Exercise 14 (Solution)

Date issued: 27th January 2025

Homework Problem 14.1. (Numerical Optimization of the Floor Heating Problem)

We will treat the following specialized version of the floor heating problem

$$\begin{aligned} &\text{Minimize} \quad \frac{1}{2}\|y - y_d\|^2_{L^2([0,1]\times[0,1])} + \frac{\gamma}{2}\|u\|^2_{L^2([0,1]\times[0,1])} \\ &\text{s. t.} \quad \begin{cases} -\operatorname{div}(\nabla y) = u & \text{in } [0,1]\times[0,1] \\ \dfrac{\partial}{\partial n} y = -y & \text{on } \partial([0,1]\times[0,1]) \end{cases} \end{aligned} \tag{P}$$

numerically.

(a) Identify and state, which parameters of the floor heating problem (7.3) have been specified to which value to obtain (P), and address which quantities remain to be fixed, before a concrete solution can be computed.

(b) Install Fenics (not FenicsX). Download the file solve_KKT_sparse.py. The file contains a python solution for solving an FEM approximation of (P) for multiple regularization parameters γ in two problem settings where the desired states y_d are computed by solving the PDE for provided u_d. The results are displayed in subplots of figures. When you run the file, you should see the desired states plotted but all other subplots empty because the code segment that solves the optimization problems has been removed. Read and fix the code (lines 77 and 80) so the optimization problems are solved and give a quick analysis of what you can observe from the solutions for various γ's.

Solution.

(a) We have:

- $\Omega = \Omega_{\text{obs}} = \Omega_{\text{ctrl}} = [0,1] \times [0,1]$
- $\kappa \equiv 1$
- $\alpha \equiv 1$
- $y_\infty \equiv 0$

and we need to specify the remaining free parameters y_d and γ to obtain a solvable problem. Additionally, the bound constraints have been removed.

(b) You can download the solution file solve_KKT_sparse-solution.py.

The section of code from the lines 77 to 80 of the initial file looks as follows:

```
# Set right hand side of the optimality system
zeros = sps.csr_matrix((1,nDoF))
RHS = [prob.M.dot(prob.yd),
       zeros,
       zeros]

# Set system matrix of the optimality system
zeros = sps.csr_matrix((nDoF,nDoF))
MAT = [[prob.M, zeros        ,  sps.csr_matrix.transpose(prob.S)],
       [zeros , gamma*prob.M , -sps.csr_matrix.transpose(prob.M)],
       [prob.S, -prob.M      ,  zeros]]
```

Figure 0.1: Code section for setting up the linear optimality system.

Note that the system matrix MAT is symmetric as a result of sorting the variables in the sequence (y, u, p) and the equations in the sequence adjoint equation, gradient equation, state equation and of the choice of adjoint equation - where we could have compute $-p$ by changing the right hand side to $y - y_d$ instead of $-(y - y_d)$ and compensated for it by entering p into the gradient equatio nsign inverted, which would make the system asymmetric. The symmetry makes the system easier to solve numerically, so it is a desirable property.

The results are as shown in Figure 0.2. You can nicely see the effect of the regularization parameter γ, where large values of γ lead so optimal states closer to 0 while small γ lead to optimal states approximating the desired state more closely. Note that you can see that the magnitude of a γ, where the majority of the behavior of the desired state is being picked up by the optimal state is dependent on the problem configuration, making it hard to predict reasonable γ values in advance.

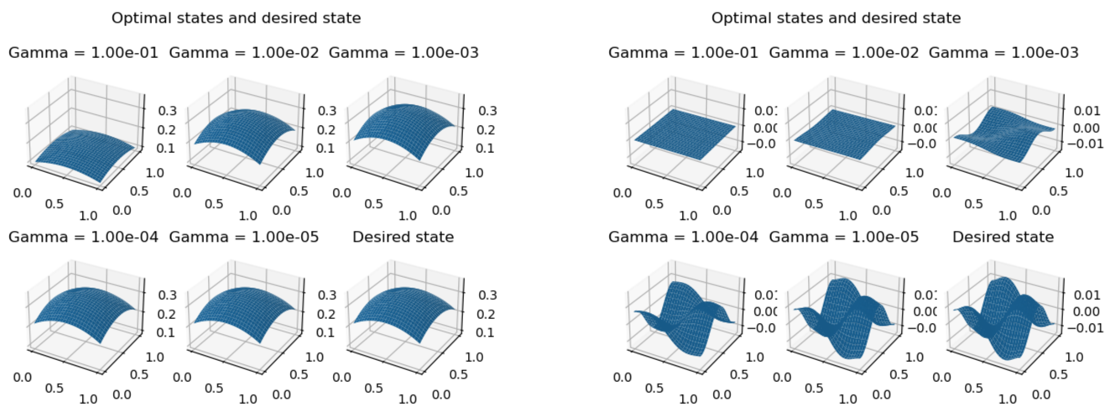

Figure 0.2: Solution visualizations.

You are not expected to turn in your solutions.