Spring 2016: Advanced Topics in Numerical Analysis:
Computational and Variational Methods for Inverse Problems
Assignment 5 (due May 12, 2016)

1. **An inverse problem for Burger's equation.** We consider inversion for a coefficient in the viscous Burger's equation.[1] Using the spatial interval $[0,1]$ and the temporal interval $[0,T]$, the solution $u = u(t,x) : [0,T] \times [0,1]$ satisfies

$$u_t + auu_x - \nu u_{xx} = f \quad \text{on } (0,T)\times(0,1), \tag{1a}$$
$$u(t,0) = u(t,1) = 0 \quad \text{for all } t \in [0,T], \tag{1b}$$
$$u(0,x) = 0 \quad \text{for all } x \in [0,1]. \tag{1c}$$

Here, $a = a(x) : [0,1] \to \mathbb{R}$ is the coefficient we invert for, the constant $\nu > 0$ controls the strength of the viscous term, and $f = f(t,x)$ is a given forcing. The conditions (1b) and (1c) are the boundary and the initial conditions, respectively.[2] We assume we are given observations $u^{\text{obs}} = u^{\text{obs}}(t,x)$ for $t \in [T_1, T]$, $(0 < T_1 < T)$. To invert for the coefficient a, we use the functional

$$J(a) := \frac{1}{2}\int_{T_1}^{T}\int_0^1 (u - u^{\text{obs}})^2\, dxdt + \frac{\beta}{2}\int_0^1 a_x^2\, dx \tag{2}$$

with a regularization parameter $\beta > 0$. To use an optimization method for (2), we require derivatives of J with respect to a.

   (a) Derive a weak form of (1) by multiplying (1a) with a test function $v : [0,T] \times [0,1]$, and integrating over space and time. There is no need to impose the initial condition explicitly via a Lagrange multiplier (as we did in class for the initial condition inverse problem), since the inversion parameter does not appear in the initial condition. Instead, you can build the initial condition into the definition of the solution space as we have done with Dirichlet boundary conditions. Use integration-by-parts on the viscous term to derive the weak form (note that since ν is a constant, $\nu u_{xx} = (\nu u_x)_x$).

   (b) Using the Lagrangian functional, derive expressions for the adjoint equation and for the gradient of J with respect to a. Give weak and strong forms of these equations. Note that a as well as its variations $\hat{a}$ are functions of space only, while u and the adjoint v are functions of space and time.

2. **Inverse elliptic parameter estimation, continued from Assignment 4.** Here, we continue the solution of the advection-diffusion equation we started in Assignment 4. There, we employed a steepest descent method to minimize the cost functional. Here, we will extend a state-of-the-art Newton-CG method, for which the source code is available from `http://math.nyu.edu/~stadler/inv16/assignment5/`. We wish to solve the inverse problem for the advection-diffusion equation on $\Omega = [0,1] \times [0,1]$:

$$\min_m J(m) := \frac{1}{2}\int_\Omega (u - u^{obs})^2\, dx + \frac{\gamma}{2}\int_\Omega \nabla m \cdot \nabla m\, dx, \tag{3}$$

[1] This equation is often considered as one-dimensional surrogate for the Navier-Stokes equations, probably the most important equation in fluid dynamics.

[2] Note that the boundary $\partial\Omega$ of the one-dimensional interval $\Omega = (0,1)$ are simply the points $x = 0$ and $x = 1$, i.e., $\partial\Omega = \{0,1\}$.

where $u(\boldsymbol{x})$ depends on the log diffusivity $m(\boldsymbol{x})$ through

$$\begin{aligned} -\nabla \cdot (\exp(m)\nabla u) + \boldsymbol{v} \cdot \nabla u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega, \end{aligned} \tag{4}$$

with the advective velocity $\boldsymbol{v} = (v_1, v_2)$, regularization parameter $\gamma > 0$ and measurement data $u^{obs}(\boldsymbol{x})$. The latter are synthesized by solving the state equation with $m(x, y) = 2$ for $(x-0.5)^2 + (y-0.5)^2 \leq 0.04$ and $m(x, y) = 8$ else (and adding noise). Note that compared to the problem we considered in class, here we have redefined $m(\boldsymbol{x})$ to be the natural log of the diffusivity field; this enforces positivity of the diffusivity field.

(a) Derive weak and strong forms for the gradient and the Hessian action for this problem.

(b) Extend the implementation `Poisson-InexactNewton.py` of the Gauss-Newton method by introducing the advection velocity $\boldsymbol{v} = (30, 0)$. Report the number of Gauss-Newton and of overall CG iterations for discretizations of the domain with 10×10, 20×20, 40×40 and 80×80 linear finite elements and give the number of unknowns used to discretize the log diffusivity function m (which is called a in the implementation) for each of these meshes. Discuss how the number of iterations changes as more parameters are used. In these experiments, the noise level should be fixed to the default value (0.01) while the mesh is refined. The "optimal" regularization parameter can be found manually (i.e., by experimenting with a few different values and finding the one that results in the "best" reconstruction, or else by the discrepancy principle, if you are so inclined.

(c) As we have discussed before, to avoid over-solving of the CG system in early Newton steps, where we are still far away from the solution and thus cannot benefit from the fast local convergence properties of Newton's method, the implementation uses the stopping criterion

$$\frac{\|Hd_k + g_k\|}{\|g_k\|} \leq \eta_k.$$

Here, H denotes the Hessian, g_k the gradient in the k-th iteration and d_k the (inexact) Newton direction. Compare the behavior of the choices[3]

- $\eta_k = 0.5$,
- $\eta_k = \min(0.5, \sqrt{\|g_k\|/\|g_0\|})$,
- $\eta_k = \min(0.5, \|g_k\|/\|g_0\|)$,

where g_0 denotes the norm of the initial gradient (see also the 2nd assignment) in terms of Newton iterations and overall CG iterations. For each run, try to estimate the number of overall PDE solves. Repeat the same experiment using the Gauss-Newton approximation of the Hessian rather than the Hessian.[4]

(d) *Optional:* The ill-posedness of inverse problems is closely related to the spectrum (i.e., the eigenvalues) of the Hessian operator.[5] Compute the eigenvalues of the reduced Hessian at the solution of the problem for a mesh with 20×20 elements. Since the Hessian is not explicitly available and can only be applied to vectors, there are 2 possibilities to access its eigenvalues:

[3] Search for `tolcg` in the implementation to find where the stopping criterion is defined.

[4] The flag `use_GaussNewton` allows to switch between the Hessian and its Gauss-Newton approximation. Recall that in the Gauss-Newton Hessian, terms that involve the adjoint variable are neglected and thus the corresponding Hessian approximation is guaranteed to be positive (semi-)definite.

[5] The eigenvalues often decay fast such that the inversion of the operator can only be done in a stable manner using regularization. Recall the experiments we did for the convolution problem at the beginning of the course.

- Build the Hessian matrix explicitly by applying it to unit vectors and compute the eigenvalues of the resulting matrix.
- **(preferred method):** Luckily, there are iterative methods[6] to compute eigenvalues of a matrix, which only requires the application of the matrix to vectors. Compute the largest 100 eigenvalues using an iterative method[7].

Plot the spectrum of the reduced Hessian with and without regularization and discuss the result.

(e) *Optional:* Try to replace the Tikhonov regularization by total variation regularization and report the results for different meshes.

[6] For instance, the Lanczos method which is closely related to the conjugate gradient method to solve linear systems. Such an iterative method is implemented in Matlab's and Python's `eigs` function.

[7] Note that the descent behavior of the eigenvalues is also the reason why only very few steps of the conjugate gradient method are needed in each Newton step. The CG method can be shown to preliminary "work" on the large eigenvalues (for a more detailed discussion see for instance J. Shewchuk: *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*). Since the eigenvalues decay rapidly for many of the operators occurring in inverse problems, the solution to the Hessian system is already well approximated after a few CG steps.