Problem Set 8

Posted 26 April 2014. Due by 5pm on Friday, 2 May 2014.

Please include your MATLAB code for all exercises.

1. [25 points: (a)=5 points; (b)=5 points; (c)=7 points; (d)=8 points]

The matrix equation \( Ax = b \) given by

\[
\begin{bmatrix}
1 & 50 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} = \begin{bmatrix}
2 \\
0
\end{bmatrix}
\]

has infinitely many solutions; the one with smallest norm is

\[
x_+ = A^+ b = \begin{bmatrix}
2/2501 \\
100/2501
\end{bmatrix} \approx \begin{bmatrix}
0.0008 \\
0.0400
\end{bmatrix}
\]

For all \( \varepsilon \neq 0 \), the perturbed equation \( A_\varepsilon x_\varepsilon = b_\varepsilon \),

\[
\begin{bmatrix}
1 & 50 \\
0 & \varepsilon
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} = \begin{bmatrix}
2 \\
\varepsilon
\end{bmatrix}
\]

has a unique solution. Since we are only making small changes (‘noise’) to the problem, we might expect to get a solution \( x_\varepsilon \) that is close to \( x_+ \). This problem explores whether that is the case.

(a) For any \( \varepsilon \neq 0 \), solve \( A_\varepsilon x_\varepsilon = b_\varepsilon \) for the unique solution \( x_\varepsilon \).

How does it compare to the best solution to the unperturbed problem, \( x_+ \)?

(b) Let \( \varepsilon = 0.01 \). In MATLAB, compute the SVD

\[
A_\varepsilon = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^*;
\]

and construct the truncated SVD solution for the perturbed problem:

\[
x_{SVD} = \frac{1}{\sigma_1} v_1^* u_1^* b_\varepsilon.
\]

How does this ‘solution’ \( x_{SVD} \) to the perturbed problem compare to \( x_\varepsilon \) and \( x_+ \)?

(c) The truncated SVD approach is expensive for large problems, since it requires computation of the singular value decomposition. It can be cheaper to solve the regularized least-squares problem

\[
\min_{x \in \mathbb{C}^2} \| A_\varepsilon x - b_\varepsilon \|^2 + \lambda^2 \| x \|^2
\]

for various values of \( \lambda \). As we saw in class, the optimal \( x_\lambda \) satisfies the least squares problem

\[
\min_{x \in \mathbb{C}^2} \left\| \begin{bmatrix} A_\varepsilon & \lambda I \end{bmatrix} x - \begin{bmatrix} b_\varepsilon \\ 0 \end{bmatrix} \right\|,
\]

where

\[
A_\lambda = \begin{bmatrix} A_\varepsilon & \lambda I \end{bmatrix} \in \mathbb{C}^{4 \times 2}, \quad \hat{b} = \begin{bmatrix} b_\varepsilon \\ 0 \end{bmatrix} \in \mathbb{C}^4.
\]

You can solve this least squares problem with \( x_{\lambda} = \text{Alambda} \backslash \text{bhat} \) in MATLAB.

Solve the regularized problem for 100 values of \( \lambda \) logarithmically spaced between \( \lambda = 10^{-5} \) and \( \lambda = 10^2 \) (try \( \lambda = \logspace(-5,2,100) \)). Produce one \text{loglog} plot that shows, for each value of \( \lambda \), a dot corresponding to \( \| A_\varepsilon x_\lambda - b_\varepsilon \| \) on the horizontal axis, and \( \| x_\lambda \| \) on the vertical axis. Your plot should show a prominent ‘L’ shape, as we shall discuss in class.

(d) Present values of \( x_\lambda \) for three values of \( \lambda \) corresponding to (i) a point well above the right-angle in the L; (ii) a point at the angle of the L; (iii) a point after the angle of the L. Which of these most closely agrees with the original unperturbed solution \( x_+ \)?
2. [75 points + 5 bonus: (a),(b),(c)=5 points each; (d)=15; (e)=25; (f)=20; (g)=5 bonus]

For this problem, you will use regularization to decode some UPC barcodes. First, we explain how UPC barcodes work (for details, see http://en.wikipedia.org/wiki/Universal_Product_Code). The UPC system encodes 12 digits through a series of 59 alternating black and white bars of varying widths. Each bar, be it black (b) or white (w), has one of four widths: either 1, 2, 3, or 4 units wide. Here is how they encode information.

<table>
<thead>
<tr>
<th>colors</th>
<th>number of bars</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bwb</td>
<td>three bars of width 1</td>
<td>start code</td>
</tr>
<tr>
<td>wwb</td>
<td>four bars of total width 7</td>
<td>first digit</td>
</tr>
<tr>
<td>wwb</td>
<td>four bars of total width 7</td>
<td>second digit</td>
</tr>
<tr>
<td>wwb</td>
<td>four bars of total width 7</td>
<td>third digit</td>
</tr>
<tr>
<td>wwb</td>
<td>four bars of total width 7</td>
<td>fourth digit</td>
</tr>
<tr>
<td>wwb</td>
<td>four bars of total width 7</td>
<td>fifth digit</td>
</tr>
<tr>
<td>wwb</td>
<td>four bars of total width 7</td>
<td>sixth digit</td>
</tr>
<tr>
<td>wwbwb</td>
<td>five bars of width 1</td>
<td>middle code</td>
</tr>
<tr>
<td>bwbwb</td>
<td>four bars of total width 7</td>
<td>seventh digit</td>
</tr>
<tr>
<td>bwbwb</td>
<td>four bars of total width 7</td>
<td>eighth digit</td>
</tr>
<tr>
<td>bwbwb</td>
<td>four bars of total width 7</td>
<td>ninth digit</td>
</tr>
<tr>
<td>bwbwb</td>
<td>four bars of total width 7</td>
<td>tenth digit</td>
</tr>
<tr>
<td>bwbwb</td>
<td>four bars of total width 7</td>
<td>eleventh digit</td>
</tr>
<tr>
<td>bwbwb</td>
<td>four bars of total width 7</td>
<td>twelfth digit</td>
</tr>
<tr>
<td>bwb</td>
<td>three bars of width 1</td>
<td>end code</td>
</tr>
</tbody>
</table>

Each of the 12 digits is encoded with four bars (two white, two black); the widths of these bars will differ, but the sum of the width of all four bars is always 7, so that all UPC codes have the same width (95 units wide). Each digit (0–9) corresponds to a particular pattern of bar widths, according to the following table. (For reasons beyond our interest, two different patterns can be used for each digit.)

<table>
<thead>
<tr>
<th>digit</th>
<th>pattern 1</th>
<th>pattern 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3–2–1–1</td>
<td>1–1–2–3</td>
</tr>
<tr>
<td>1</td>
<td>2–2–2–1</td>
<td>1–2–2–2</td>
</tr>
<tr>
<td>2</td>
<td>2–1–2–2</td>
<td>1–1–2–2</td>
</tr>
<tr>
<td>3</td>
<td>1–4–1–1</td>
<td>2–2–1–2</td>
</tr>
<tr>
<td>4</td>
<td>1–1–3–2</td>
<td>1–3–1–1</td>
</tr>
<tr>
<td>5</td>
<td>1–2–3–1</td>
<td>4–1–1–1</td>
</tr>
<tr>
<td>6</td>
<td>1–1–1–4</td>
<td>2–3–1–1</td>
</tr>
<tr>
<td>7</td>
<td>1–3–1–2</td>
<td>1–3–2–1</td>
</tr>
<tr>
<td>8</td>
<td>1–2–1–3</td>
<td>3–1–2–1</td>
</tr>
<tr>
<td>9</td>
<td>3–1–1–2</td>
<td>2–1–1–3</td>
</tr>
</tbody>
</table>

For example, if the four bars (read left-to-right) have the pattern 3–2–1–1 or 1–1–2–3, the corresponding UPC digit is 0. To check that you understand the system, try decoding the UPC below (for a can of Coke). To start you out, we give the code for the first three digits. It can be tricky to judge the widths – you can appreciate the accuracy of optical scanners!

<table>
<thead>
<tr>
<th>color</th>
<th>width</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>w</td>
<td>start code</td>
</tr>
<tr>
<td>w</td>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>w</td>
<td>1</td>
</tr>
<tr>
<td>w</td>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>b</td>
<td>w</td>
<td>3</td>
</tr>
<tr>
<td>w</td>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>w</td>
<td>b</td>
<td>9</td>
</tr>
</tbody>
</table>

We want to simulate the reading of this UPC code by an optical scanner, e.g., in a supermarket checkout line. The barcode is represented mathematically as a function \( f(t) \) that takes the values zero and one: zero corresponds to white bars, one corresponds to black bars. The function corresponding to the Coke bar code is shown below.
The optical scanner can only acquire a function \( b \) (sampled at discrete points in the vector, \( b \in \mathbb{C}^n \)), a blurred version of \( f \), where we assume the blurring kernel is \( h(s, t) = e^{(t-s)^2/z^2} \) for \( z = 0.01 \). The Coke UPC function, blurred by this kernel, is shown below. From this blurred function, it would be difficult to determine the widths of the bars, and hence to interpret the barcode. We shall try to improve the situation by solving the inverse problem \( Af = b \) for the vector \( f \) that samples the function \( f(t) \) at the points \( t_k = (k - 1/2)/n \).

The function \([A,b,b\text{noise},f\text{true}] = \text{coke_upc} \) from the class website generates, for \( n = 500 \): the blurring matrix for the specified kernel \( A \in \mathbb{C}^{500 \times 500} \); the blurred function sampled at 500 points, \( b \in \mathbb{C}^{500} \); the blurred vector with 5\% random noise, \( b\text{noise} \in \mathbb{C}^{500} \); the exact bar code solution \( f\text{true} \).

(We generated \( b \) as \( b = A*f\text{true} \).)

Some of these questions are intentionally open-ended. You will be graded primarily on the thoroughness of your experiments, rather than for recovering a particular value for the barcodes. Include plenty of plots and label what they show, and describe what you learn from them.

(a) Produce a plot showing the vector \( f\text{inv} = A^{-1}b \) one obtains by directly inverting (\( f\text{inv} = A\backslash b \)).

To plot with the same aspect ratio shown above, you can use the following commands:

\[
\begin{align*}
    t &= (1:n-1)/n; \\
    \text{figure, axes('position', [.075 .1 .85 .2]);} \\
    \text{plot(t,ftrue,'k-','linewidth',1); hold on} \\
    \text{plot(t,ftrue,'r-','linewidth',1)}
\end{align*}
\]

(b) Repeat part (a), but now using the noisy, blurred vector: \( f\text{noise} = A^{-1}b\text{noise} \).

Plot your recovered barcode \( f\text{noise} \) along with the true value \( f\text{true} \).

(c) Now see if you can do any better using the truncated singular value decomposition.

If \( A = \sum_{j=1}^n \sigma_j u_j v_j^* \), then the truncated SVD solution from the leading rank-\( r \) part of \( A \) is:

\[
f_r = \left( \sum_{j=1}^r \frac{1}{\sigma_j} v_j u_j^* \right) b = \sum_{j=1}^r \frac{1}{\sigma_j} v_j (u_j^* b).\]

Experiment with different several values of \( r \). (Start with \( r = 150 \), say.) Describe how varying \( r \) affects the quality of the solution. Is there any difference if you use the exact blurred vector \( b \), versus the noisy blurred vector \( b\text{noise} \)? Illustrate your experiments by producing plots like the
ones you constructed in parts (a) and (b). Are any of these recoveries good enough that you can estimate the value of the barcode?

(d) As an alternative to the truncated SVD, we shall explore the solutions obtained from the regularized least squares problem

\[
\min_{f \in \mathbb{C}^n} \| Af - b \|^2 + \lambda^2 \| f \|^2
\]

for various \( \lambda \). Recall that you can find the optimal value \( f_\lambda \) by solving the least squares problem

\[
\min_{f \in \mathbb{C}^n} \left\| \begin{bmatrix} A & \lambda I \end{bmatrix} f - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|.
\]

(i) Using the value \( b_{\text{noise}} \), create the L-curve plot described in Problem 1 using parameter values \( \lambda = 10^{-8}, \ldots, 10^0 \). (Use \texttt{logspace(-8,0,100)} to generate 100 logarithmically-spaced values of \( \lambda \) in this range.) Recall that the L-curve is a loglog plot with \( \| Af_\lambda - b_{\text{noise}} \| \) on the horizontal axis and \( \| f_\lambda \| \) on the vertical axis.

(ii) Pick a \( \lambda \) value corresponding to the right-angle in the L-curve, and plot the recovered solution \( f_\lambda \) as done in earlier parts of the problem. Feel free to show plots for several different \( \lambda \) values that vary over one or two orders of magnitude. Also show the \( f_\lambda \) you get for the same value of \( \lambda \) if you instead use the true data \( b \) instead of \( b_{\text{noise}} \).

(e) Using your recovered \( f_r \) and/or \( f_\lambda \) from parts (c) and (d), attempt to reconstruct the barcode for the can of Coke. This is not trivial – just do your best, using insight from the structure of the UPC code to help.

(f) The function \([b, b_{\text{noise}}] = \text{mystery}\_\text{upc}\) generates the blurred \( b \) and blurred, noisy \( b_{\text{noise}} \) barcodes for a mystery product (again with \( n=500 \)). Use the techniques in parts (c) and (d) to attempt to recover the mystery barcode. Show samples of the results from your various attempts.

(g) \textbf{5 point bonus: Please do not discuss your answer with your classmates.}

What is the product described by the mystery UPC? (Once you have found the correct UPC, a search on Amazon will turn up the product.)