Inverse problems course, spring 2014 Exercise 5 (February 18-21, 2014)
University of Helsinki
Department of Mathematics and Statistics
Samuli Siltanen, Esa Niemi and Teemu Saksala

Note that this exercise has more than one page and contains *two* types of exercises: theoretical (T) and computational (M).

Please complete the theoretical exercises T1 and T2 before the exercise session and be prepared to present your solution there. Exercise T3 is more complicated and needs not be done beforehand. (Of course you *can* do T3 beforehand if you want!) You will get credit for it by understanding it in the exercise session.

T1. Explain how the conjugate gradient method can be used to solve the linear equation $(A^TA + \alpha L^TL)f = A^Tm$ without actually constructing any of the matrices $A, A^T, L$ or $L^T$.

**Answer**: We have to make two standing assumptions if we want to use theorems of sections 5.5.1 and 5.5.2 of textbook. Assume that $\alpha > 0$ and that square matrix

$$\widetilde{A} := A^TA + \alpha L^TL$$

is invertible. For instance it's safe to assume that square matrix $L$ is invertible, then the invertibility of $\widetilde{A}$ follows from T1 of exercise set 4.

If we want to use the Conjugate gradient method we must first clarify what is the quantity we want to minimize. In our case the quantity is

$$\|\widetilde{A}f - A^Tm\|^2. \tag{1}$$

Calculate

$$\begin{aligned}\|\widetilde{A}f - A^Tm\|^2 &= \langle \widetilde{A}f, \widetilde{A}f\rangle - 2\langle \widetilde{A}f, A^Tm\rangle + \langle A^Tm, A^Tm\rangle \\ &= f^T\widetilde{A}^T\widetilde{A}f - (2m^TA\widetilde{A})f + \|A^Tm\|^2. \end{aligned}\tag{2}$$

It follows that $\widetilde{A}^T\widetilde{A}$ is positive definite and symmetric, since for any $v \in \mathbb{R}^n \neq 0$ it holds that

$$v^T\widetilde{A}^T\widetilde{A}v = (\widetilde{A}v)\cdot(\widetilde{A}v) = \| \underbrace{\widetilde{A}v}_{\neq 0,\ \widetilde{A}\text{ invertible}} \|^2 > 0.$$

Write

$$Q := 2\widetilde{A}^T\widetilde{A} \text{ and } \mathbf{b}^T := (2m^TA\widetilde{A}).$$

Then it holds by equation (2) that we minimize the quantity (1) iff we minimize

$$\frac{1}{2}f^TQf - \mathbf{b}^Tf. \tag{3}$$

But now quantity (3) is the quantity (5.23) of the textbook and as the exercise T2 will show us it can be minimized by conjugate gradient method. We should

also note that in order to use the conjugate gradient method we need only to know the vectors

$$\mathbf{b} \text{ and } Q\mathbf{d}_k, \text{ for every } k \in \{0, \dots, n-1\}.$$

not the matrices $L$ or $\widetilde{A}$.

T2. Prove that the vectors $d_k$ constructed in the conjugate gradient algorithm in Section 5.5.2 of the textbook satisfy the assumptions of Theorem 5.4 of the textbook.

**Answer**: Suppose that we are given a symmetric and positive definite matrix $Q$. We must show that the set $(\mathbf{d}_k)_{k=0}^{n-1}$ is $Q$ orthogonal i.e.

$$\mathbf{d}_k^T Q \mathbf{d}_j = 0, \text{ if } k \neq j.$$

To prove this we must use induction. The proof given here is from a book Numerical Optimization by Nocedal and Wright Theorem 5.3. You can find a part of the book copied in the same folder as the textbook. First we note that by Conjugate gradient algorithm it holds that

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \\ \mathbf{g}_k = Q\mathbf{x}_k - \mathbf{b} \end{cases} \Rightarrow \mathbf{g}_{k+1} = \mathbf{g}_k + \alpha_k Q \mathbf{d}_k. \tag{4}$$

Here we multiplied the first row with $Q$ and used the second row for $k$ and $k+1$. Let $k \in \mathbb{N}$ and suppose that $x_k - x^* \neq 0$. We will prove the following three claims, where the one we are interested is the last one.

$$\text{span}(\mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_k) = \text{span}(\mathbf{g}_0, Q\mathbf{g}_0 \dots, Q^k \mathbf{g}_0) \tag{5}$$
$$\text{span}(\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_k) = \text{span}(\mathbf{g}_0, Q\mathbf{g}_0 \dots, Q^k \mathbf{g}_0) \tag{6}$$
$$\mathbf{d}_k^T Q \mathbf{d}_i = 0, \text{ for } i = 0, \dots, k-1 \tag{7}$$

Since we assumed that $\mathbf{d}_0 = -\mathbf{g}_0$ the equations (5) and (6) hold trivially. By Conjugate Gradient algorithm it holds that

$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k \text{ and } \beta_k(\mathbf{d}_k^T Q \mathbf{d}_k) = -\mathbf{g}_{k+1}^T Q \mathbf{d}_k$$

Then it holds that

$$\beta_k \mathbf{d}_k^T Q \mathbf{d}_k = (\beta_k \mathbf{d}_k - \mathbf{d}_{k+1})^T Q \mathbf{d}_k \Rightarrow -\mathbf{d}_{k+1}^T Q \mathbf{d}_k = 0.$$

I.e. equation (7) holds for $k = 1$.

Next we prove by induction equation (5). We assume that equations (5) and (6) hold for $k$. Then it holds that

$$\mathbf{g}_k \in \text{span}(\mathbf{g}_0, Q\mathbf{g}_0 \dots, Q^k \mathbf{g}_0) \text{ and } \mathbf{d}_k \in \text{span}(\mathbf{g}_0, Q\mathbf{g}_0 \dots, Q^k \mathbf{g}_0)$$

$$\Rightarrow Q\mathbf{d}_k \in \text{span}(Q\mathbf{g}_0, Q^2\mathbf{g}_0 \dots, Q^{k+1}\mathbf{g}_0).$$

By equation (4) and induction hypothesis we have that

$$\mathbf{g}_{k+1} \in \text{span}(\mathbf{g}_0, Q\mathbf{g}_0, Q^2\mathbf{g}_0 \ldots, Q^{k+1}\mathbf{g}_0).$$

By induction hypothesis of equation (5) we have now

$$\text{span}(\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_k, \mathbf{g}_{k+1}) \subset \text{span}(\mathbf{g}_0, Q\mathbf{g}_0 \ldots, Q^{k+1}\mathbf{g}_0)$$

By induction hypothesis of (6) we have that

$$Q^{k+1}\mathbf{g}_0 = QQ^k\mathbf{g}_0 \in \text{span}(Q\mathbf{d}_0, Q\mathbf{d}_1, \ldots, Q\mathbf{d}_k)$$

By equation (4) it holds that

$$Q\mathbf{d}_i = \frac{\mathbf{g}_{i+1} - \mathbf{g}_i}{\alpha_i} \text{ for all } i = 0, 1, \ldots, k.$$

Therefore we have that

$$Q^{k+1}\mathbf{g}_0 \in \text{span}(Q\mathbf{d}_0, Q\mathbf{d}_1, \ldots, Q\mathbf{d}_k) \subset \text{span}(\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k+1}).$$

By induction hypothesis we have now proved that

$$\text{span}(\mathbf{g}_0, Q\mathbf{g}_0 \ldots, Q^{k+1}\mathbf{g}_0) \subset \text{span}(\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_k, \mathbf{g}_{k+1})$$

This ends the proof of equation (5) for $k+1$. Next we prove that (6) holds for $k+1$. Recall that in Conjugate gradient algorithm we have assumed that

$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k \tag{8}$$

Write

$$\text{span}(\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{k+1}) \overset{(8)}{=} \text{span}(\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_k, \mathbf{g}_{k+1}) \overset{induction\ hyp}{=} \text{span}(\mathbf{g}_0, Q\mathbf{g}_0 \ldots, Q^k\mathbf{g}_0, \mathbf{g}_{k+1})$$

$$\overset{(5)for\ k}{=} \text{span}(\mathbf{g}_0, \mathbf{g}_1 \ldots, \mathbf{g}_k, \mathbf{g}_{k+1}) \overset{(5)for\ k+1}{=} \text{span}(\mathbf{g}_0, Q\mathbf{g}_0 \ldots, Q^k\mathbf{g}_0, Q^{k+1}\mathbf{g}_0)$$

This ends the proof of equation (6) for $k+1$.

Next we prove that (7) holds for $k+1$. Multiply equation (8) by $Q\mathbf{d}_i$ $i = 0, \ldots, k$ to get

$$\mathbf{d}_{k+1}^T Q\mathbf{d}_i = -\mathbf{g}_{k+1}^T Q\mathbf{d}_i + \beta_k \mathbf{d}_k^T Q\mathbf{d}_i. \tag{9}$$

By definition of $\beta_k$ it holds that right hand side of (9) is zero if $i = k$. By induction hypothesis for equation (7) it holds that the second term of right hand side of (9) vanishes. Therefore it suffices to show that

$$\mathbf{g}_{k+1}^T Q\mathbf{d}_i = 0, \text{ for all } i = 0, \ldots, k.$$

First we note that it holds

$$\mathbf{g}_{k+1}^T \mathbf{d}_i = 0, \text{ for all } i = 0, \ldots, k$$

as it is indicated in textbook. If one wants to find a rigorous proof for this claim it is Theorem 5.2. in book mentioned earlier. Now it holds that

$$Q\mathbf{d}_i \in Q\ \mathrm{span}(\mathbf{g}_0, Q\mathbf{g}_0 \ldots, Q^i\mathbf{g}_0) = \mathrm{span}(Q\mathbf{g}_0, Q^2\mathbf{g}_0 \ldots, Q^{i+1}\mathbf{g}_0)$$

$$\overset{(6)}{\subset} \mathrm{span}(\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{i+1})$$

Therefore it holds that

$$\mathbf{g}_{k+1}^T Q\mathbf{d}_i = \mathbf{g}_{k+1}^T \left( \sum_{j=0}^{i+1} a_j \mathbf{d}_j \right) = 0, \text{ for } i = 0, \ldots, k-1.$$

This proves our claim.

T3. Show that truncated SVD (as formulated in Definition 4.2.1) gives a regularization method in the sense of Definition 3.4.1.

**Answer**: Let $\alpha > 0$. In our case we are interested about the situation

$$A : \mathbb{R}^n \to \mathbb{R}^k, A \text{ is linear and one-to-one,}$$

As we know already from linear algebra every Euclidean space is a finite dimensional Hilbert space and by this it also holds that $A$ is bounded. Write $A$ into SVD as $A = UDV^T$. Since $A$ is one-to-one it must hold that the singular values $d_1 \geq d_2 \geq \ldots \geq d_r$ are all strictly positive and $r = \min\{n, k\}$.

Let $\frac{1}{D}$ be that matrix where we have changed $d_i \mapsto \frac{1}{d_i}$ and kept all the zero elements of $D$ intact. Then it holds that

$$D^+ = \frac{1}{D}^T.$$

By definition we have that $D_\alpha^+ = D^+$ if $\alpha < d_r$, since

$$r_\alpha := \min\left\{r, \max\{j | 1 \leq j \leq \min(k, n), d_j > \alpha\}\right\}.$$

We also defined that

$$\mathcal{L}_\alpha := V D_\alpha^+ U^T.$$

Therefore it holds by orthogonality of $U$ and $V$ that

$$\mathcal{L}_\alpha A f = (V D_\alpha^+ U^T)(U D V^T) f = (V D_\alpha^+ D V^T) f \overset{\alpha \to 0}{\longrightarrow} (V D^+ D V^T) f = f.$$

For the second claim we can consider function $\alpha(t) = t$, $t \in (0, 1)$. We have to prove that

$$\alpha(t) \overset{t \to 0}{\longrightarrow} 0 \text{ and } \sup_m \left\{ \|\mathcal{L}_{\alpha(t)} m - f\| : \|Af - m\| \leq t \right\} \overset{t \to 0}{\longrightarrow} 0.$$

The first claim holds trivially by the choice of $\alpha$. Write $m = Af + \epsilon$, $\|\epsilon\| < t$. Then it holds that

$$\|Af - m\| = \|\epsilon\| \leq t.$$

Calculate

$$\|\mathcal{L}_{\alpha(t)}m - f\| \le \|\mathcal{L}_{\alpha(t)}Af - f\| + \|\mathcal{L}_{\alpha(t)}\epsilon\| = \|\mathcal{L}_{\alpha(t)}Af - f\| + \|(VD_\alpha^+U^T)\epsilon\|$$
$$= \|\mathcal{L}_{\alpha(t)}Af - f\| + \|(D_\alpha^+U^T)\epsilon\| \le \|\mathcal{L}_{\alpha(t)}Af - f\| + \|(D^+U^T)\epsilon\|$$
$$\le \|\mathcal{L}_{\alpha(t)}Af - f\| + \frac{1}{d_r}\|(U^T)\epsilon\| = \|\mathcal{L}_{\alpha(t)}Af - f\| + \frac{t}{d_r} \xrightarrow{t\to 0} 0.$$

You can work on these Matlab exercises (marked with M) in the exercise session.

M1. Implement the conjugate gradient method can be used to solve the linear equation $(A^TA + \alpha L^TL)f = A^Tm$ for the one-dimensional deconvolution problem. Let $L$ be the first-order finite difference matrix; you can implement it in a matrix-free fashion using the Matlab command `diff`.

**M1. Answer**

In this exercise there are many options. One can do it by using explicit matrices $A, L$ or matrix-freely by replacing $A$ and $L$ by appropriate routines (e.g. $A$ by `conv.m` and $L$ by `diff.m`). Also, one can either implement the conjugate gradient method by him/herself or just use the Matlab's built-in (preconditioned) conjugate gradient routine `pcg.m`. In the following we present three possible solutions:

```
% Note that the files DC1_cont_data_comp.m and DC2_discretedata_comp.m must
% be run before this code.

% Load the data (N.B. Add the last three variables (PSF, Ca and Dx)
% to the corresponding save command in file DC2_discretedata_comp.m and run
% it)
load DC2_discretedata A x xx n m mn mIC sigma PSF Ca Dx

% Choose the regularization parameter
alpha = 0.1;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 1. THE EASIEST WAY

% Use Matlab's function pcg.m and explicit matrices A and L

L = eye(n);
L = L-[L(:,end),L(:,1:end-1)];
AA  = A.'*A+alpha*(L.'*L);
b   = A.'*mn(:);
```

```
rec = pcg(AA,b);

% Take a look at the reconstruction
figure(1)
clf
plot(xx,DC_target(xx),'k');
hold on
plot(x,rec,'r')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 2. IMPLEMENT CONJUGATE GRADIENT METHOD WITHOUT USING PCG.M

Q = A.'*A+alpha*(L.'*L);
b = A.'*mn(:);

% Choose the number of iterations (we remark that there are many
% possible stopping conditions for the iteration; for details, see
% literature)
maxIter = 10;

% Choose initial point
f_prev = zeros(n,1);

% Start iteration
d_prev = b-Q*f_prev;
g_prev = -d_prev;
for k=1:maxIter
    Qd      = Q*d_prev;
    dQd     = d_prev.'*Qd;
    lambda  = (-g_prev.'*d_prev)/dQd;
    f_next  = f_prev + lambda*d_prev;
    g_next  = Q*f_next-b;
    beta    = -(g_next.'*Qd)/dQd;
    d_next  = -g_next + beta*d_prev;
    f_prev  = f_next;
    g_prev  = g_next;
    d_prev  = d_next;
end
rec = f_prev;

% Take a look at the reconstruction
figure(2)
clf
plot(xx,DC_target(xx),'k');
hold on
```

```
plot(x,rec,'r')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% 3. MATRIX-FREE IMPLEMENTATION (let's use pcg.m here for simplicity)

% Let's take the discretized PSF from DC2_discretedata_comp.m and compute
% the convolution Af using the PSF and Matlab's function conv.m (without
% matrices)
Afun = @(f) conv(f,Dx*PSF,'same');

% Define similarly the operations Lf and L^Tf matrix-freely (assume
% periodic boundary conditions)
Lfun  = @(f) diff([f;f(1)]);
LTfun = @(f) -diff([f(end);f]);

% Finally define the function that computes the operation
% (A^T*A + alpha*L^T*L)f matrix-freely
Qfun = @(f) Afun(Afun(f)) + alpha*LTfun(Lfun(f));

% Now use pcg.m to find the solution (reconstruction)
rec = pcg(Qfun,Afun(mn(:)));

% Take a look at the reconstruction
figure(3)
clf
plot(xx,DC_target(xx),'k');
hold on
plot(x,rec,'r')
```

M2. Use Morozov discrepancy principle for automatic determination of regularization parameter in M1. You can take the file `DC08_Tikhonov_Morozov1.m` as a starting point. Note that the file `DC08_Tikhonov_Morozov1.m` is not consistent with the previous deconvolution files shared at the course website and will not work properly without appropriate modification.

**M2. Answer**

We first remark that the Morozov discrepancy principle as presented in the textbook pp. 72–73 and implemented in `DC08_Tikhonov_Morozov1.m` is not applicable as such to the regularization problem in M1 with $L$ being a finite difference matrix; in order to use the Morozov principle for M1, we have to choose $L = I$.

```
% Load the data
load DC2_discretedata A x xx n m mn mIC sigma

% Choose the regularization parameter alpha by copying lines 36-42, 47 and
% 52-60 from DC08_Tikhonov_Morozov1.m

% How many singular values are zero?
[U,D,V] = svd(A);
svals   = diag(D);
r       = max(find(svals>1e-10));

% Here we implement finding the zero in a naive but simple way.
mpilkku  = U.'*mn(:);

delta = sqrt(n)*sigma;

alphavec = [0.0000001:.00001:.01];
objfun   = zeros(size(alphavec));
dvec     = zeros(size(mpilkku));
for iii = 1:length(alphavec)
    dvec(1:r)     = alphavec(iii)./(alphavec(iii) + svals(1:r).^2).*mpilkku(1:r);
    dvec(r+1:end) = mpilkku(r+1:end);
    objfun(iii)   = dvec.'*dvec - delta^2;
end
alpha = alphavec(min(find(objfun>0)));


% Let's use method 1 from M1 to solve the Tikhonov problem with the
% alpha chosen above
AA  = A.'*A+alpha*eye(n);
b   = A.'*mn(:);
rec = pcg(AA,b);

% Take a look at the reconstruction
figure(1)
clf
plot(xx,DC_target(xx),'k');
hold on
plot(x,rec,'r')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% OPTION 2: Apply Matlab's built-in optimization routine fminbnd.m

% Another way to determine alpha: use fminbnd.m function to find the
% minimizer of the function abs(f)=abs(f(alpha)), where function f as
```

```
% defined on p. 73 in the textbook
[U,D,V] = svd(A);
svals   = diag(D);
mpilkku = U.'*mn(:);
delta = sqrt(n)*sigma;
f = @(a) abs(sum(((a./(svals.^2+a)).^2).*(mpilkku.^2)) - delta^2);
alpha = fminbnd(f,0,100);

% Compute and take a look at the reconstruction
AA  = A.'*A+alpha*eye(n);
b   = A.'*mn(:);
rec = pcg(AA,b);
figure(2)
clf
plot(xx,DC_target(xx),'k');
hold on
plot(x,rec,'r')
```