Numerical Algorithms

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

universitätbonn

Exercise sheet 5.

Submission on **Tuesday, 2018-11-20, before lecture**.

Exercise 15. (Schwarz iterations with approximate subdomain solvers)

In the lecture's section on subdomain solvers we have replaced A_λ^{-1} by W_λ^{-1}, an approximation of A_λ^{-1}. It can, for instance, be obtained from using an inner iteration for $A_\lambda^{-1} y = c$. We assume that W_λ stems from a symmetric iteration and that

$$\delta W_\lambda \leq A_\lambda \leq \Delta W_\lambda \,, \qquad\qquad 0 < \delta \leq \Delta \,.$$

Reminder: An iteration with $M = I - NA = I - W^{-1}A$ is called symmetric if $W > 0$ and A symmetric.
This yields and inner iteration Φ_λ and global iteration matrices M^s_{mult}, M^s_{add}

$$\Pi_\lambda = p_\lambda W_\lambda^{-1} r_\lambda A \qquad\qquad \Phi_\lambda(y,c) := y - p_\lambda W_\lambda^{-1} r_\lambda (Ay - c) = y - \Pi_\lambda (y - x) \,,$$
$$M^s_{mult} = (I - \Pi_{\lambda_{max}}) \cdots (I - \Pi_{\lambda_1}) \,, \qquad M^s_{add} = I - \sum_{\lambda \in \Lambda} \Pi_\lambda \,.$$

Show the following, yet unproven, results from the lecture.

a) It holds that
$$\sigma(\Pi_\lambda) \subset \sigma(W_\lambda^{-1} A_\lambda) \cup \{0\}$$
with equality instead of $\subset$ if $\dim X_\lambda < \dim X$.

b) We have that
$$\rho(\Pi_\lambda) \leq \Delta \Leftrightarrow A_\lambda \leq \Delta W_\lambda \,.$$

c) M^s_{add} yields a symmetric linear iteration.

d) With W^s_{add} being the matrix of the third normal form to the iteration matrix M^s_{add} we have
$$A \leq \#\Lambda \Delta W^s_{add} \,.$$

e) For sufficiently small θ the dampened additive Schwarz method with approximate subdomain solvers W_λ^{-1} converges.

(5 Points)

Exercise 16. (Disjoint Schwarz iterations)

If a non-overlapping decomposition Λ of I is given *without* the assumptions of ordering of indices or shape of the prolongations as

$$(p_\lambda)_{k,l} = \delta_{k,l} \in \mathbb{R}^{N \times \#\lambda} , \qquad k \in I, l \in \lambda$$

we can still transform the system into an equivalent one that reproduces a block-Jacobi or block-Gauss-Seidel structure of the additive and multiplicative Schwarz iteration. That, is we can treat the Schwarz iteration in the general disjoint subdomain case as block-Gauss-Seidel and block-Jacobi iterations as well.

We view $x_\lambda \in X_\lambda$ as single block of a vector in X and gather these all together into a block vector $\hat{x} = (x_\lambda)_{\lambda \in \Lambda}$. Note that in the previously considered, ordered case we would have $x = \hat{x}$.

The map

$$T : X \to X \qquad \hat{x} = (x_\lambda)_{\lambda \in \Lambda} \mapsto \sum_{\lambda \in \Lambda} p_\lambda x_\lambda$$

describes a regular transformation $x = T\hat{x}$. This map reorders the space X such that the resulting partition has the right ordering to yield block vectors and matrices. Hence, define

$$\hat{p}_\lambda : X_\lambda \to X \qquad (\hat{p}_\lambda)_{k,l} := \delta_{k,l} \in \mathbb{R}^{N \times \#\lambda} , k \in I, l \in \lambda .$$

This yields

$$p_\lambda = T\hat{p}_\lambda , \qquad x = \sum_{\lambda \in \Lambda} p_\lambda x_\lambda = T\hat{x} .$$

Using T and $\hat{p}_\lambda$ we can transform the system $Ax = b$ into

$$\underbrace{T^T A T}_{=:\hat{A}} \hat{x} = \underbrace{T^T b}_{=:\hat{b}} .$$

Let $\hat{\Phi}$ be the iterative method defined via $\hat{p}_\lambda, \hat{A}$ and $\hat{r}_\lambda = \hat{p}_\lambda^T$.

Show that the following hold.

a) For the consistent single subdomain iteration Φ_λ defined via $M_\lambda = (I - P_\lambda)$ it holds that $\Phi_\lambda = T \circ \hat{\Phi}_\lambda \circ T^T$. Moreover, the iteration $\hat{\Phi}$ is linear and consistent itself.

b) $\hat{\Phi}_{mult} = \Phi_{GS}$, $\hat{\Phi}^\theta_{add} = \Phi^\theta_J$ with Φ_{GS} denoting the block-Gauss-Seidel and Φ^θ_J the dampened block Jacobi method.

c) $\Phi_{mult} = T \circ \Phi_{GS} \circ T^T$ and $\Phi^\theta_{add} = T \circ \Phi^\theta_J \circ T^T$.

(2 Points)

Exercise 17. (Disjoint additive Schwarz iteration)

For the following assume that the conditions of the previous exercise hold, that $A > 0$ and that

$$\gamma W_{add} \leq A \leq \Gamma W_{add} \, .$$

and that we are restricted to the case of two $k = \#\Lambda = 2$ disjoint domains without any assumptions on ordering of $p_i = p_{\lambda_i}$. For a simplified notation we use only $1, 2$ instead of λ_1, λ_2.

a) Show that

$$\hat{A} = \begin{pmatrix} A_1 & A_{1,2} \\ A_{1,2}^T & A_2 \end{pmatrix} \, , \qquad\qquad A_{1,2} = r_1^T A p_2 \, .$$

Moreover show that $\hat{A} > 0$ and $\hat{A}_{i,j} > 0$ for $i, j = 1, 2$.

b) It holds that

$$\delta := \| A_1^{-\frac{1}{2}} p_1^T A p_2 A_2^{-\frac{1}{2}} \|_2 < 1$$

is the best bound in the strengthened Cauchy-Schwarz-inequality

$$| < x, y >_A | \leq \delta \|x\|_A \|y\|_A$$

Hint: Insert $x = p_1 x_1, y = p_2 x_2$ and show and use $\|p_\lambda x_\lambda\|_A = \|A^{\frac{1}{2}} x_\lambda\|_2$.

c) Use c) to show that $\gamma W_{add} \leq A \leq \Gamma W_{add}$ is equal to

$$\gamma D \leq \hat{A} \leq \gamma D$$

with D being the block-diagonal of $\hat{A}$ formed by A_1, A_2.

d) Apply lemma

> For the spectrum of the (weakly) 2-cyclic matrix
>
> $$B = \begin{pmatrix} 0 & B_1 \\ B_2 & 0 \end{pmatrix}$$
>
> it holds that
>
> $$\sigma(B) = \pm\sqrt{\sigma(B_1 B_2)} \cup \pm\sqrt{\sigma(B_2 B_1)} \, , \quad \sigma(B_1 B_2) \setminus \{0\} = \sigma(B_2 B_1) \setminus \{0\} \, .$$

to $D^{-\frac{1}{2}}(\hat{A} - D)D^{-\frac{1}{2}}$ to conclude that

$$\lambda_{min}(B) = -\delta \, , \qquad \lambda_{max}(B) = \delta \, , \qquad \gamma = 1 - \delta \qquad \Gamma = 1 + \delta \, .$$

e) Follow from d) that

$$\theta = 1 \qquad\qquad \rho(M_{add}^1) = \|M_{add}^1\|_A = \delta$$

hold for the optimal damping and the resulting convergence rate.

(5 Points)

Programming exercise 6. (Efficient sparse prolongation and restriction)

In the previous programming exercise we have already decomposed the vertexes into 3 (non)-overlapping subdomains. However, for the application to our linear system, we need a decomposition of the indices that are actually active in the resulting linear system, i.e. with Dirichlet boundary conditions already introduced and respective vertices removed from vectors and matrices. Since we want to keep using the same decomposition for different boundary condition (and different Γ_D), we will focus on an approach of adapting the decomposition to the nodes to the Dirichlet boundary.
We will keep working with the mesh and domain decomposition introduced in the last programming exercise.

a) Write a function to remove indices i of nodes $x_i \in \Gamma_D$ on the Dirichlet boundary from $\lambda \in \Lambda$.

   Hint: Programming exercise 3a).

b) Write a function to construct the prolonged matrices $p_\lambda B_\lambda r_\lambda$ for some matrix $B_\lambda : X_\lambda \to X_\lambda$, in particular K_λ.

   Hint: It is easier to prolong to the whole space and then restrict to the free nodes.

Since all the matrices obtained from the Galerkin discretization are sparse, it is highly inefficient to construct our matrices using full matrices of `zeros` as a starting point or array slices that might trigger dense matrices in the background. `scipy.sparse` offers the sparse `coo_matrix` matrix format. It offers 3 constructors

- `A = coo_matrix(B)` with `B` being either a dense or sparse matrix creating a COO format copy of `B`
- `A = coo_matrix(N,M)` creating an empty COO format matrix of size $N \times M$.
- `A = coo_matrix((data, (i,j)), (N,M))` creating a sparse matrix from the 3 equally sized array `data,i,j` such that `A[i[k],j[k]] = data[k]`

This format is inefficient for computation but is easily usable for the construction of sparse matrices. Moreover, it can be efficiently converted into the `csc_matrix` or `csr_matrix` formats that are better suited for computations. Also, the underlying arrays are accessible via `A.data,A.row,A.col`

c) Change the construction of the global stiffness matrix to use `coo_matrix`. Be aware that the COO format allows multiple `data,i,j` entries for the same $a_{i,j}$. When converting to another format, usually CSR or CSC, duplicate entries are summed together, which facilitates construction of finite element matrices.

d) Write a function for an efficient computation of $K_\lambda = r_\lambda K p_\lambda$ from the stiffness matrix K.

e) Write a function for an efficient computation of $p_\lambda B_\lambda r_\lambda$ for any block matrix $B_\lambda \in \mathbb{R}^{\#\lambda \times \#\lambda}$.

f) Use d) to write a function for the computation of $\Pi_\lambda = p_\lambda W_\lambda^{-1} r_\lambda A$ for a given regular W_λ, in particular $W_\lambda = A_\lambda$.

g) Compare the times to construct Π_λ by using `coo_matrix` compared to array slices (that might trigger background construction of dense matrices) when using the L-shaped domain and PDE from the previous programming exercise to give K.

   Hint: `IPython %timeit` magic builtin or `timeit` module.

(4 Points)

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