CS320/CSE302/ECE392: INTRODUCTION TO PARALLEL PROGRAMMING FOR SCIENTISTS AND ENGINEERS

Spring 2000

Machine Problem 5

Assigned: April 18, 2000
Due: May 3, 2000.

The objective of this MP is to implement a parallel version of gaussian elimination using MPI. The objective of gaussian elimination is to convert a general system of linear equations into a triangular system of equations. A system of equations is represented in matrix form as $Ax = b$. The method uses the characteristic of linear equations that any row can be replaced by that row added to another row multiplied by a constant. The procedure starts at the first row and works towards the bottom row. At the *i*th row, each row *j* below the *i*th row of the form

$$a_{j,1}, a_{j,2}, a_{j,3}, \ldots a_{j,n}$$

is replaced by the row

$$a_{j,1} + a_{i,1}\left(-\frac{a_{j,i}}{a_{i,i}}\right), a_{j,2} + a_{i,2}\left(-\frac{a_{j,i}}{a_{i,i}}\right), a_{j,3} + a_{i,3}\left(-\frac{a_{j,i}}{a_{i,i}}\right), \ldots, a_{j,n} + a_{i,n}\left(-\frac{a_{j,i}}{a_{i,i}}\right)$$

The constant used for row j is $-\frac{a_{j,i}}{a_{i,i}}$. This has the effect of making all elements in the *i*th column below the *i*th row zero because

$$a_{j,i} = a_{j,i} + a_{i,i}\left(-\frac{a_{j,i}}{a_{i,i}}\right) = 0$$

Figure 1 shows the situation when row *i* is being processed. Row *i* is called the pivot row. At this stage, all columns to the left of column *i* below the diagonal have already been set to zero by previous actions. Once the procedure has reached the last row, the resultant system will be an upper triangular system.

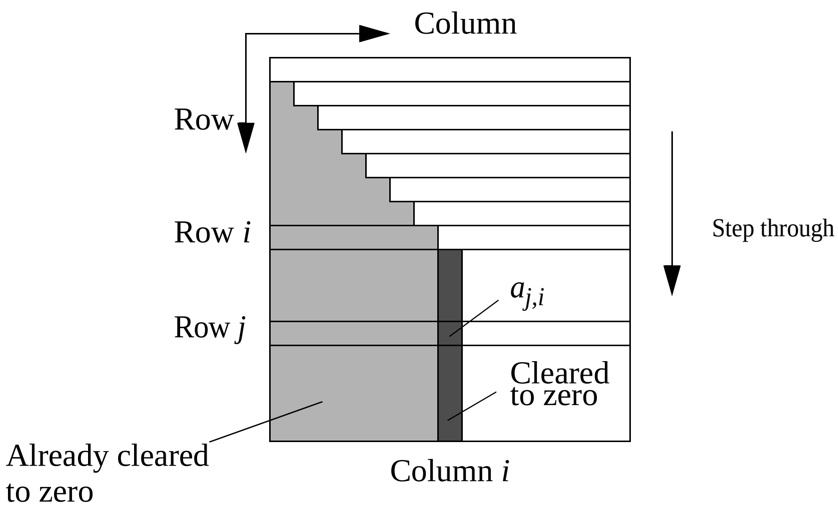

Figure 1. Gaussian Elimination

The following is an outline of the sequential code to implement Gaussian elimination

```
do i=1,n-1
     do j=i+1, n
          m=a(j,i)/a(i,i)
          do k=i,n
               a(j,k)=a(j,k)-a(i,k)*m
          end do
          b(j)=b(j)-b(i)*m
     end do
end do
```

Looking at the sequential code, clearly there is scope for parallelization. The inner loop modifying row *j* is a parallel loop. One way to partition the problem is to arrange that one processor holds one row of elements and operates on this row. This requires *n* processors for *n* equations. Before a processor can operate on its row, it must receive the elements from row *i*, which can be achieved through a broadcast operation. First, processor P_0 (holding row 1) broadcasts elements of its row to each of the other *n-1* processors. Then these processors compute their multiplier and modify their row. The procedure is repeated with each of $P_1, P_2 ... P_{n-2}$ broadcasting elements of their rows. The elements of row *i* broadcast by P_i are `a(i,i+1)`, `a(i,i+2)`, `...`, `a(i,n)`, and `b(i)`, and a total of *n-i+1* elements for $1 \le i < n$, as shown in Figure 2. Processors P_{i+1} to P_{n-1}

inclusive receive the broadcast message (at total *n-i-1* processors) and they operate upon *n-i+1* elements of their rows.

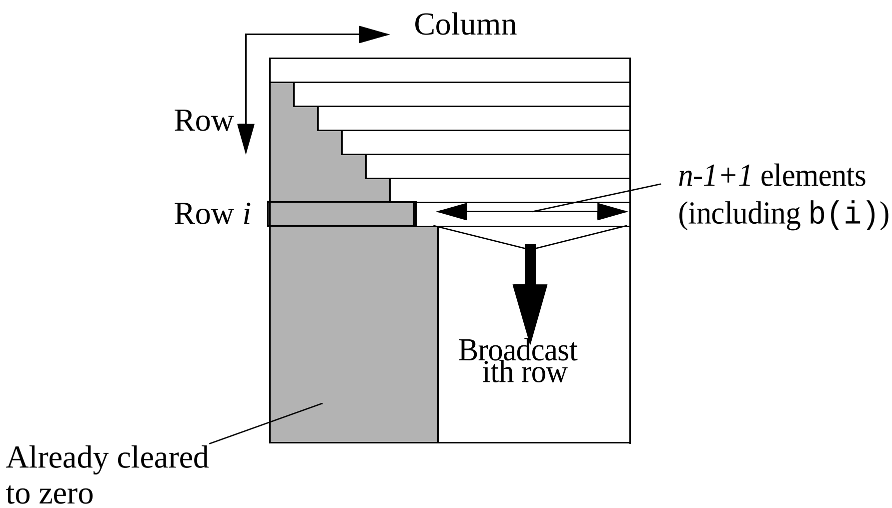

Figure 2. Broadcast in parallel implementation of Gaussian elimination

To reduce the number of processors to be less than *n*, we can partition the matrix into groups of rows for each processor, such as the so-called strip partitioning shown in Figure 3. With *p* processors, each processor holds *n/p* rows. Unfortunately, again processors do not participate in the computation after their last row is processed. An alternative, which actually equalizes the processor workload is the *cyclic-striped partitioning*, as shown in Figure 4. Now each processor will have some activity for a greater period. For example, P_0 is assigned rows *1, n/p+1, 2n/p+1, 3n/p* and is active until row *3n/p+1* is passes. P_1 is assigned rows *2, n/p+2, 2n/p+2, 3n/p+2* and is active until row *3n/p+2* is passed. A final way the matrix could be partitioned is in a checkerboard fashion.

In your program you should implement the *cyclic-striped partitioning* distribution. Processor 0 should read the value *n* and the input matrix (*n* x *n+1*) and print the resulting triangular matrix. Those registered for one unit should also implement the checkerboard distribution.