CMDA 3606 · MATHEMATICAL MODELING II

Problem Set 8 · Solutions

Posted 22 October 2022. Due at 11:59pm on Thursday, 27 October 2022.

1. [30 points: 6 points per part]
   Consider the 1×2 matrix $\mathbf{A}$ and "vector" $\mathbf{b}$ given by

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \end{bmatrix} \in \mathbb{R}^{1\times 2}, \qquad \mathbf{b} = \begin{bmatrix} 1 \end{bmatrix} \in \mathbb{R}^{1}.$$

   The linear system $\mathbf{Ax} = \mathbf{b}$ has infinitely many solutions: Any

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$$

   that satisfies $x_1 + x_2 = 1$ is a solution.

   (a) Use the SVD of $\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$ to compute the pseudoinverse

$$\mathbf{A}^+ = \frac{1}{\sigma_1} \mathbf{v}_1 \mathbf{u}_1^T$$

   of this rank $r = 1$ matrix, and compute the minimum norm solution $\mathbf{x}_+ = \mathbf{A}^+\mathbf{b}$ to $\mathbf{Ax} = \mathbf{b}$. What is $\|\mathbf{x}_+\|^2$?

This $\mathbf{x}_+$ is an exact solution to the system $\mathbf{Ax} = \mathbf{b}$ and it has the smallest norm of all solutions. Suppose we want to use regularization to find a vector $\mathbf{x}$ of smaller norm that was only an approximate solution of $\mathbf{Ax} = \mathbf{b}$. We can do this using Tikhonov regularization. For a fixed $\lambda > 0$, we will find the unique $\mathbf{x} \in \mathbb{R}^2$ that minimizes

$$\|\mathbf{b} - \mathbf{Ax}\|^2 + \lambda^2 \|\mathbf{x}\|^2.$$

(In lectures we will focus on the $m \geq n$ case, but the formulas work $m < n$ as well.) We define

$$\mathbf{A}_\lambda = \begin{bmatrix} \mathbf{A} \\ \lambda \mathbf{I} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ \lambda & 0 \\ 0 & \lambda \end{bmatrix}, \qquad \widehat{\mathbf{b}} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

The minimizer to the regularized problem, denoted $\mathbf{x}_\lambda$, can be computed by solving a standard least squares problem involving the augmented matrices:

$$\min_{\mathbf{x}\in\mathbb{R}^2} \|\widehat{\mathbf{b}} - \mathbf{A}_\lambda \mathbf{x}\|.$$

Parts (b), (c), and (d) explore three ways to solve the regularized problem; all should arrive at the same solution. *You must show your work to get credit.*

   (b) Form the solution $\mathbf{x}_\lambda$ to the regularized equation by solving the normal equations

$$(\mathbf{A}_\lambda^T \mathbf{A}_\lambda)\mathbf{x}_\lambda = \mathbf{A}_\lambda^T \widehat{\mathbf{b}}.$$

   Compute (by hand) $(\mathbf{A}_\lambda^T \mathbf{A}_\lambda)^{-1}$ and then form

$$\mathbf{x}_\lambda = (\mathbf{A}_\lambda^T \mathbf{A}_\lambda)^{-1} (\mathbf{A}_\lambda^T \widehat{\mathbf{b}}).$$

   (c) Since $\mathbf{A}$ is rank-1, the solution $\mathbf{x}_\lambda$ satisfies an easy formula in terms of the SVD of $\mathbf{A}$:

$$\mathbf{x}_\lambda = \frac{\sigma_1}{\sigma_1^2 + \lambda^2} (\mathbf{u}_1^T \mathbf{b}) \mathbf{v}_1.$$

   Compute $\mathbf{x}_\lambda$ using this formula.

(d) We should also be able to compute $\mathbf{x}_\lambda$ using multivariable calculus, which might seem entirely different from the linear algebra approach we take in the lectures.

- Define $f(x_1, x_2) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|^2 + \lambda^2\|\mathbf{x}\|^2$ for the particular $\mathbf{A}$ and $\mathbf{b}$ in this problem, where $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.
- Work out a simple formula for $f(x_1, x_2)$ involving x_1, x_2, and λ.
- Compute the partial derivatives $\partial f/\partial x_1$ and $\partial f/\partial x_2$ (holding λ constant).
- Set these two partial derivatives to zero simultaneously (to minimize f), showing that you can arrange the two resulting equations in the form
$$\mathbf{H}\mathbf{x} = \mathbf{c}$$
for a matrix $\mathbf{H} \in \mathbb{R}^{2\times 2}$ and $\mathbf{c} \in \mathbb{R}^2$ that you should state ($\mathbf{H}$ and/or $\mathbf{c}$ could contain the variable λ).
- Solve $\mathbf{H}\mathbf{x} = \mathbf{c}$ for the solution, $\mathbf{x}_\lambda$, minimizes $f(x_1, x_2)$.

(e) Now consider Tikhonov regularization for general $\mathbf{A} \in \mathbb{R}^{m\times n}$. Does calculus provide the same equations that linear algebra gave us? Explore this question with the following exercise.

- Define $f(\mathbf{x}) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|^2 + \lambda^2\|\mathbf{x}\|^2$. Multiply out the inner products in
$$f(\mathbf{x}) = (\mathbf{b} - \mathbf{A}\mathbf{x})^T(\mathbf{b} - \mathbf{A}\mathbf{x}) + \lambda^2\mathbf{x}^T\mathbf{x}$$
to get an expression for $f(\mathbf{x})$ involving simple terms like $\mathbf{b}^T\mathbf{b}$.
- Recall that the gradient is the vector of partial derivatives:
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \partial f/\partial x_1(\mathbf{x}) \\ \vdots \\ \partial f/\partial x_n(\mathbf{x}) \end{bmatrix}.$$
Compute $\nabla f(\mathbf{x})$ for the specific $f(\mathbf{x})$ you have just computed.
Hint: Do not compute the individual partial derivatives; everything can be done using gradients, if you recall these rules of multivariable calculus: the gradient is a linear operator, so
$$\nabla(f(\mathbf{x}) + cg(\mathbf{x})) = \nabla f(\mathbf{x}) + c\nabla g(\mathbf{x}),$$
for constant $c \in \mathbb{R}$ and, for any matrix $\mathbf{B} \in \mathbb{R}^{n\times m}$ and symmetric $\mathbf{S} \in \mathbb{R}^{n\times n}$,
$$\nabla(c) = \mathbf{0}, \qquad \nabla(\mathbf{x}^T\mathbf{B}\mathbf{y}) = \mathbf{B}\mathbf{y}, \qquad \nabla(\mathbf{x}^T\mathbf{S}\mathbf{x}) = 2\mathbf{S}\mathbf{x}.$$
- To minimize f, set $\nabla f(\mathbf{x}) = \mathbf{0}$ and show why this implies
$$(\mathbf{A}^T\mathbf{A} + \lambda^2\mathbf{I})\mathbf{x} = \mathbf{A}^T\mathbf{b}.$$
- Show that this last equation is equivalent to
$$\mathbf{A}_\lambda^T\mathbf{A}_\lambda\mathbf{x} = \mathbf{A}_\lambda^T\widehat{\mathbf{b}},$$
for the usual definition of $\mathbf{A}_\lambda$ and $\widehat{\mathbf{b}}$. (Thus, calculus has taken us to the same equation we obtained for $\mathbf{x}_\lambda$ via linear algebra.)

Solution.

(a) The SVD of $\mathbf{A}$ is simply
$$\mathbf{A} = \sqrt{2}\begin{bmatrix} 1 \end{bmatrix}\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$
and so the pseudoinverse is
$$\mathbf{A}^+ = \frac{1}{\sqrt{2}}\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}\begin{bmatrix} 1 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}.$$
Thus the minimum norm solution of the linear system is
$$\mathbf{x}_+ = \mathbf{A}^+\mathbf{b} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}\begin{bmatrix} 1 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix},$$
with $\|\mathbf{x}_+\|^2 = 1/2$.

(b) To solve the normal equations, first form

$$\mathbf{A}_\lambda^T \mathbf{A}_\lambda = \begin{bmatrix} 1+\lambda^2 & 1 \\ 1 & 1+\lambda^2 \end{bmatrix}, \qquad \mathbf{A}_\lambda^T \widehat{\mathbf{b}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Compute

$$\mathbf{A}_\lambda^{-1} = \begin{bmatrix} \frac{\lambda^2+1}{\lambda^4+2\lambda^2} & \frac{-1}{\lambda^4+2\lambda^2} \\ \frac{-1}{\lambda^4+2\lambda^2} & \frac{\lambda^2+1}{\lambda^4+2\lambda^2} \end{bmatrix}.$$

Thus

$$\mathbf{x}_\lambda = (\mathbf{A}_\lambda^T \mathbf{A}_\lambda)^{-1} \mathbf{A}_\lambda^T \widehat{\mathbf{b}} = \frac{1}{\lambda^2+2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

(c) Using ingredients from part (a), we can compute

$$\mathbf{x}_\lambda = \frac{\sqrt{2}}{(\sqrt{2})^2+\lambda^2} \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = \frac{1}{\lambda^2+2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

(d) Write out

$$\|\mathbf{b}-\mathbf{A}\mathbf{x}\|^2 = \big(1-(x_1+x_2)\big)^2 = 1-2(x_1+x_2)+x_1^2+2x_1x_2+x_2^2, \qquad \lambda^2\|\mathbf{x}\|^2 = \lambda^2(x_1^2+x_2^2),$$

and so

$$f(x_1,x_2) = 1-2(x_1+x_2)+x_1^2+2x_1x_2+x_2^2+\lambda^2(x_1^2+x_2^2).$$

Differentiate to obtain

$$\frac{\partial f}{\partial x_1}(x_1,x_2) = -2+2x_1+2x_2+2\lambda^2 x_1$$

$$\frac{\partial f}{\partial x_2}(x_1,x_2) = -2+2x_1+2x_2+2\lambda^2 x_2$$

Rearrange to get the system $\mathbf{H}\mathbf{x}=\mathbf{c}$, which can take the form

$$\begin{bmatrix} 2+2\lambda^2 & 2 \\ 2 & 2+2\lambda^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix},$$

or (cancelling a common 2):

$$\begin{bmatrix} 1+\lambda^2 & 1 \\ 1 & 1+\lambda^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Notice that this last equation is exactly the system $\mathbf{A}_\lambda^T \mathbf{A}_\lambda \mathbf{x} = \mathbf{A}_\lambda^T \widehat{\mathbf{b}}$, which we can solve again (it is not necessary to work out explicitly again) that

$$\mathbf{x}_\lambda = \mathbf{H}^{-1}\mathbf{c} = (\mathbf{A}_\lambda^T \mathbf{A}_\lambda)^{-1} \mathbf{A}_\lambda^T \widehat{\mathbf{b}} = \frac{1}{\lambda^2+2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

(e) Compute

$$f(\mathbf{x}) = \mathbf{b}^T\mathbf{b} - 2\mathbf{x}^T\mathbf{A}^T\mathbf{b} + \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} + \lambda^2\mathbf{x}^T\mathbf{x}$$

and thus, using the rules provided,

$$\nabla f(\mathbf{x}) = \mathbf{0} - 2\mathbf{A}^T\mathbf{b} + 2\mathbf{A}^T\mathbf{A}\mathbf{x} + 2\lambda^2\mathbf{x}.$$

Set this equal to zero, cancel the factor of two, and rearrange to obtain

$$(\mathbf{A}^T\mathbf{A}+\lambda^2)\mathbf{x} = \mathbf{A}^T\mathbf{b}.$$

Finally, note that

$$\mathbf{A}_\lambda^T \mathbf{A}_\lambda = \begin{bmatrix} \mathbf{A}^T & \lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \lambda\mathbf{I} \end{bmatrix} = \mathbf{A}^T\mathbf{A} + \lambda^2\mathbf{I}$$

and

$$\mathbf{A}_\lambda^T \widehat{\mathbf{b}} = \begin{bmatrix} \mathbf{A}^T & \lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} = \mathbf{A}^T\mathbf{b},$$

and so the equation obtained from multivariable calculus recovers the normal equations for the augmented system.

2. [20 points: 8 points each for (a) and (b); 4 points for (c)]

This problem picks up right where Problem 3 on Problem Set 7 left off, still using the Gaussian kernel with $n = 500$ and $z = 0.05$, and the same $\mathbf{f}$ and $\mathbf{b}$ vectors you used on that problem. (We will provide solution code for this earlier problem on Canvas after the late due date for Problem Set 7, or you can use your own code.) On that earlier Problem 3(f), you formed $\mathbf{f}_{\text{rec}} = \mathbf{A}^{-1}\mathbf{b}$, and found that the answer looked nothing like the original function $\mathbf{f}$ from which you formed $\mathbf{b} = \mathbf{A}\mathbf{f}$. Does regularization help?

(a) Recover $\mathbf{f}$ using truncated SVD regularization. For the five values $k = 5, 10, 25, 50$ and 100, form

$$\mathbf{f}_k = \sum_{j=1}^{k} \frac{\mathbf{u}_j^T \mathbf{b}}{\sigma_j} \mathbf{v}_j.$$

To compute $\mathbf{f}_k$ efficiently in Python (using `import scipy.linalg as la`) for a desired k:

```
U,S,Vt = la.svd(A)
fk = (Vt[0:k,:].T)*(1/(S[0:k]))@(U[:,0:k].T@b)
```

Plot your results; produce a separate plot for each value of k.

(b) Recover $\mathbf{f}$ using Tikhonov regularization. For the four values $\lambda = 10^{-6}$, 10^{-2}, 10^{-1}, and 10^{0}, in Python form

$$\texttt{Alam} = \begin{bmatrix} \mathbf{A} \\ \lambda \mathbf{I} \end{bmatrix} \in \mathbb{R}^{1000 \times 500}, \qquad \texttt{bhat} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{1000}.$$

You can then solve for $\mathbf{f}_\lambda$ via

```
flam = la.lstsq(Alam,bhat)[0]
```

Plot the results; produce a separate plot for each value of λ.

Notes: You can create, say, 10^{-6} via `1e-6` in Python. If you have two matrices $\mathbf{C} \in \mathbb{R}^{m_1 \times n}$ and $\mathbf{D} \in \mathbb{R}^{m_2 \times n}$ that both have n columns, then you can create the augmented matrix

$$\begin{bmatrix} \mathbf{C} \\ \mathbf{D} \end{bmatrix} \in \mathbb{R}^{(m_1+m_2) \times n}$$

with the command `np.concatenate((C,D))`.

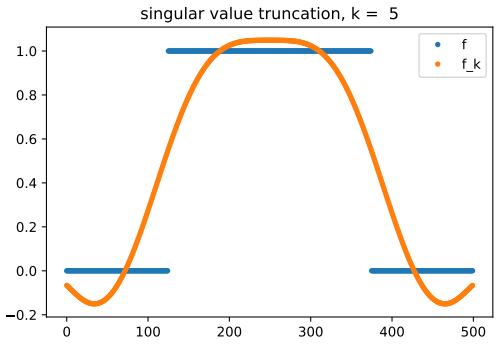
(c) Analyze your results from parts (a) and (b).

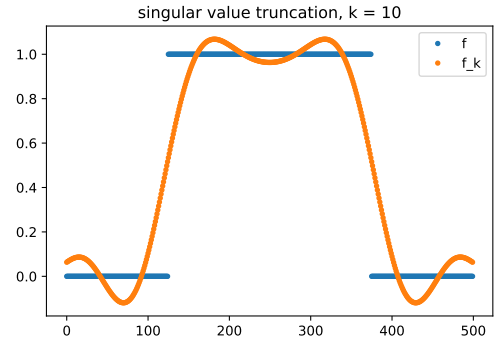
- Which value of k gave you the best results? (This is a qualitative judgement.)
- What happens if k is too large?
- Which value of λ gave you the best results? (This too is a qualitative judgement.)
- What happens if λ is too large?
- How do your best $\mathbf{f}_k$ and $\mathbf{f}_\lambda$ compare to the $\mathbf{f}_{\text{rec}}$ you computed in Problem 3(f) of Problem Set 7?

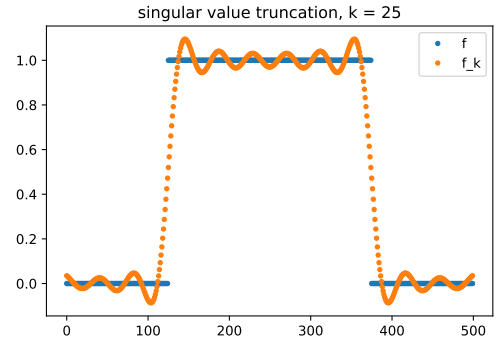
Note: You might naturally want to form an "L curve" to analyze Tikhonov regularization for this example. Because of the nature of $\mathbf{A}$, when we use the exact $\mathbf{b} = \mathbf{A}\mathbf{f}$, most solutions $\mathbf{f}$ (even some rather bad ones!) give us very small values of $\|\mathbf{b} - \mathbf{A}\mathbf{f}\|$, spoiling the usual L shape. If you replace $\mathbf{b}$ by a noisy version, e.g., `b + 1e-8*np.random.randn(n)`, you can get a more natural L curve. You do not need to make such a plot for this problem; you will have the chance to form an L curve in Problem 3. . . .

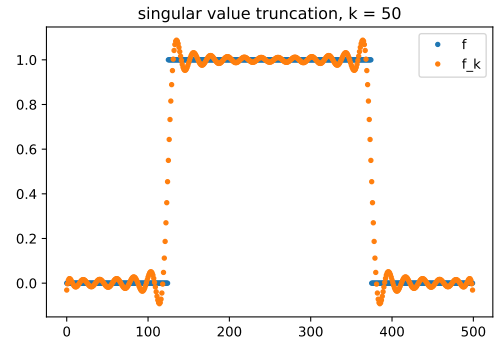
Solution.

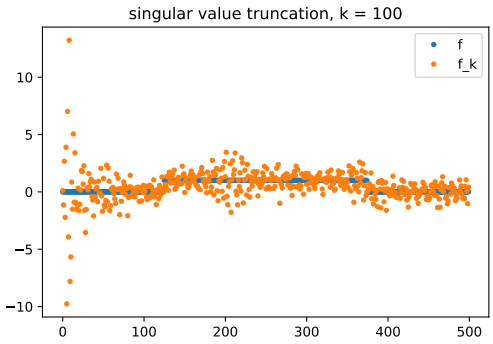
- The following plots show the solutions $\mathbf{f}_k$ obtained with truncated SVD regularization.


singular value truncation, k = 5
f
f_k
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
100
200
300
400
500

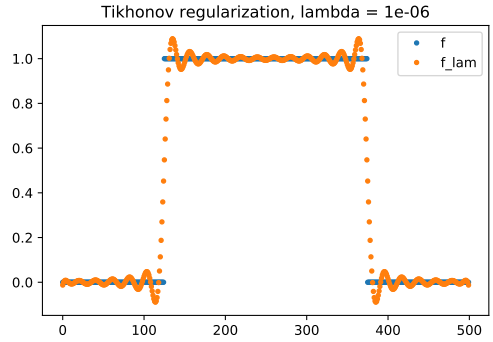

singular value truncation, k = 10
f
f_k
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500

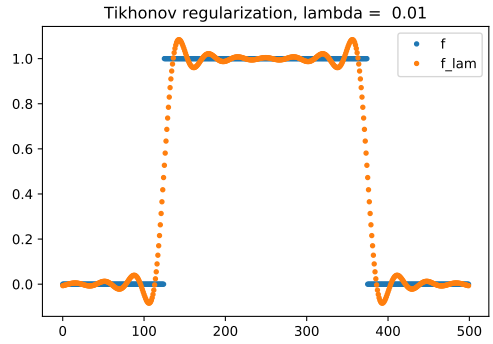

singular value truncation, k = 25
f
f_k
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500

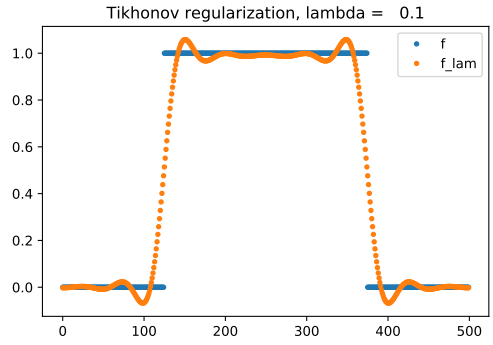

singular value truncation, k = 50
f
f_k
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500

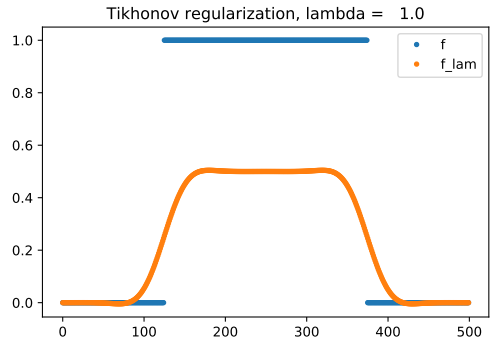


- The following plots show the solutions $\mathbf{f}_\lambda$ obtained with Tikhonov regularization.

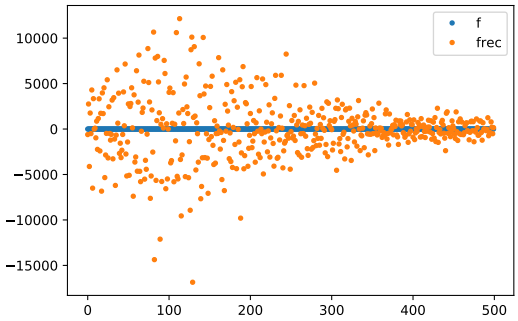








- For reference, recall that the solution obtained in Problem 3(f) on Problem Set 7 is as follows. (Student answers might have varied slightly but will have been similar in spirit – it is not necessary to include this in the solution to this probelm.)



  - Of these values, $k = 50$ seems best; one could also make a good case for $k = 25$.
  - When k gets too large, we again get "garbage" solutions.
  - Of these values, $\lambda = 10^{-6}$ seems best; $\lambda = 10^{-2}$ is not bad either.
  - If λ is too large, the solution starts converging toward zero.
  - The best approximations $\mathbf{f}_k$ and $\mathbf{f}_\lambda$ are much better than the vector $\mathbf{f}_{\text{rec}}$ recovered in problem 3(f) of the last problem set.

---

3. [50 points: 5 each for (a), (b), (e); 10 each for (c), (d); 15 for (f); +5 bonus for (g)]
   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.

colors	*number of bars*	*description*
bwb	three bars of width 1	start code
wbwb	four bars of total width 7	first digit
wbwb	four bars of total width 7	second digit
wbwb	four bars of total width 7	third digit
wbwb	four bars of total width 7	fourth digit
wbwb	four bars of total width 7	fifth digit
wbwb	four bars of total width 7	sixth digit
wbwbw	five bars of width 1	middle code
bwbw	four bars of total width 7	seventh digit
bwbw	four bars of total width 7	eighth digit
bwbw	four bars of total width 7	ninth digit
bwbw	four bars of total width 7	tenth digit
bwbw	four bars of total width 7	eleventh digit
bwbw	four bars of total width 7	twelfth digit
bwb	three bars of width 1	end code

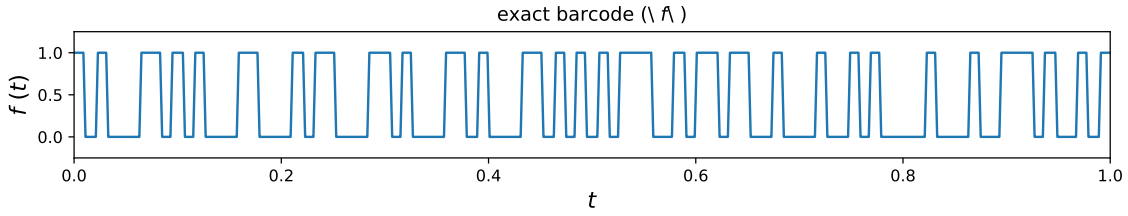
Each of the 12 digits is encoded with four bars (two white, two black); the widths of these bars will differ, but the *sum* of the width of all four bars is always 7, so that all UPC codes have the same width (95 units wide). Each digit (0–9) corresponds to a particular pattern of bar widths, according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

digit	0	1	2	3	4	5	6	7	8	9
pattern 1	3–2–1–1	2–2–2–1	2–1–2–2	1–4–1–1	1–1–3–2	1–2–3–1	1–1–1–4	1–3–1–2	1–2–1–3	3–1–1–2
pattern 2	1–1–2–3	1–2–2–2	2–2–1–2	1–1–4–1	2–3–1–1	1–3–2–1	4–1–1–1	2–1–3–1	3–1–2–1	2–1–1–3

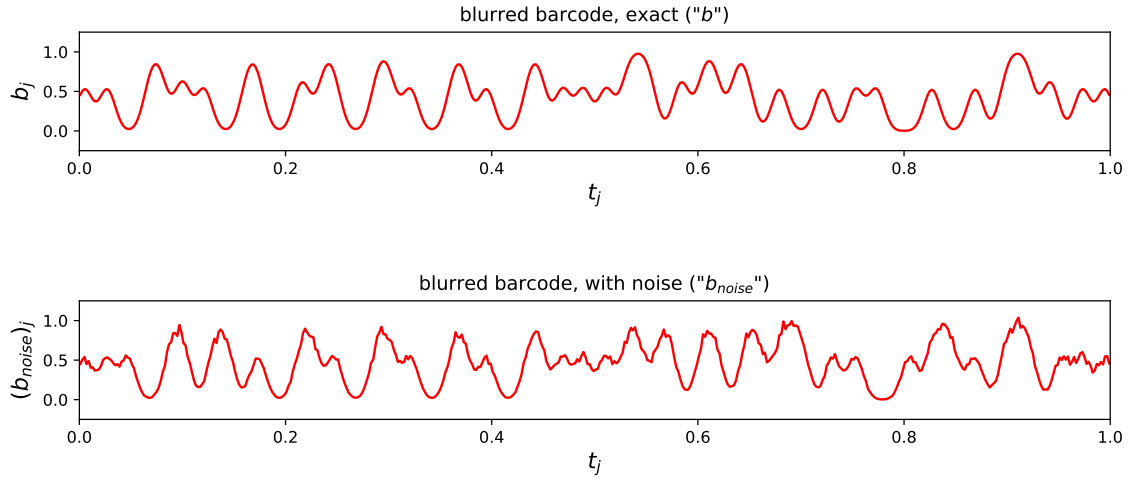
For example, if the four bars (read left-to-right) have the pattern 3–2–1–1 or 1–1–2–3, the corresponding UPC digit is 0. To check that you understand the system, try decoding the UPC below (for a can of Coke). To start you out, we give the code for the first three digits. It can be tricky to judge the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

color	b	w	b	w	b	w	b	w	b	w	b	w	b	w	b
width	1	1	1	3	2	1	1	1	1	3	2	3	1	1	2
meaning	start code			0				4				9			

We want to simulate the reading of this UPC code by an optical scanner, e.g., in a supermarket check-out 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, $\mathbf{b} \in \mathbb{R}^n$), a version of f blurred by the matrix $\mathbf{A}$. 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 $\mathbf{Af} = \mathbf{b}$ for the vector $\mathbf{f}$ that samples the function $f(t)$ at the points $t_k = (k - 1/2)/n$.



The Jupyter notebook `coke_upc.ipynb` posted on Canvas defines a function `coke_upc` with the interface `A,b,bnoise,ftrue = coke_upc()`. This code generates, for $n = 500$: the blurring matrix $\mathbf{A} \in \mathbb{R}^{500\times 500}$ for a particular kernel; the blurred function sampled at 500 points, $\mathbf{b} \in \mathbb{R}^{500}$; the blurred vector with 5% random noise, $\mathbf{b}_{\text{noise}} \in \mathbb{R}^{500}$; the exact bar code solution $\mathbf{f}_{\text{true}}$. (We generated $\mathbf{b}$ as `b = A@ftrue`, then polluted it with random noise. Since different noise is generated each time you call the routine, you could get slightly different answers, but the qualitative results will be the same.)

Some of the following 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 use markdown cells in your Jupyter notebook to describe what you learn from them. Your explanations will count toward your grade.

(a) Produce a plot showing the vector $\mathbf{f}_{\text{rec}}$ one obtains by directly solving $\mathbf{A}\mathbf{f}_{\text{rec}} = \mathbf{b}$ (`frec = la.solve(A,b)`). (First, `from scipy import linalg as la`.) Adapt the plotting commands from the `coke_upc.ipynb` Jupyter notebook, to produce an elongated plot like the ones shown above.

(b) Repeat part (a), but now using the noisy, blurred vector: solve $\mathbf{A}\mathbf{f}_{\text{noise}} = \mathbf{b}_{\text{noise}}$ for $\mathbf{f}_{\text{noise}}$.
Plot your recovered barcode $\mathbf{f}_{\text{noise}}$. Does this resemble $\mathbf{f}_{\text{true}}$?

(c) Now see if you can do any better using the truncated singular value decomposition.
If $\mathbf{A} = \sum_{j=1}^{n} \sigma_j \mathbf{u}_j \mathbf{v}_j^T$, then the truncated SVD solution from the leading rank-k part of $\mathbf{A}$ is:

$$\mathbf{f}_k = \Big(\sum_{j=1}^{k} \frac{1}{\sigma_j} \mathbf{v}_j \mathbf{u}_j^T \Big) \mathbf{b} = \sum_{j=1}^{k} \frac{\mathbf{u}_j^T \mathbf{b}}{\sigma_j} \mathbf{v}_j .$$

(Recall the code given in Problem 2(a) for computing the truncated SVD approximation efficiently.)

Experiment with different several values of k. (Start with $k = 50$ or $k = 75$, say.) Describe how varying k affects the quality of the solution. Is there any difference if you use the exact blurred vector $\mathbf{b}$, versus the noisy blurred vector $\mathbf{b}_{\text{noise}}$? 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, explore the solutions obtained from the regularized least squares problem

$$\min_{\mathbf{f}\in\mathbb{R}^n} \|\mathbf{A}\mathbf{f} - \mathbf{b}\|^2 + \lambda^2 \|\mathbf{f}\|^2$$

for various λ. Recall that you can find the optimal value $\mathbf{f}_\lambda$ by solving the least squares problem

$$\min_{\mathbf{f}\in\mathbb{R}^n} \|\widehat{\mathbf{b}} - \mathbf{A}_\lambda \mathbf{f}\|,$$

where

$$\mathbf{A}_\lambda = \begin{bmatrix} \mathbf{A} \\ \lambda \mathbf{I} \end{bmatrix} \in \mathbb{R}^{2n\times n}, \qquad \widehat{\mathbf{b}} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n}.$$

(i) Using the value $\mathbf{b}_{\text{noise}}$ (*be sure to use* $\mathbf{b}_{\text{noise}}$), create an L-curve plot using parameter values $\lambda = 10^{-6}, \ldots, 10^{1}$. (Use `lam = np.logspace(-6,1,100)` to generate 100 logarithmically-spaced values of λ in this range.) Recall that the L-curve is a `plt.loglog` plot with $\|\mathbf{b}_{\text{noise}} - \mathbf{A}\mathbf{f}_\lambda\|$ on the horizontal axis and $\|\mathbf{f}_\lambda\|$ on the vertical axis. (*Be sure you are using* $\mathbf{b}_{\text{noise}}$ *to compute the norm of the residual,* $\|\mathbf{b}_{\text{noise}} - \mathbf{A}\mathbf{f}_\lambda\|$.)

(ii) Pick a λ value corresponding to the right-angle in the L-curve, and plot the regularized solution $\mathbf{f}_\lambda$. (See the sample code using `la.lstsq` in Problem 2.) Feel free to show plots for several different λ values that vary over several orders of magnitude. Also show the $\mathbf{f}_\lambda$ you get for the same value of λ if you instead use the true data $\mathbf{b}$ instead of $\mathbf{b}_{\text{noise}}$.

(e) Using your recovered $\mathbf{f}_k$ and/or $\mathbf{f}_\lambda$ from parts (c) and (d), attempt to reconstruct the barcode for the can of Coke. This is a little tricky – do your best, using insight from the structure of the UPC code to help.

(f) The file `mystery_bnoise1.csv` specifies the blurred, noisy $\mathbf{b}_{\text{noise}}$ barcode for a mystery product (again with `n=500` and the same $\mathbf{A}$ as for the Coke barcode).

You can load the file with the simple command `bnoise = np.loadtxt('mystery_bnoise1.csv')`.

Use the techniques in parts (c) and (d) to attempt to recover the mystery barcode. Show samples of the results from your various attempts.

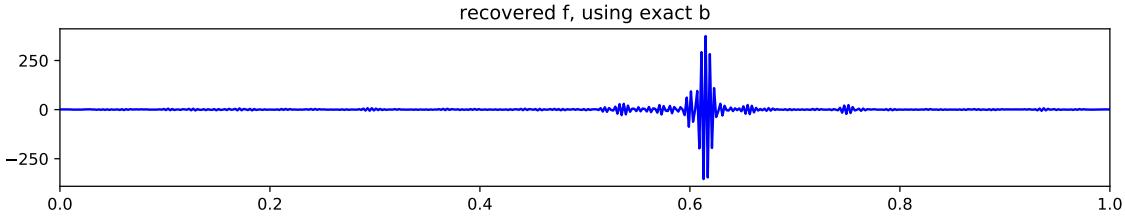
(g) **5 point bonus**: *You may not discuss your answer with your classmates.*
What is the product described by the mystery UPC? (Once you have found the correct UPC, a search on Amazon or `https://www.barcodelookup.com` will turn up the product.)

(h) **10 point bonus**: *You may not discuss your answer with your classmates.*
Repeat the deblurring process for the blurred barcode stored in `mystery_bnoise2.csv`, which is slightly noisier that the previous mystery UPC (but again uses `n=500` and the same $\mathbf{A}$ as for the two previous barcodes).

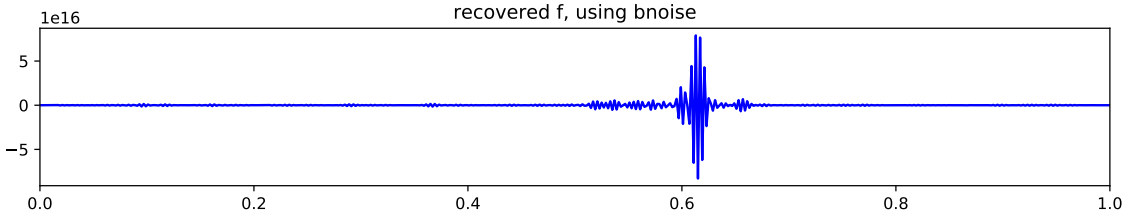
To what product does this blurry barcode correspond? Include your evidence.

Solution.

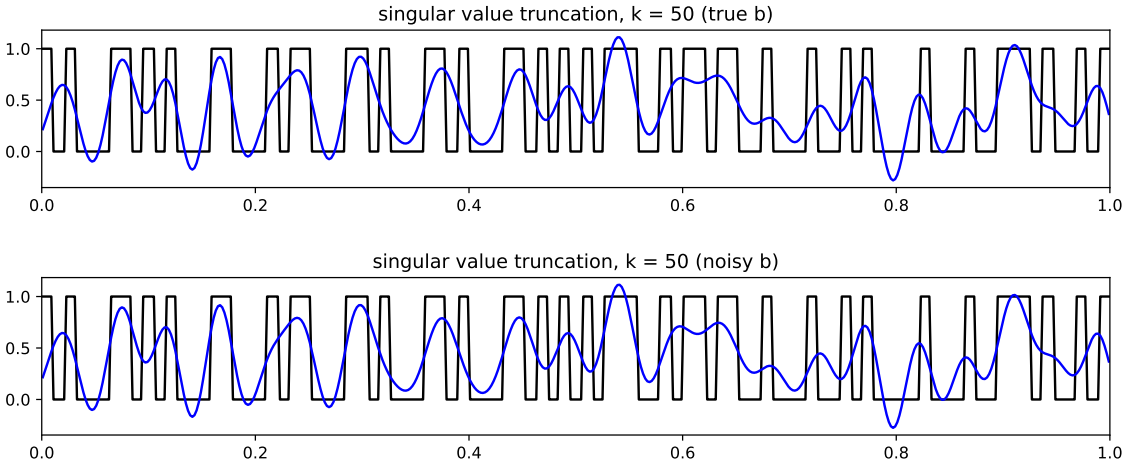
(a) The "exact" inverse gives $_{\text{inv}}$, shown in red in the plot below. (This is quite far from what we expect, due to small rounding errors in the Gaussian elimination. For $n = 500$ this problem is quite sensitive!)

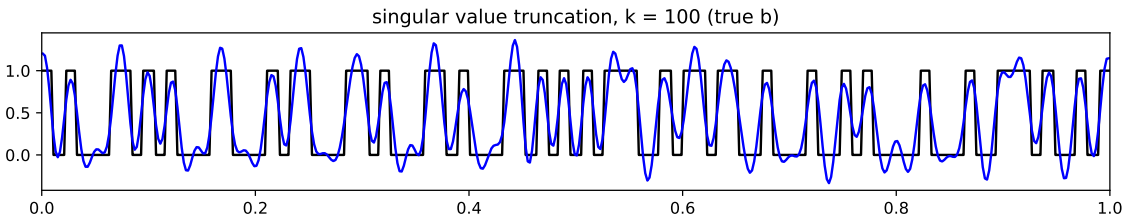


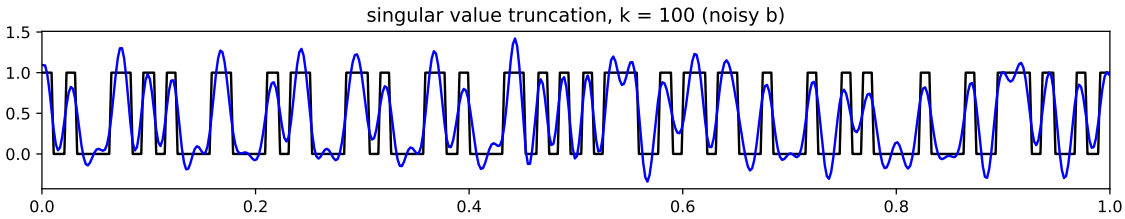
(b) The recovery from $\mathbf{b}_{\text{noise}}$ is even worse. Notice the vertical scale on this plot.

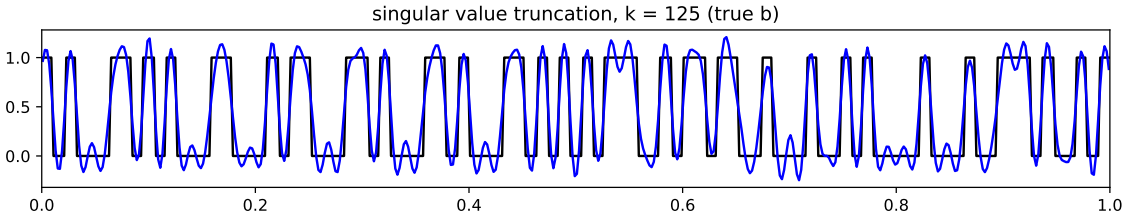


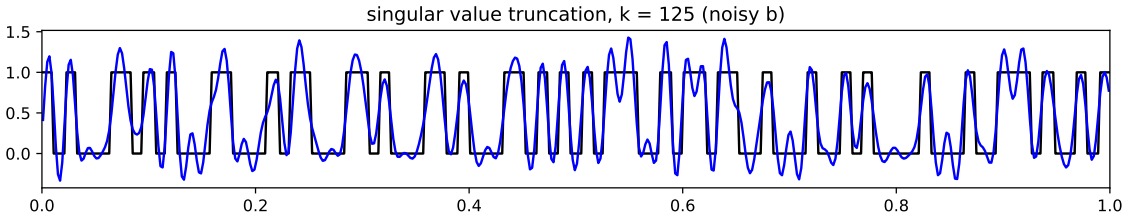
(c) The truncated SVD produces the following output for five different values of r. When r is too small, the answer is too smooth; for values of r around 100 to 125, the answer is best. For the larger values of r shown here, the recovery for $\mathbf{b}$ are good, while those for the noisy data $\mathbf{b}_{\text{noise}}$ are not nearly as good.

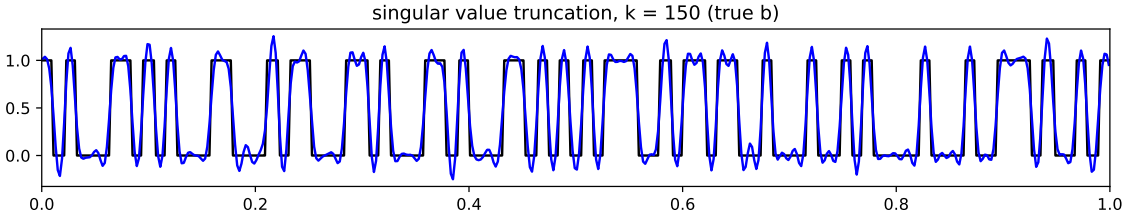


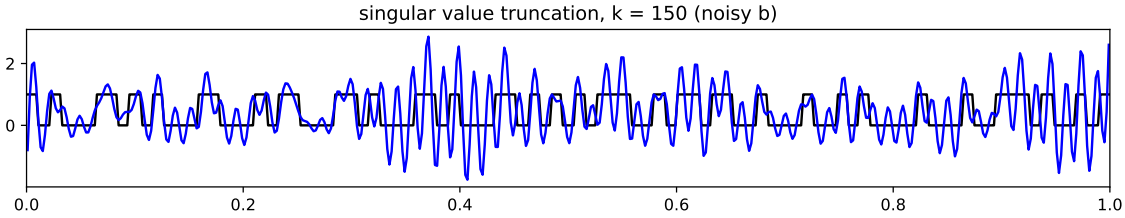

singular value truncation, k = 100 (true b)
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

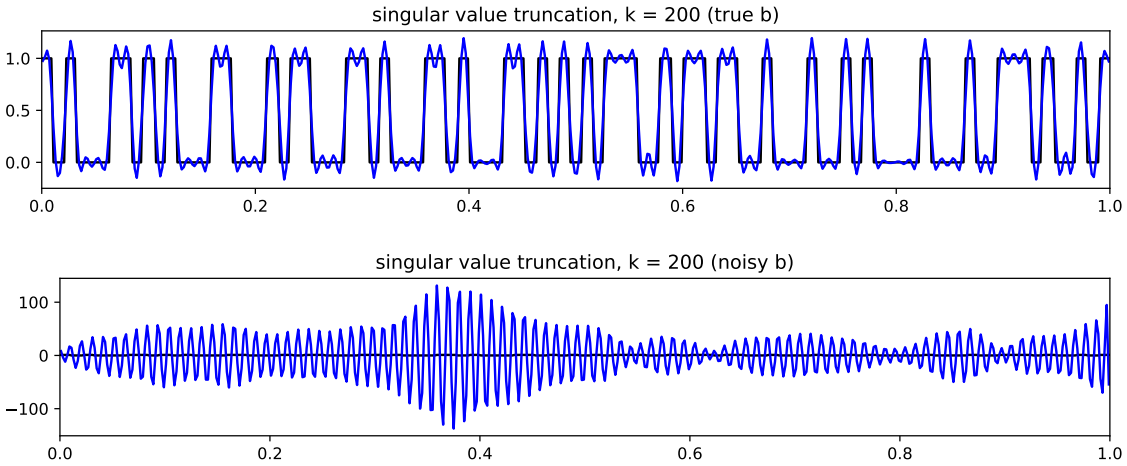

singular value truncation, k = 100 (noisy b)
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


singular value truncation, k = 125 (true b)
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

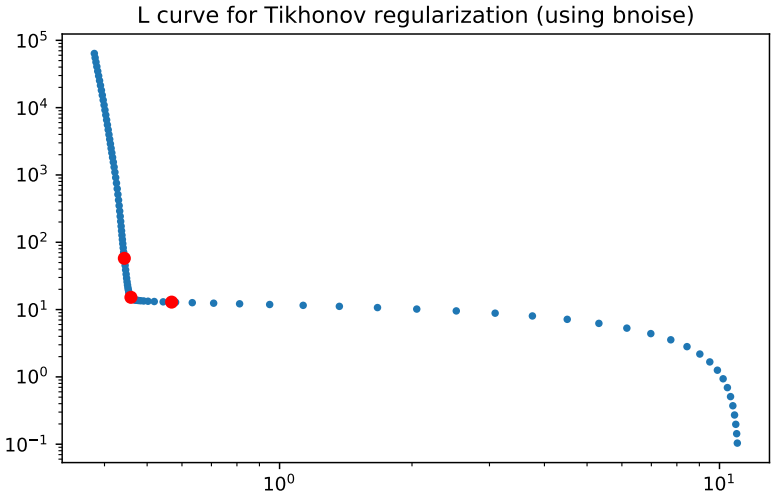

singular value truncation, k = 125 (noisy b)
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

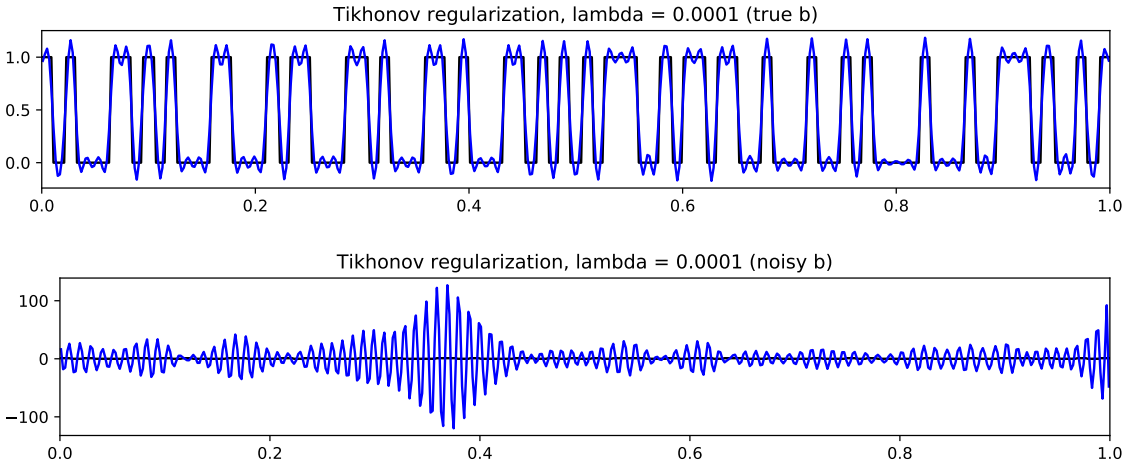

singular value truncation, k = 150 (true b)
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

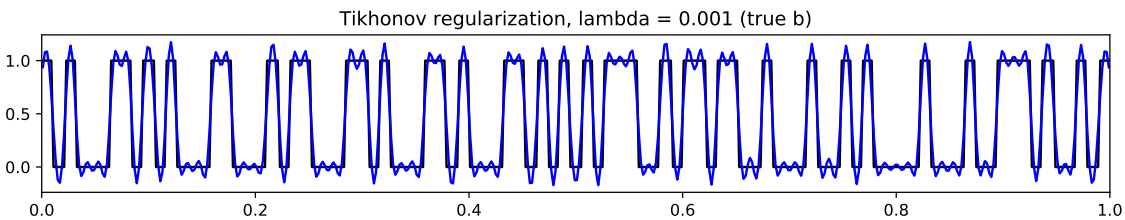

singular value truncation, k = 150 (noisy b)
2
0
0.0
0.2
0.4
0.6
0.8
1.0

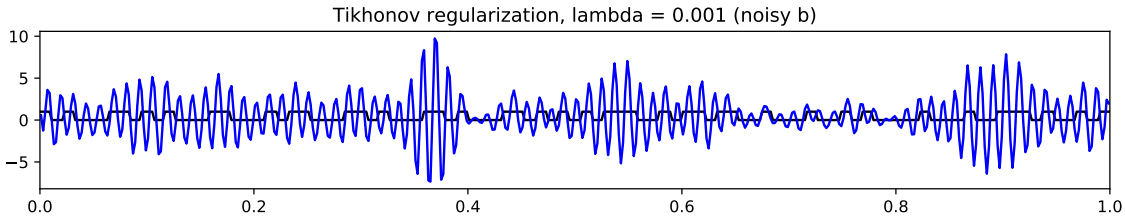


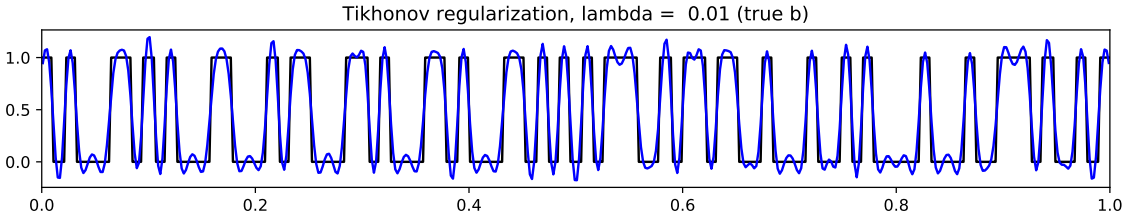
(d) The following plot shows the L curve generated by applying Tikhonov regularization to $\mathbf{b}_{\text{noise}}$ for $\lambda \in [10^{-6}, 10^{1}]$. The three red dots on this plot highlight the values $\lambda = 0.001$ (before the bend), $\lambda = 0.01$ (near the bend), and $\lambda = 0.1$ (after the bend). These values, along with a couple other λ values, are shown afterwards (for both $\mathbf{b}$ and $\mathbf{b}_{\text{noise}}$) .

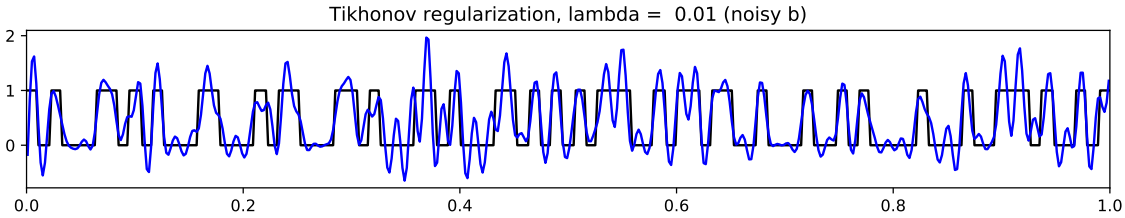


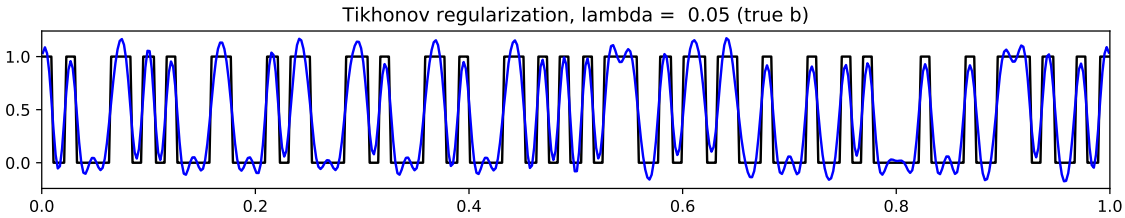


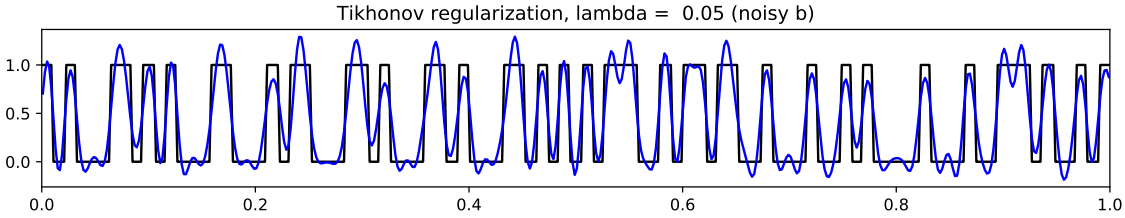

Tikhonov regularization, lambda = 0.001 (true b)
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

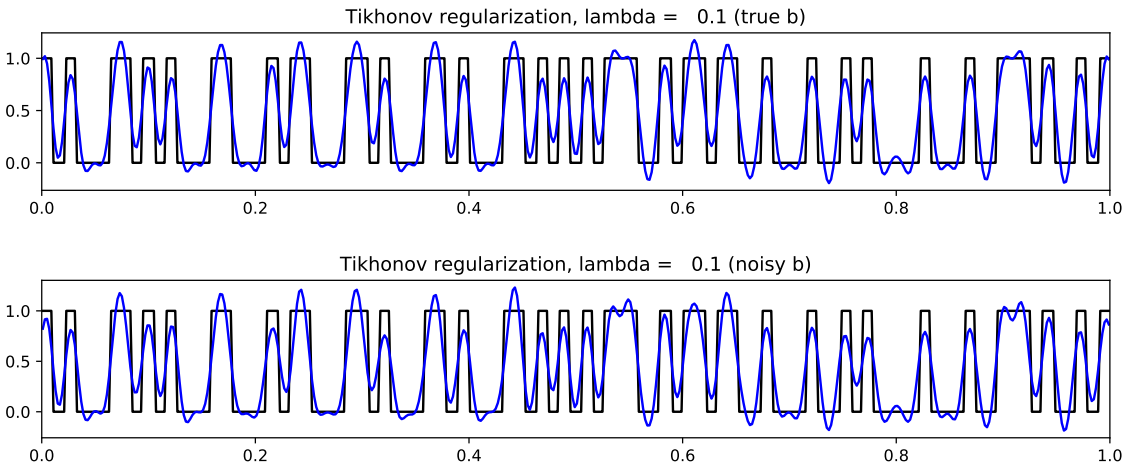

Tikhonov regularization, lambda = 0.001 (noisy b)
10
5
0
−5
0.0
0.2
0.4
0.6
0.8
1.0


Tikhonov regularization, lambda = 0.01 (true b)
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Tikhonov regularization, lambda = 0.01 (noisy b)
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0


Tikhonov regularization, lambda = 0.05 (true b)
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

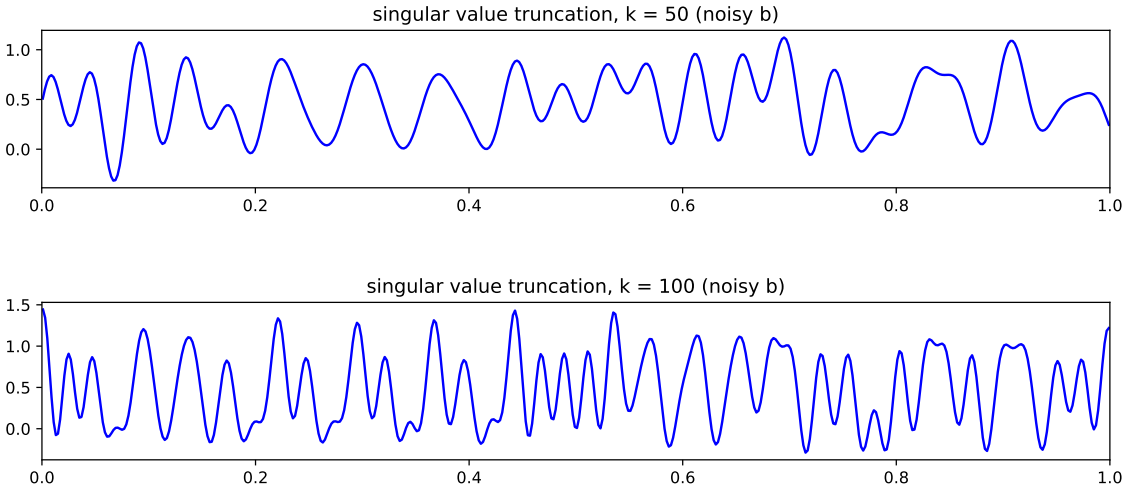

Tikhonov regularization, lambda = 0.05 (noisy b)
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

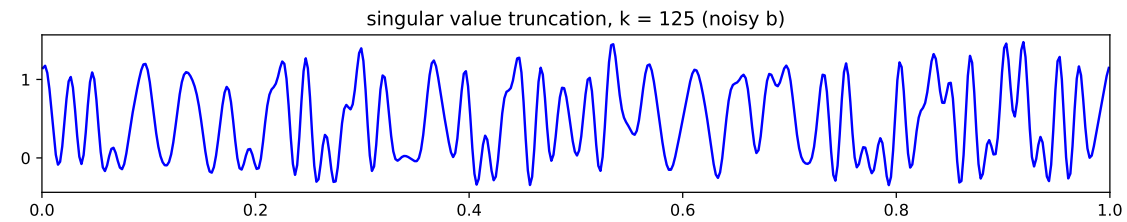


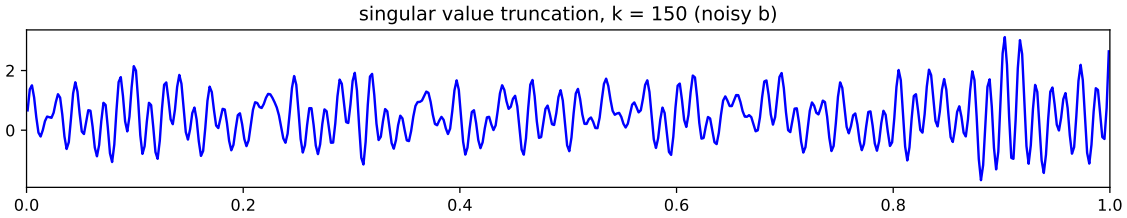
(e) We use the $\mathbf{f}_\lambda$ recovered from $\mathbf{b}_{\text{noise}}$ with $\lambda = 0.05$ to recover the Coke barcode. We recover the bars of the following length (arranged according to digits):

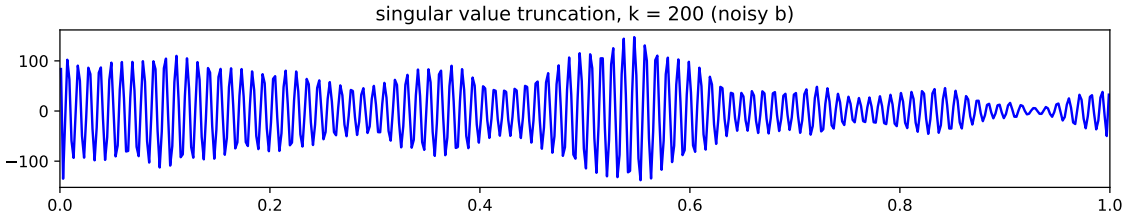
start	1	1	1		
digit 1	3	2	1	1	
digit 2	1	1	3	2	
digit 3	3	1	1	2	
digit 4	3	2	1	1	
digit 5	3	2	1	1	
digit 6	3	2	1	1	
middle	1	1	1	1	1
digit 7	3	2	1	1	
digit 8	2	1	2	2	
digit 9	1	3	1	2	
digit 10	1	1	1	4	
digit 11	1	3	1	2	
digit 12	3	1	1	2	
end	1	1	1		

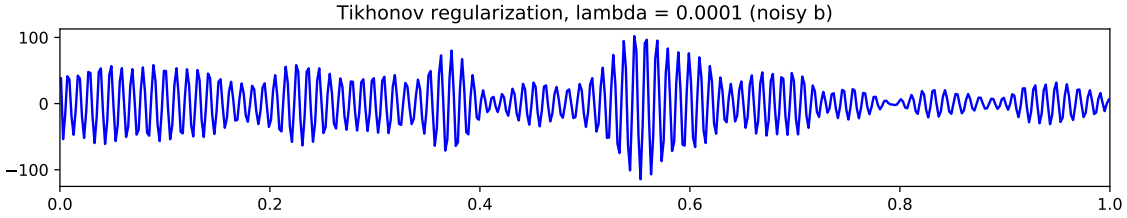
(f) Using the mystery $\mathbf{b}_{\text{noise}}$, we use the truncated SVD and regularization to obtain the following estimates.

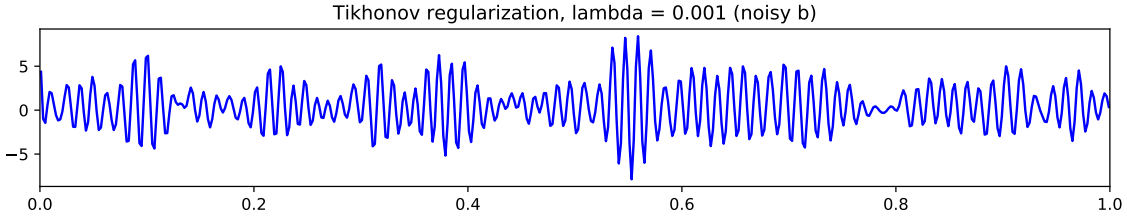


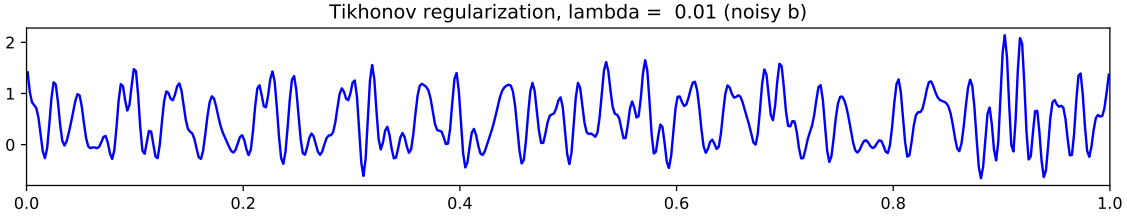

singular value truncation, k = 125 (noisy b)
1
0
0.0
0.2
0.4
0.6
0.8
1.0

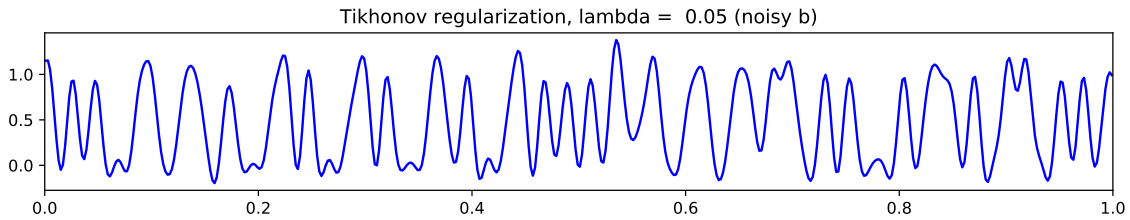

singular value truncation, k = 150 (noisy b)
2
0
0.0
0.2
0.4
0.6
0.8
1.0

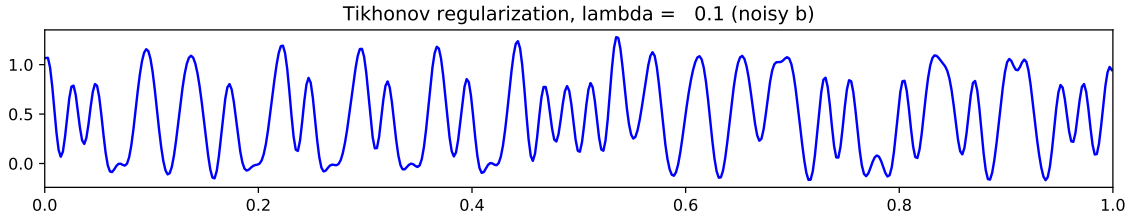

singular value truncation, k = 200 (noisy b)
100
0
−100
0.0
0.2
0.4
0.6
0.8
1.0


Tikhonov regularization, lambda = 0.0001 (noisy b)
100
0
−100
0.0
0.2
0.4
0.6
0.8
1.0


Tikhonov regularization, lambda = 0.001 (noisy b)
5
0
−5
0.0
0.2
0.4
0.6
0.8
1.0


Tikhonov regularization, lambda = 0.01 (noisy b)
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0





(g) [+5 points] We will leave this bonus a mystery.... Suffice to say, we can recover the correct code using regularization applied to the noisy vector $\mathbf{b}_{\text{noise}}$.

(h) [+10 points] The second bonus has a bit more noise, so recovering it might be slightly more subtle. Graders, please only reward the full 10 points if the answer is justified by computational evidence.
