Spring 2016: Advanced Topics in Numerical Analysis:
Computational and Variational Methods for Inverse Problems
Assignment 2 (due Mar. 10, 2016)

1. **[An example without explicitly available Hessian]** This problem gives an example of a (finite-dimensional) optimization problem in which the Hessian is not available explicitly, but its application to vectors is. Inverse problems governed by partial differential equations (PDEs) result, upon regularization and discretization, in optimization problems of the following form:

$$\min_{\boldsymbol{m}\in\mathbb{R}^p,\boldsymbol{u}\in\mathbb{R}^n} \frac{1}{2}(O\boldsymbol{u}-\boldsymbol{d})^T(O\boldsymbol{u}-\boldsymbol{d}) + \frac{\alpha}{2}\boldsymbol{m}^T R\boldsymbol{m} \tag{1a}$$

$$\text{subject to } \; K\boldsymbol{u} = B\boldsymbol{m}. \tag{1b}$$

   Here, $\boldsymbol{u}\in\mathbb{R}^n$ is the discretized *state variable*, i.e., the PDE solution, $O\in\mathbb{R}^{k\times n}$ can be thought of as the *observation operator* that extracts observation data from the state, $\boldsymbol{d}\in\mathbb{R}^k$ are *observation data*, $R\in\mathbb{R}^{p\times p}$ is the symmetric and positive definite *regularization* matrix, and $\alpha>0$ the regularization parameter. Moreover, $B\in\mathbb{R}^{n\times p}$ and $K\in\mathbb{R}^{n\times n}$. We think of (1b) as originating from the discretization of a PDE. In particular, K is invertible but we are not able to compute its inverse explicitly—however, for given $\boldsymbol{m}$, we can solve the system (1b) using a PDE solver.[1]

   (a) Eliminate the equality constraint (1b) from the optimization problem (1) using[2] $\boldsymbol{u} = K^{-1}B\boldsymbol{m}$ and thus write (1) as unconstrained optimization problem over the parameters $\boldsymbol{m}$ only (this is sometimes called the *reduced problem*).

   (b) Compute expressions for the gradient and the Hessian of this reduced problem.[3] Argue that the Hessian is positive definite.

   (c) How many solves with K are necessary to compute the gradient and to apply the Hessian matrix to a vector, as required in the conjugate gradient method?

   (d) The Hessian has two parts, one coming from the regularization and one from the *data misfit term* in (1a). Show that the rank of the data misfit part cannot be larger than the rank of the observation matrix O.[4]

2. **[Write your own simple CG solver]** A conjugate gradient (CG) method for solving the linear system $H\boldsymbol{x}=\boldsymbol{b}$ solves the equivalent minimization problem

$$\min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T H\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}, \tag{2}$$

   provided H is symmetric and positive definite. Implement your own CG solver, which terminates when a direction of zero or negative curvature is detected. The syntax should be as follows:

[1] For instance, the inverse problem for the initial condition in the time-dependent advection-diffusion problem we discussed in class results, when discretized, in a system like (1). In this case, solving (1b) corresponds to integrating a time-dependent PDE for given initial condition $\boldsymbol{m}$, and n corresponds to the number of discretization points in space and time and is thus very large. While the space-time operator K is sparse, K^{-1} is dense and cannot be computed in reasonable time, nor stored in memory.

[2] Note that K^{-1} is merely a notation here. Since this inverse matrix is not available to us, K^{-1} means to *solve a system with the matrix* K.

[3] Note that since this is a quadratic problem, the Hessian does not depend on m.

[4] In particular, if only a small number of observation data are available (i.e., k is small), the Hessian is dominated by the regularization matrix R.

```
function x = mycg(H,b,maxiter,tol,x0)
% Conjugate Gradient Method.
% [X,iter] = MYCG(H,B,maxiter,tol,x0) solves the system of linear
% equations H*X=B for X. The N-by-N coefficient matrix H must be
% symmetric and the right hand side column vector B must have length
% N. X0 is the starting vector for the iteration, MAXITER the
% maximum number of iterations, and TOL the absolute tolerance.
% As output, the function returns the solution vector X and the
% number of required iterations ITER
```

For our tests, we use the following matrix, which arises from the discretization of the negative second derivative operator ($-u''$) with homogeneous Neumann boundary conditions in one dimension:[5]

$$R = \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

For simplicity, we choose $\boldsymbol{b} = 0$ and initialize the CG iteration with a non-zero vector. We will also use a sparse symmetric matrix with a few random[6] diagonal entries generated by

```
rng('default');
M = spdiags(max(0,randn(n,1)-1.5),0,n,n);
```

(a) Verify your CG implementation by comparing it to a direct solver (e.g., MATLAB's backslash) and an available CG implementation (e.g., MATLAB's `pcg`).

(b) Due to the optimization formulation (2), the CG method can also be applied to problems with semi-definite matrices. The result of CG then depends on the initialization, and the difference between solutions found with different initializations is in the null space of the matrix H. Show that R has constant vectors in its null space, and apply your CG implementation to solve $R\boldsymbol{x} = \boldsymbol{b}$ with different non-zero initializations. Show (numerically) that the difference of solutions found with different initializations is in the null space of R.

(c) Let us solve the system

$$(M + \gamma R)\boldsymbol{x} = \boldsymbol{b}, \tag{3}$$

for $\gamma = 1, 10^{-4}, 10^{-8}$ with a fixed CG tolerance (e.g., `tol`$=10^{-6}$). Plot the spectra of these matrices and report the number of CG iterations. Try to explain the behavior.[7]

(d) Let us now consider a preconditioned system. Instead of (3), we solve a transformed system for $\hat{\boldsymbol{x}} := L\boldsymbol{x}$:

$$L^{-T}(M + \gamma R)L^{-1}\hat{\boldsymbol{x}} = L^T\boldsymbol{b}, \tag{4}$$

where the idea is to choose L that can be inverted efficiently, and such that CG converges faster for the transformed matrix $L^{-T}(M + \gamma R)L^{-1}$ than for the original matrix $M +$

[5] In MATLAB, the matrix R can, for $n = 100$, be generated as follows: `n=100; e = ones(n,1); R = spdiags([-e 2*e -e], -1:1, n, n); R(1,1) = 1; R(n,n) = 1;`.

[6] It's always a good idea to seed the random number generator to be able to reproduce results

[7] Chapter 5 in the book of Nocedal/Wright has a nice summary of the properties of the CG method, as well as J. Shewchuck's "painless" introduction to CG. As I have mentioned in class, CG has a preference for targeting the error in the directions of the eigenvalues corresponding to the large eigenvalues, and is very efficient in targeting errors corresponding to directions that have identical eigenvalues, i.e., when eigenvalues are clustered.

γR.[8] Test the two following preconditioning ideas by solving (4) for different values of γ. Report iteration numbers, plot spectra of the transformed matrix and try to explain your observations.

- Use $L = \text{diag}(\sqrt{l_i})$, where l_i are the diagonal entries of $M + \gamma R$.[9]
- L is a Choleski factor of γR, i.e., $\gamma R = LL^T$.[10]

3. **[Inexact Newton-CG]** Write a program that implements the inexact Newton-conjugate gradient method as described in class[11] and use it to solve the following problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^4} f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T(\boldsymbol{I} + \mu\boldsymbol{A})\boldsymbol{x} + \frac{\sigma}{4}\left(\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}\right)^2$$

with parameters $\sigma > 0, \mu \geq 0$, the identity matrix $\boldsymbol{I}$, and the matrix $\boldsymbol{A}$ given by

$$\boldsymbol{A} = \begin{pmatrix} 5 & 1 & 0 & 0.5 \\ 1 & 4 & 0.5 & 0 \\ 0 & 0.5 & 3 & 0 \\ 0.5 & 0 & 0 & 2 \end{pmatrix}.$$

Your implementation should have the following features:

- Terminate the CG iterations when $\|H_k p_k + g_k\| \leq \eta_k \|g_k\|$, and implement the following three options for η:
  - $\eta_k = 0.5$
  - $\eta_k = \min(0.5, \sqrt{\|g_k\|})$
  - $\eta_k = \min(0.5, \|g_k\|)$
- Also terminate the CG iterations when a direction of negative curvature is detected within the CG iteration.
- For a line search, implement a backtracking Armijo line search as described in class.

(a) Please turn in code listings of your implementation.

(b) Compare the performance of Newton and steepest descent for $\sigma = 1, \mu = 0$, as well as for $\sigma = 1, \mu = 10$. Use the starting point $\boldsymbol{x} = (\cos 70°, \sin 70°, \cos 70°, \sin 70°)^T$. Can you explain the different behavior?

(c) Experiment with the different choices of η for $\sigma = 1$ and $\mu = 10$. Verify the theoretical convergence rates for these different choices.

[8] One can write the CG algorithm such that one does not explicitly solve systems with L, but with the symmetric positive definite matrix $P = LL^T$.

[9] This obviously assumes that the diagonal entries of both matrices are explicitly available, which is not always the case (see the problem above). This simple preconditioner, for which L can be inverted easily, is called a *diagonal preconditioner*.

[10] In the inverse problem context, where R originates from the regularization and M is a product of operators, this is called *preconditioning by the regularization operator*. Ideally, this leads to clustering of eigenvalues, which can speed up the convergence of the CG method substantially. Since R is only positive semi-definite, you can add εI to R, with, e.g., $\varepsilon \sim 10^{-10}$, to ensure that the matrix is positive and thus the Choleski factorization can be computed.

[11] References: J. Nocedal and S. Wright, *Numerical Optimization*, Springer, 2006; For more details, see S.C. Eisenstat and H.F Walker, *Globally convergent inexact Newton's method*, SIAM Journal on Optimization, Vol. 4, p.393–422, 1994.

4. **[FEniCS warmup]** An anisotropic Poisson problem in a two-dimensional domain Ω is given by the strong form

$$-\nabla \cdot (A(x)\nabla u) = f \quad \text{in } \Omega, \tag{5a}$$
$$u = u_0 \quad \text{on } \partial\Omega, \tag{5b}$$

where $A(x) \in \mathbb{R}^{2\times 2}$ is assumed to be symmetric and positive definite for every x, f is a given force, and u_0 are Dirichlet boundary data.

(a) Derive the variational/weak form corresponding to the above problem, and give the energy functional that is minimized by the solution u of (5).

(b) Solve problem (5) in FEniCS using quadratic finite elements. Choose Ω to be a disc with radius 1 around the origin and use[12]

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

Use the diffusion matrices $A(x)$ given by

$$A_1 = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} \text{ and } A_2 = \begin{pmatrix} 1 & -5 \\ -5 & 100 \end{pmatrix}$$

and compare the results obtained with A_1 and A_2 in (5). The mesh for the circle geometry can be found in the directory http://math.nyu.edu/~stadler/inv16/assignment2/. You can load the mesh into FEniCS using the command

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

To define the tensors $A(x)$, use

```
A1 = Expression(((''10'',''0''), (''0'',''10'')))
A2 = Expression(((''1'',''-5''), (''-5'',''100''))).
```

[12] If you think of u in (5) as a temperature, f is a heat source located at the origin of the domain, and the boundary condition means that the temperature at the boundary $\partial\Omega$ is set to 0. Then, (5) describes how the heat diffuses in a steady state.