Allen Suner
CSCI 322
Assignment 6: Write-Up
3/15/2014

a. No, each row must be solved from bottom to top. For example, if you have a linear system you are trying to solve, the bottom row of the x-matrix must be solved first because each the value of each row in the x-matrix depends on its predecessors.

b. Yes, if we create a reduction variable for the inner for loop. Since each row performs computations that are not dependant on the other, we are theoretically able to parallelize it. However, the subtraction presents a problem. We are only able to parallel loops with assignments that are associative, and the subtraction operator is not. So, we must create a reduction variable to add up the cumulative sum from the particular row in the A-matrix and then after summation we are able to subtract from the x-matrix safely. Change the assignment of this reduction variable to as follows.

c. No, there are inherent race conditions built into the column solution's outer loop if we were to parallelize it. In particular, pay close attention to fourth line and sixth line of that method. If one thread were to stop at line 4 and another thread continued on in its own iteration of that loop, and got to the sixth line of that method we are definitely going to overwrite an x-matrix index that is referenced in line 4 from the first thread.

d. Yes, for the same reason as response b, we can parallelize the inner loop of the column solution. Look at the sixth line in this solution. The reference to x[col] is constant for all iterations of this loop, so there are no race conditions present for this particular loop in this method. So, we are able to parallelize it again without the same precautions as were present in response b. Since, each iteration of the inner loop is referencing a different index (i.e. x[row] will be a different location in each iteration), we don't have to worry about the subtraction operator not being associative since it will only ever reference that index once, per execution of the inner loop since the outer loop is going sequentially.

e. (Response is presented in source code provided)

f. Testing was performed with sys/time.h library to get the times for all solutions of different types. Comparisons were performed only by observation and each type of solution was run multiple times to guarantee that the times received were accurate and reliable. So, after all tests were performed, the every row-based solution has outperformed the column-based solution on every type of scheduling. The scheduling paradigm that performed the best was the static scheduling paradigm.