Spring 2016: Advanced Topics in Numerical Analysis: Computational and Variational Methods for Inverse Problems Assignment 3 (due Mar. 31, 2016)

The assignment requires/uses some FEniCS files, which you can find in the directory http://math.nyu.edu/~stadler/inv16/assignment3/. As reference for both, basics of the finite element method as well as FEniCS, I recommend the FEniCS Tutorial http://hplgit.github.io/fenics-tutorial/doc/web/index.html.

1. **[Minimal surfaces]** We consider a minimal surface problem over a bounded domain $\Omega \subset \mathbb{R}^2$ with boundary Γ (see also Figure 1). This amounts to finding minimizers $u^\star$ of the energy functional

$$\Pi(u) = \int_\Omega \sqrt{1 + |\nabla u|^2}\, d\boldsymbol{x}. \tag{1}$$

over all functions $u = u(\boldsymbol{x}) : \Omega \to \mathbb{R}$ from a space X of sufficiently smooth[1] functions, which satisfy the boundary condition $u(\boldsymbol{x}) = u_0(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Gamma$.

(a) Compute the first variation $\delta\Pi(u)(\hat{u})$ in directions $\hat{u} \in X_0 := \{u \in X, u = 0 \text{ on } \Gamma\}$.

(b) From the Euler-Lagrange equations, we know that $\delta\Pi(u^\star)(\hat{u}) = 0$ for all $\hat{u} \in X_0$. Use integration by parts to derive the nonlinear partial differential equation a minimizer $u^\star$ must satisfy.

(c) Compute the second variations of Π and give the Newton system at a point $u^k \in X$ in weak form, i.e., find $\tilde{u} \in X_0$ such that

$$\delta^2\Pi(u^k)(\hat{u}, \tilde{u}) = -\delta\Pi(u^k)(\hat{u}) \quad \text{for all } \hat{u} \in X_0. \tag{2}$$

Note that the corresponding Newton update with unit step length is then $u^{k+1} = u^k + \tilde{u}$.

(d) Use integration by parts to derive the strong (i.e., PDE) form of the Newton system (2).

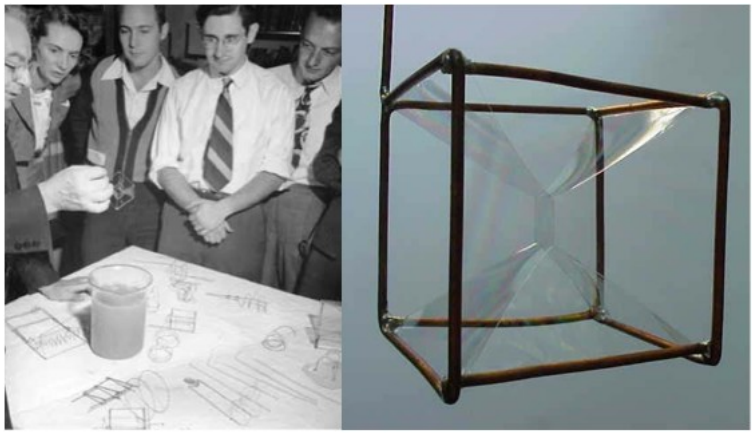

Figure 1: Already Richard Courant (left) was fascinated by the shapes of soap film surfaces. Soap films are surfaces with minimal area. For complex wires, this minimization problem can have multiple local minimia $u^\star$, corresponding to different soap film configurations. For simplicity, in our setting, we use simple "wire frames" described by the Dirichlet boundary data u_0, and find minimal surfaces that are functions over the domain Ω.

[1]The proper space here is the space of functions of bounded variations.

2. **[Minimal surfaces, cont'd]** Modify the provided FEniCS example code `MinPiNonLin.py` to solve a minimal surface problem with $\Omega \subset \mathbb{R}^2$ the unit circle and

$$u_0(\boldsymbol{x}) = u_0(x, y) = x^4 - y^2. \tag{3}$$

To specify the boundary nodes of the circle mesh provided with the 2nd homework assignment, you can use (due to round-off some of the boundary nodes have a radius slightly smaller than 1):

```
def Dirichlet_bdry(x,on_boundary)
    if (x[0]*x[0] + x[1]*x[1]) > 0.99:
        return True
    else:
        return False
```

Adapt the FEniCS nonlinear solver, and the infinite-dimensional Newton step solver from the example problem to solve this minimal surface problem.[2]

3. **[Total variation image denoising]** 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 $u_0(x, y)$ 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

$$\mathcal{F}_{LS} := \int_\Omega (u - u_0)^2 \, d\boldsymbol{x},$$

while also removing noise, which is assumed to comprise very "rough" components of the image. This latter goal can be incorporated as an additional term in the objective, in the form of a penalty, i.e.,

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

where $k(\boldsymbol{x})$ acts as a "diffusion" coefficient that controls how strongly we impose the penalty, 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*,

$$\mathcal{R}_{TV} := \int_\Omega k(\boldsymbol{x}) \sqrt{(\nabla u \cdot \nabla u)} \, d\boldsymbol{x}$$

where (we will see that) taking the square root is the key to preserving edges. Since $\mathcal{R}_{TV}$ is not differentiable when $\nabla \boldsymbol{u} = \boldsymbol{0}$, it is usually modified to include a positive parameter ε as follows:

$$\mathcal{R}_{TV}^{\varepsilon} := \int_\Omega k(\boldsymbol{x}) \sqrt{\nabla u \cdot \nabla u + \varepsilon} \, d\boldsymbol{x}.$$

We wish to study the performance of the two denoising functionals $\mathcal{F}_{TN}$ and $\mathcal{F}_{TV}^{\varepsilon}$, where

$$\mathcal{F}_{TN} := \mathcal{F}_{LS} + \mathcal{R}_{TN}$$

[2] FEniCS's nonlinear equation solver implements Newton's method. FEniCS is able to symbolically linearize variational forms and thus evaluates Hessians analytically. Even though the nonlinear solver works for many problems, it does not work for every nonlinear system of PDEs, which is why it is often necessary to derive and implement Newton algorithms manually. For instance, providing the weak form of the first variation, FEniCS does not know the underlying optimization problem, which is useful to globalize algorithms through line search. Moreover, it does not enforce that the Hessian (approximation) is positive definite as it also does not use that the variational problem origins from an energy minimization. As an example where FEniCS's nonlinear solver does not converge, try multiplying the boundary condition (3) by 5. This results in a more nonlinear problem and the FEniCS solver does not converge anymore—try it!

and

$$\mathcal{F}_{TV}^{\varepsilon} := \mathcal{F}_{LS} + \mathcal{R}_{TV}^{\varepsilon}.$$

We will prescribe the homogeneous Neumann condition $\nabla u \cdot \boldsymbol{n} = 0$ on the four sides of the square, which amounts to assuming that the image intensity does not change normal to the boundary.

(a) For both $\mathcal{F}_{TN}$ and $\mathcal{F}_{TV}^{\varepsilon}$, 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 $\mathcal{R}_{TV}^{\varepsilon}$ is.

(c) For both $\mathcal{F}_{TN}$ and $\mathcal{F}_{TV}^{\varepsilon}$, derive the infinite-dimensional Newton step, in both weak and strong form. For consistency of notation, please use $\tilde{u}$ as the differential of u (i.e., the Newton step). The strong form of the second variation of $\mathcal{F}_{TV}^{\varepsilon}$ will give an anisotropic diffusion operator of the form $-\nabla \cdot (\boldsymbol{A}(u)\nabla\tilde{u})$, where $\boldsymbol{A}(u)$ is an anisotropic tensor[3] that plays the role of the diffusivity coefficient[4]. (In contrast, you can think of the second variation of $\mathcal{F}_{TN}$ giving an *isotropic* diffusion operator, i.e. with $\boldsymbol{A} = \alpha\boldsymbol{I}$ for some α.)

(d) Derive expressions for the two eigenvalues and corresponding eigenvectors of $\boldsymbol{A}$. Based on these expressions, give an explanation of why $\mathcal{F}_{TV}^{\varepsilon}$ is effective at preserving sharp edges in the image, while $\mathcal{F}_{TN}$ is not. Consider a single Newton step for this argument.

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

4. **[Total variation image denoising, cont'd]** Implement the denoising problem with Tikhonov (TN) and total variation (TV) regularizations in FEniCS. To this end, set $k(\boldsymbol{x}) = \alpha$ with small $\alpha > 0$ in $\mathcal{R}_{TV}$ and $\mathcal{R}_{TN}$, choose small $\varepsilon > 0$ and zero Neumann boundary conditions for u[5]. The file (`tntv.py`) contains lines to start the implementation.

   (a) Compute a reconstruction using TN regularization. Since for TN regularization the first variation is linear (in u), you can use FEniCS's solver `solve`. Choose an $\alpha > 0$ such that you obtain a reasonable reconstruction[6], i.e., a reconstruction that removes noise from the image but also does not smooth the image too much.

   (b) Compute a reconstruction using TV regularization. Since the first variation is nonlinear in u, FEniCS would try to use its nonlinear solver within the `solve` function,[7] but you will observe that this solver does not converge. Use the `InexactNewtonCG` nonlinear solver provided in the file `unconstrainedMinimization.py`. This solver uses the inexact Newton CG method with Armijo 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 appropriate here since the image denoising objective functionals for both TN and TV result in positive

[3]Compare with problem 4 on the previous assignment.

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

[5]This amounts to minimizing over all H^1 functions without restricting values at the boundary. Since this is the "natural" boundary condition, you can simply not define any boundary condition.

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

[7]As discussed before, this function implements a Newton method, which uses symbolic variations of the weak form.

definite Hessians.[8] Find an appropriate value for α[9]. You will have to increase the number of nonlinear iterations in `InexactNewtonCG`[10] How does the number of nonlinear iterations behave for decreasing ε (e.g., between 10 and 10^{-4})? Try to explain this behavior[11].

(c) Compare the reconstructions obtained with TN and TV regularization using the results from the previous problem.

[8]See the file `energyMinimization.py` for an example of how to use this nonlinear solver. Compared to FEniCS's build-in solver, implementing our own nonlinear solver allows us to globalize Newton's method via an Armijo line search based on knowledge of the cost functional.

[9]Try out a few different values for α. Using the L-curve or Morozov's criterion in not justified here, since compared to TN regularization, TV-regularization is not quadratic, which makes the automatic choice of α harder.

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

[11]There are more efficient, so called primal-dual Newton algorithms for TV-regularized problems (see, for instance, Chan-Golub-Mulet, SIAM Journal on Scientific Computing, 20(6), 1999). The efficient solution of total variation problems is still an active field of research.