PHYS52015 Coursework
Part 1: OpenMP

CIS Username: bgtt65

1 Gauss-Seidel Parallelisation

The Gauss-Seidel iteration requires successive updating of solution components in the natural ordering (see Fig 1). For the dependency of natural ordering, a node's calculation depends on new values of the previous nodes and old values of the subsequent nodes. Therefore, the results of the parallelisation using natural ordering is untrustworthy since the order of updating nodes is different in every iteration.

To solve this problem, red-black ordering (see Fig. 2) has a unique characteristic of dependency that can be applied in parallelisation. For 5-point discretization on a square grid, colour alternate nodes in each dimension are red and others black. Each red node stencil does not depend on any other red nodes, then all the red nodes can be calculated in parallel. This situation is the same for black nodes.

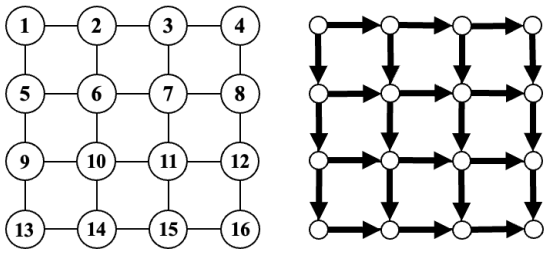

Figure 1: Natural ordering and its dependency

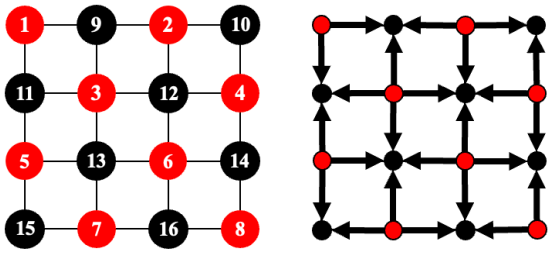

Figure 2: Red-black ordering and its dependency

2 Algorithm and Implementation

Parallelisation using red-black ordering is given in pseudocode in Alg. 1. This algorithm updates all red nodes simultaneously and updates all black nodes afterwards using new red nodes data. Therefore, it proceeds in alternating phases, first updating all nodes of one colour, then those of another colour. The procedure of this algorithm can be repeated until convergence.

In the implementation, using `#pragma omp parallel` to create a parallel region in the beginning. It should be careful to set variables inside the parallel region. In my testing, data races occurred because of the improper use of variables. The solution is using `private(i,j)` to ensure two variables i and j for loops in the algorithm are private. After that, add `#pragma omp for` before the loops of updating all red nodes and updating all black nodes. Then, the implementation is completed.

Algorithm 1 Parallelisation using red-black ordering

```
1:  Input: A matrix M with NY rows and NX columns
2:  parallel for i = 0,1,…, NY-1 do                    ▷ Update all red nodes
3:      if i = even then j = 0 else j = 1
4:      for j, j+2, j+4 to j < NX do
5:          Modify M(i,j) using black nodes data
6:      end for
7:  end parallel for
8:  parallel for i = 0,1,…, NY-1 do                    ▷ Update all black nodes
9:      if i = even then j = 1 else j = 0
10:     for j, j+2, j+4 to j < NX do
11:         Modify M(i,j) using new red nodes data
12:     end for
13: end parallel for
```