Solving Linear Equations by Jacobi Iteration

This assignment is due at class time Thursday, 6 November 2001.

You are to write a program to do Jacobi iteration in parallel.

Your goal should be to solve by Jacobi iteration a matrix equation $Ax = b$.

You may assume that the matrix A is square with the number of rows and columns a power of 2 up to $2^8 = 256$.

Your program should be able to run on 4, 8, or 16 processors.

Your program must be able to deal with the number of rows, the number of columns, and the number of processors as variables. Hard coding options for any of these variables will result in zero credit.

You should be able to use the maximal parallelism permissible by dividing up the matrix by rows into blocks of rows whose computation is handled by different processors.

As before, use your social security number to separate yourself from other people in the class with regard to machine use.

You will find two sample input files in `/usr/public/buellexamples` as files `jacobi.small` and `jacobi.large`.

The input format for your program should be as in these files:

1. The maximum number of iterations to attempt
2. The error tolerance (ϵ)
3. The number of rows
4. The number of columns
5. The matrix A

6. The right hand side b

7. An initial approximate solution

Your program should iterate a solution vector $(x_1^{(t)}, x_2^{(t)}, ..., x_n^{(t)})$ for iterations $t = 0, 1, ...,$ until successive iterations have converged, with convergence defined to be

$$\sqrt{\sum_{i=1}^{n}(x_i^{(t)} - x_i^{(t-1)})^2} < \epsilon.$$

(Note that there's no reason for something simple like this that the maximum iteration count ever be reached.)

Some hints.

First, write a noddy that does Jacobi iteration as a sequential program. That way, you'll have the answer to compare against when you do the parallel version.

Second, to generate a matrix that is guaranteed to be solvable by Jacobi iteration, just generate a matrix with random numbers scaled between 0.0 and 1.0 and then add at least the max of the number of rows and the number of columns to the element on the diagonal. Then the matrix is guaranteed to be strongly diagonally dominant and Jacobi iteration is guaranteed to work.