OpenMP Implementation of 2D Poisson Solver

Carlos Vazquez Gomez

April 10, 2019

1 Presentation of the Algorithm

This algorithm is the Jacobi version of the MPI and Single threaded implementations (for Project 1 and 2, respectively). We use a main WHILE loop inside of which we use a "#pragma parallel" section, and a "#pragma for" which splits the grid into blocks and dispatches threads to calculate the updated Temperature values, which we store in a TEMPORARY BUFFER T_temp, and we memcpy this T_temp into T (the main array) at the before the next iteration of the WHILE loop. We use "#pragma critical" sections in order to compare local error values with the global value, and replace the global it the local is greater. This is how we obtain the convergence and absolute errors every 1000 cycles.

2 Verification

We verify the second order accuracy and the ability to solve heterogeneous grids (grids with different numbers of x and y points) of our 2D Poisson solver. For a simple visual inspection see Figure 1, where the output of 100 processes solving a 1x2 grid with source terms $x \cdot e^y$ with error threshold $1e$-12 are displayed in ParaView.

2.1 Second Order Accuracy

We test the grid convergence behavior of our algorithm by solving 20x20, 40x40, 60x60, ..., 220x220, 240x240 grids with four processes on a 1x2 sized rectangular region with source values $x \cdot e^y$ and convergence error threshold $1e$-12. The infinity-normed error of analytic to computed solution is shown to be linearly proportional to Δx^2 via the constant 62.7. Figure 2 reveals the decay of the absolute error as grid points increase. The linear proportionality of the absolute error and the square of the step size confirms that the algorithm is second-order accurate.

2.2 Solving Heterogeneous Grids

Now we verify that our solver works for cases where the x and y dimensions have different number of grid points by running our program on the same 1x2

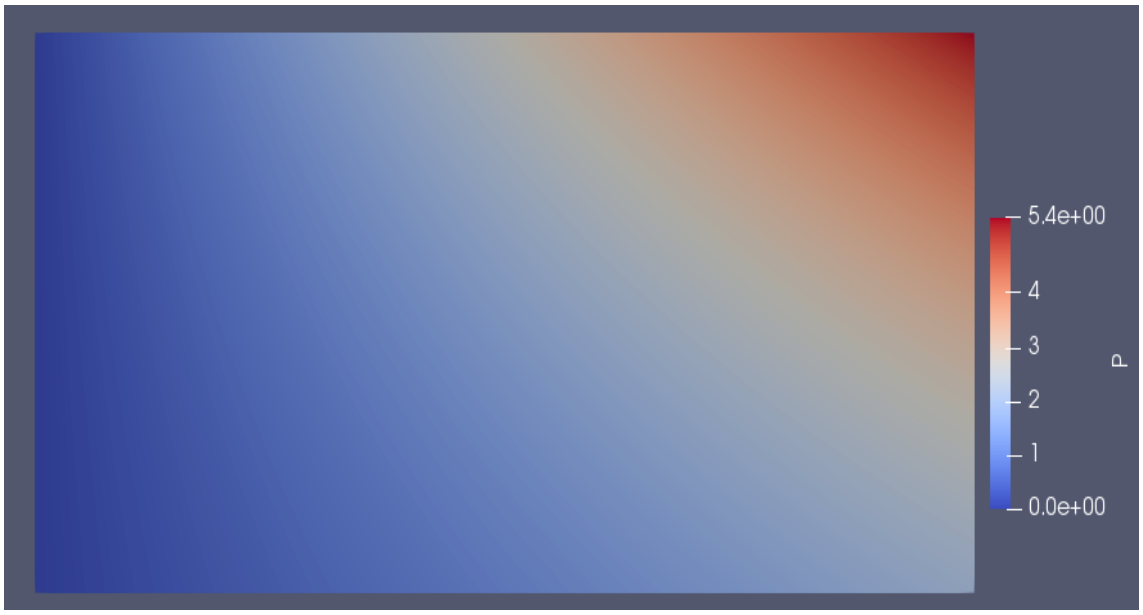

Figure 1: 200x200 grid result in ParaView

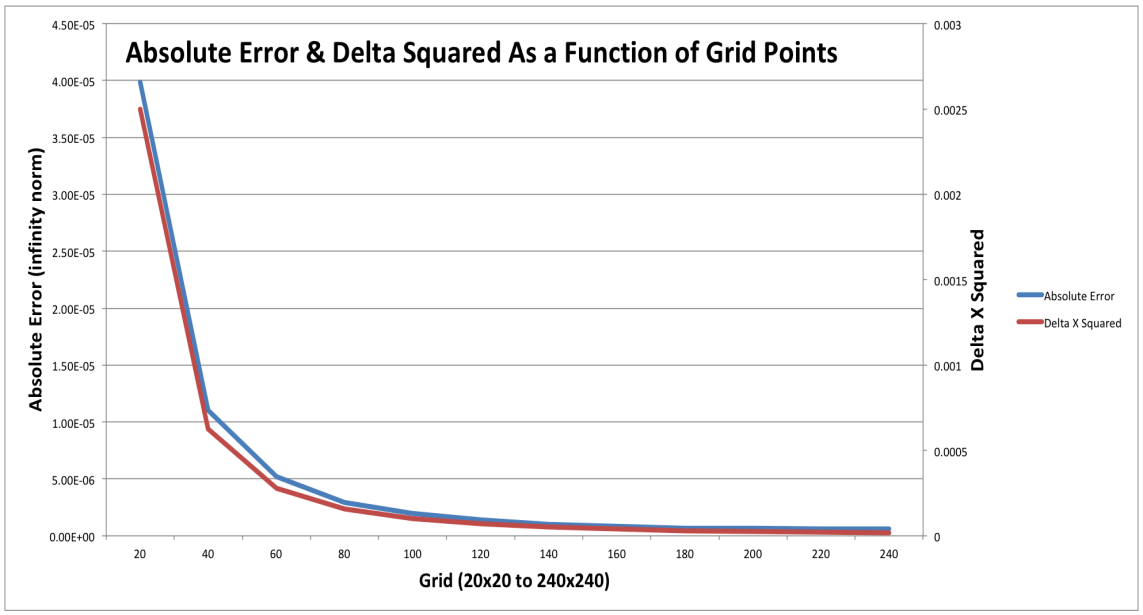

Figure 2: Absolute Error (red line, infinity normed) superimposed with Delta X squared (blue line), showing direct proportionality

Grid	Absolute Error
20x100	1.84e-06
40x80	2.89e-06
60x60	5.22e-06
80x40	1.13e-05
100x20	4.26e-05

Table 1: Solving on heterogeneous grids yields similar errors to solving on a homogeneous grid of 60x60

sized rectangular region on various heterogeneous grids and checking the absolute error. From Table 1, we can see that the computation errors from solving heterogeneous grids in the vicinity of the homogeneous grid 60x60 are comparable to that of 60x60. This proves that our solver can handle heterogeneous grids.

3 Convergence Analysis

Two understand how the number of processes affects the convergence behavior of our algorithm, we plot the normed (maximum) difference between a grid and its immediate next version ($||T^{k+1} - T^k||$), and we do this every 1000 update cycles. We again use the 1x2 rectangular region with and a 200x200 grid with error threshold 1e-12. Figure 3 shows the decay of the convergence error by plotting the natural log of $||T^{k+1} - T^k||$ against the cycle number the error was measured. The Project 1 implementation had periodic occurrences of faster convergence, but the reason for this is still not known. Figure 3 shows the results of the grid convergence analysis.

4 Performance

Lastly, we time the program for 50x50, 100x100, and 200x200 grids with 1, 2 and 4 threads. We display the total execution time, the times per thread, and the times per thread per cycle.

As you can see, the Time per thread per cycle (right-most column) doesn't change too much throughout. This means that this problem has good weak scaling (it scales well), but the time per thread explodes as it takes 11,000, 42,000, and 158,000 cycles to converge for 50x50, 100x100, and 200x200 grids, respectively.

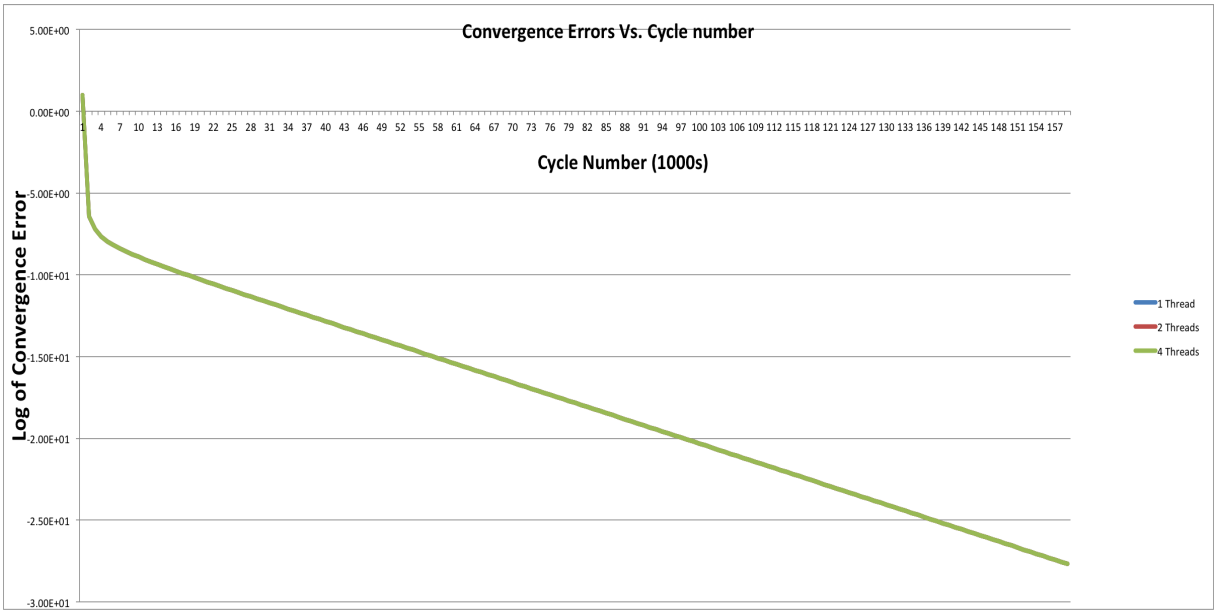

Figure 3: Convergences for different numbers of processes, and for the Project 1 implementation. All appear as one line since they are identical

Grid	Threads	Time/Thread (s)	Also per cycle (s)
50x50	1	1.85	0.000168
100x100	1	28.4	0.000676
200x200	1	443	0.00280
50x50	2	0.985	0.0000895
100x100	2	13.8	0.000329
200x200	2	204	0.00129
50x50	4	0.504	0.000046
100x100	4	7.33	0.000175
200x200	4	114	0.000722

Table 2: Timing analysis