# Notes on JacobiBC

A bug reported by Robert Ammon

# Notes on Jacobi_BC

Data layout

- Consider a square Jacobi relaxation mesh of length

$$MATRIX\_DIM = 3$$

- Assume that the boundary values are added to this mesh around the border of the 3 x 3 mesh.
- The resulting augmented mesh is characterized by a size given by

$$mesh\_size = MATRIX\_DIM + 2$$

# Notes on Jacobi_BC

- The augmented mesh is constructed in procedure

  void MakeinitalMatrix

# Notes on Jacobi_BC

- In file jacobi.c notice that both JacobiBC and JacobiBC2 update the x[i] iterate using the following statements

```
nModelDim = matrix_size - 2;

int nOA_size = nModelDim  * nModelDim;
for (m=0; m<nOA_size; m++)
{    /* get next one from calc order vector */
   i = o[m];
   x[i] = 0.25 * ( last_x[i-1] + last_x[i+1]
          + last_x[i-nModelDim]  + last_x[i+nModelDim]);
```

# Notes on Jacobi_BC

- Notice that the iterates are updated using a linear array x[i]
- For illustration purposes, suppose

$$\text{MATRIX\_DIM} = 3$$

- In order to construct the linear array x[i], the augmented mesh that embeds the relaxation mesh is organized in a row-wise linear wraparound fashion

# Layout of the Augmented Mesh

| | | | | |
|---|---|---|---|---|
| $x[0,0]$ | $x[0,1]$ | $x[0,2]$ | $x[0,3]$ | $x[0,4]$ |
| $x[1,0]$ | $x[1,1]$ | $x[1,2]$ | $x[1,3]$ | $x[1,4]$ |
| $x[2,0]$ | $x[2,1]$ | $x[2,2]$ | $x[2,3]$ | $x[2,4]$ |
| $x[3,0]$ | $x[3,1]$ | $x[3,2]$ | $x[3,3]$ | $x[3,4]$ |
| $x[4,0]$ | $x[4,1]$ | $x[4,2]$ | $x[4,3]$ | $x[4,4]$ |

The boundary elements are highlighted in blue; whereas, the relaxation elements are highlighted in red

# Construction of the Linear Array x[i]

The construction of the linear array x[i] is performed in the procedure:

*void GetCalcOrder*(int o[MATRIX_DIM*MATRIX_DIM])

- In this procedure, the "natural" coordinate ordering of the relaxation mesh is established using the indices ki and kj that forms the tuple (ki,kj)
- The linear distance between relaxation update elements are implicitly based on the augmented mesh dimension mesh_size.

# “Natural” Coordinate Ordering of the Relaxation Mesh

$x[0,0]$ $x[0,1]$ $x[0,2]$

$x[1,0]$ $x[1,1]$ $x[1,2]$

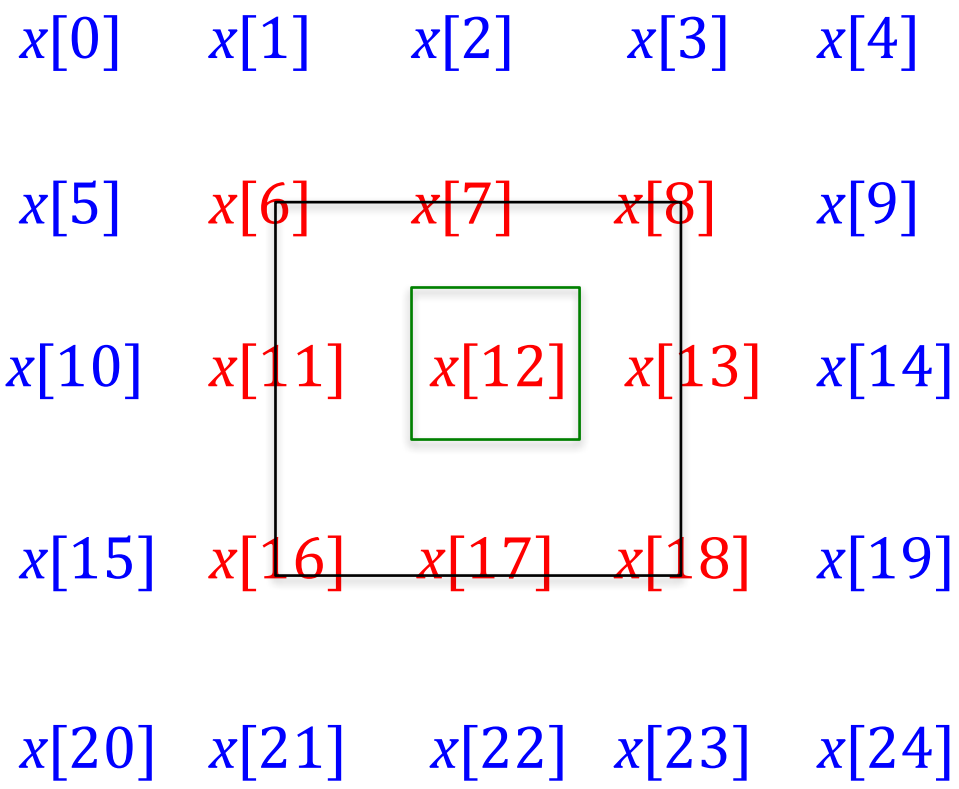
$x[2,0]$ $x[2,1]$ $x[2,2]$

# Construction of the Linear Array x[i]

- For MATRIX_DIM = 3, *GetCalcOrder* returns:

| *M* = 0 (Ring Level 0) | | *M* = 1 (Ring Level 1) | |
|---|---|---|---|
| (ki, kj) | | | |
| (0, 0) | o(0) = 6 | (0, 0) | NoOp |
| (0, 1) | o(1) = 7 | (0, 1) | NoOp |
| (0, 2) | o(2) = 8 | (0, 2) | NoOp |
| (1, 0) | o(3) = 11 | (1, 0) | NoOp |
| (1, 1) | NoOp | (1, 1) | o(8) = 12 |
| (1, 2) | o(4) = 13 | (1, 2) | NoOp |
| (2, 0) | o(5) = 16 | (2, 0) | NoOp |
| (2, 1) | o(6) = 17 | (2, 1) | NoOp |
| (2, 2) | o(7) = 18 | (2, 2) | NoOp |

# Ring Level Linear Array x[i] Structure




The **black** square represents the m = 0 ring level of updates; whereas, the **green** square represents the m = 1 ring level of updates

# Construction of the Linear Array x[i]

- The linear array x[i] is numbered in a row-wise linear wraparound fashion based on the size of the augmented mesh = matrix_size * matrix_size.
- The elements above and below the relaxation mesh point to be updated are separated by exactly –mesh_size and +mesh_size steps away.
- The relaxation mesh size determines the number of updates to be performed.
- For this reason, the JacobiBC and JacobiBC2 requires a straight forward realignment:

# Notes on Jacobi_BC

- Based on this example, the iterate update assignment should state (as reported by Robert Ammon)

```
nModelDim = matrix_size – 2;

int nOA_size = nModelDim  * nModelDim;
for (m=0; m<nOA_size; m++)
{   /* get next one from calc order vector */
   i = o[m];
   x[i] = 0.25 * ( last_x[i-1] + last_x[i+1]
          + last_x[i-matrix_size]  + last_x[i+matrix_size]);
```

# Standard Laplace Iteration Methods

Jacobi row wised + individual update (JacobiR2)

x11, x12 , x13 ... x21, x22, x23 .....

```
For i =1 to n
    For j=1 to n
        x[i,j] = 0.25 * ( xold[i-1,j] + xold [i+1,j]
                        + xold [i,j-1] + xold[i,j+1] )
For i =1 to n
    For j=1 to n
        Xold(i,j) = x(i,j)
```

# Standard Laplace Iteration Methods

Gauss-Seidel row wised (Gauss-seidel)

x11, x12 , x13 ... x21, x22, x23 .....

```
For i =1 to n
    For j=1 to n
        x[i,j] = 0.25 * ( x[i-1,j] + x [i+1,j]
                        + x [i,j-1] + x[i,j+1] )
```