Institute for Analysis and Scientific Computing
TU Wien
Philip L. Lederer

Numerical methods for partial differential equations

Exercise 8 – 12. May 2020

Example 8.1

Consider the finite element problems:

- Find $u \in V_h$, s.t.
$$\int_\Omega uv \,\mathrm{d}x = \int fv \,\mathrm{d}x \quad \forall\ v \in V_h$$
- Find $u \in V_h$, s.t.
$$\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x = \int fv \,\mathrm{d}x \quad \forall\ v \in V_h$$

where V_h is a continuous finite element space (with hom. Dirichlet bnd-cond.) on the unit square $\Omega = (0,1)^2$. Set up and solve such a problem with NGSolve. Afterwards, add the lines

```
c = Preconditioner(a,"local",test = True)
c.Update()
```

which sets up a Jacobi-preconditioner C (diagonal of the stiffness matrix) and evaluates (numerically) $\kappa_2(C^{-1}A)$ (where A is the stiffness matrix).
Make several mesh refinements for both problems and investigate how the condition number of the preconditioned problem behaves under mesh refinement. Does the behavior match the findings of the lecture?

Example 8.2

1. Implement the conjugate gradient method from the lecture (page 98). Use the numerically more stable versions for α_k and β_k. The input parameters should be
   - The matrix of the bilinearform
   - The matrix of the preconditioner
   - The vector of the solution gridfunction and the right hand side (linear form)
   - The maximum number of iterations
   - The tolerance (error calculated as in the lecture)
   - Optional parameter that enables to print the iteration number and the error in each step

   The function call could look something like

```
SolveCG(mat = a.mat, pre = c.mat, sol = gfu.vec, rhs = f.vec, \
        maxsteps = 200, tol = 1e-8, printrate = True)
```

2. Test your solver with the Jacobi-preconditioner (previous example) and with (the exact inverse) `c=Preconditioner(a, "direct")` for the Poisson problem with homogeneous Dirichlet boundary conditions.

Example 8.3 (AFW-preconditioner)

Let $\Omega \subset \mathbb{R}^3$ be a bounded domain. We want to solve the (Maxwells equation, magnetostatic) problem (with appropriate boundary conditions):

$$\int_\Omega \nu \operatorname{curl} u \cdot \operatorname{curl} v \, \mathrm{d}x + \int_\Omega \nu\varepsilon \; u \cdot v \, \mathrm{d}x = \int_\Omega f \cdot v \, \mathrm{d}x \quad \forall v$$

with the Sobolev space $H(\mathrm{curl}) := \{u \in L^2(\Omega, \mathbb{R}^3) : \operatorname{curl} u \in L^2(\Omega, \mathbb{R}^3)\}$. Where $\operatorname{curl} u$ is the weak curl of u (compare with the definition of $H(\mathrm{div})$). This space is discretized with basis functions that are associated to edges. The corresponding Nédélec Finite element (on the reference tetrahedron/triangle)) $(\hat{T}, V_{\hat{T}}, \Psi_{\hat{T}})$ is defined by $V_T = P^1(\hat{T})$ and $\Psi_{\hat{T}} = \{\psi_{i0}^{\hat{T}}, \psi_{i1}^{\hat{T}}\}_{i=1}^{N_E}$ where

$$\psi_{i0}^{\hat{T}}(v) := \int_{\hat{E}_i} v \cdot \hat{t} \, \mathrm{d}s \quad \text{and} \quad \psi_{i1}^{\hat{T}}(v) := \int_{\hat{E}_i} sv \cdot \hat{t} \, \mathrm{d}s$$

where $\hat{t}$ is the tangential vector on E_i and N_E is the number of edges of $\hat{T}$. We want to solve the problem that is predefined in `magnet.py` with a conjugate gradient method (with relative error tolerance 10^{-8}):

1. Try to solve the example with the Jacobi preconditioner. Present a plot (choose appropriate logarithmic scales) to present the relative error over the number of iterations for the cases $\varepsilon = 10^{-i}$ with $i = 0, \dots, 6$. The list `data` already stores the error and iteration numbers.

2. We want to apply the AFW (Arnold-Falk-Winther) preconditioner. This is a (overlapping) Block-Jacobi preconditioner where every block contains the degrees of freedoms that lie on the edges that are connected to this vertex. A sketch is given in Figure 1. Implement the AFW preconditioner in NGSolve.

   - `a.mat.CreateBlockSmoother(blocks)` creates a block preconditioner of the bilinear form `a` where `blocks` is a list of lists including the dofs of each block.
   - `mesh.vertices` gives you the vertices (that's an iterate-object).
   - `mesh[v].edges` gives you the edges connected to the vertex `v`.
   - `fes.GetDofNrs(...)` gives you the dofs of ... of the finite element space `fes`.

   Present a similar plot as before. Is the AFW-preconditioner robust with respect to the mesh size h?

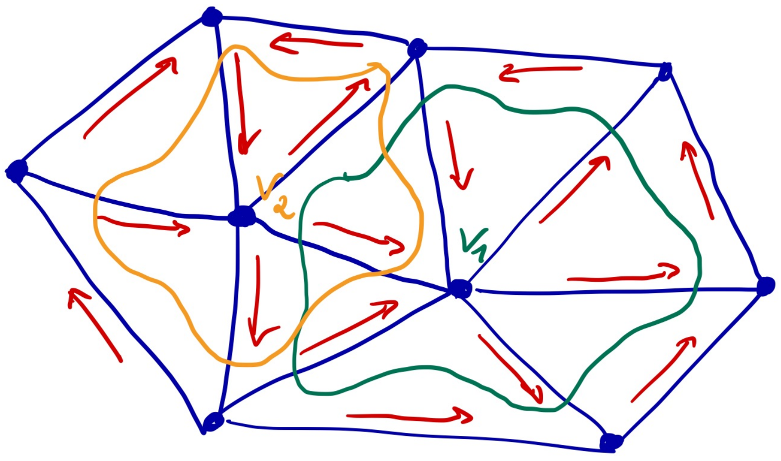

Figure 1: Blocks for AFW preconditioner

Example 8.4 (Multilevel preconditioner)

Solve the Poisson problem with $f = 1$ and homogeneous Dirichlet boundary conditions on the unit cube $\Omega = (0,1)^3$. Define the bilinear form `a`, the linear form `f` and a grid function `gfu` on the initial mesh (choose maxh = 0.5). To solve the system, we want to apply a multilevel preconditioner with `L` levels. To this end add the line `c = Preconditioner(a, "multigrid")` BEFORE calling any `a.Assemble()`. Then, implement the following (pseudo) loop:

```
for l in range(L):
     if not initial mesh: mesh.Refine()

     fes.Update()
     gfu.Update()

     a.Assemble()
     f.Assemble()
     gfu = solve with CG and c.mat as preconditioner and f.vec as rhs
     save number of iterations
```

Present a plot of number of the number of iterations over the levels and describe what the this plot says about the condition number κ of the multilevel preconditioner. Try to solve this problem with $L = 0, \ldots, 6$ (or even higher, check your memory...).