Numerical Algorithms

Winter semester 2018/2019
Prof. Dr. Marc Alexander Schweitzer
Denis Duesseldorf

universitätbonn

Exercise sheet 4.

Submission on **Tuesday, 2018-11-13, before lecture**.

Exercise 12. (Computation rules for symmetric, positive (semi)definite matrices)

Recall that with $< \cdot, \cdot >$ denoting an inner product on $\mathbb{R}^n = \mathbb{R}^I$ we call a *symmetric* matrix $A = (a_{i,j})_{i,j \in I} \in \mathbb{R}^{n \times n}$

1. positive definite, if $\forall 0 \neq x \in \mathbb{R}^n :< Ax, x >> 0$. We use the notation $A > 0$. Analogously we have $A < 0$.
2. positive semidefinite, if $\forall x \in \mathbb{R}^n :< Ax, x > \geq 0$. We use the notation $A \geq 0$. Analogously we have $A \leq 0$.
3. If B is another *symmetric* matrix, we introduce the notation $A > B$ if $A - B > 0$ i.e. if $A - B$ is positive definite. Anologously we have $A < B$.
4. If A, B are (possibly non-symmetric) matrices, we write $A \geq B$ if $A - B \geq 0$. Anologously we have $A \leq B$.

Recall that $A > 0 \Leftrightarrow \sigma(A) \subset (0, \infty)$, $A \geq 0 \Leftrightarrow \sigma(A) \subset [0, \infty)$ and that $A > 0$ implies that A is regular.
Show that the following rules for this notation hold.

a) $A > 0 \Leftrightarrow CAC^T > 0$, $A > B \Leftrightarrow CAC^T > CBC^T$ for all regular matrices $C \in \mathbb{R}^{n \times n}$,

b) $A \geq 0 \Rightarrow CAC^T \geq 0$, $A \geq B \Rightarrow CAC^T \geq CBC^T$ for all matrices $C \in \mathbb{R}^{n \times n}$,

c) $A, B \geq 0 \Rightarrow A + B \geq 0$

d) $A > 0, B \geq 0 \Rightarrow A + B > 0, A + B \geq A, A + B > B$.

e) $A > 0 \Rightarrow \xi A > 0$ for all $\xi > 0$,

f) $\zeta I \leq A \leq \xi I \Leftrightarrow \sigma(A) \subset [\zeta, \xi]$, $-\xi I \leq A \leq \xi I \Leftrightarrow \|A\|_2 \leq \xi$ for symmetric A,

g) $A \geq B > 0 \Rightarrow 0 < A^{-1} \leq B^{-1}$,

h) $A > 0 \Leftrightarrow A^{-1} > 0$,

i) All principal submatrices of a positive (semi)definite matrix are positive (semi)definite,
$A > 0 \Rightarrow \forall J \subset I : (a_{i,j})_{i,j \in J} > 0$ and $A \geq 0 \Rightarrow \forall J \subset I : (a_{i,j})_{i,j \in J} \geq 0$,

j) All diagonals of a positive (semi)definite matrix are positive (non-negative),
$A > 0 \Rightarrow \forall i \in I : a_{i,i} > 0$ and $A \geq 0 \Rightarrow \forall i \in I : a_{i,i} \geq 0$,

k) The diagonal and each block-diagonal submatrix of a positive (semi)definite matrix are positive (semi)definite.

(4 Points)

Exercise 13. (Convergence of block-Jacobi and block-Gauss-Seidel)

In the lecture and exercise 11 we have already established a link between Gauss-Seidel / Jacobi methods and Schwarz methods if the partition Λ is ordered and non-overlapping. In this case the convergence proofs are also easier and can be directly adapted from the classical Gauss-Seidel and Jacobi convergence proofs.

These classical proofs use (variations of) the following convergence criterium for an iterative method with matrices $M = I - W^{-1}A = I - NA$ in third normal form.

> If
> $$W + W^T > A > 0$$
> then the iteration $x^{m+1} = x^m - W^{-1}(Ax^m - b)$ converges monotonously in the energy norm $\|\cdot\|_A$
> $$\rho(M) \leq \|M\|_A < 1 \ .$$

Let $A > 0$ and Λ be ordered and nonoverlapping, yielding the results from exercise 11. Show that

a) The multiplicative Schwarz/block-Gauss-Seidel method converges.

b) The damped additive Schwarz/block-Jacobi-method converges for θ small enough. A sufficient condition is $\theta < \frac{2}{\#\Lambda}$.

Hint: Use exercise 12 and the lemma from the lecture establishing $A \leq \#\Lambda W_{\text{add}}$.

(4 Points)

Exercise 14. (Nested iterations)

In the lecture's section on subdomain solvers we have already seen a case of a nested iteration. An iterative method Φ with matrices $M = I - NA = I - W^{-1}A$

$$x^{m+1} = x^m - W^{-1}(Ax^m - b)$$

which is computed as

$$W\delta = (Ax^m - b) \; , \qquad\qquad x^{m+1} = x^m - \delta$$

hinges on an efficient solution of $W\delta = d$.
We, thus, introduce another iteration Φ_W with $M_W = I - N_W W = I - W_W^{-1}W$, namely

$$\delta^{m+1} = \delta^m - W_W^{-1}(W\delta^m - d) = M_W\delta^m + N_W d$$

for the solution of $W\delta = d$.
Let Φ_k, M_k, N_k, W_k be the resulting iterative method when using k steps of the nested iteration.
Assume that Φ, Φ_W are consistent. Show that

a) $M_k = I - \sum_{q=0}^{k-1} M_W^q N_W A = M + M_W^k W^{-1} A$,

b) $N_k = (I - M_W^k)W^{-1}$,

c) If M_W has and eigenvalue λ with $\lambda^k = 1$ then Φ_k diverges. Otherwise we have $W_k = W(I - M_W^k)^{-1}$.

d) Φ_k is a consistent linear iteration.

Hint: For an iteration $x^{m+1} = Mx^m + Nb$ It holds that

$$x^m = M^m x^0 + \sum_{q=0}^{m-1} M^q N b \; .$$

(4 Points)

Programming exercise 5. (L-Shape domain decomposition)

For the application of the domain decomposition methdos from the lecture we now need to first establish a mesh with a splitting of the index set I of the nodes $x_i, i \in I$ into subsets

$$\Lambda = \{\lambda_1, \ldots, \lambda_K\} \, .$$

a) Write a function to create a mesh of

$$\Omega_L := (-1,1)^2 \setminus (0,1)^2 \, , \quad \Gamma_D := \partial[-1,1]^2 \cap \partial\Omega_L \, , \quad \Gamma_N := \partial\Omega_L \setminus \Gamma_D = [0,1]^2 \cap \partial\Omega_L \, .$$

Hint: Modify the functions for creating the meshes in programming exercises 1d) and 2b). Take special care of indices of shared nodes when creating the arrays for the triangles and boundary edges.

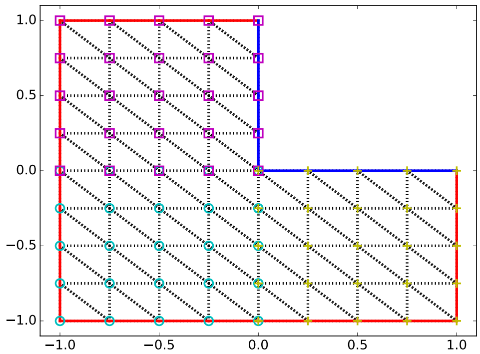
b) Create an overlapping partition of the index set I of the vertices into

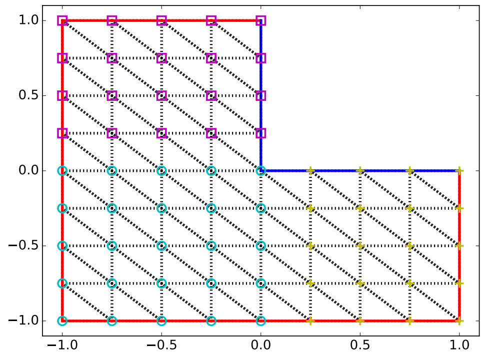
$$\begin{aligned} \lambda_1 &:= \{i \in I : x_i \in [-1,0]^2\} \, , \\ \lambda_2 &:= \{i \in I : x_i \in [-1,0] \times [0,1]\} \, , \\ \lambda_3 &:= \{i \in I : x_i \in [0,1] \times [-1,0]\} \, . \end{aligned}$$

c) Create an non-overlapping partition Λ_n

$$\begin{aligned} \lambda_1 &:= \{i \in I : x_i \in [-1,0]^2\} \, , \\ \lambda_2 &:= \{i \in I : x_i \in [-1,0] \times (0,1]\} \, , \\ \lambda_3 &:= \{i \in I : x_i \in (0,1] \times [-1,0]\} \, . \end{aligned}$$

d) Plot the resulting mesh, marking the vertices according to each of the partitions.





e) For both cases, compute the local problem (sub)matrices $K_\lambda = r_\lambda K p_\lambda$ of the global stiffness matrix K computed from programming exercise 3. *Hint*: This can be done using only `array` slices.

(4 Points)

Send to `duesseld@ins.uni-bonn.de`