High Performance Numerical Libraries

Sathish Vadhiyar

Gaussian Elimination - Review

Version 1

for each column i

  zero it out below the diagonal by adding multiples of row i to later rows

for i= 1 to n-1

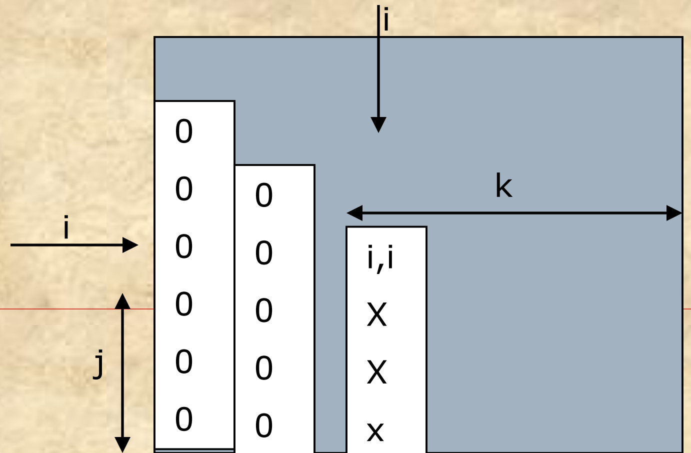
  for each row j below row i

  for j = i+1 to n

    add a multiple of row i to row j

    for k = i to n

      A(j, k) = A(j, k) – A(j, i)/A(i, i) * A(i, k)



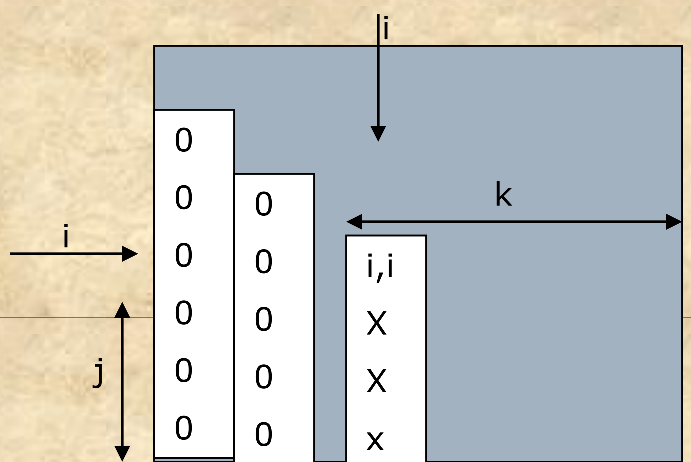
Gaussian Elimination - Review

Version 2 – Remove A(j, i)/A(i, i) from inner loop

for each column i

  zero it out below the diagonal by adding multiples of row i to later rows

```
for i= 1 to n-1
  for each row j below row i
  for j = i+1 to n
    m = A(j, i) / A(i, i)
    for k = i to n
      A(j, k) = A(j, k) – m* A(i, k)
```

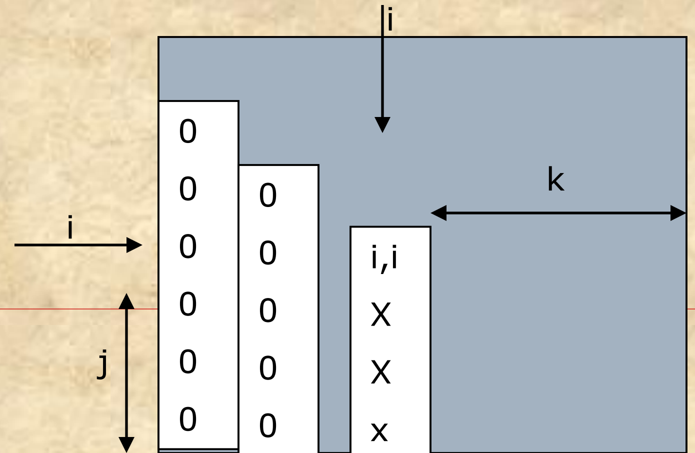


Gaussian Elimination - Review

Version 3 – Don't compute what we already know

for each column i
  zero it out below the diagonal by adding multiples of row i to later rows

```
for i= 1 to n-1
  for each row j below row i
  for j = i+1 to n
    m = A(j, i) / A(i, i)
    for k = i+1 to n
      A(j, k) = A(j, k) – m* A(i, k)
```



Gaussian Elimination - Review

Version 4 – Store multipliers m below diagonals

for each column i

  zero it out below the diagonal by adding multiples of row i to later rows

for i= 1 to n-1

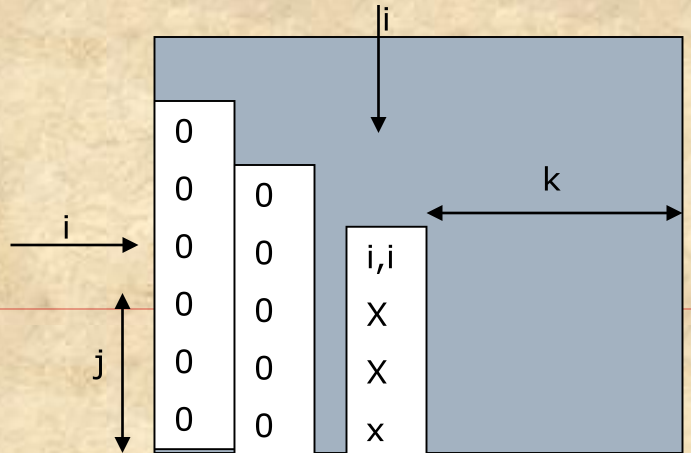
  for each row j below row i

  for j = i+1 to n

    A(j, i) = A(j, i) / A(i, i)

    for k = i+1 to n

      A(j, k) = A(j, k) – A(j, i)* A(i, k)



GE - Runtime

- Divisions

$$1 + 2 + 3 + \ldots (n-1) = n^2/2 \text{ (approx.)}$$

- Multiplications / subtractions

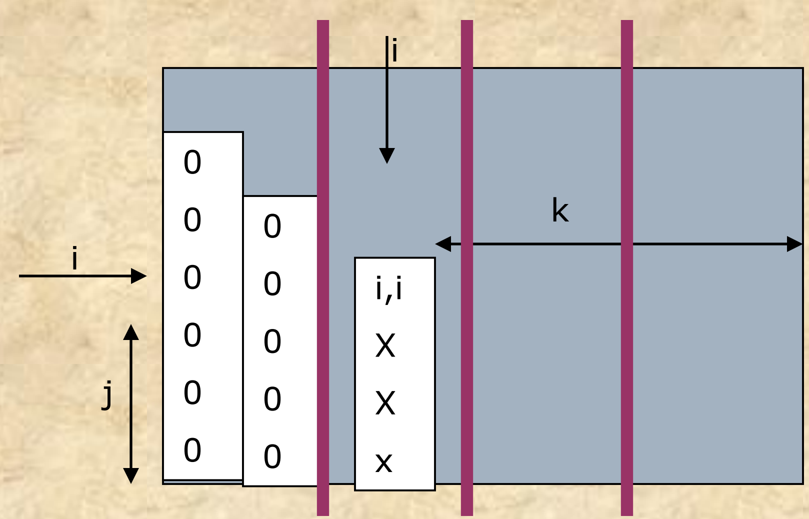
$$1^2 + 2^2 + 3^2 + 4^2 + 5^2 + \ldots. (n-1)^2 = n^{3/}3 - n^2/2$$

- Total

$$2n^3/3$$

Parallel GE

- 1st step – 1-D block partitioning along blocks of n columns by p processors



1D block partitioning - Steps

1. Divisions

$n^2/2$

2. Broadcast

$x\log(p) + y\log(p-1) + z\log(p-3) + \ldots \log 1 < n^2 \log p$

3. Multiplications and Subtractions

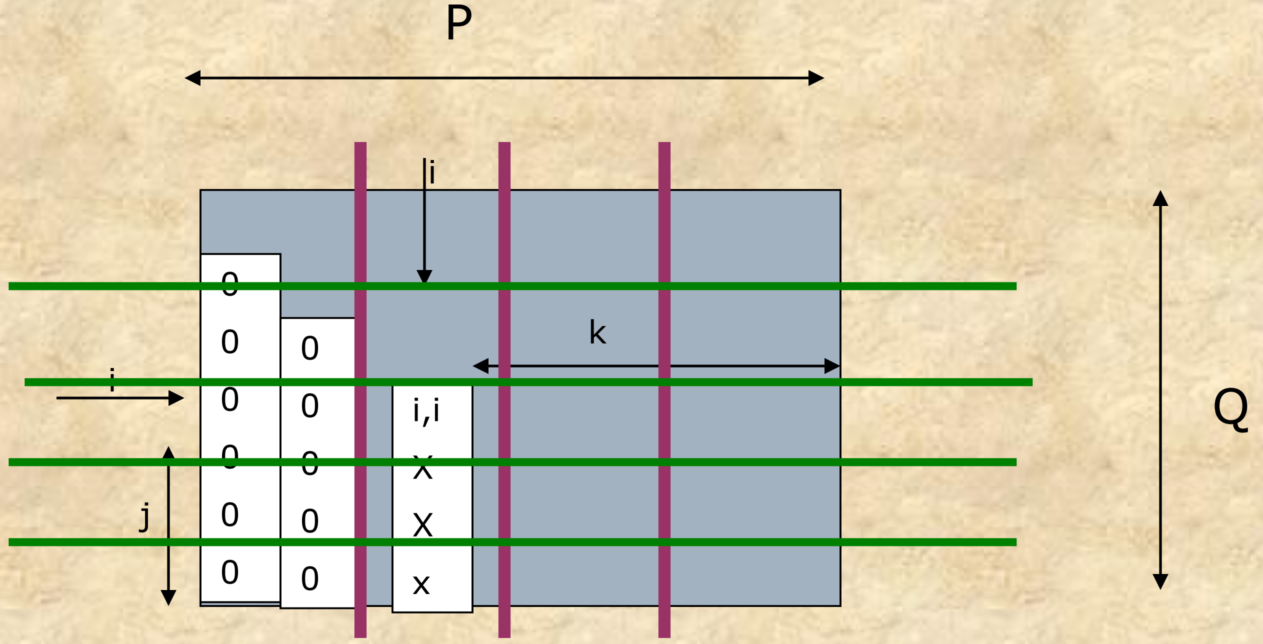
$(n-1)n/p + (n-2)n/p + \ldots .\ 1x1 = n^3/p$ (approx.)

Runtime:

$< n^2/2 + n^2\log p + n^3/p$

2-D block

- To speedup the divisions



2D block partitioning - Steps

1. Broadcast of (k,k)

   logQ

2. Divisions

   n^2/Q (approx.)

3. Broadcast of multipliers

   $x\log(P) + y\log(P-1) + z\log(P-2) + ... = n^2/Q \log P$

4. Multiplications and subtractions

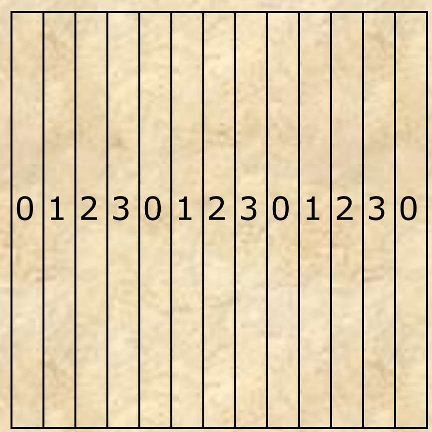
   n^3/PQ (approx.)

Problem with block partitioning for GE

- Once a block is finished, the corresponding processor remains idle for the rest of the execution
- Solution? -

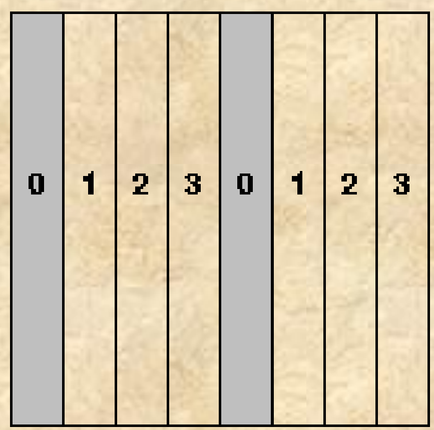
Onto cyclic

- The block partitioning algorithms waste processor cycles. No load balancing throughout the algorithm.
- Onto cyclic



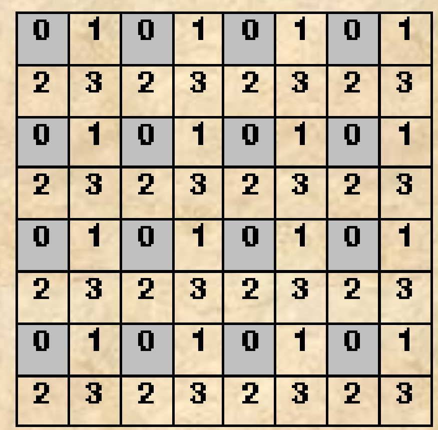
cyclic

Load balance



1-D block-cyclic

Load balance, block operations, but column factorization bottleneck



2-D block-cyclic

Has everything

Block cyclic

- Having blocks in a processor can lead to block-based operations (block matrix multiply etc.)
- Block based operations lead to high performance

GE: Miscellaneous
GE with Partial Pivoting

- 1D block-column partitioning: which is better? Column or row pivoting

•Column pivoting does not involve any extra steps since pivot search and exchange are done locally on each processor. O(n-i-1)

•The exchange information is passed to the other processes by piggybacking with the multiplier information

• Row pivoting

• Involves distributed search and exchange – O(n/P)+O(logP)

- 2D block partitioning: Can restrict the pivot search to limited number of columns

Sparse Iterative Methods

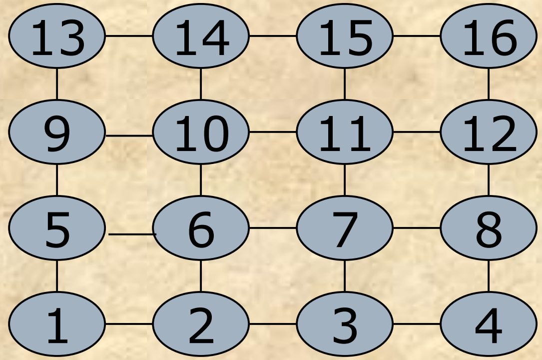
Iterative & Direct methods – Pros and Cons.

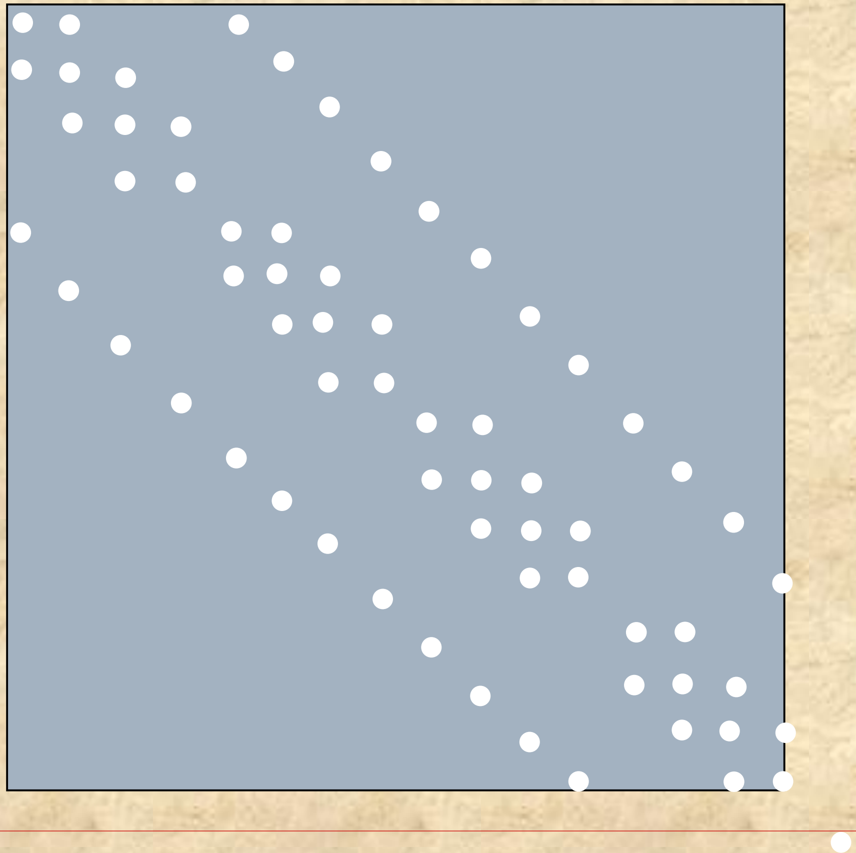
- Iterative methods do not give accurate results.
- Convergence cannot be predicted
- But absolutely no fills.

Parallel Jacobi, Gauss-Seidel, SOR

- For problems with grid structure (1-D, 2-D etc.), Jacobi is easily parallelizable
- Gauss-Seidel and SOR need recent values. Hence ordering of updates and sequencing among processors
- But Gauss-Seidel and SOR can be parallelized using red-black ordering or checker board

2D Grid example

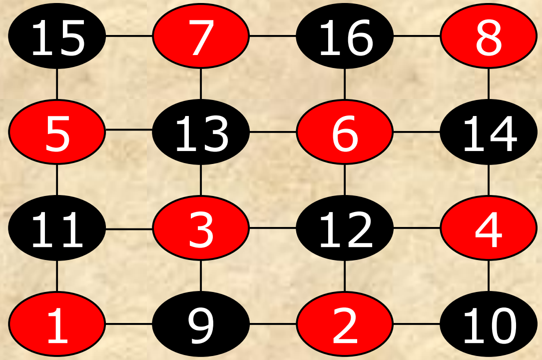


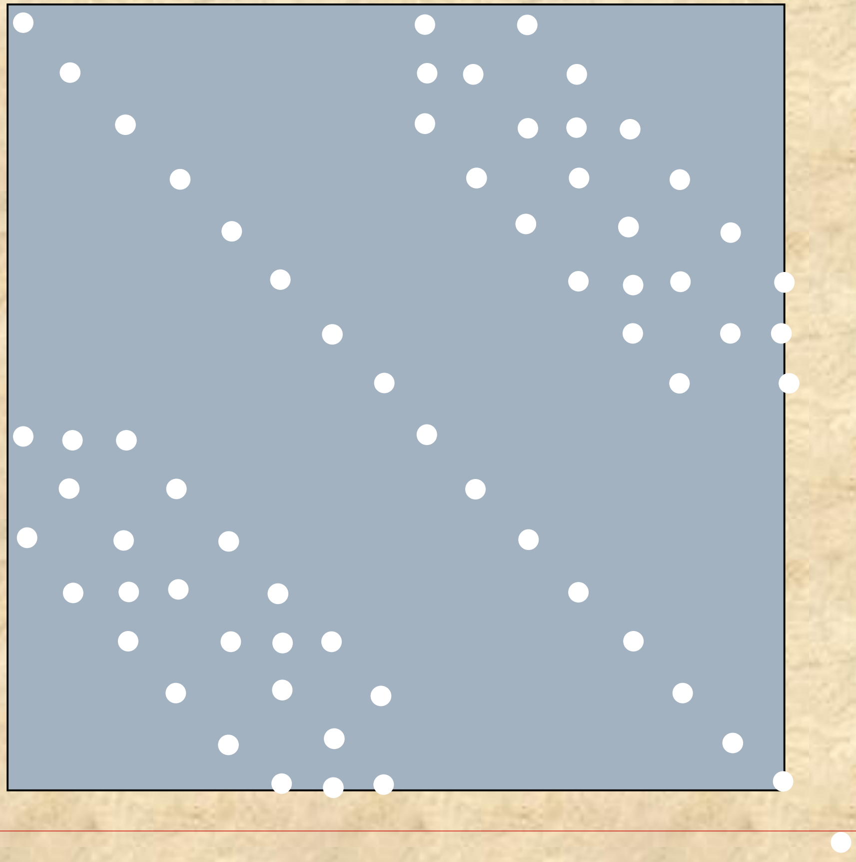


Red-Black Ordering

- Color alternate nodes in each dimension red and black
- Number red nodes first and then black nodes
- Red nodes can be updated simultaneously followed by simultaneous black nodes updates

2D Grid example – Red Black Ordering





- In general, reordering can affect convergence

Graph Coloring

- In general multi-colored graph coloring Ordering for parallel computing of Gauss-Seidel and SOR