

1 Introduction & Instructions

When you are finished submit all your work through the MyClasses page for this class. Create a directory called Homework05, put each programming exercise into its own subdirectory of this directory, zip the entire Homework05 directory up into the file Homework05.zip, and then submit this zip file to Homework #5. Make sure all the code and document files are included in the zip file.

2 Programming Exercises

1. In this exercise set we will continue our work with parallelizing some matrix operations. Early on in your linear algebra class you learned Gauss-Jordan elimination to reduce a matrix to reduced row-echelon form. If that is a little rusty you may want to look up the method either from your linear algebra text or online.
 - (a) Think about how you could divide the problem up to do matrix reduction in parallel. What operations of the reduction process are independent? What processes in the reduction process need to be sent to other processors? What needs to be given to all processors and what can be local to individual processors? Put these thoughts into a document and devise a way (algorithm) to parallelize Gauss-Jordan elimination on an arbitrary number of processors.
 - (b) Create a program from your algorithm that will do Gauss-Jordan elimination using MPI that will run on an arbitrary number of processors. The matrix sizes can be arbitrary, not just square.
 - (c) Also write a serial version for a single processor.
 - (d) Test this on small and large matrices and obviously compare the two results.
 - (e) Test and report the speedup and efficiency on the HPCL machines and your cluster.
2. In this exercise you will use your matrix-vector multiplication functions to implement the power method for finding the eigenvector corresponding to the largest eigenvalue of a matrix. Below is a method for doing this, this is not the best algorithm around but in many cases it will get the job done.
 - (a) Let A be an $n \times n$ matrix with real valued entries, and let x be the n -dimensional vector with all 1's.
 - (b) Update: $x \leftarrow Ax$
 - (c) Normalize: $x \leftarrow x/\|x\|_2$. Note that $\|x\|_2$ is the standard Euclidean norm and is defined as $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$.
 - (d) From this, x will converge (usually quickly) to the eigenvector corresponding to the largest eigenvalue. You may obtain an estimate of this eigenvalue by computing $\|Ax\|_2/\|x\|_2$.

- i. Make sure you test for convergence by choosing a small difference tolerance (say 10^{-16} to 10^{-14}) and stopping when the new value of x differs from the previous one by less than that amount.
- ii. You will also want to set an upper-bound on the number of iterations to perform and report an error if it has not converged after that limit.

Write an MPI program that will do this method in both serial and parallel. As usual, test this on small and large matrices and obviously compare the two results. Test and report the speedup and efficiency on the HPCL machines and your cluster.