Homework 6

Due: Wednesday 11/18/2015.

Reading assignment: Chapter 14 (Constrained least squares) in the textbook and chapter 2 (Cholesky factorization) in the EE133A Lecture Notes.

Homework problems: Exercises 8.15, 8.16, 10.1, 11.7 (b, c), and the following two additional problems.

1. Regularized least squares image deblurring. This exercise is on the image deblurring problem of lecture 9, pages 26–29. Note that while the problem description is long, the assignment is short and requires only a few lines of MATLAB code.

   The purpose is to develop a fast method for solving the regularized least-squares problem on page 9-27:

   \[ \text{minimize } \|Ax - y\|^2 + \lambda(\|D_v x\|^2 + \|D_h x\|^2). \]  

   (1)

   Notation. A black-and-white image of size \( n \times n \) is represented as an \( n \times n \) matrix \( X \) with \( X_{ij} \) the intensity of pixel \( i, j \), or as an \( n^2 \)-vector \( x \). We use column-major order when converting a matrix \( X \) to a vector \( x \):

   \[
   x = \begin{bmatrix}
   X_{1:n,1} \\
   X_{1:n,2} \\
   \vdots \\
   X_{1:n,n}
   \end{bmatrix}.
   \]

   In MATLAB the conversion can be done by the command \( x = X(:) \) or, equivalently, \( x = \text{reshape}(X, n^2, 1) \). To convert an \( n^2 \)-vector \( x \) to an \( n \times n \) matrix \( X \) we use \( X = \text{reshape}(x, n, n) \).

   In (1), the vectors \( x \) and \( y \) have length \( n^2 \). The vector \( y \) is given and represents an observed, noisy and blurred image \( Y \) of size \( n \times n \). The variable \( x \) is the reconstructed \( n \times n \) image \( X \) in vector form.
The following notation will be used to express the matrices $A$, $D_v$, $D_h$ in (1). As in homework 4, $T(v)$, with $v$ an $n$-vector, denotes the $n \times n$ circulant matrix with $v$ as its first column:

$$T(v) = \begin{bmatrix} v_1 & v_n & v_{n-1} & \cdots & v_3 & v_2 \\ v_2 & v_1 & v_n & \cdots & v_4 & v_3 \\ v_3 & v_2 & v_1 & \cdots & v_5 & v_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ v_{n-1} & v_n & v_{n-2} & \cdots & v_1 & v_n \\ v_n & v_{n-1} & v_{n-2} & \cdots & v_2 & v_1 \end{bmatrix}.$$

We also define $T(U)$ for an $n \times n$ matrix $U$. If the columns of $U$ are $u_1$, $u_2$, $\ldots$, $u_n$, then $T(U)$ is the $n^2 \times n^2$ matrix:

$$T(U) = \begin{bmatrix} T(u_1) & T(u_n) & T(u_{n-1}) & \cdots & T(u_3) & T(u_2) \\ T(u_2) & T(u_1) & T(u_n) & \cdots & T(u_4) & T(u_3) \\ T(u_3) & T(u_2) & T(u_1) & \cdots & T(u_5) & T(u_4) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ T(u_{n-1}) & T(u_{n-2}) & T(u_{n-3}) & \cdots & T(u_1) & T(u_n) \\ T(u_n) & T(u_{n-1}) & T(u_{n-2}) & \cdots & T(u_2) & T(u_1) \end{bmatrix}.$$

This structure is called block-circulant with circulant blocks (BCCB). Each block $T(u_i)$ is a circulant $n \times n$ matrix. The matrix $T(U)$ is block-circulant because each block column is a circular downward shift of the previous block column.

**Image blurring.** A linear, spatially invariant blurring operation is defined by a set of coefficients $P_{kl}$, with $k$ and $l$ integers ranging from $-r$ to $r$. We call $P$ (or, more accurately, the function that maps a pair $(k, l)$ to the coefficient $P_{kl}$) the point spread function (PSF). The integer $r$ is the width of the PSF. We will assume that $2r + 1 \leq n$ (usually, $r \ll n$). The blurring operation defined by the PSF $P$ is a two-dimensional convolution. It transforms an $n \times n$ image $X$ into a blurred $n \times n$ image $Y$ defined by

$$Y_{ij} = \sum_{k=-r}^{r} \sum_{l=-r}^{r} P_{kl} X_{i-k,j-l}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n. \quad (2)$$

Note that the sum in this definition references some values of $X_{i-k,j-l}$ with indices outside the interval $[1, n]$. To fully specify the blurring operation we therefore need to make assumptions about the image $X$ outside the frame. These assumptions are called boundary conditions. In this exercise we will use periodic boundary conditions. Periodic boundary conditions assume that the image $X$ is repeated periodically outside the frame. One way to write this is to replace the convolution (2) with

$$Y_{ij} = \sum_{k=-r}^{r} \sum_{l=-r}^{r} P_{kl} X_{i-k,j-l}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n, \quad (3)$$
where $\bar{X}$ denotes the larger, bordered image

$$\bar{X} = \begin{array}{ccc} X & X & X \\ X & X & X \\ X & X & X \end{array}$$

and the indices $i, j$ in $\bar{X}_{ij}$ run from $-n + 1$ to $2n$. For example, if $n = 3$, $r = 1$, the convolution with periodic boundary conditions gives

$$\begin{bmatrix} Y_{11} \\ Y_{21} \\ Y_{31} \\ Y_{12} \\ Y_{22} \\ Y_{32} \\ Y_{13} \\ Y_{23} \\ Y_{33} \end{bmatrix} = \begin{bmatrix} P_{00} & P_{-1,0} & P_{1,0} \\ P_{10} & P_{00} & P_{-1,0} \\ P_{-1,0} & P_{10} & P_{00} \\ P_{01} & P_{-1,1} & P_{11} \\ P_{11} & P_{01} & P_{-1,1} \\ P_{-1,1} & P_{11} & P_{01} \\ P_{0,-1} & P_{-1,-1} & P_{1,-1} \\ P_{1,-1} & P_{0,-1} & P_{-1,-1} \\ P_{-1,-1} & P_{1,-1} & P_{0,-1} \end{bmatrix} \begin{bmatrix} X_{11} \\ X_{21} \\ X_{31} \\ X_{12} \\ X_{22} \\ X_{32} \\ X_{13} \\ X_{23} \\ X_{33} \end{bmatrix}.$$

(In this example, $2r + 1$ happens to be equal to $n$, while in general $2r + 1 \leq n$.)

We note that the matrix on the right-hand side is a BCCB matrix. More generally, if we write the convolution with periodic boundary conditions \(3\) in matrix form as $y = Ax$, then $A$ is the $n^2 \times n^2$ matrix $A = T(B)$, where $B$ is the following $n \times n$ matrix constructed from the PSF coefficients.

$$B = \begin{bmatrix} P_{00} & P_{01} & \cdots & P_{0r} & 0 & \cdots & 0 & P_{0,-r} & \cdots & P_{0,-1} \\ P_{10} & P_{11} & \cdots & P_{1r} & 0 & \cdots & 0 & P_{1,-r} & \cdots & P_{1,-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ P_{r0} & P_{r1} & \cdots & P_{rr} & 0 & \cdots & 0 & P_{r,-r} & \cdots & P_{r,-1} \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ P_{-r,0} & P_{-r,1} & \cdots & P_{-r,r} & 0 & \cdots & 0 & P_{-r,-r} & \cdots & P_{-r,-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ P_{-1,0} & P_{-1,1} & \cdots & P_{-1,r} & 0 & \cdots & 0 & P_{-1,-r} & \cdots & P_{-1,-1} \end{bmatrix}.$$

As we will see, periodic boundary conditions are very convenient mathematically. Although the assumption of periodicity is not realistic, it is an acceptable approximation if $r \ll n$.

**Vertical and horizontal differencing.** Next we discuss the matrices $D_v$ and $D_h$ in \(\text{[1]}\). The vector $D_v x$ is the image obtained by subtracting from $X$ the image $X$ shifted up over one pixel, so that

$$\|D_v x\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} (X_{ij} - X_{i+1,j})^2.$$
Here too, the sum references values of $X_{ij}$ outside the frame (namely, the values $X_{n+1,j}$ are needed when $i = n$). To be consistent with the boundary conditions used for the blurring operation, we use periodic boundary conditions and assume $X_{n+1,j} = X_{1j}$. With this assumption, $D_v$ is an $n^2 \times n^2$ matrix

$$D_v = \begin{bmatrix} 0 & \cdots & D \\ \vdots & \ddots & \vdots \\ D & \cdots & 0 \end{bmatrix},$$

with all blocks of size $n \times n$ and $D$ defined as the $n \times n$ matrix

$$D = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -1 \\ -1 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}.$$

Note that $D_v$ is a BCCB matrix: $D_v = T(E)$ where $E$ is the $n \times n$ matrix

$$E = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ -1 & 0 & \cdots & 0 \end{bmatrix}.$$ (4)

(The matrix $E$ is zero, except for the elements 1, $-1$ in the first column.)

The last term $\|D_h x\|^2$ in (1) is a penalty on the horizontal differences in the image: $D_h$ is defined by

$$\|D_h x\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} (X_{ij} - X_{i,j+1})^2.$$

We again use periodic boundary conditions and define $X_{i,n+1} = X_{i1}$. This gives

$$D_h = \begin{bmatrix} I & -I & 0 & \cdots & 0 & 0 & 0 \\ 0 & I & -I & \cdots & 0 & 0 & 0 \\ 0 & 0 & I & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & I & -I & 0 \\ 0 & 0 & 0 & \cdots & 0 & I & -I \\ -I & 0 & 0 & \cdots & 0 & 0 & I \end{bmatrix}.$$

This is a BCCB matrix: $D_h = T(E^T)$ with $E$ the matrix defined in (4).
Two-dimensional discrete Fourier transform. The two-dimensional DFT of a complex $n \times n$ matrix $U$ is the complex $n \times n$ matrix
\begin{equation}
V = WUW, 
\end{equation}
where $W$ has the same meaning as in the lectures and homework (lecture 3, page 33). The inverse two-dimensional DFT maps an $n \times n$-matrix $V$ to the matrix
\begin{equation}
U = \frac{1}{n^2}W^HVW^H. 
\end{equation}
In MATLAB the two-dimensional DFT and its inverse are implemented in the functions \texttt{fft2} and \texttt{ifft2}. We use $V = \texttt{fft2}(U)$ to evaluate (5) and $U = \texttt{ifft2}(V)$ to evaluate (6). The complexity is roughly $n^2 \log n$.

The two-dimensional DFT can also be interpreted as a matrix-vector product. Suppose $u$ and $v$ are the matrices $U$ and $V$ converted to $n^2$-vectors (in column-major order). Then the relation $V = WUW$ can be written as
\begin{equation}
v = \tilde{W}u, \quad \tilde{W} = \begin{bmatrix}
W_{11}W & W_{12}W & \cdots & W_{1n}W \\
W_{21}W & W_{22}W & \cdots & W_{2n}W \\
\vdots & \vdots & \ddots & \vdots \\
W_{n1}W & W_{n2}W & \cdots & W_{nn}W
\end{bmatrix}. 
\end{equation}
The matrix $\tilde{W}$ is $n^2 \times n^2$ and has $n$ block rows and columns. The $i,j$ block of $\tilde{W}$ is $W_{ij}W$ where $W_{ij}$ is the $i,j$ element of $W$. (The operation that constructs $\tilde{W}$ from $W$ is known as a Kronecker product and written $\tilde{W} = W \otimes W$.) Using (7) and the property $W^HW = (1/n)I$ one can show that
\begin{equation}
\tilde{W}^H\tilde{W} = n^2I.
\end{equation}
Therefore $\tilde{W}^{-1} = (1/n^2)\tilde{W}^H$ and $\tilde{W}\tilde{W}^H = n^2I$.

It is important to keep in mind that we never need the matrix $\tilde{W}$ explicitly, since we use the \texttt{fft2} and \texttt{ifft2} functions for the matrix-vector multiplications with $\tilde{W}$. The product $v = \tilde{W}u$ can be evaluated by combining \texttt{fft2} and the \texttt{reshape} function:
\begin{eqnarray*}
v &=& \texttt{reshape( fft2( reshape( u, n, n ) ), n^2, 1) ;}
\end{eqnarray*}
Similarly, the matrix-vector product $u = \tilde{W}^{-1}v = (1/n^2)\tilde{W}^Hv$, can be computed as
\begin{eqnarray*}
u &=& \texttt{reshape( ifft2( reshape( v, n, n ) ), n^2, 1) ;}
\end{eqnarray*}
Finally, we need the two-dimensional counterpart of the factorization result for circulant matrices proved in homework 4. The result is as follows. Suppose $U$ is an $n \times n$ matrix and $u$ is the corresponding $n^2$-vector (with the elements of $U$ in column-major order). Then the $n^2 \times n^2$ BCCB matrix $T(U)$ can be factored as
\begin{equation}
T(U) = \frac{1}{n^2}\tilde{W}^H\texttt{diag}(\tilde{W}u)\tilde{W}. 
\end{equation}
Assignment.

(a) The normal equations for (1) are
\[(A^T A + \lambda D_v^T D_v + \lambda D_h^T D_h)x = A^T y.\]
Assume \(A, D_v, D_h\) are BCCB matrices
\[A = T(B), \quad D_v = T(E), \quad D_h = T(E^T),\]
where \(B\) is a given \(n \times n\) matrix and \(E\) is the \(n \times n\) matrix defined in (4). Use the factorization (8) to derive a fast algorithm (with order \(n^2 \log n\) complexity) for solving the normal equations. Implement the fast algorithm in MATLAB or Octave.

(b) Download the file deblur.mat on the course website and load it in MATLAB or Octave (load deblur). The file contains two matrices \(Y\) and \(B\). The matrix \(Y\) is the blurred image and can be displayed using the command imshow(Y). The matrix \(B\) defines the blurring matrix \(A = T(B)\). (The image is from the USC-SIPI Image Database at [http://sipi.usc.edu/database](http://sipi.usc.edu/database).)
Test your deblurring code with several values of \(\lambda\), plot the reconstructed image for each value (using imshow(X)), and determine the value of \(\lambda\) that gives the best result (visually, in your judgment). It is best to search for a good value of \(\lambda\) over a large range, by using a series of values evenly spaced on a logarithmic scale, for example, \(\lambda = 10^{-6}, 10^{-5}, 10^{-4}, \ldots\).

2. Least squares fit of piecewise-polynomial function. This exercise is on the piecewise-polynomial fitting problem of lecture 10, page 13, and page 208 of the textbook. Download the file splinefit.m and run it in MATLAB/Octave to create two arrays \(t\) and \(y\) of length 100. The vector \(t\) contains 100 points in the interval \([-1, 1]\). The first 50 points are in \([-1, 0]\), the last 50 in \([0, 1]\). You are asked to find two cubic polynomials
\[p(x) = \theta_1 + \theta_2 x + \theta_3 x^2 + \theta_4 x^3, \quad q(x) = \theta_5 + \theta_6 x + \theta_7 x^2 + \theta_8 x^3,\]
that give the best least squares fit on the intervals \([-1, 0]\) and \([0, 1]\), respectively, and satisfy the continuity constraints \(p(0) = q(0), p'(0) = q'(0)\):
\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{50} (p(t_i) - y_i)^2 + \sum_{i=51}^{100} (q(t_i) - y_i)^2 \\
\text{subject to} & \quad p(0) = q(0), \quad p'(0) = q'(0).
\end{align*}
\]
The variables are the coefficients \(\theta_1, \ldots, \theta_8\). Formulate this as a constrained least squares problem
\[
\begin{align*}
\text{minimize} & \quad \|Ax - b\|^2 \\
\text{subject to} & \quad Cx = d
\end{align*}
\]
and solve it using the method of page 19 of lecture 10. In MATLAB/Octave:
\[
\begin{align*}
sol &= [ A'*A, C'; C, zeros(p,p) ] \backslash [ A'*b; d ]; \\
x &= sol(1:n);
\end{align*}
\]