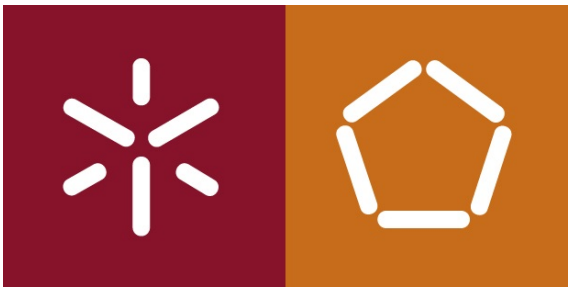


University of Minho

Mestrado Integrado em Engenharia Informática

Parallel Algorithms

Implementation of Parallel SOR Red-Black Gauss-Seidel method for solving Poisson's Equation

Adriana Meireles (A82582) Shahzod Yusupov (A82617)

15 de Maio de 2020

Conteúdo

1 Introduction

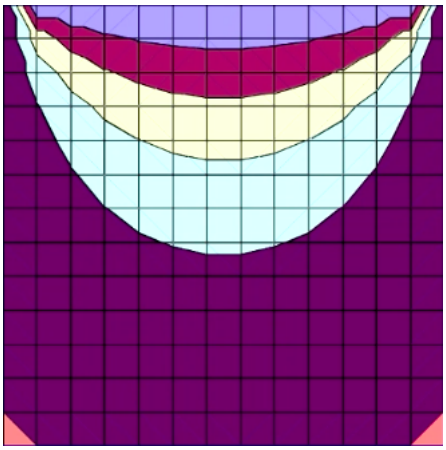
In this paper we will study the application of Poisson's equation in a termodinamic problem, the Steady State Heat Distribution, which is very interesting and worth studying.

There are a couple methods to solve this problem, but our case of study is the iterative method called SOR Red-Black, where SOR comes from sucessive over relaxation, which will be explained later, and will also be implemented a parallel version of this method and some measurements will be done in order to make some comparisons and draw some conclusions.

2 Problem Description

2.1 The Steady State Heat Distribution

Giving a two-dimensional physicall domain, where different sources of heat are applied on the borders, the aim is to find the equilibrium temperature at each interior point of the grid, after a certain amount of time.



At the upper limit a 0 degree heat source is applied , while at the lower and lateral limits the temperature of the heat source is 100 degree. Each grid point represents the value of the state solution at particular **(x,y)** location in the plate. To solve this problem the *Poisson's equation* will be used.

2.2 Discretization of the Poisson's Equation

Poisson's equation is a partial differential equation of elliptic type with broad utility in mechanical engineering and theoretical physics.

To solve numerically the Poisson's equation in 2D (two-dimensional) we use finite differences to discretize the equation.

$$u_{xx} + u_{yy} = f(x, y)$$

Figura 1: Poisson's equation

Where the first **u** stands for the second derivative of u in order of **x** and the second **u** stands for the second derivative of u in order of **y**. In this particular problem the function u is the temperature at each interior point and to simplify things we can assume that the function **f** is zero, which turns this equation into the **LaPlace's Equation**.

As mentioned before, after a certain amount of time the system reaches an equilibrium, and knowing that the temperature at the boundaries are always constant, we want to compute the values of **u**(aproximations) at every point of the domain. It is important to highlight that **u** is independent of **time** because the time varies depending on the physical properties of the domain, that is, the conductivity.

The finite difference method use aproximations of derivatives in the partial differential equation by linear combinations of function values at the grid points. What comes out from the discretization of the problem is the following simple relation:

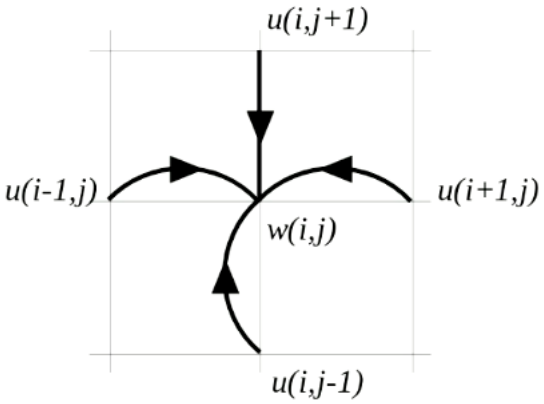


Figura 2: heart of the sequencial problem

Where **w[i][j] = (u[i-1][j] + u[i+1][j] + u[i][j-1] + u[i][j+1])/4**

We can see that the temperature at each point only depends of the temperature of the 4 neighbours that are vertical(2) and horzontal(2). To solve this five points stencil we will use the **SOR Red-Black** ordering , which is an particular order of the **Gauss-Seidel** iterative method.

2.3 SOR Red-Black Ordering

For Gauss-Seidel and SOR methods, the order in which the variables are processed matters. To explain this ordering we can make an analogy to the chessboard. Each red square only has black squares to its north, south, east, and west; similarly, each black square is only adjacent to red squares. If we are thinking about doing Gauss-Seidel, then, it is convenient to similarly color nodes red or black depending on whether i + j is even or odd, then process all the red squares first followed by all the black squares.

Because no red node depends on information from any other red node, the red nodes can be processed in any order without changing results; they just have to be processed before the black nodes. Similarly, the black nodes can be processed in any order, followed by the red nodes.[0]

As we do not know the initial values of the interior points, we can assume an initial approximation of 50 degrees, because the convergence of this iterative method does not deppend on starting values

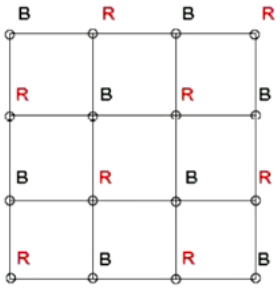


Figura 3: Red-Black ordering

To solve this problem a constant **p** is also used for relaxation of values, that is, the correction

made to the values of the previous iteration to obtain the values of the new iteration, and varies between 1 and 2, because values greater than 2 may jeopardize the convergence.

2.4 Parallelization of the algorithm

Analysing the way how the temperature at each point is computed we can verify that there are no dependencies between updates of values of the same color, making it interesting to analyse its version in shared or distributed memory.

We choose to parallelize in shared memory using OpenMP, because we can predict that in distributed memory the performance will be damaged by the overhead caused by the ammount of communication that this method will need.

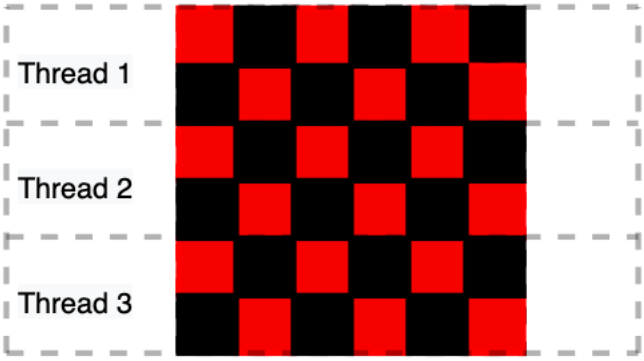


Figura 4: distribution of work p/thread using OpenMP

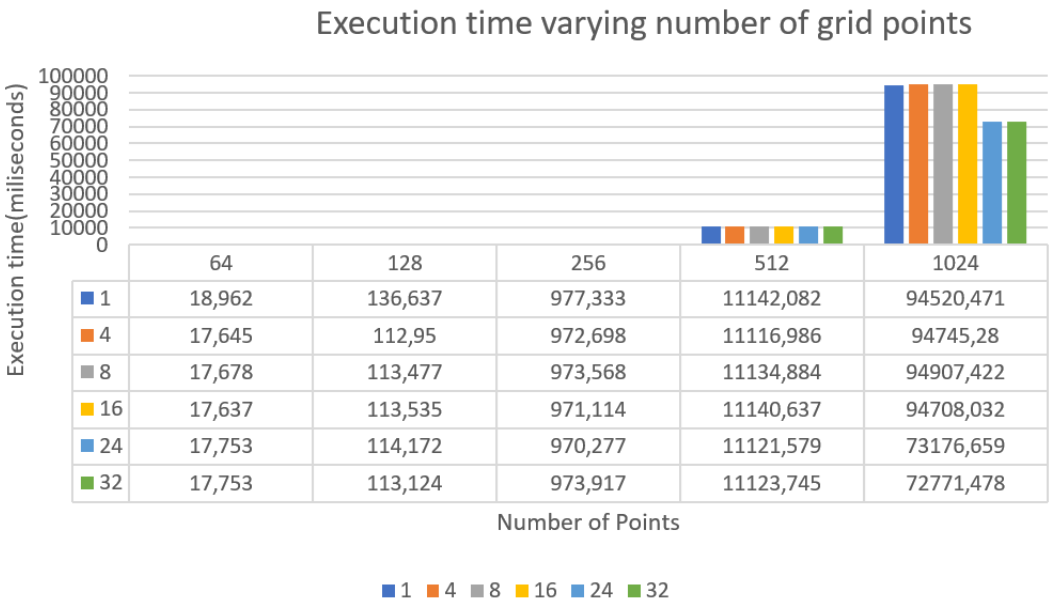
We adopted a very simple strategy that consists of distributing the work equally among the number of available threads. In this example each thread is responsible for 2 lines, so first each thread updates the values of the red points and after all red points have been updated, the same process is done for the black ones.

3 Measurements

In order to test the algorithm, we used 662 node from the SeARCH cluster atUniversity of Minho. We took several important parameters for performance such as execution time and number of iterations.

We can observe bellow the execution time(in miliseconds) increasing the number of grid points.

As far as the number of threads increases, the execution time is slightly faster, however the difference in relation to the sequencial(1 thread) version isn't that much.



	64	128	256	512	1024
1	18,962	136,637	977,333	11142,082	94520,471
4	17,645	112,95	972,698	11116,986	94745,28
8	17,678	113,477	973,568	11134,884	94907,422
16	17,637	113,535	971,114	11140,637	94708,032
24	17,753	114,172	970,277	11121,579	73176,659
32	17,753	113,124	973,917	11123,745	72771,478

Figura 5:

The graph bellow shows the speedup over the sequential version. We have chosen the best thread time for each grid point.

As we can see, the maximum speedup is 1.3 for 1024 points. So we conclude that it doesn't make up for to parallelize due to the inherent cost of multiple executions.

This kind of itherative methods doesn't take advantage of data locality. Performing successive global sweeps, as mentioned above this method performs one complete sweep through the grid updating the red points and another entire sweep to update all the black ones. When using data sizes that doesn't fit entirely in the cache, the data of the lower part of the grid is no longer in the cache after update sweep for the red ones, since they have been replaced by the data corresponding to the nodes in the upper part of the grid. Hence the data must be reloaded from the slower main memory into the cache again and this is a very costly process.

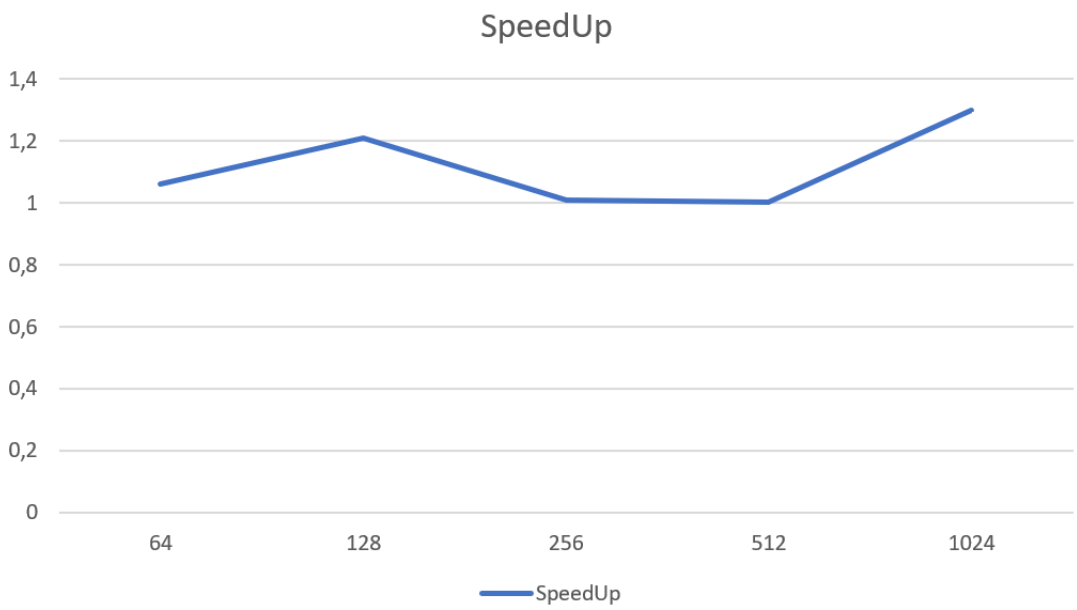


Figura 6:

We now present the graph that shows the iterations increasing the number of grid points. We can see that as far as the number of points increases,more iterations we obtain, as expected.

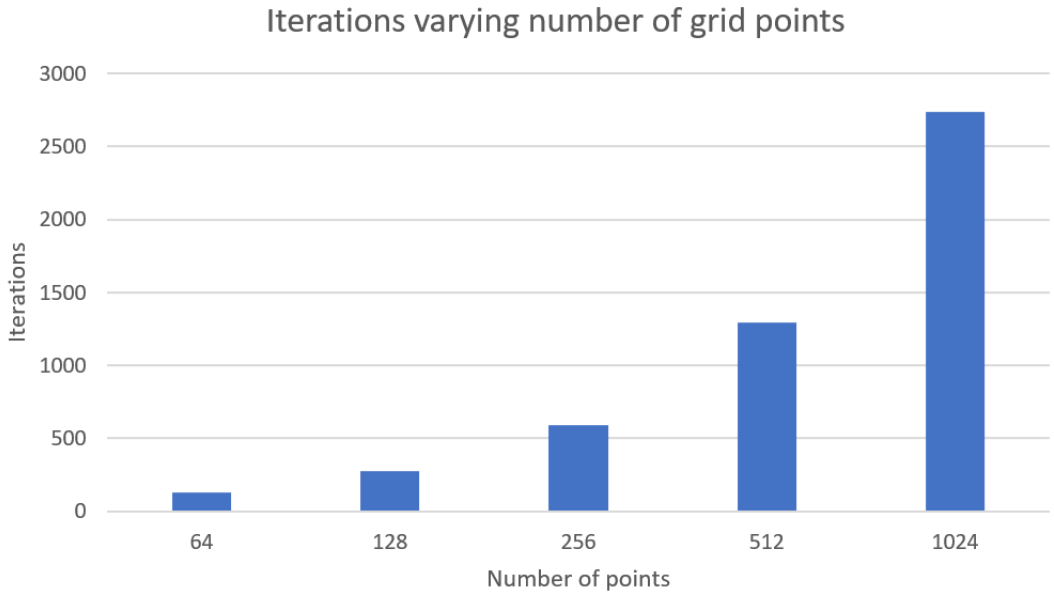


Figura 7:

4 Conclusion

With this work became evident that the utility of iterative algorithms solving linear system equations. Sometimes, the computational cost of conventional methods is too heavy, adding the fact that the system can be scattered. The iterative methods are presented as viable alternatives to the traditional method,obtaining solutions with good results and a much lower cost.

In our case of heat diffusion, the Poisson's equations is solved using the method SOR Red-Black. The results obtained previously allowed us to conclude that parallelizing this method isn't efficient.

5 Bibliography

Math.la.asu.edu. 2020. [online] Available at: ¡https://math.la.asu.edu/ kuiper/502files/Laplace.pdf?fbclid=IwAR3f_-
c9EfzPYnx2SE3SZBNC-HiWAeIuYAdGyA5sH_zJpHHaDrca7Kg6VUNo¿ [Accessed 15 May 2020].

2020. [online] Available at: ¡https://www.researchgate.net/publication/220260980_Cache-Aware_-
Multigrid_Methods_for_Solving_Poisson's_Equation_in_Two_Dimensions¿ [Accessed 15 May 2020].

Iue.tuwien.ac.at. 2020. 4.1.1 Discretization Of The POISSON Equation. [online] Available at:
¡https://www.iue.tuwien.ac.at/phd/pourfath/node68.html?fbclid=IwAR2g_QJCAWwbJm9JpZjFUCGhvMEV9EjSX
yJNz8rsI28BzgFHDr_o5U9s¿ [Accessed 15 May 2020].

http://www.cs.cornell.edu/ bindel/class/cs6210-f12/notes/lec33.pdf