image reconstructing with the Gauss-Seidel method.

2 Parallelisation

Parallelisation with OpenMP which is API for programming shared memory computers.

2.1 Scheme

Planned to use parallelisation (#pragma omp parallel) for every calculation to get a quick result. Therefore my parallelisation scheme was parallel to every loop, which does not suffer data race.

Code 1 #OpenMP

```
1: #pragma omp parallel for
```

2.2 Problem

With (#pragma omp parallel for) located before the nested for loops activated only outer loop. If the code parallelise s all nested for loops in the main.c file, it printed the value in output.txt file with under 1 value at the 0 times r/r0. However, according to the convergence check, the value must be 1 in 0 times. Furthermore, another problem was that the calculation speed was slower than the original.

2.3 Reason

- It revealed that (Code 1) only recognise the outer loop and no recognised variables.
- The printed value of the output.txt file was under 1 in 0 times because of the parallelising residual norm function. Residual could be changed by parallelising
- Serial method by multi threads make the calculation more complex.

2.4 Solve

Use (collapse(2)) to parallelise two overlapped nested loops, and the variables were not declared and set the default to avoid danger shown below Code2.

Code 2 #OpenMP

```
1: #pragma omp parallel default(none) shared(NX,NY,edges,new,r0,r)
2 #pragma omp parallel for collapse(2)
```

Moreover, eliminate all other parallels without reconstructing from the edge function to solve the under 1 value in the output.txt file.

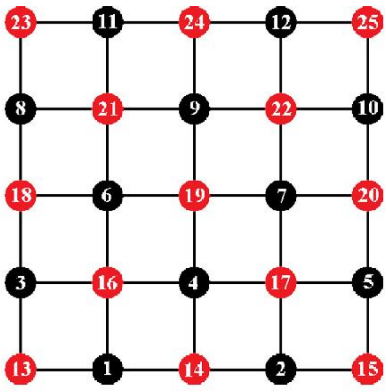
Break the original loop to the Red-Black ordering sequence.

Code 3 Red-Black ordering

```
1: if ((j+i)%2==0) //to find red dots with even number
2 if ((j+i)%2!=0) //to find black dots with odd number
```

3 Conclusion

Using Parallel with OpenMP API should check the structure of serial codes. Without selective parallelise could mislead the result and running time. It would be better to use breaking loops into independent sets (Red-Black ordering)



(fig 3.1)

The idea is to update all the black points, followed by all the red points. The black grid points are connected only to red grid points and so can all be updated simultaneously using the most recent red values. Then the red values can be updated using the most recent black values. In two parallel steps, one for updating the red points and one for the black. As a result, Red-Black ordering with parallelising makes calculations more quickly than before.