Numerical Simulation

Summer Semester 2015
Lecturer: Prof. Dr. André Uschmajew
Assistent: Bastian Bohn

universitätbonn

Excercise sheet 9.

Closing date **23.06.2015**.

Theoretical exercise 1. (Distributed source control [8 points])

In an analogous fashion to Corollary 3.3, Lemma 3.4 and Theorem 3.5, derive the operator S^*, the adjoint state p and the optimality conditions in variational form for the problem of distributed source control on $Q = (0,1) \times (0,T)$:

$$\min J(y,u) := \frac{1}{2}\int_0^T \int_0^1 (y(x,t) - y_Q(x,t))^2 \mathrm{d}x\mathrm{d}t + \frac{\lambda}{2}\int_0^T \int_0^1 u(x,t)^2 \mathrm{d}x\mathrm{d}t$$

subject to

$$\begin{aligned} y_t(x,t) &= y_{xx}(x,t) + u(x,t) && \text{in } Q \\ y_x(0,t) &= 0 && \text{in } (0,T) \\ y_x(1,t) + \alpha y(1,t) &= 0 && \text{in } (0,T) \\ y(x,0) &= 0 && \text{in } (0,1) \end{aligned}$$

with $u_a \le u \le u_b$ almost everywhere and $u, y_Q, u_a, u_b \in L_2(Q)$ and $\alpha, \lambda \ge 0$.

Theoretical exercise 2. (Uniqueness of optimal bang-bang control [4 points])

Prove the uniqueness statement in Theorem 3.6, i.e. show that the optimal Bang-Bang control is unique.

Programming exercise 1. (Elliptic optimal control with Dirichlet boundary conditions)

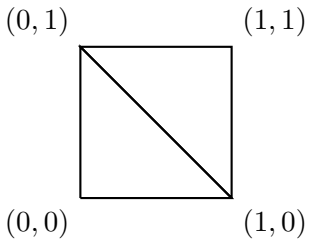
The programming exercises have to be done in C/C++. Please mail your code to *bohn@ins.uni-bonn.de* by 30th of June.
Write a C/C++ code to solve the optimal control problem

$$\min_{y,u} \|y - y_\Omega\|^2_{L_2([0,1]^2)} + \frac{\lambda}{2}\|u\|^2_{L_2([0,1]^2)}$$

subject to

$$\begin{aligned} -\Delta y &= u && \text{in } [0,1]^2 \\ y &= 0 && \text{on } \partial[0,1]^2 \\ -1 \le\ u &\le 1 && \text{almost everywhere on } [0,1]^2 \end{aligned}$$

by the projected gradient method using a Finite Element PDE solver. To discretize $y \in H^1([0,1]^2)$ and the adjoint state $p \in H^1([0,1]^2)$ use piecewise linear Finite Elements. $u \in L_2([0,1]^2)$ should be discretized by piecewise constant Finite Elements. The FEM-mesh is created by 5-times uniform (red-)refinement of



PDE-solver The code framework provided in the zip-archive on the website is the FEM-code from the lecture Scientific Computing I (WS 13/14). The code is compiled (on Linux-operating systems) by running `make` in your console. However, if you prefer to use any other or your own PDE solver, feel free to do so. The code you should implement yourself is just the projected gradient method. The PDE solver can be treated as a black-box-algorithm. The results should then be written to two VTK-files which can be visualized by e.g. paraview (this is already done in the reference code): One file for the controls and one for the state.

In the reference code the reading and refinement of the geometry and the PDE solver itself already exist and do not need to be changed. The only thing that needs to be done in order to solve the state and adjoint equations is setting up the correct right hand sides for the poisson solver. To do this you can use the function `resetRHSByVec` which resets the right hand side elementwise. For the adjoint equation you need to compute the L_2-projection of the piecewise linear $y - y_\Omega$ onto the space of piecewise constant functions on the elements. To this end, you can just use a simple midpoint rule. Here the function `Mesh::getNodesToElement` will be of good use to get the indices of the nodes which belong to a certain element.

After the right hand side has been set, the actual PDE solving is done by consecutive calls to

- `generateGlobalStiffnessMatrixAndLoadVector`,
- `incorporateBoundaryConditions`,
- `pcCG`.

See for example the corresponding calls in `solvePDEFromFile` as a reference.

To implement the projected gradient method for the control problem, you only need to edit the file `PDE_exercise.cc` where it states "YOUR CODE HERE". The coarse structure of the function `solveControlProblem` is already given. The parameters `maxCGIt` and `eps` are only relevant for the CG-algorithm of the PDE-solver.

When you send in your results, only send the files you actually changed (in the best case, this would just be `PDE_exercise.cc`).

Example-Problem Test your code with $\lambda = 0.01$ and $\lambda = 0.001$ for $y_\Omega(a, b) = a(1 - a)b^2(1 - b)$. The stepsize for the projected gradient method can be chosen as 100. The algorithm should terminate when the ℓ_2 norm of the coefficients of $u^{(k-1)} - u^{(k)}$ is smaller than 10^{-10}.