Fall 2021:
Computational and Variational Methods for Inverse Problems
CSE 397/GEO 391/ME 397/ORI 397
Assignment 3 (due 8 November 2021)

The problems below require a mix of paper-and-pencil work and FEniCS implementation. Jupyter notebooks that provide example implementations in FEniCS of minimizing functionals—or equivalently solving (weak form of) PDEs—have been posted on Canvas under *Class 15*. Feel free to build on these in solving Problems 3 and 4 below. Please turn in printouts of your FEniCS implementations together with the results and discussion (i.e., no need to submit actual Jupyter notebooks).

1. The problem of removing noise from an image without blurring sharp edges can be formulated as an infinite-dimensional minimization problem. Given a possibly noisy image $d(x, y)$ defined within a square domain Ω, we would like to find the image $u(x, y)$ that is closest in the L^2 sense, i.e. we want to minimize the "data misfit"

$$\mathcal{F}_{LS}(u) := \frac{1}{2}\int_\Omega (u-d)^2 \, d\boldsymbol{x},$$

while also removing noise.[1] This noise is assumed to comprise very "rough" (i.e. highly oscillatory) components of the image. Denoising can be addressed through an additional term in the objective, in the form of a regularization that penalizes steeper gradients, i.e.,

$$\mathcal{R}_{TN}(u) := \frac{\alpha}{2}\int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} u \, d\boldsymbol{x},$$

where $\alpha > 0$ is the regularization parameter that controls how strongly we impose the regularization, i.e. how much smoothing occurs. Unfortunately, if there are sharp edges in the image, this so-called *Tikhonov (TN) regularization* will blur them. Instead, in these cases we prefer the so-called *total variation (TV) regularization*,

$$\mathcal{R}_{TV}(u) := \alpha\int_\Omega (\boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} u)^{\frac{1}{2}} \, d\boldsymbol{x}$$

where taking the square root is the key to preserving edges, as we will see in this assignment. Since $\mathcal{R}_{TV}$ is not differentiable when $\boldsymbol{\nabla} u = \boldsymbol{0}$, it is usually modified to include a small parameter $\delta > 0$ as follows:

$$\mathcal{R}^\delta_{TV}(u) := \alpha\int_\Omega (\boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} u + \delta)^{\frac{1}{2}} \, d\boldsymbol{x}.$$

Here and in Problem 3, we wish to study the performance of the two denoising functionals $\mathcal{F}_{TN}(u)$ and $\mathcal{F}^\delta_{TV}(u)$ by solving the minimization problems

$$\min_{u\in\mathcal{U}} \mathcal{F}_{TN}(u) := \mathcal{F}_{LS}(u) + \mathcal{R}_{TN}(u),$$

and

$$\min_{u\in\mathcal{U}} \mathcal{F}^\delta_{TV}(u) := \mathcal{F}_{LS}(u) + \mathcal{R}^\delta_{TV}(u),$$

where $\mathcal{U}$ is the space of admissible functions for each problem. We prescribe the homogeneous Neumann boundary condition $\boldsymbol{\nabla} u \cdot \boldsymbol{n} = 0$ on the four sides of the square domain, where $\boldsymbol{n}$ is the outward normal. This amounts to assuming that the image intensity does not change normal to the boundary of the image.

[1] We are focusing on greyscale images, since $u(x, y)$ and $d(x, y)$ are scalar functions.

(a) For both $\mathcal{F}_{TN}$ and $\mathcal{F}_{TV}^{\delta}$, derive the first-order necessary condition for optimality using calculus of variations, in both weak form and strong form. Use $\hat{u}$ to represent the variation of u.

(b) Show that when $\boldsymbol{\nabla} u$ is zero, $\mathcal{R}_{TV}$ is not differentiable, but $\mathcal{R}_{TV}^{\delta}$ is.

(c) For $\mathcal{F}_{TV}^{\delta}$, derive the infinite-dimensional Newton step, in both weak and strong form. For consistency of notation, use $\tilde{u}$ as the differential of u (i.e. the Newton step). The strong form of the second variation of $\mathcal{F}_{TV}^{\delta}$ will give an anisotropic diffusion operator of the form $-\text{div}(\boldsymbol{A}(u)\boldsymbol{\nabla}\tilde{u})$, where $\boldsymbol{A}(u)$ is a 2×2 anisotropic tensor that plays the role of the diffusion coefficient.[2] (In contrast, you can think of the second variation of $\mathcal{F}_{TN}$ giving an *isotropic* diffusion operator, i.e. with $\boldsymbol{A} = \alpha \boldsymbol{I}$.

(d) Derive expressions for the two eigenvalues and corresponding eigenvectors of $\boldsymbol{A}$.[3] Based on these expressions, give an explanation of why $\mathcal{F}_{TV}^{\delta}$ is effective at preserving sharp edges in the image, while $\mathcal{F}_{TN}$ is not. This argument can be made by considering just a single Newton step.

(e) Show that for large enough δ, $\mathcal{R}_{TV}^{\delta}$ behaves like $\mathcal{R}_{TN}$, and for $\delta = 0$, the Hessian of $\mathcal{R}_{TV}^{\delta}$ is singular. This suggests that δ should be chosen small enough that edge preservation is not lost, but not too small that ill-conditioning occurs.

(f) *Optional:* Show the equivalence between the following two approaches:

- *Discretize-then-optimize* (also known as the Ritz method in this context): First make a finite element approximation of the infinite-dimensional functional $\mathcal{F}_{TV}^{\delta}$, and then derive the finite-dimensional Newton step.
- *Optimize-then-discretize* (also known as the Galerkin method): Directly make a finite element approximation of the (weak form of the) infinite-dimensional Newton step you derived above.

2. Install FEniCS on your computer.

3. An anisotropic Poisson problem in a two-dimensional domain Ω is given by the strong form

$$-\boldsymbol{\nabla}\cdot(\boldsymbol{A}\boldsymbol{\nabla}u) = f \qquad \text{in } \Omega, \tag{1a}$$

$$u = u_0 \qquad \text{on } \Gamma, \tag{1b}$$

where the conductivity tensor $\boldsymbol{A}(\boldsymbol{x}) \in \mathbb{R}^{2\times 2}$ is assumed to be symmetric and positive definite for all $\boldsymbol{x}$, $f(\boldsymbol{x})$ is a given distributed source, and $u_0(\boldsymbol{x})$ is the source on the boundary Γ.[4]

(a) Derive the weak form corresponding to the above problem, and give the energy functional that is minimized by the solution u of (1).

(b) Solve problem (1) in FEniCS using quadratic finite elements. Choose Ω to be a disc with radius 1 around the origin and take the source terms to be

$$f = \exp(-100(x^2+y^2)) \quad \text{and} \quad u_0 = 0.$$

[2] Hint: For vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^n$, note the identity $(\boldsymbol{a}\cdot\boldsymbol{b})\boldsymbol{c} = (\boldsymbol{c}\boldsymbol{a}^T)\boldsymbol{b}$, where $\boldsymbol{a}\cdot\boldsymbol{b} \in \mathbb{R}$ is the inner product and $\boldsymbol{c}\boldsymbol{a}^T \in \mathbb{R}^{n\times n}$ is a matrix of rank one.

[3] This problem is a lot easier if you determine the eigenvalues/eigenvectors by inspection (i.e., just look at $\boldsymbol{A}$ and see if you can deduce the two eigenvectors that make it singular). If you write out the characteristic polynomial of $\boldsymbol{A}$ and try to determine its roots, it will be a lot more complicated.

[4] One interpretation of the boundary value problem (1) is that it describes the steady state conduction of heat in a solid body. In this case, u is the temperature, $\boldsymbol{A}$ is the anisotropic thermal conductivity tensor, f is the distributed heat source, and the temperature on the boundary Γ is maintained at u_0.

Use conductivity tensors $\boldsymbol{A}(\boldsymbol{x})$ given by

$$\boldsymbol{A}_1 = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} \text{ and } \boldsymbol{A}_2 = \begin{pmatrix} 1 & -5 \\ -5 & 100 \end{pmatrix}$$

and compare the results obtained using $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ in (1).

To construct a mesh for the unit circle domain, one can use `mshr`, the mesh generation component of FEniCS. To install `mshr` with anaconda run the following:

```
conda install -c conda-forge mshr
```

After that, the commands below can be used to generate the mesh:

```
import mshr
mesh = mshr.generate_mesh(mshr.Circle(Point(0.,0.), 1.), 40)
```

Alternatively, a mesh for this example can be loaded from the file `circle.xml`, available on Canvas:

```
mesh = Mesh("circle.xml")
```

To define the conductivity tensor $\boldsymbol{A}(\boldsymbol{x})$ use the commands:

```
A1 = Constant(((10.0, 0.0),(0.0, 10.0)))
A2 = Constant(((1.0, -5.0),(-5.0, 100.0)))
```

4. Implement the image denoising method from Problem 1 above in FEniCS using Tikhonov (TN) and total variation (TV) regularizations for the image given in the file `image.dat`. The file `tntv.py` contains some code to get you going (in particular, definition of the mesh, finite element space, and an expression to evaluate the true image and the noisy image at each point of the mesh).

   (a) Solve the denoising inverse problem using TN regularization. Since for TN regularization, the gradient is linear in u, you can use FEniCS's linear solver, `solve`. Choose an $\alpha > 0$ such that you obtain a reasonable reconstruction, i.e., a reconstruction that removes noise from the image but does not overly smooth the image.[5]

   (b) Solve the denoising inverse problem defined above using TV regularization. Since in this case the gradient is nonlinear in u, use the `InexactNewtonCG` nonlinear solver provided in the file `unconstrainedMinimization.py`. This solver uses the inexact Newton CG method with Eisenstat-Walker early termination condition and Armijo-based backtracking line search. It uses FEniCS's built-in CG algorithm, which does not support early termination due to detection of negative curvature; this is fine, since the image denoising objective functionals for both TN and TV result in positive definite Hessians.[6] Use appropriate values for α and δ by experimenting with a few choices, with the goal of achieving the best denoising while preserving edges in the image. You will have to increase the default number of nonlinear iterations in `InexactNewtonCG`.[7] How does the number of nonlinear iterations behave for

[5] Either experiment manually with a few values for α or use the L-curve criterion.

[6] See the file `energyMinimization.py` for an example of how to use this nonlinear solver. FEniCS's built-in nonlinear solver is not appropriate for this problem, since it does not know that the nonlinear equation representing the vanishing of the gradient stems from an optimization problem. On the other hand, implementing our own nonlinear solver allows us to globalize Newton's method via an Armijo line search based on knowledge of the objective functional.

[7] Typing "`help(InexactNewtonCG)`" will show you solver options. You should set the relative tolerance `rel_tolerance` to 10^{-5} and increase the value of `max_iter`, which defaults to 20.

decreasing δ (e.g., from 10 to 10^{-4})? *Optional:* Try to explain this behavior. Note that for small values of δ, there are more efficient methods for solving TV-regularized inverse problems than the basic Newton method we use here, though they are more involved.[8]

(c) Compare and contrast the denoised images obtained with TN and TV regularizations, using the insight derived from your answers to Problem 1.

[8] In particular, *primal-dual* Newton methods are very efficient; see T.F. Chan, G.H. Golub, and P. Mulet, A nonlinear primal-dual method for total variation-based image restoration, *SIAM Journal on Scientific Computing*, 20(6):1964–1977, 1999. The efficient solution of TV-regularized inverse problems remains an active field of research.