Implementation

Example:

Consider the boundary-value problem[1]

$$-\nabla \cdot D\nabla u = f \quad \text{on } \Omega, \tag{1}$$
$$u = 0 \quad \text{on } \partial\Omega, \tag{2}$$

and the corresponding variational problem (weak formulation)

$$a(u,v) = (f,v) \quad \forall \quad v \in H^1(\Omega),$$
$$a(u,v) := \int_\Omega (\nabla v)^T D \nabla u dx,$$

where

$$D := \epsilon \cdot I + \mathbf{d} \cdot \mathbf{d}^T, \quad \mathbf{d} = (\cos\phi, \sin\phi)^T, \quad 0 < \epsilon \le 1, \quad \phi \in [0, \pi/2],$$

$\Omega = (0,1) \times (0,1)$, and $f = 1$.

A discretization of this problem on a (coarse) mesh $\mathcal{T}_{h_0}$ with N_0 nodes, consisting of triangular elements, using linear shape functions, results in a system $A^{(0)}\mathbf{u}^{(0)} = \mathbf{b}^{(0)}$ of linear equations, which we want to solve exactly, for instance using the "\" operator in MATLAB, i.e., `u_0=A_0\b_0`.

Instructions for the usage of the MATLAB PDE TOOLBOX:

- The graphical user interface (GUI) of the PDE TOOLBOX is activated with the command `pdetool`, which has to be entered on the command line of the Command Window of MATLAB. Using the Draw and Edit menu options of the GUI, we can easily draw the square domain $\Omega = (0,1) \times (0,1)$.
- Next, we select the type of the PDE application from the pop-up menu: The correct type for our example is Diffusion.
- Now, we specify the PDE using the menu option PDE $\rightarrow$ PDE Specification....
  In the field for the Diffusion coefficient we enter the entries of the matrix $D = (d_{ij})$ which can be computed via the following simple (user-defined) function:

```
function d = D(epsilon,phi)
d11=epsilon+cos(phi).*cos(phi);
d12=cos(phi).*sin(phi);
d21=cos(phi).*sin(phi);
d22=epsilon+sin(phi).*sin(phi);
d=[d11,d12,d21,d22];
```

[1]Equation (1) is sometimes referred to as an anisotropic (for small ϵ) rotated (depending on the angle ϕ) diffusion equation.

For instance, calling our function with the arguments $\epsilon = 0.01$ and $\phi = \pi/12$ yields

```
>> D(0.01,pi/12)
ans =
0.9430     0.2500     0.2500     0.0770
```

which is exactly the form to be entered in the field specifying the PDE coefficients. The Volume source is set to 1.0.

- The mesh is generated using the menu option Mesh.
  First we set the Mesh → Parameters... to
  - Maximum edge size: 0.75
  - Mesh growth rate: 1.5
  - do NOT jiggle mesh
  - Jiggle mode: optimize minimum
  - Refinement method: regular

  By clicking the Mesh → Initialize Mesh menu option a uniform mesh consisting of 8 triangles and 9 nodes is produced. Note that the boundary nodes are numbered first and the interior nodes (in this case there is only one) are numbered last, which can be seen from using the Mesh → Show Node Labels menu option.
  Using the menu option Mesh → Refine Mesh produces a regular refinement of the mesh such that the resulting triangulation consists of 32 triangles and 25 nodes. Note that the *newly added nodes* are numbered last, that is, just opposite to what we need in the construction of two- and multilevel preconditioners, as discussed in the Lecture.

- After this we are ready to solve the PDE using the menu option Solve → Solve PDE. Note that in general there is one more preprocessing step that is related to the specification of the boundary conditions. However, in our example we may skip this step because homogenous Dirichlet boundary conditions are imposed per default.

- For creating problems on a hierarchy of nested meshes we switch into the Mesh → Mesh Mode, then refine the mesh (using the menu option Mesh → Refine Mesh) and solve the problem (using Solve → Solve PDE) as often as required. Note that it is also possible to coarsen a certain mesh (which was obtained by regular refinement) by using the option Mesh → Undo Mesh Change in the Mesh Mode.

- After having solved the PDE (at a certain level k of refinement) we use
  - PDE → Export PDE Coefficients... to export the PDE coefficients `c a f d` to the workspace of MATLAB.
  - Mesh → Export Mesh... to export the mesh data `p e t`.
  - Boundary → Export Decomposed Geometry, Boundary Cond's... to save the variables `g b` in the workspace.

  Then using the command

```
>> [A_k,b_k]=assempde(b,p,e,t,c,a,f)
```

in the MATLAB Command Window assembles the PDE problem (by approximating the Dirichlet boundary condition with stiff springs, see "The Elliptic System" in the PDE TOOLBOX Help menu for details). Here `A_k` and `b_k` are the stiffness matrix and right-hand side, respectively. The solution to the FEM formulation of the PDE problem is `u_k=A_k\b_k`.

Remark 1: In order to transform `A_k` into 2×2 block form

$$A^{(k)} = \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A_{22}^{(k)} \end{bmatrix} \begin{matrix} \}\mathcal{N}_{h_k \setminus h_{k-1}} \\ \}\mathcal{N}_{h_k} \end{matrix}$$

where the upper-left block $A_{11}^{(k)}$ corresponds to the newly added nodes at level k, we have to reverse the numbering (ordering) of nodes (unknowns) in `A_k` and consequently in `b_k`. However, this permutation of rows and columns of the level-k stiffness matrix–in order to reverse the ordering of unknowns–can also be applied at level ℓ of the finest mesh, i.e.,

$$A^{(\ell)} \leftarrow P^T A^{(\ell)} P \quad \text{where } P = (p_{ij})_{i,j=1}^N \text{ and } p_{ij} = \begin{cases} 1 & \text{if } j = N+1-i \\ 0 & \text{else} \end{cases}$$

Then the newly added nodes of at level ℓ will have the smallest numbers, followed by the newly added nodes at level $\ell - 1$, ..., and, the nodes of the coarsest mesh will be numbered last.

Remark 2: Note that `A_k` is stored in a sparse matrix format, where for every nonzero matrix entry two integers (the row and the column index) and one double (the value of the matrix entry) are stored. All matrices, including the permutation matrix P mentioned in the previous Remark, as well as the hierarchical basis transformation $J^{(k)}$ at level k, should be stored in this sparse matrix format.

Remark 3: For the construction of two- and multilevel methods we need the (two-level) hierarchical basis representation

$$\widetilde{A}^{(k)} = \begin{bmatrix} \widetilde{A}_{11}^{(k)} & \widetilde{A}_{12}^{(k)} \\ \widetilde{A}_{21}^{(k)} & \widetilde{A}_{22}^{(k)} \end{bmatrix} = \begin{bmatrix} A_{11}^{(k)} & \widetilde{A}_{12}^{(k)} \\ \widetilde{A}_{21}^{(k)} & A^{(k-1)} \end{bmatrix}$$

of the stiffness matrix a level k. As we know from the Lecture, it is possible to compute $\widetilde{A}^{(k)}$ from the relation $\widetilde{A}^{(k)} = (J^{(k)})^T A^{(k)} J^{(k)}$ where the transformation matrix $J^{(k)}$ at level k relates the nodal point vectors for the hierarchical and the standard basis in the following way:

$$\mathbf{v} := \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} = J^{(k)} \begin{bmatrix} \tilde{\mathbf{v}}_1 \\ \tilde{\mathbf{v}}_2 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{v}}_1 + J_{12}^{(k)} \tilde{\mathbf{v}}_2 \\ \tilde{\mathbf{v}}_2 \end{bmatrix}, \text{ i.e., } \quad J^{(k)} := \begin{bmatrix} I_1 & J_{12}^{(k)} \\ 0 & I_2 \end{bmatrix}.$$

Moreover, in case of linear shape functions and triangular elements the (sub)matrix $J_{12}^{(k)}$ has a very simple form; It has exactly two nonzero entries in each row. All these nonzeros are equal to $1/2$ and they occur in those positions that correspond to connections between fine nodes (newly added nodes) and coarse nodes (associated with $A^{(k-1)}$) of $\mathcal{T}_{h_k}$.[2]

Exploiting the sparsity, the triple matrix product $(J^{(k)})^T A^{(k)} J^{(k)}$ (the evaluation in MATLAB is as simple as `J_k'*A_k*J_k`) requires only $\mathcal{O}(N_k)$ arithmetic operations.

[2] If $A_{12}^{(k)}$ has two nonzero entries per row then the nonzero patterns of $A_{12}^{(k)}$ and $J_{12}^{(k)}$ agree.

Exercise 21

Generate a linear system

$$\widetilde{A}^{(k)}\widetilde{\mathbf{u}}^{(k)} = \widetilde{\mathbf{b}}^{(k)}$$

arising from discretization of the abovementioned problem (see the Example at the beginning) where the matrix $\widetilde{A}^{(k)}$ and the right-hand side $\widetilde{\mathbf{b}}^{(k)}$ are given with respect to the two-level hierarchical basis. Choose the level k of refinement properly (such that the total number of degrees of freedom is in the range 10^2 to 10^4).

(a) Study the condition number of the additive (cf. Equation (3.12) in the Lecture Notes) and of the multiplicative (cf. Equation (3.15)) two-level preconditioner assuming that $A_{11}^{(k)}$ is inverted exactly, i.e., $C_{11}^{(k)} = A_{11}^{(k)}$. How do the theoretical bounds compare to the observed results (for both preconditioners)?

(b) Use an incomplete factorization in order to approximate $A_{11}^{(k)}$, i.e., $A_{11}^{(k)} \approx C_{11}^{(k)} = L^{(k)}U^{(k)}$. How does this affect the condition number in case of both two-level preconditioners.

(c) Compute the condition number for different values of ϵ and ϕ in (1). How does the condition number (of both preconditioners) depend on these parameters.

Exercise 22

Use the preconditioned conjugate gradient (PCG) method, cf. Algorithm (1.4.3), with the linear algebraic multilevel iteration (AMLI), cf. Algorithm (3.5.1), as a preconditioner to solve the linear system

$$\widetilde{A}^{(\ell)}\widetilde{\mathbf{u}}^{(\ell)} = \widetilde{\mathbf{b}}^{(\ell)}$$

associated with the fine-grid problem on some mesh $\mathcal{T}_{h_\ell}$, which is obtained from a coarse mesh $\mathcal{T}_{h_0}$ by several ($\ell = 2, 3, \ldots$) refinement steps. Assume that the coarse(st)-grid problem (at level 0) is solved exactly and an ILU preconditioner is used to approximate the solutions of the linear systems with $A_{11}^{(k)}$, cf. Exercise 21 (b).

(a) Implement the linear AMLI V-cycle and study its convergence behavior. Choose a proper stopping criteria. Does the number of iterations depend on the number ℓ of levels?

(b) Implement the linear AMLI W-cycle and study its convergence behavior. Find a proper second-degree polynomial for stabilization of the condition number. How does the W-cycle compare to the V-cycle?

Exercise 23

Use the nonlinear AMLI method, cf. Algorithm 3.6.1, to solve the same linear system(s) as in Exercise 22. Again the system at level 0 should be solved exactly, and the systems with $A_{11}^{(k)}$ should be solved approximately by using an ILU factorization for $k = \ell, \ell - 1, \ldots, 1$.

(a) How does the W-cycle variant (using two inner GCG iterations at every intermediate level) of this parameter-free algorithm compare to the W-cycle of the linear AMLI method (combined with PCG)?

(b) Is the method robust with respect to the parameters ϵ and ϕ?