1.1. Project 3: Heat equation

1.1.1. Motivation

Gauss-Seidel and SOR are iterative methods to approximate the solution of a system of partial differential equations (PDEs). In this assignment we model the 2D heat equation, i.e. how would energy distribute on a sheed of metal.

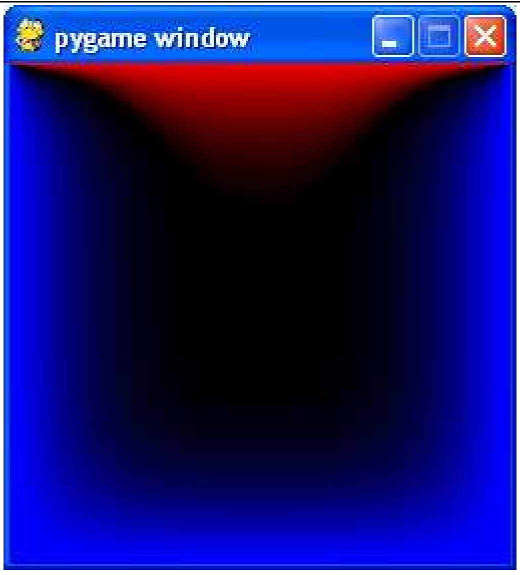

Figure 1 The stable heat equation

1.1.2. Considerations on the parallel version

The technique that we use to simulate the temperature development in the cube is called Successive Over-Relaxation, SOR. SOR is a method for solving very large systems of partial differential equations by successive approximations. The general idea is to approximate each element in a matrix to its neighbors until the sum of all changes within a single iteration converges below a given value.

```
do {
  diff=0.0;
  for i = 1 to n-1
    for j = 1 to n-1 {
      temp = A[i,j]
      A[i,j] = 0.2 * ( A[i,j] + A[i+1,j] + A[i-1,j] + A[i,j+1] + A[i,j-1] )
      diff = diff + abs( A[i,j] – temp )
    }
} while ( diff>epsilon )
```

Example 1. Sequential version of Successive Over-Relaxation

The sequential algorithm is given in Example 1.

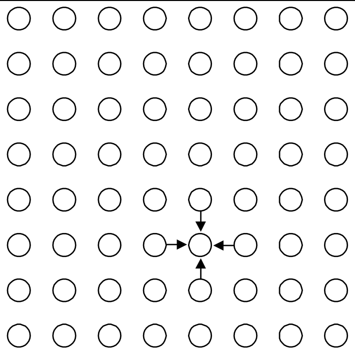

Figure 2 Two dimensional finite-differences method for the heat distribution problem

This algorithm may appear hard to parallelize as each calculation depends on two previous results from the same iteration, namely A[i-1, j] and A[i, j-1]. One solution is to implement a diagonal wave-front calculation, but that is not efficient, since the work available is highly unbalanced and frequent communication is required. However, the correctness of the algorithm does not depend on the sequential order reflected in the sequential algorithm. Each point may be updated using results from any iteration. This slows down the convergence slightly, but this is more than compensated by the parallelism gained. Using results from any iteration has the disadvantage that, since this is a successive approximation method, the convergence of the algorithm varies with the number of CPU's, the communication speed, and other parameters. The result is that different configurations will return different, but still correct, results.

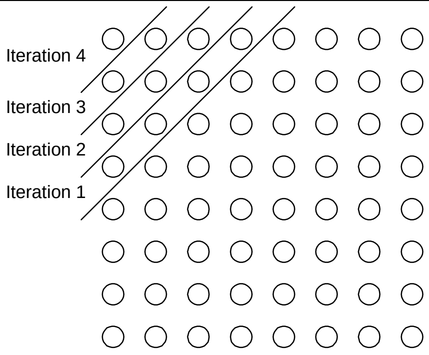

Figure 3 Wave-front approach to parallel two-dimensional finite differences method

Another solution, Red-Black ordering, returns identical results for the same system of equations; independent of the actual computing environment, while at the same time providing sufficient parallelism that real speedup is achieved. With Red-Black ordering, the equation system is divided into alternating red and black points in a checkerboard fashion. Updating a red point depends only on black neighboring points and vice versa. Using this, an algorithm is derived where each worker-process updates all its red points, exchanges the red point values on its borders with the red points from its neighbors. Each worker then updates its black points and repeats the exchange with the black points.

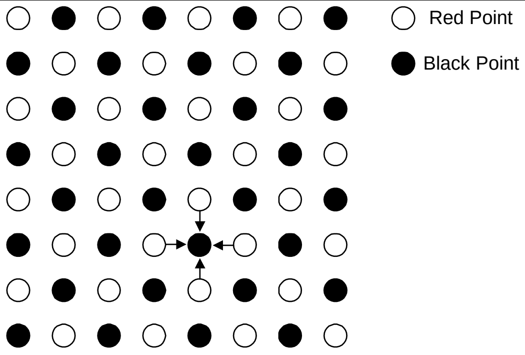

Figure 4 Red-Black checker-pointing for parallel SOR

1.1.3. Programming Task

This problem should be implemented using MPI, to ensure correct results the supplied version should be converted to a Red-Black checker pointing method. The code should be run on from one through 32 nodes. The cube should be modeled as a 500x500 grid.

The report should identify the various choices that have been made as well as individual techniques that has been applied to improve performance. And the impact of each should be documented. In addition the scalability of your implementation should be discussed and the achieved performance curve should be discussed.

1.1.4. Real World Relevance

Widely used in scientific applications

- Weather forecasting
- Ocean current simulation
- Climate modeling