

## Serie 11

*Remark: Programs shall be executed on altix!*

**Exercise 1** *In this exercise, we want to parallelize the Gauss-Seidel method using the red-black (or checkerboard) numbering. The example is the matrix for the Laplacian on a square, discretized with the usual five-points stencil and a number of nodes  $N$  on the square sides. Recall that, by numbering first the red nodes and then the black ones, we get the block matrix (see the course, Chapter 4):*

$$A = \begin{pmatrix} D_r & A_{rb} \\ A_{br} & D_b \end{pmatrix} \quad (1)$$

with  $D_r, D_b$  diagonal and  $A_{rb}, A_{br}$  with dense bands. One iteration for updating the red (resp. black) nodes in  $\mathbf{z}_r$  (resp. in  $\mathbf{z}_b$ ) reads

$$\begin{aligned} \mathbf{z}_r^{k+1} &= D_r^{-1}(\mathbf{b}_r - A_{rb}\mathbf{z}_b^k), \\ \mathbf{z}_b^{k+1} &= D_b^{-1}(\mathbf{b}_b - A_{br}\mathbf{z}_r^{k+1}). \end{aligned} \quad (2)$$

1. Play with the Matlab script `rb.m` for discovering the sparsity structure of  $A$  (this is for the case  $N$  odd).
2. Draw the red-black numbering for  $N=4$ .
3. At each iteration, one has to solve the two equations of (2) sequentially. Explain why each equation can be parallelized.
4. The program `gsrb.c` implements the sequential red-black Gauss-Seidel method for  $N$  odd in the case where the r.h.s is made of ones. Since the matrices  $D_r, D_b, A_{rb}$  and  $A_{br}$  contain constants, a simplified Morse data structure is used (no arrays to store the matrices). Parallelize this code with the help of OpenMP.
5. Compare, for  $N = 25, 65, 81$ , the CPU times of the non parallel and the parallel versions (with a number of processors equal to 1, 2, 4, 8, 12, 16). Notice that the number of iterations increases with  $N$ .

**Exercise 2** *In the course, the first version of the conjugate gradient algorithm for solving a linear system with a symmetric positive definite matrix  $A$  is given as follows:*

1. Initialization:  $\mathbf{x}^0 = \mathbf{0}, \mathbf{p}^0 = \mathbf{r}^0, k = 0$
2. Iterations:

$\alpha_k = (\mathbf{r}^k, \mathbf{p}^k) / (A\mathbf{p}^k, \mathbf{p}^k)$
$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$
$\mathbf{r}^{k+1} = \mathbf{b} - A\mathbf{x}^{k+1}$
$\beta_k = -(A\mathbf{p}^k, \mathbf{r}^{k+1}) / (A\mathbf{p}^k, \mathbf{p}^k)$
$\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \beta_k \mathbf{p}^k$

The last presented version, which is the one used for programming the method, is given by

1. Initialization:  $\mathbf{x}^0 = \mathbf{0}, \mathbf{p}^0 = \mathbf{g}^0, k = 0$
2. Iterations:

$\alpha_k = \ \mathbf{g}^k\ ^2 / (A\mathbf{p}^k, \mathbf{p}^k)$
$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$
$\mathbf{g}^{k+1} = \mathbf{g}^k - \alpha_k A\mathbf{p}^k$
$\beta_k = \ \mathbf{g}^{k+1}\ ^2 / \ \mathbf{g}^k\ ^2$
$\mathbf{p}^{k+1} = \mathbf{g}^{k+1} + \beta_k \mathbf{p}^k$

For establishing the second version from the first one, prove the following results:

1. the residual  $\mathbf{r}^k = -\mathbf{g}^k$  where  $\mathbf{g}^k$  corresponds to the gradient of the error  $E(\mathbf{x})$  at the point  $\mathbf{x}^k$ ,
2.  $\mathbf{g}^{k+1} = A\mathbf{x}^{k+1} - \mathbf{b} = \mathbf{g}^k - \alpha_k A\mathbf{p}^k$ ,
3.  $\alpha_k = (\mathbf{r}^k, \mathbf{p}^k) / (A\mathbf{p}^k, \mathbf{p}^k) = \|\mathbf{g}^k\|^2 / (A\mathbf{p}^k, \mathbf{p}^k)$ ,
4.  $\beta_k = -(A\mathbf{p}^k, \mathbf{r}^{k+1}) / (A\mathbf{p}^k, \mathbf{p}^k) = \|\mathbf{g}^{k+1}\|^2 / \|\mathbf{g}^k\|^2$ .
5. Discuss the advantages of the second version.