Additional Exercises for

*Vectors, Matrices, and Least Squares*

Stephen Boyd    Lieven Vandenberghe

December 13, 2017

This is a collection of additional exercises for the book *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*, by Stephen Boyd and Lieven Vandenberghe. They are used in EE103 (Stanford) and EE103 (UCLA). We will be updating this file frequently, so be sure to download it often, for example, before starting your homework assignment. Some of the exercises appearing here have been moved into the book; we are in the process of deleting or modifying those exericses. The first 19 sections follow the book chapters.

*Stephen Boyd and Lieven Vandenberghe*

# Contents

# 1 Vectors

**1.1** *Julia timing test.* Determine how much time it takes for your computer to compute the inner product of two vectors of length $10^8$ (100 million), and use this to estimate (very crudely) how many Gflops/sec your computer can carry out. The following code generates two (random) vectors of length $10^8$, and times the evaluation of the inner product. (You might run it a few times; the first time might be a bit slower.)

```
a = randn(10^8);
b = randn(10^8);
tic(); s=dot(a,b); toc();
```

How long would it take a person to carry this out, assuming the person can carry out a floating point operation every 10 seconds for 8 hours each day?

**1.2** *Creating vectors in Julia.* In each of the parts below, use Julia to create the described vector $a$. In each case, check that $a^T x$ gives the correct result, for a random vector $x$.

(a) $a^T x$ extracts (is equal to) the 5th entry of the 10-vector $x$.

(b) $a^T x$ is the weighted average of a 3-vector $x$, assigning weights 0.3 to the first component, 0.4 to the second, and 0.3 to the third.

(c) $a^T x$ (with $x$ a 22-vector) is the sum of $x_i$ for $i$ a multiple of a 4, minus the sum of $x_i$ for $i$ a multiple of 7.

(d) $a^T x$ (with $x$ an 11-vector) is the average of the middle five entries of $x$, *i.e.*, entries 4 to 8.

## 2 Linear functions

**2.1** *Deviation of middle element value from average.* Suppose $x$ is a $n$-vector, with $n = 2m - 1$ and $m \geq 1$. We define the middle element value of $x$ as $x_m$. Define

$$f(x) = x_m - \frac{1}{n}\sum_{i=1}^{n} x_i,$$

which is the difference between the middle element value and the average of the coefficients in $x$. Express $f$ in the form $f(x) = a^T x$, where $a$ is an $n$-vector.

**2.2** *Nonlinear functions.* Show that the following two functions $f : \mathbf{R}^3 \to \mathbf{R}$ are *not* linear.

(a) $f(x) = (x_1 - x_2 + x_3)^2$.

(b) $f(x) = (x_1 + 2x_2 - x_3)_+$, where for any real number $a$, $(a)_+$ is the *positive part* of $a$, defined as $(a)_+ = \max\{a, 0\}$. *Remark.* This function is widely used in neural networks, where is it called the *relu* function, a shortening of *rectified linear unit.*

*Hint.* To show a function $f$ is not linear, it is enough to find specific vectors $x$, $y$, and scalars $\alpha$, $\beta$, for which superposition does not hold, *i.e.*,

$$f(\alpha x + \beta y) \neq \alpha f(x) + \beta f(y).$$

It's polite to the reader, but not formally required, to find simple values $x, y, \alpha, \beta$ for your counterexample.

**2.3** *Net present value.* Suppose that the $n$-vector $c$ represents a cash flow over $n$ periods. The NPV (net present value) of the cash flow, with (positive) per-period interest rate $r$, is defiend as $\mathrm{NPV}(c, r) = c_1 + (1 + r)^{-1}c_2 + \cdots + (1 + r)^{-n+1}c_n$.

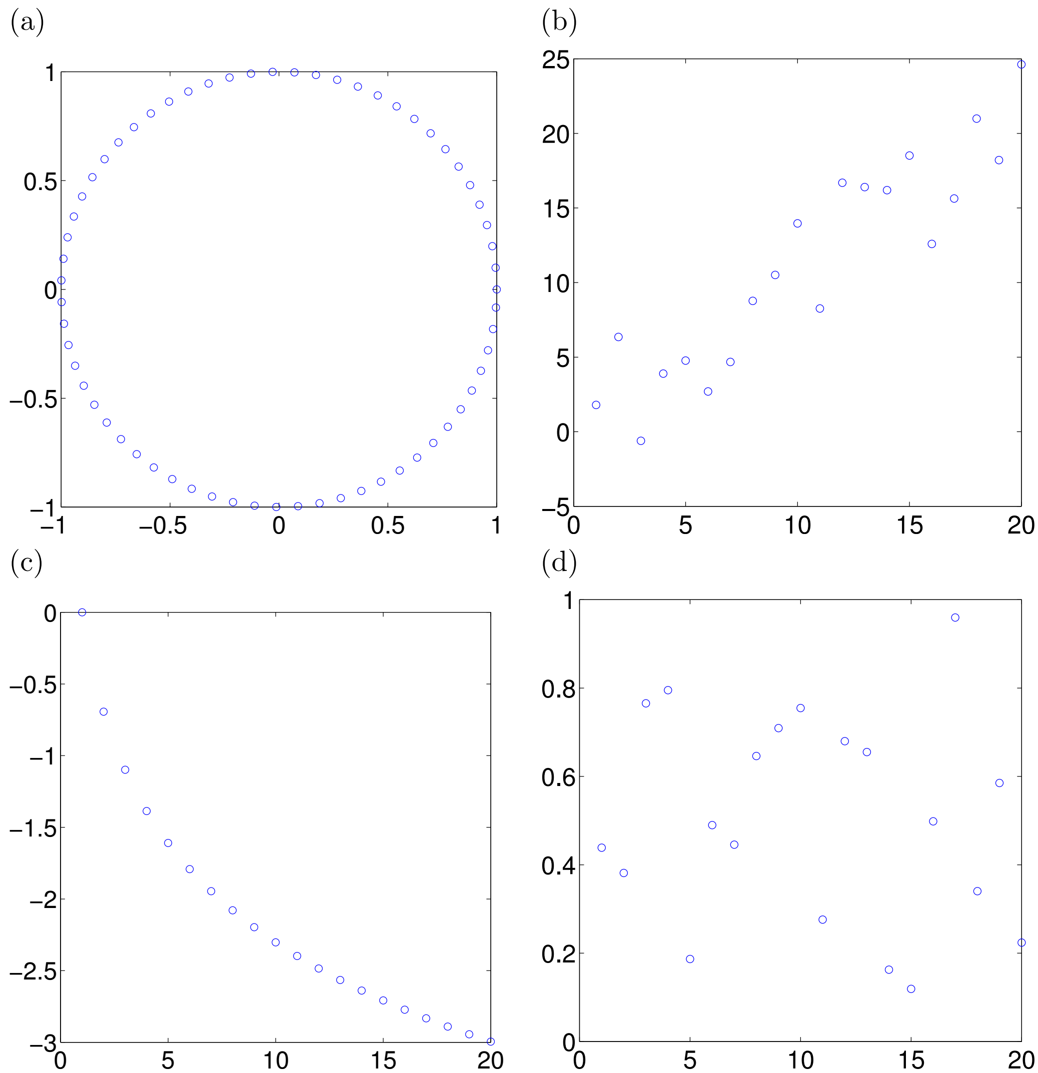
(a) How are $\mathrm{NPV}(c, 0.05)$ and $\mathrm{NPV}(2c, 0.05)$ related? (The second case is twice the cash flow, with the same 5% per-period interest rate.)

(b) How are $\mathrm{NPV}(c, 0.05)$ and $\mathrm{NPV}(c, 0.10)$ related? (The second case is the same cash flow, with twice the per-period interest rate.)

In both cases, your response can either be a specific and simple formula relating the two quantities, or the response 'It's complicated', which means that you cannot say what the relationship is, without knowing more about the entries of $c$.

# 3 Norm and distance

**3.1** *Correlation coefficient.* Each of the following plots shows the points corresponding to two vectors $x$ and $y$ of the same size, *i.e.*, we plot points at the locations $(x_i, y_i)$. In each case, determine whether the correlation coefficient $\rho$ of the two vectors is positive (and, say, $\geq 0.5$), negative (say, $\leq -0.5$), or near zero (say, less than 0.3 in absolute value). (You must choose one of these options.)




**3.2** *Nearest neighbor and smallest angle.* Using Julia, find the nearest neighbor of $a = (1, 3, 4)$ among the vectors

$$x_1 = (4, 3, 5), \quad x_2 = (0.4, 10, 50), \quad x_3 = (1, 4, 10), \quad x_4 = (30, 40, 50).$$

Report the minimum distance of $a$ to $x_1, \ldots, x_4$. Also, find which of $x_1, \ldots, x_4$ makes the smallest angle with $a$ and report that angle.

**3.3** *Orthogonality.* Suppose the $n$-vectors $a$, $b$, and $c$ satisfy $a \perp c$ and $b \perp c$. Which of the following statements must hold? (That is, are true for any $a$, $b$, and $c$ that satisfy $a \perp c$, $b \perp c$.)

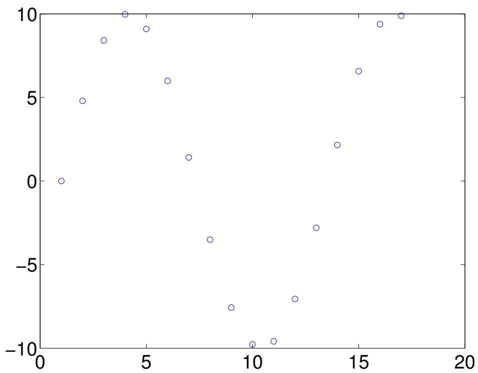
(a) $a \perp b$.

(b) $(a+b) \perp c$.

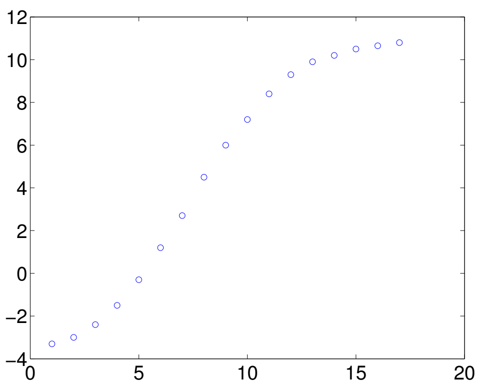
(c) $(a+c) \perp (b+c)$.

**3.4** *Guessing means and standard deviations.* Each of the plots below shows a vector $x$, with $x_i$ plotted on the vertical axis and $i$ on the horizontal axis. For each case, estimate $\mathbf{avg}(x)$ and $\mathbf{std}(x)$. We are looking for a crude guess, say, within a factor of two.

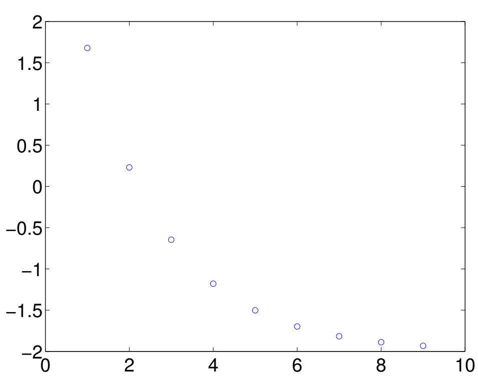
(a)




(b)




(c)




**3.5** Let $\alpha$, $\beta$, and $\gamma$ be scalars and let $a$, $b$, and $c$ be pairwise orthogonal $n$-vectors. (This means that $a \perp b$, $a \perp c$, and $b \perp c$.) Express

$$\|\alpha a + \beta b + \gamma c\|$$

in terms of $\|a\|$, $\|b\|$, $\|c\|$, $\alpha$, $\beta$, and $\gamma$.

**3.6** *True or false.* Determine whether each of the following statements is true or false.

(a) If $n$-vectors $x$ and $y$ make an acute angle, then $\|x+y\| \geq \max\{\|x\|, \|y\|\}$.

(b) For any vector $a$, $\mathbf{avg}(a) \leq \mathbf{rms}(a)$.

**3.7** *Perfect correlation.* Suppose nonzero vectors $x$ and $y$ are perfectly correlated, which means their correlation coefficient is one. Show that this implies there are numbers $a$ and $b$ for which $y = ax+b$.

**3.8** *Triangle inequality for angles.* Show that $\angle(x,y) \leq \angle(x,z) + \angle(z,y)$ for any nonzero vectors $x$, $y$, $z$. In other words, angles satisfy the triangle inequality. (Recall that angles are normalized to lie between 0 and $\pi$.)

*Hints.*

- We can just as well assume that $\|x\| = \|y\| = \|z\| = 1$. This will greatly simplify the formulas for the angles.
- When $\|x\| = \|y\| = 1$, $\|x-y\| = \sqrt{2(1-\cos\angle(x,y))}$.
- You might find the identity $\cos^2(\alpha+\beta) = \cos^2\alpha + \cos^2\beta - 1$ useful.

**3.9** Let $x = (1,2,3)$ and $y = (3,2,1)$. For parts (a)–(e), give the number, which can involve standard mathematical functions like squareroot, arc-cosine, and so on. For part (f), your answer must be *True* or *False*. You do not need to show any computations or give any justification.

(a) $\mathbf{rms}(x)$.

(b) $\mathbf{avg}(y)$.

(c) $\mathbf{std}(x)$.

(d) The correlation coefficient of $x$ and $y$.

(e) The distance between $x$ and $y$.

(f) The vectors $x$ and $y$ are linearly dependent.

**3.10** *Time to find the nearest neighbor.* We have a collection of 10000 different feature vectors, each of dimension 1000. A new feature vector (of dimension 1000) is given, and we need to determine which of the 10000 given feature vectors is its nearest neighbor. About how long would this take, on a computer capable of 10 Gflop/s? (That is, $10^{10}$ floating point operations per second.) Circle one of the responses below. You do not have to justify your choice.

- Well under one second.
- A few seconds.
- A few minutes.
- Around an hour.

# 4 Clustering

**4.1** *Building a recommendation engine using $k$-means.* A set of $N$ users of a music-streaming app listens to songs from a library of $n$ songs over some period (say, a month). We describe user $i$'s listening habits by her playlist vector, which is the $n$-vector $p_i$ defined as

$$(p_i)_j = \begin{cases} 1 & \text{user } i \text{ has played song } j \\ 0 & \text{user } i \text{ has not played song } j, \end{cases}$$

for $j = 1, \ldots, n$. (Note that $p_i$ is an $n$-vector, while $(p_i)_j$ is a number.) You can assume that if a user listens to a song, she likes it.

Your job (say, during a summer internship) is to design an algorithm that recommends to each user 10 songs that she has not listened to, but might like. (You can assume that for each user, there are at least 10 songs that she has not listened to.)

To do this, you start by running $k$-means on the set of playlist vectors $p_1, \ldots, p_N$. (It's not relevant here, but a reasonable choice of $k$ might be 100 or so.) This gives the centroids $z_1, \ldots, z_k$, which are $n$-vectors.

Now what do you do? You can explain in words; you do not need to give a formula to explain how you make the recommendations for each user.

**4.2** *Topic discovery via $k$-means.* In this problem you will use $k$-means to cluster 300 Wikipedia articles selected from 5 broad groups of topics. The Julia file `wikipedia_corpus.jl` contains the histograms as a list of 300 1000-vectors in the variable `article_histograms`. It also provides the list of article titles in `article_titles` and a list of the 1000 words used to create the histograms in `dictionary`.

The file `kmeans.jl` provides a Julia implementation of the $k$-means algorithm in the function `kmeans`. The `kmeans` function accepts a list of vectors to cluster along with the number of clusters, $k$, and returns three things: the centroids as a list of vectors, a list containing the index of each vector's closest centroid, and a list of the value of $J$ after each iteration of $k$-means. Each time the function `kmeans` is invoked it initializes the centroids by randomly assigning the data points to $k$ groups and taking the $k$ representatives as the means of the groups. (This means that if you run `kmeans` twice, with the same data, you might get different results.)

For example, here is an example of running $k$-means with $k = 8$ and finding the 30th article's centroid.

```
include("wikipedia_corpus.jl")
include("kmeans.jl")
using Kmeans

centroids, labels, j_hist = kmeans(article_histograms, 8)
centroids[labels[30]]
```

The list `labels` contains the index of each vector's closest centroid, so if the 30th entry in `labels` is 7, then the the 30th vector's closest centroid is the 7th entry in `centroids`.

There are many ways to explore your results. For example, you could print the titles of all articles in a cluster.

```
julia> article_titles[labels .== 7]
16-element Array{UTF8String,1}:
 "Anemometer"
 "Black ice"
 "Freezing rain"
 ...
```

Alternatively, you could find a topic's most common words by ordering `dictionary` by the size of its centroid's entries. A larger entry for a word implies it was more common in articles from that topic.

```
julia> dictionary[sortperm(centroids[7],rev=true)]
1000-element Array{ASCIIString,1}:
 "wind"
 "ice"
 "temperature"
 ...
```

(a) For each of $k = 2$, $k = 5$, and $k = 10$ run $k$-means twice, and plot $J$ (vertically) versus iteration (horizontally) for the two runs on the same plot. Create your plot by passing a vector containing the value of $J$ at each iteration to PyPlot's `plot` function. Comment briefly on your results.

(b) Choose a value of $k$ from part (a) and investigate your results by looking at the words and article titles associated with each centroid. Feel free to visit Wikipedia if an article's content is unclear from its title. Give a short description of the topics your clustering discovered along with the 3 most common words from each topic. If the topics do not make sense pick another value of $k$.

**4.3** *Centroid interpretations.* The $n$-vectors $x_1, \ldots, x_N$ contain $n$ attributes of $N$ patients admitted to a hospital. The first component, $(x_i)_1$, is the age of patient $i$. The second component, $(x_i)_2$, is 1 if the patient is having trouble breathing, and 0 if not. (The other components give other attributes.) An EE103 graduate carries out $k$-means on this data set, with $k = 22$. She finds the 5th centroid or group representative is $z_5 = (41.6, 0.37, 1.55, \ldots, 29.6)$.

Give a simple short interpretation in English of the first two components, *i.e.*, $(z_5)_1 = 41.6$ and $(z_5)_2 = 0.37$.

# 5 Linear independence

**5.1** *Linear independence under combination.* Suppose $S = \{a, b, c\}$ and $T = \{d, e, f\}$ are two linearly independent sets of $n$-vectors. For each of the sets given below, determine which statement is correct. You may not use a computer to answer the questions. (Only one is correct in each case.)

(a) $\{a, b, c, d, e, f\}$

- is always linearly independent.
- is always linearly dependent.
- could be linearly independent or linearly dependent, depending on the values of $a, \ldots, f$.

(b) $\{a + d, b + e, c + f\}$

- is always linearly independent.
- is always linearly dependent.
- could be linearly independent or linearly dependent, depending on the values of $a, \ldots, f$.

(c) $\{a, a + b, a + b + c\}$

- is always linearly independent.
- is always linearly dependent.
- could be linearly independent or linearly dependent, depending on the values of $a, \ldots, f$.

**5.2** *Order of vectors in the Gram-Schmidt algorithm.* Suppose $a_1, a_2$ is a list of two linearly independent $n$-vectors. When we run the Gram-Schmidt algorithm on this list, we obtain the orthonormal vectors $q_1, q_2$.

Now suppose we run the Gram-Schmidt algorithm on the list of vectors $a_2, a_1$ (*i.e.*, the same vectors, in reverse order). Do we get the orthonormal vectors $q_2, q_1$ (*i.e.*, the orthonormal vectors obtained from the original list, in reverse order)?

If you believe this is true, give a very brief explanation why. If you believe it is not true, give a simple counter-example.

# 6 Matrices

**6.1** *Checking superposition in Julia.* Generate a random $20 \times 10$ matrix $A$, as well as 10-vectors $x$ and $y$, and scalars $\alpha$ and $\beta$. Evaluate the two 20-vectors $A(\alpha x + \beta y)$ and $\alpha(Ax) + \beta(Ay)$, and verify that they are very close by printing the norm of the difference. (If the numerical calculations were done exactly, they would be equal. Due to very small rounding errors made in the floating-point calculations, they will not be exactly equal.)

*Hint.* The Julia function `rand` can be used to generate random scalars, vectors, and matrices. `rand()` generates a random number, `rand(n)` generates a random $n$-vector, and `rand(m,n)` generates a random $m \times n$ matrix.

**6.2** *Vandermonde matrices in Julia.* Write a function that takes a positive integer $n$ and an $m$-vector $t$ as inputs and generates the corresponding $m \times n$ Vandermonde matrix.

**6.3** *Linear independence.* For each of the following matrices, determine which response is correct. You may not use a computer to answer the questions.

(a)

$$\begin{bmatrix} 428 & 973 & -163 & 245 & -784 & 557 \\ 352 & 869 & 0 & 781 & -128 & 120 \\ 1047 & 45 & -471 & 349 & -721 & 781 \end{bmatrix}$$

- The columns are linearly independent.
- The columns are linearly dependent.
- This is not an appropriate question.

(b)

$$\begin{bmatrix} 768 & 1121 & 3425 & 8023 \\ -2095 & -9284 & 5821 & -6342 \\ 4093 & -3490 & -7249 & 8241 \\ 834 & 1428 & 4392 & 5835 \\ -7383 & 1435 & 2345 & -293 \end{bmatrix}$$

- The columns are linearly independent.
- The columns are linearly dependent.
- This is not an appropriate question.

**6.4** *Difference from trailing three-day rolling average.* The $n$-vector $p$ gives the daily time series of the price of an asset over $n$ trading days, with $n \geq 4$. The $(n-3)$-vector $d$ gives the difference of the current asset price and the average asset price over the previous three trading days, starting from the fourth day. Specifically, for $i = 1, \dots, n-3$, we have $d_i = p_{i+3} - (p_i + p_{i+1} + p_{i+2})/3$. (Note that $d$ is an $(n-3)$-vector.)

Give the matrix $A$ for which $d = Ap$, for the specific case $n = 6$. Be sure to give its size and all entries.

*Remark.* Time series similar to $d$ are used in some simple trading schemes, which buy or sell the asset depending on whether the price is high or low compared to a trailing multi-day rolling average.

## 7 Matrix examples

**7.1** *Equalization in communication.* Run the file `channel_equalization_data.jl`, which will define a message $s$, a channel $c$, and an equalizer $h$. (Your are welcome to look inside the file to see how we designed the equalizer.)

Plot $c$, $h$, and $h * c$. Make a brief comment about the channel and equalized channel impulse responses.

Plot $s$, $y$, and $\tilde{y}$ over the index range $i = 1, \ldots, 100$. Is it clear from this plot that $\hat{s} = \mathbf{round}(y_{1:N})$ will be worse estimate of $s$ than $\hat{s}^{\mathrm{eq}} = \mathbf{round}(\tilde{y}_{1:N})$?

Report the BER for $\hat{s}$ (estimating the message without equalization), and for $\hat{s}^{\mathrm{eq}}$ (estimating the message with equalization).

*Hint:* To round a real vector `x` to $\{0, 1\}$ in Julia you can use `(x .> 0.5)`, which yields a Boolean vector. You can convert it to an integer vector (say, for plotting) by multiplying by 1. That is, `1*(x .> 0.5)`.

**7.2** *Convolution in Julia.* Use Julia's `conv()` function to find the coefficients of the polynomial $(1 - x + 2x^2)^4$. *Hint.* Convolution gives the coefficients of the product of two polynomials.

**7.3** *Audio filtering.* When the vector $x$ represents an audio signal, and $h$ is another (usually much shorter) vector, the convolution $y = h * x$ is called the *filtered* version of $x$, and $h$ is called the *filter impulse response.* Filters can be used to smooth out audio signals (which reduces high frquency sounds and enhances low frequency sounds), or to sharpen them (which enhances high frequency sounds and reduces low frequency sounds), as in audio bass and treble tone controls. In this problem you will experiment with, and listen to, the effects of several audio filters.

The file `audio_filtering_original.wav` contains a 10-second recording with sample rate of $f = 44100$/sec. We let $x$ denote the 441000-vector representing this recording. You can read in $x$ and the sample rate $f$ using the following code:

```
Pkg.add("WAV")
using WAV
x, f = wavread("audio_filtering_original.wav");
x = vec(x);
```

To play the signal, run:

```
wavplay(x, f);
```

If this not supported on your system, you can write the signal into a file, download the file from JuliaBox if you are using that, and then listen to it on your machine:

```
wavwrite(x, f, "filename.wav");
```

(a) *1ms smoothing filter.* Let $h^{\mathrm{smooth}}$ be the 44-vector $h^{\mathrm{smooth}} = \frac{1}{44}\mathbf{1}_{44}$. (The subscript 44 gives the length of the vector.) The signal $h^{\mathrm{smooth}} * x$ is the 1ms moving average of the input $x$. We can construct the vector $h^{\mathrm{smooth}}$ and compute the output signal as follows:

```
h_smooth = 1 / 44 * ones(44);
output = conv(h_smooth, x);
wavplay(output, f);
```

Listen to the output signal and briefly describe the effect of convolving $h^{\text{smooth}}$ with $x$ in one sentence.

(b) *Echo filter.* What filter (*i.e.*, vector) $h^{\text{echo}}$ has the property that $h^{\text{echo}} * x$ consists of the original recording, plus an echo of the original recording 0.25 seconds delayed, with half the original amplitude? Since sound travels at about 340m/s, this is equivalent to the effect of hearing an echo from a wall about 42.5m away. Construct $h^{\text{echo}}$ using Julia and listen to the output signal $h^{\text{echo}} * x$ to confirm the effect. Form and listen to the signal $h^{\text{echo}} * h^{\text{echo}} * x$ and very briefly describe what you hear.

*Hint.* The entries of the output signal $y = h^{\text{echo}} * x$ satisfy $y_i = x_i + 0.5x_{i-k}$, where we take $x_j = 0$ for $j$ outside the range $1, \ldots, 441000$, and $k$ is the number of samples in 0.25 seconds.

**7.4** *Another property of convolution.* Suppose that $a$ is an $n$-vector and $b$ is an $m$-vector that satisfy $a * b = 0$. Is it true that either $a = 0$ or $b = 0$? If yes, explain why. If no, give a specific example of $a$ and $b$, not both zero, with $a * b = 0$.

**7.5** *Convolution.* What is $(1, -1) * (1, 0, 1)$?

## 8 Linear equations

**8.1** Suppose the 5-vector $c$ gives the coefficients of a quartic (degree four) polynomial $p(x) = c_1 + c_2 x + c_3 x^2 + c_4 x^3 + c_5 x^4$. Express the conditions

$$p(0) = p(1), \quad p'(0) = p'(1)$$

as a set of linear equations of the form $Ac = b$. Give the sizes of $A$ and $b$, as well as their entries.

**8.2** *Linear function?* The function $f : \mathbf{R}^n \to \mathbf{R}^m$ is linear. Define another function $g : \mathbf{R}^n \to \mathbf{R}$ by $g(x) = (f(x))_1$. Is $g$ a linear function? If so, briefly explain why. If not, give a simple counter-example.

# 9 Linear dynamical systems

## 10 Matrix multiplication

**10.1** *Matrix multiplication Julia timing test.* Determine how long it takes your computer to compute the product of two $n \times n$ matrices for $n = 500, 1000, 2000, 4000$, and use your result for $n = 4000$ to estimate (very crudely) how many Gflops/sec your computer can carry out. (Hopefully your results for the different values of $n$ will give roughly consistent estimates of computer speed.)

The follow code generates two random $500 \times 500$ matrices and times the evaluation of their product. (You might run it a few times; the first time might be a bit slower, since the matrix multiplication code has to be loaded and compiled.)

```
A = randn(500,500); B = randn(500,500);
tic(); C=A*B; toc();
```

How long would it take a person to carry this out, assuming the person can carry out a floating point operation every 10 seconds for 8 hours each day?

**10.2** *Customer purchase history matrix.* A store keeps track of its sales of products from $K$ different product categories to $N$ customers over some time period, like one month. (While it doesn't matter for this problem, $K$ might be on the order of 1000 and $N$ might be 100000.) The data is stored in an $N \times K$ matrix $C$, with $C_{ij}$ being the total dollar purchases of product $j$ by customer $i$. All the entries of $C$ are nonnegative. The matrix $C$ is typically sparse, *i.e.*, many of its entries are zero.

(a) What is $C\mathbf{1}$?

(b) What is $C^T\mathbf{1}$?

(c) Give a short matrix-vector expression for the total dollar amount of all purchases, by all customers.

(d) What does it mean if $(CC^T)_{kl} = 0$? Your answer should be simple English.

(e) Suppose you run $k$-means on the rows of $C$, with $k = 100$. How would you interpret the centroids $z_1, \ldots, z_{100}$?

**10.3** *State feedback control.* Consider a time-invariant linear dynamical system with $n$-vector state $x_t$ and $m$-vector input $u_t$, with dynamics

$$x_{t+1} = Ax_t + Bu_t, \quad t = 1, 2, \ldots.$$

The entries of the state often represent deviations of $n$ quantities from their desired values, so $x_t \approx 0$ is a goal in operation of the system. The entries of the input $u_t$ are deviations from the standard or nominal values. For example, in an aircraft model, the states might be the deviation from the desired altitude, climb rate, speed, and angle of attack; the input $u_t$ represents changes in the control surface angles or engine thrust from their normal values.

In *state feedback control*, the states are measured and the input is a linear function of the state, $u_t = Kx_t$. The $m \times n$ matrix $K$ is called the *state feedback gain matrix.* The state feedback gain matrix is very carefully designed, using several methods. State feedback control is very widely used in many application areas (including, for example, control of airplanes).

(a) *Open and closed-loop dynamical system.* With $u_t = 0$, the system satisfies $x_{t+1} = Ax_t$ for $t = 1, 2, \ldots$, which is called the *open-loop dynamics*. When $u_t = Kx_t$, the system dynamics can be expressed as $x_{t+1} = \tilde{A}x_t$, for $t = 1, 2, \ldots$, where the $n \times n$ matrix $\tilde{A}$ is the *closed-loop dynamics matrix*. Find an expression for $\tilde{A}$ in terms of $A$, $B$, and $K$.

(b) *Aircraft control.* The longitudinal dynamics of a 747 flying at 40000 ft at Mach 0.81 is given by

$$A = \begin{bmatrix} .99 & .03 & -.02 & -.32 \\ .01 & .47 & 4.7 & .00 \\ .02 & -.06 & .40 & -.00 \\ .01 & -.04 & .72 & .99 \end{bmatrix}, \qquad B = \begin{bmatrix} 0.01 & 0.99 \\ -3.44 & 1.66 \\ -0.83 & 0.44 \\ -0.47 & 0.25 \end{bmatrix},$$

where the sampling time is one second. (The state and control variables are described in more detail in the lecture on control.) We will use the state feedback matrix

$$K = \begin{bmatrix} -.038 & .021 & .319 & -.270 \\ -.061 & -.004 & -.120 & .007 \end{bmatrix}.$$

(The matrices $A$, $B$, and $K$ can be found in `747_cruise_dyn_data.jl`, so you don't have to type them in.) Plot the open-loop and closed-loop state trajectories from several nonzero initial states, such as $x_1 = (1, 0, 0, 0)$, or ones that are randomly generated, from $t = 1$ to $t = 100$ (say). (In other words, plot $(x_t)_i$ versus $t$, for $i = 1, 2, 3, 4$.) Would you rather be a passenger in the plane with the state feedback control turned off (*i.e.*, open-loop) or on (*i.e.*, closed-loop)?

**10.4** *Student-course matrix.* The Stanford registrar has the complete list of courses taken by each graduating student over several graduating classes. This data is represented by an $m \times n$ matrix $C$, with $C_{ij} = 1$ if student $i$ took class $j$, and $C_{ij} = 0$ otherwise, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. (Thus, there are $m$ students in the data set, and $n$ different courses. For simplicity, we ignore the possibility that in some circumstances a student can take a course multiple times.)

Answer each of the questions below in English, with no equations, references to matrices or vectors, and so on. (You can refer to student $i$ and course $j$, though.)

(a) What is $(C^TC)_{kl}$?

(b) What is $(CC^T)_{rs}$?

(c) What is $(C^T\mathbf{1})_p$?

(d) Suppose you cluster the columns of $C^T$ using $k$-means, with $k = 50$ (say). What do you think the results might look like? (Your response can be a bit vague, but not more than one or two sentences.)

(e) (Continuation of part (d).) Suppose $z_1$ is the cluster representative for group 1. What does $(z_1)_{143} = 0.01$ mean?

**10.5** Suppose $A$ is a $5 \times 10$ matrix, $B$ is a $20 \times 10$ matrix, and $C$ is a $10 \times 10$ matrix. Determine whether each of the following expressions make sense. If the expression makes sense, give its dimensions.

(a) $A^TA + C$.

(b) $BC^3$.

(c) $I + BC^T$.

(d) $B^T - [C\ I]$.

(e) $B \left[ \begin{array}{c} A \\ A \end{array} \right] C$.

**10.6** *Friend matrix.* We consider a collection of $n$ people who participate in a social network in which pairs of people can be connected, by 'friending' each other. The $n \times n$ matrix $F$ is the friend matrix, defined by $F_{ij} = 1$ if persons $i$ and $j$ are friends, and $F_{ij} = 0$ if not. We assume that the friend relationship is symmetric, *i.e.*, person $i$ and person $j$ are friends means person $j$ and person $i$ are friends. We will also assume that $F_{ii} = 0$. Express the following in matrix/vector notation, briefly justifying your expression.

(a) $t$ is the $n$-vector with $t_i$ being the total number of friends of person $i$.

(b) $C$ is the $n \times n$ matrix with $C_{ij}$ equal to the number of friends persons $i$ and $j$ have in common. (Person $k$ is a friend in common of persons $i$ and $j$ if she is a friend of both person $i$ and person $j$. The diagonal entry $C_{ii}$, which is the total number of friends person $i$ has in common with herself, is the total number of friends of person $i$.)

**10.7** Suppose $F^T G = 0$, where $F$ and $G$ are $n \times k$ matrices. Determine whether each of the following statements must always be true, or can be false. 'Must be true' means the statement holds for any $n \times k$ matrices $F$ and $G$ that satisfy $F^T G = 0$, without any further assumptions; 'can be false' means that there are $n \times k$ matrices $F$ and $G$ that satisfy $F^T G = 0$, but the statement does not hold.

(a) Either $F = 0$ or $G = 0$.

(b) The columns of $F$ are orthonormal.

(c) Each column of $F$ is orthogonal to each column of $G$.

(d) The matrices $F$ and $G$ are square or tall, *i.e.*, $n \geq k$.

(e) The columns of $F$ are linearly dependent.

**10.8** *Matrix with acute columns.* Suppose $A$ is an $m \times n$ matrix with nonzero columns $a_1, \ldots, a_n$, and in addition, any pair of columns makes an acute angle, *i.e.*, $|\angle(a_i, a_j)| < 90^\circ$ for all $i, j = 1, \ldots, n$. What can you say about the entries of the Gram matrix $G = A^T A$?

## 11 Matrix inverses

**11.1** *Rows and columns of a matrix and its inverse.* Suppose the $n \times n$ matrix $A$ is invertible, with inverse $B = A^{-1}$. We let the $n$-vectors $a_1, \ldots, a_n$ denote the columns of $A$, and $b_1^T, \ldots, b_n^T$ the rows of $B$. Determine whether each of the following statements is true or false, and justify your answer. *True* means the statement always holds, with no further assumptions. *False* means the statement does not always hold, without further assumptions.

(a) For any $n$-vector $x$, we have $x = \sum_{i=1}^n (b_i^T x) a_i$.

(b) For any $n$-vector $x$, we have $x = \sum_{i=1}^n (a_i^T x) b_i$.

(c) For $i \neq j$, $a_i \perp b_j$.

(d) For any $i$, $\|b_i\| \geq 1/\|a_i\|$.

(e) For any $i$ and $j$, $b_i + b_j \neq 0$.

(f) For any $i$, $a_i + b_i \neq 0$.

**11.2** *Solving linear equations in Julia.* Generate a random $20 \times 20$ matrix $A$ and a random 20-vector $b$ using the following code:

```
A = rand(20, 20)
b = rand(20)
```

(It's very likely that the matrix you generate will be invertible.)

We solve the linear equation $Ax = b$, *i.e.*, compute the solution $x = A^{-1}b$ in Julia using several methods. In each case, you should check that the $x$ you compute satisfies the equations by evaluating and reporting the norm of the residual, $\|Ax - b\|$. (This should be very small.)

(a) Using the backslash operator:

```
x = A \ b
```

(b) Computing the inverse of $A$ explicitly:

```
x = inv(A) * b
```

(c) Using QR factorization, from the formula $x = R^{-1}Q^T b$:

```
Q, R = qr(A)
x = R \ (Q' * b)
```

(You should check that the matrix $Q$ obtained is very nearly orthogonal, $R$ is an upper triangular matrix, and that $A$ is very near $QR$.)

**11.3** *Julia timing test for linear equations.*

(a) Determine how long it takes for your computer to solve a system of $n = 2000$ linear equations in $n = 2000$ variables (with invertible coefficient matrix) using Julia's \ operator. You may use the following code.

```
A = 1 + rand(2000, 2000)
b = ones(2000)
@time A\b;
```

(b) Julia is rather clever about how it solves systems of equations with \. Determine how long it takes for your computer to solve the following system of $n = 2000$ linear equations in $n = 2000$ variables.

```
L = 1 + rand(2000, 2000)
for i = 1:2000
  for j = i+1:2000
    L[i, j] = 0
  end
end
b = ones(2000)
@time L\b;
```

(c) Can you explain why the times differ by so much between the two systems, *i.e.*, what is special about the matrix $L$ as opposed to $A$? Make a hypothesis about what you think Julia is doing behind the scenes.

**11.4** *Sensitivity of solution of linear equations.* Let $A$ be an invertible $n \times n$ matrix, and $b$ and $x$ be $n$-vectors satisfying $Ax = b$. Suppose we now perturb the $j$th entry $b$ by $\epsilon \neq 0$ (which is a traditional symbol for a small quantity), which means that $b$ becomes $\tilde{b} = b + \epsilon e_j$. Let $\tilde{x}$ be the $n$-vector that satisfies $A\tilde{x} = \tilde{b}$, *i.e.*, the solution of the linear equations using the perturbed right-hand side. We are interested in $\|x - \tilde{x}\|$, which is how much the solution changes due to the change in the right-hand side.

(a) Show that $\|x - \tilde{x}\|$ does not depend on $b$; it only depends on the matrix $A$, $\epsilon$, and $j$.

(b) How would you find the index $j$ that maximizes the value of $\|x - \tilde{x}\|$? By part (a), your answer should be in terms of $A$ (or quantities dervied from $A$) and $\epsilon$ only.

(c) Try this out in Julia with the following values:

$$A = \begin{bmatrix} 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \\ 1/4 & 1/5 & 1/6 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \qquad \epsilon = 0.1.$$

To prevent numerical imprecision errors, use the following code to build the data.

```
A = [1/2 1/3 1/4; 1/3 1/4 1/5; 1/4 1/5 1/6]
b = [1.0, 1.0, 1.0]
epsilon = 0.1
```

Which $j$ do you pick to maximize $\|x - \tilde{x}\|$, and what value do you get for $\|x - \tilde{x}\|$? Check your answer from part (b) by direct calculation (*i.e.*, simply finding $\tilde{x}$ after perturbing entry $j = 1, 2, 3$ of $b$).

**11.5** Let $A$ and $B$ be distinct, square $n \times n$ matrices that satisfy $A^5 = B^5$ and $AB^4 = BA^4$. Determine whether the matrix $A^4 + B^4$ is invertible. If it is, give an explicit formula to compute its inverse. If it is not, provide an explanation.

*Hint:* Recall that matrix multiplication distributes over addition.

## 12 Least squares

**12.1** *Solving least squares problems in Julia.* Generate a random $20 \times 10$ matrix $A$ and a random 20-vector $b$.

(a) Compute the solution $\hat{x}$ of the associated least squares problem using the methods listed below, and verify that the solutions found are the same, or more accurately, very close to each other; they will be very slightly different due to small roundoff errors in the computations.

- Using the Julia backslash operator.
- Using $\hat{x} = (A^T A)^{-1} A^T b$.
- Using $\hat{x} = A^\dagger b$.

*Hints.* In Julia, `inv()` computes the inverse matrix, `pinv()` computes the pseudo-inverse matrix, and `A\b` directly solves the least squares problem.

(b) Let $\hat{x}$ be one of the solutions found in part (a). Generate a random nonzero 10-vector $\delta$ and verify that $\|A(\hat{x} + \delta) - b\|^2 > \|A\hat{x} - b\|^2$. Repeat several times with different values of $\delta$; you might try choosing a small $\delta$ (say, by scaling the original random vector).

Be sure to submit your code, including the code that checks if the solutions in part (a) are close to each other, and whether the expected inequality in part (b) holds.

**12.2** *Julia timing test for least squares.* Determine how long it takes for your computer to solve a least squares problem with $m = 100000$ equations and $n = 100$ variables. (You can use the backslash operator.)

*Remark.* Julia compiles just in time, so you should run the code a few times to get the correct time.

**12.3** *Active noise cancellation.* Active noise cancellation is used to reduce the noise level heard at a specific reference location. This is done using several microphones that pick up ambient noise, a loudspeaker near the reference loction, and a microphone that monitors the sound at the reference location. The signals from the ambient noise microphones are combined, added to the desired signal, and played through the loudspeaker in such a way that the noise heard at the reference location is cancelled, or at least reduced. In real active noise cancellation systems the noise microphone signals are convolved with filters before being added, but we will consider here a simplified setup in which a simple linear combination of them is played through the speaker. We will also assume here that the speaker is perfect.

Here is a mathematical description of the method. We let the scalar time series $y_1, y_2, \ldots$ denote the desired signal, which we know. We let the scalar time series $c_1, c_2, \ldots$ denote the noise cancellation signal, which we will create. We play the signal $y_t + c_t$ through the loudspeaker. At the reference location we hear $y_t + c_t + n_t$, where $n_1, n_2, \ldots$ is a noise signal that we do not know. If we can choose $c_t$ so that $c_t + n_t \approx 0$, then we have achieved (approximate) noise cancellation.

The $K$ ambient noise signals are given by the $K$-vector time series $a_1, a_2, \ldots$. We use $c_t = \theta^T a_t$, where the $K$-vector $\theta$ gives the coefficients for combining the ambient noise microphone signals to form the noise cancellation signal. Our goal is to choose $\theta$ so that $\theta^T a_t + n_t \approx 0$.

Here is how we will choose $\theta$. We use a training period, say, $t = 1, \ldots, T$. We send nothing (*i.e.*, the zero signal) to the loudspeaker, and record the noise $n_1^{\text{train}}, \ldots, n_T^{\text{train}}$ using the reference microphone. We also record the ambient noise signals $a_1^{\text{train}}, \ldots, a_T^{\text{train}}$. We then choose $\theta$ to minimize $\sum_{t=1}^{T}(\theta^T a_t^{\text{train}} + n_t^{\text{train}})^2$.

Once we have $\hat{\theta}$ we are ready to deploy noise cancellation. We send $y_t + \hat{\theta}^T a_t$ to the loudspeaker, and we hear $y_t + \hat{\theta}^T a_t + n_t$ at the reference location.

(In real active noise cancellation systems, the weights are updated continuously, while the desired signal is playing.)

The file `anc_data.jl` contains data for an active noise cancellation problem. When you execute this file it will include the audio files. It also contains some helper code for writing out a vector as an audio file, which you can then listen to. You will find the following signals:

- For training, a $T$-vector `n` and the $K \times T$ matrix `a`, which you will use to determine $\theta$. Be sure to listen to the noise signal, and the individual rows of `a`, which give the ambient noise microphone signals. You can check your active noise cancellation by forming and then listening to $n_t + \hat{\theta}^T a_t$. (This should sound quieter than $n_t$, which is what you would hear without active noise cancellation.)
- For testing, we give you some test signals `y_plus_n_test`, which gives $y_t + n_t$ (over the test interval), and `a_test`, a matrix which gives $a_t$ (over the test interval). To test your $\hat{\theta}$, you should form and listen to $y_t + \hat{\theta}^T a_t + n_t$. Compare this to what $y_t + n_t$ sounds like.

**12.4** *Transit system tomography.* An urban transit system (say, a subway) consists of $n$ links between pairs of stations. Each passenger enters the system at an origin station, traverses over a sequence of the links, and then exits the system at their destination station. The fare collection system keeps track of the following information for passenger $i$: $s_i$, the trip starting time (when she enters the origin station), measured in minutes after 6AM, $f_i$, the trip finishing time (when she leaves the destination station), and the sequence of links over which she traveled. For example, $s_i = 128$, $f_i = 144$, and link list $(3, 7, 8, 10, 4)$ means the passenger entered the origin station at 8:08AM, left the destination station at 8:24AM, and traversed links $3, 7, 8, 10, 4$, in that order. The total trip time is $f_i - s_i$.

We model the trip time as the sum of the delays over the links the passenger traverses. We let $d_i$ denote the delay associated with link $i$, for $i = 1, \ldots, n$. We do not know the link delays, but wish to estimate them, based on the passenger information described above, for a very large number $m$ of passengers passing through the system.

We will do this using least-squares. We choose our estimate $\hat{d}$ of the $n$-vector of link delays as the minimizer of the norm squared of the residual of our trip time model. In other words, we choose $\hat{d}$ to minimize $\|Rd - c\|^2$, for some $m \times n$ matrix $R$ and some $m$-vector $c$.

Say what $R$ and $c$ are. (That is, give all their entries.)

## 13 Least squares data fitting

**13.1** *Saving TA time using midterm score prediction.* The TAs very carefully graded all problems on all students' midterms. But suppose they had just graded the first half of the exam, *i.e.*, problems 1–5, and used a regression model to predict the total score on each exam.

The matrix of midterm scores is available on the midterm page on the course web site. There rows correspond to students (in random order, anonymized), and the columns to midterm questions.

(a) Find the average and standard deviation of the total midterm scores. Find the average and standard deviation for each individual midterm problem.

(b) Here is a very simple way to predict the total score based on the 5 scores for first half of the exam: Sum the 5 first half scores, and double the result. What RMS prediction error does this simple method achieve?

(c) Find a regression model that predicts the total score based on the 5 scores for problems 1–5. Give the coefficients and briefly interpret them. What is the RMS prediction error of your model?

(Just for fun, evaluate the predictor on your own exam score. Would you rather have your actual score or your predicted score?)

*Remark.* Your dedicated EE103 teaching staff would *never* do anything like this. Really.

**13.2** *Moore's law.* These numbers are available in the Julia file `moore_data.jl` In this data file, `t` is the first column of the table (introduction year) and `N` is the second column (number of transistors). *Hint.* In Julia, the function $\log_{10}$ is `log10`.

**13.3** *Auto-regressive time series prediction.* Suppose that $x$ is an $N$-vector representing time series data. The (one step ahead) prediction problem is to guess $x_{t+1}$, based on $x_1, \ldots, x_t$. We will base our prediction $\hat{x}_{t+1}$ of $x_{t+1}$ on the previous $M$ values, $x_t, x_{t-1}, \ldots, x_{t-M+1}$. (The number $M$ is called the *memory length* of our predictor.) When the prediction is a linear function,

$$\hat{x}_{t+1} = \beta_1 x_t + \beta_2 x_{t-1} + \cdots + \beta_M x_{t-M+1},$$

it is called an *auto-regressive predictor*. (It is possible to add an offset to $\hat{x}_{t+1}$, but we will leave it out for simplicity.) Of course we can only use our auto-regressive predictor for $M \leq t \leq N-1$.

Some very simple and natural predictors have this form. One example is the predictor $\hat{x}_{t+1} = x_t$, which guesses that the next value is the same as the current one. Another one is $\hat{x}_{t+1} = x_t + (x_t - x_{t-1})$, which guesses what $x_{t+1}$ is by extrapolating a line that passes through $x_t$ and $x_{t-1}$.

We judge a predictor (*i.e.*, the choice of coefficients $\beta_i$) by the mean-square prediction error

$$J = \frac{1}{N-M} \sum_{t=M}^{N-1} (\hat{x}_{t+1} - x_{t+1})^2.$$

A sophisticated choice of the coefficients $\beta_i$ is the one that minimizes $J$. We will call this the least-squares auto-regressive predictor.

(a) Find the matrix $A$ and the vector $b$ for which $J = \|A\beta - b\|^2/(N - M)$. This allows you to find the coefficients that minimize $J$, *i.e.*, the auto-regressive predictor that minimizes the mean-square prediction error on the given time series. Be sure to give the dimensions of $A$ and $b$.

(b) For $M = 2, \ldots, 12$, find the coefficients that minimize the mean-square prediction error on the time series `x_train` given in `time_series_data.jl`. The same file has a second time series `x_test` that you can use to test or validate your predictor on. Give the values of the mean-square error on the train and test series for each value of $M$. What is a good choice of $M$? Also find $J$ for the two simple predictors described above.

*Hint.* Be sure to use the `toeplitz` function contained in `time_series_data.jl`. It'll make your life alot easier. Documentation for the function is also contained in `time_series_data.jl`.

**13.4** *Modeling class attendance.* A university registrar keeps track of class attendance, measured as a percentage (so 100 means full attendance, and 30 means 30% of the students attend). For each class lecture, she records the attendance $y$, and several features:

- $x_1$ is the day of week, with Monday coded as 1 and Friday coded as 5.
- $x_2$ is the week of the quarter, coded as 1 for the first week, and 10 for the last week.
- $x_3$ is the hour of the lecture, with 8AM coded as 8, and 4PM coded as 16.
- $x_4 = \max\{T - 80, 0\}$, where $T$ is the outside temperature (so $x_4$ is the number of degrees above 80°F).
- $x_5 = \max\{50 - T, 0\}$, where $T$ is the outside temperature (so $x_5$ is the number of degrees below 50°F).

(These features were suggested by a professor who is an expert in the theory of class attendance.) An EE103 alumna carefully fits the data with the following regression model,

$$\hat{y} = -1.4x_1 - 0.3x_2 + 1.1x_3 - 0.6x_4 - 0.5x_5 + 68.2,$$

and validates it properly.

Give a short story/explanation, in English, of this model.

**13.5** *Nonlinear auto-regressive model.* We have a (scalar) time series $z_1, z_2, \ldots, z_T$. The following one step ahead prediction model is proposed:

$$\hat{z}_{t+1} = \theta_1 z_t + \theta_2 z_{t-1} + \theta_3 z_t z_{t-1},$$

where $\theta = (\theta_1, \theta_2, \theta_3)$ is the model parameter vector. The sum of the squares of the prediction error of this model on the given time series is

$$\sum_{t=2}^{T-1} (\hat{z}_{t+1} - z_{t+1})^2.$$

(Note that we must start this sum with $t = 2$, since $z_0$ and $z_{-1}$ are not defined.) Express this quantity as $\|A\theta - b\|^2$, where $A$ is a $(T-2) \times 3$ matrix and $b$ is a $(T-2)$-vector. (You must say what the entries of $A$ and $b$ are. They can involve the known data $z_1, \ldots, z_T$.)

*Remark.* Finding $A$ and $b$ is the first step in fitting the parameters $\theta$ to the data. We are not asking you to find $\theta$, but only to set up the least squares problem you'd solve to carry out the least squares fit.

**13.6** *Lecture attendance, laptops in lecture, and final grade.* A study collects data on a large number of students in a lecture course, with the goal of predicting the effect (or at least the association) of lecture attendance and laptop use on the final exam grade. The regressor is the 2-vector $x$, where $x_1$ is the student's lecture attendance, expressed as a number between 0 and 1 (with, say, 0.78 meaning the student attended 78% of the lectures), and $x_2$ is a Boolean feature that codes whether or not the student routinely used a laptop during lecture, with $x_2 = 0$ meaning she did not, and $x_2 = 1$ meaning that she did. The outcome variable $y$ is the student's final exam grade, scaled to be between 0 and 100 points. A basic regression model $\hat{y} = x^T\beta + v$ is fit to the data, and checked with out-of-sample validation.

(a) Give a one sentence interpretation of $\beta_1$.

(b) Give a one sentence interpretation of $\beta_2$.

(c) Suggest a value of $\beta_1$ that would not surprise you, and give a one sentence explanation that might be appropriate if the value of $\beta_1$ were your value.

(d) Suggest a value of $\beta_2$ that would not surprise you, and give a one sentence explanation that might be appropriate if the value of $\beta_2$ were your value.

For parts (c) and (d) we are looking for a plausible guess of the values, along with a plausible story that would go along with your guessed value.

## 14 Least squares classification

**14.1** *Trading off false positive and false negative rates.* In this problem you use the Boolean least squares classifier with skewed decision point, described on pages 223-224 of the textbook, to alter the false positive and false negative rates. The classifier is $\hat{f}(x) = \mathbf{sign}(\tilde{f}(x) - \alpha)$, where $\tilde{f}$ is the (real-valued) least squares predictor of the labels encoded as $\pm 1$, and $\alpha$ is a constant that skews the decision point. We will use basic regression, *i.e.*, $\tilde{f}(x) = x^T\beta + v$.

(a) *Basic classifier.* The file `classifier_with_tradeoff_data.jl` contains train and test sets `X_train`, `X_test`, and the corresponding labels `y_train` and `y_test`. Find the model parameters $\beta$ and $v$ using the training data set, and give the confusion matrix for the associated classifier (with $\alpha = 0$) for both the training and test sets. Give the error rate for the train and test sets.

(b) *ROC curve from skewed decision point classifier.* The ROC curve plots the performance of different classifiers with the vertical axis showing the true positive rate, and the horizontal axis showing the false positive rate. (See textbook, page 226.) Give ROC plots for the training and test data sets, for the skewed decision point classifier, for 20 values of $\alpha$ ranging from $-0.3$ to $+0.3$. (You will use the values of $\beta$ and $v$ found in part (a).)

*Julia hints.*

- If `y_til` is the vector $\tilde{y}$ (the continuous regression prediction), you can find $\mathbf{sign}(\tilde{y} - \alpha)$ using `sign(y_til-alpha)`.
- To find the number of true positives, you can use `sum((y_true .== 1) & (y_hat .== 1))`. (Similar constructions give the number of false positives, false negatives, and so on.)

**14.2** *Equalizer design from training message.* (Continues book exercise 14.10.) The file `eq_design_data.jl` contains the training message $s^{\text{tr}}$, the value of $n$, and the signal received from the training message, $y^{\text{tr}}$. Your first task is to design an equalizer $h$ using this data, and plot it.

The file also includes the received signal $y$ from a message $s$ that is sent. First, take the sign of $y$ to get an estimate of the message, and print it as a string. Then estimate the message $s$ using your equalizer, and print it as text. You'll know when your equalizer is working.

*Hints.*

- In Julia you can take the sign of a vector using `sign.(x)`.
- You can turn a Boolean vector with entries encoded as $-1$ and 1 into a string using the function `binary2string`. (While not needed for this problem, the function `string2binary` converts a string to a Boolean vector.)

**14.3** *Iris classification.* In this exercise you will develop and test a 3-class classifier for classifying the flower Iris into three species: *iris setosa*, *iris versicolor*, and *iris virginica*, which we will refer to as classes 1, 2, and 3, respectively. The classifier will use $n = 4$ features:

$$x = (\text{sepal length},\ \text{sepal width},\ \text{petal length},\ \text{petal width})\,.$$

(You don't need to know what these are.) The data set, which contains 150 examples, is a classic one, first published in 1936 by the famous statistician Ronald Fisher. (At the very least, you might think about how much fun it would be to carry out the calculations needed in this exercise in 1936, by hand.)

You will find the data set in `iris_flower_data.jl`. You can use the first 100 examples (2/3) for training, and the last 1/3 for testing. (We have randomized the order of the examples in the data set given.)

(a) Find 3 least squares regression classifiers, each one classifying one of the species against the other two. Give the error rate for each classifier, on both the train and test sets.

(b) Combine the three classifiers of part (a) into a 3-class classifier, and give the $3 \times 3$ confusion matrix for the train and test sets.

*Hints.* The Julia code snippet `2(y .== k) - 1` might be useful. This checks if each entry of vector `y` is equal to `k`. If it is, then the result for that entry is `1`; otherwise it is `-1`. Additionally, it may be useful to consult our helper file, `iris_multiclass_helpers.jl`. We have provided you two functions. The first is `argmax_by_row(A)`, which computes and returns a column vector $x$, where $x_i$ is the index of the maximum entry in the $i$th row of `A`. The second is `confusion_matrix(y_hat, y_true)`, which computes and returns a confusion matrix $C$. For example, $C_{11}$ is the number of predictions where your classifier correctly predicted that the flowers belong to class 1, *iris setosa*.

**14.4** *Modifying a classifier.* A co-worker develops a classifier of the form $\hat{y} = \mathbf{sign}(x^T\beta + v)$, with $v < 0$, where the $n$-vector $x$ is the feature vector, and the $n$-vector $\beta$ and scalar $v$ are the classifier parameters. The classifier is evaluated on a given test data set. The false positive rate is the fraction of the test data points with $y = -1$ for which $\hat{y} = 1$. (We will assume there is at least one data point with $y = -1$.)

Are each of the following statements true or false? True means it always holds, with no other assumptions on the data set or model; false means that it need not hold.

(a) Replacing $v$ with zero will reduce, or not increase, the false positive rate.

(b) Replacing $\beta$ with zero will reduce, or not increase, the false positive rate.

(c) Halving $v$ (*i.e.*, replacing $v$ with $v/2$) will reduce, or at least not increase, the false positive rate.

(d) Halving $\beta$ (*i.e.*, replacing $\beta$ with $(1/2)\beta$) will reduce, or at least not increase, the false positive rate.

## 15 Multi-objective least squares

**15.1** *Trading off tracking error and input size in control.* A system that we want to control has input (time series) $n$-vector $u$ and output (time series) $y$, related by convolution: $y = h * u$, where $h$ is an $m$-vector. (So $y$ is an $(n+m-1)$-vector.) (See §15.2 in the textbook.) We are given $y^{\text{des}}$, the (time series of) desired or target output values, and we will choose the input $u$ to minimize $\|y - y^{\text{des}}\|^2 + \lambda\|u\|^2$, where $\lambda > 0$ is a parameter we use to trade off tracking error (*i.e.*, $\|y - y^{\text{des}}\|^2$) and input size (*i.e.*, $\|u\|^2$).

We will consider the case where $n = 100$, $m = 7$, with

$$h = (0.3, 0.5, 0.6, 0.4, 0.3, 0.2, 0.1).$$

The desired output is the 106-vector

$$y_t^{\text{des}} = \begin{cases} 10 & 10 \leq t < 40 \\ -5 & 40 \leq t < 80 \\ 0 & \text{otherwise.} \end{cases}$$

Plot a trade off curve of the RMS tracking error ($\textbf{rms}(y - y^{\text{des}})$) versus the input RMS value ($\textbf{rms}(u)$). As usual with trade-off curves, you should vary $\lambda$ over a wide range, using values that are logarithmically spaced (*i.e.*, by a constant factor). You will find the Julia function `logspace` useful for creating a set of values of $\lambda$. You can use the function `toeplitz`.

Pick 3 values of $\lambda$ that correspond to too little regularization, too much regularization, and a reasonable amount of regularization. (Reasonable might correspond to RMS tracking error around 0.3) Plot the input $u$ found for each choice of $\lambda$ on the same figure. Also plot the output $y$ found for each $\lambda$ on the same figure, along with $y^{\text{des}}$.

**15.2** *Least squares classification with regularization.* The file `lsq_classifier_data.jl` contains feature $n$-vectors $x_1, \ldots, x_N$, and the associated binary labels, $y_1, \ldots, y_N$, each of which is either $+1$ or $-1$. The feature vectors are stored as an $n \times N$ matrix $X$ with columns $x_1, \ldots, x_N$, and the labels are stored as an $N$-vector $y$. We will evaluate the error rate on the (training) data $X$, $y$ and (to check if the model generalizes) a test set $X_{\text{test}}$, $y_{\text{test}}$, also given in `lsq_classifier_data.jl`. You may use `LinearLeastSquares` for all parts of the problem. Include your Julia code in your solution.

(a) *Least squares classifier.* Find $\beta, v$ that minimize $\sum_{i=1}^N (x_i^T\beta + v - y_i)^2$ on the training set. Our predictions are then $\hat{f}(x) = \textbf{sign}(x^T\beta + v)$. Report the classification error on the training and test sets, the fraction of examples where $\hat{f}(x_i) \neq y_i$. There is no need to report the $\beta, v$ values.

(b) *Regularized least squares classifier.* Now we add regularization to improve the generalization ability of the classifier. Find $\beta, v$ that minimize

$$\sum_{i=1}^N (x_i^T\beta + v - y_i)^2 + \lambda\|\beta\|^2,$$

where $\lambda > 0$ is the regularization parameter, for a range of values of $\lambda$. We suggest the range $10^{-1}$ to $10^4$, say, 100 values logarithmically spaced. The function `logspace` may be useful.

Use it to plot the training and test set errors against $\log_{10}(\lambda)$. Suggest a reasonable choice of $\lambda$. Again, there is no need to report the $\beta, v$ values, just attach the plot and a reasonable value of $\lambda$.

**15.3** *Estimating the elasticity matrix.* In this problem you create a standard model of how demand varies with the prices of a set of products, based on some observed data. There are $n$ different products, with (positive) prices given by the $n$-vector $p$. The prices are held constant over some period, say, a day. The (positive) demands for the products over the day is given by the $n$-vector $d$. The demand in any particular day varies, but it is thought to be (approximately) a function of the prices. The units of the prices and demands don't really matter in this problem. Demand could be measured in 10000 units, and prices in \$100.

The *nominal prices* are given by the $n$-vector $p^{\mathrm{nom}}$. You can think of these as the prices that have been charged in the past for the products. The *nominal demand* is the $n$-vector $d^{\mathrm{nom}}$. This is the average value of the demand, when the prices are set to $p^{\mathrm{nom}}$. (The actual daily demand fluctuates around the value $d^{\mathrm{nom}}$.) You know both $p^{\mathrm{nom}}$ and $d^{\mathrm{nom}}$. We will describe the prices by their (fractional) variations from the nominal values, and the same for demands. We define $\delta^p$ and $\delta^d$ as the (vectors of) relative price change and demand change:

$$\delta_i^p = \frac{p_i - p_i^{\mathrm{nom}}}{p_i^{\mathrm{nom}}}, \quad \delta_i^d = \frac{d_i - d_i^{\mathrm{nom}}}{d_i^{\mathrm{nom}}}, \quad i = 1, \ldots, n.$$

So $\delta_3^p = +0.05$ means that the price for product 3 has been increased by 5% over its nominal value, and $\delta_5^d = -0.04$ means that the demand for product 5 in some day is 4% below its nominal value.

Your task is to build a model of the demand as a function of the price, of the form

$$\delta^d \approx E\delta^p,$$

where $E$ is the $n \times n$ elasticity matrix.

You don't know $E$, but you do have the results of some experiments in which the prices were changed a bit from their nominal values for one day, and the day's demands were recorded. This data has the form

$$(p_1, d_1), \ldots, (p_N, d_N),$$

where $p_i$ is the price for day $i$, and $d_i$ is the observed demand.

Explain how you would estimate $E$, given this price-demand data. Be sure to explain how you will test for, and (if needed) avoid over-fit.

*Hint.* You might find it easier to separately fit the models $\delta_i^d \approx \tilde{e}_i^T \delta^p$, where $\tilde{e}_i^T$ is the $i$th row of $E$. (We use the tilde above $e_i$ to avoid conflict with the notation for unit vectors.)

Carry out your method using the price and demand data in the matrices `Prices` and `Demands`, found in `price_elasticity.jl`. Give your estimate $\hat{E}$, and guess (roughly) how accurate your model $\delta^d = \hat{E}\delta^p$ might be (in terms of RMS prediction error) on unseen data.

Here are some facts about elasticity matrices that might help you check that your estimates make sense (but you don't need to incorporate this information into your estimation method). The diagonal entries of $E$ are always negative, and typically on the order of one. (This means that when you raise the price of one product only, demand for it goes down by a similar fractional amount as the price increase.) The off-diagonal entries can have either sign, and are typically (but not always) smaller than one in magnitude.

**15.4** *Regularized least squares in Julia.* You are asked to write some Julia code to compute the $\hat{x}$ that minimizes $\|Ax-b\|^2 + \lambda\|x\|^2$, where $A$ is an $m \times n$ matrix, $b$ is an $m$-vector, and $\lambda$ is a positive scalar. These are given as the Julia quantities `A`, `b`, and `lambda`, respectively, and the dimensions $m$ and $n$ are given as `m` and `n`. You are to put the value of $\hat{x}$ in `xhat`.

Which of the following Julia snippets will carry this out correctly? Circle the correct response for each code snippet below. You do not need to justify your responses. (You are welcome to try to run the snippets in Julia, but you should make your guesses before doing this.)

(a) `xhat = [A sqrt(lambda)*eye(n)]\b` *Works.* *Doesn't work.*

(b) `xhat = [A sqrt(lambda)*eye(n)]\[b; zeros(m)]` *Works.* *Doesn't work.*

(c) `xhat = inv(A'*A+lambda*eye(n))*(A'*b)` *Works.* *Doesn't work.*

(d) `xhat = [A; sqrt(lambda)*eye(n)]\b` *Works.* *Doesn't work.*

(e) `xhat = [A; sqrt(lambda)*eye(n)]\[b; zeros(n)]` *Works.* *Doesn't work.*

**15.5** *Fitting models to two different but similar populations.* We wish to fit two different models to data from two different but similar populations, for example males and females. The models are given by $\hat{y} = x^T\beta$ for the first group and $\hat{y} = x^T\tilde{\beta}$ for the second group, where $x$ is the $n$-vector of features, $\hat{y}$ is the prediction, $\beta$ is the $n$-vector of model parameters for the first group and $\tilde{\beta}$ is the $n$-vector of model parameters for the second group. (We can include an offset in the two models by including a feature that is always one.)

Our training data consists of $x^{(1)}, \ldots, x^{(N)}$ and $y^{(1)}, \ldots, y^{(N)}$ from the first population, and $\tilde{x}^{(1)}, \ldots, \tilde{x}^{(N)}$ and $\tilde{y}^{(1)}, \ldots, \tilde{y}^{(N)}$ from the second group. (For simplicity we assume that we have an equal number of training data points in the two groups.)

Our main goal in choosing the parameter $n$-vectors $\beta$ and $\tilde{\beta}$ is that the sum of squares of the prediction errors for the first group (using the first model) and the sum of squares of the prediction errors for the second group (using the second model) is small. Our secondary objective is that the two parameter vectors $\beta$ and $\tilde{\beta}$ are not too different. (This desire is based on our idea that the two groups are similar, so the associated models should not be too different.)

Capture the goals expressed above as a bi-objective least squares problem with variable $\theta = (\beta, \tilde{\beta})$. Identify the primary objective $J_1 = \|A_1\theta - b_1\|^2$ and the secondary objective $J_2 = \|A_2\theta - b_2\|^2$. Give $A_1$, $A_2$, $b_1$ and $b_2$ explicitly. Your solution can involve the $n \times N$ data matrices $X$ and $\tilde{X}$, whose columns are $x^{(i)}$ and $\tilde{x}^{(i)}$, respectively, and the two $N$-vectors $y^{\rm d}$ and $\tilde{y}^{\rm d}$, whose entries are $y^{(i)}$ and $\tilde{y}^{(i)}$, respectively.

# 16 Constrained least squares

**16.1** *Computing least-norm solutions.* Generate a random $10 \times 100$ matrix $A$ and a 10-vector $b$. Use Julia to compute the least norm solution for $Ax = b$ using the methods listed below, and verify that the solutions found are the same (or more accurately, very close to each other). Be sure to submit your code, including the code that checks if the solutions are close to each other.

- Using the formula $\hat{x} = A^T(AA^T)^{-1}b$.
- Using the pseudo inverse: $\hat{x} = A^\dagger b$.
- Using the Julia backslash operator.
- Using the Julia package `LinearLeastSquares`.

**16.2** *Julia timing test for least-norm.* Determine how long it takes for your computer to solve a least-norm problem with $m = 600$ equations and $n = 4000$ variables. (You can use the backslash operator.) What approximate flop rate does your result suggest?

*Remark.* Julia compiles just in time, so you should run the code a few times to get the correct time.

**16.3** *Constrained least squares in Julia.* You are asked to write some Julia code to compute the $\hat{x}$ that minimizes $\|Ax - b\|^2$ subject to $Cx = d$, where $A$ is an $m \times n$ matrix, $b$ is an $m$-vector, $C$ is a $p \times n$ matrix, and $d$ is a $p$-vector. These are given as the Julia quantities `A`, `b`, `C`, and `d`, and the dimensions $m$, $n$, and $p$ are given as `m`, `n`, and `p`. You are to put the value of $\hat{x}$ in `xhat`. (You can assume that the associated KKT matrix is invertible.)

Write two lines of Julia code below that carries this out. Your code should be simple and clear. You do not need to justify your answer.

*Hint.* Recall that the optimality conditions for this constrained least squares problem are

$$2A^TAx + C^Tz = 2A^Tb, \qquad Cx = d,$$

where $z$ is the vector of Lagrange multipliers.

# 17 Constrained least squares applications

**17.1** *Portfolio optimization.* In this problem you will optimize a set of holdings to minimize risk for various average returns. Download the files `portfolio_data.jl` and `asset_prices.csv` and place them both in your working directory. Include `portfolio_data.jl` in your code, it defines the return matrices `R_train` of dimension `T_train` by `n` and `R_test` of dimension `T_test` by `n` (where `n` is the number of assets and `T_train`, `T_test` are number of days), and the `asset_names` array. (The train data covers the time period 2005-2013, the test data 2013-2015.) Using `R_train` find asset allocation weights for the portfolios that minimize risk for annualized returns of 10% and 20%. (To obtain annualized return multiply the daily return by $P = 250$ trading days.)

Plot the cumulative value for each portfolio over time, starting from the conventional initial investment of \$10000, for both the train and test sets of returns. To compute the product $10^4(1 + r_1^T w)(1 + r_2^T w)\cdots(1 + r_T^T w)$ you can use `1e4*cumprod(1+returns*w)` where `returns` is a returns matrix and `w` an allocation vector.

For each of the 2 portfolios report

- the annualized return on the training and test sets;
- the annualized risk on the training and test sets;
- the asset with the minimum allocation weight (can be the most negative), and its weight;
- the asset with the maximum allocation weight, and its weight;
- the leverage, defined as $|w_1| + \cdots + |w_n|$. (Several other definitions of leverage are used.) This number is always at least one, and it is exactly one only if the portfolio has no short positions. You can use `sum(abs(w))`, where `w` is your allocation vector.

Comment briefly.

**17.2** *Rendezvous.* The dynamics of two vehicles, at sampling times $t = 1, 2, \ldots$, are given by

$$x_{t+1} = Ax_t + Bu_t, \qquad z_{t+1} = Az_t + Bv_t$$

where the $n$-vectors $x_t$ and $z_t$ are the states of vehicles 1 and 2, and the $m$-vectors $u_t$ and $v_t$ are the inputs of vehicles 1 and 2. The $n \times n$ matrix $A$ and the $n \times m$ matrix $B$ are known.

The position of vehicle 1 at time $t$ is given by $Cx_t$, where $C$ is a known $2 \times n$ matrix. Similarly, the position of vehicle 2 at time $t$ is given by $Cz_t$.

The initial states of the two vehicles are fixed and given:

$$x_1 = x_{\text{start}}, \qquad z_1 = z_{\text{start}}.$$

We are interested in finding a sequence of inputs for the two vehicles over the time interval $t = 1, \ldots, T-1$ so that they *rendezvous* at time $t = T$, *i.e.*, $x_T = z_T$. You can select the inputs to the two vehicles,

$$u_1, u_2, \ldots, u_{T-1}, \qquad v_1, v_2, \ldots, v_{T-1}.$$

Among choices of the sequences $u_1, \ldots, u_{T-1}$ and $v_1, \ldots, v_{T-1}$ that satisfy the rendezvous condition, we want the one that minimizes the weighted sum of squares of the two vehicle inputs,

$$J = \sum_{t=1}^{T-1} \|u_t\|^2 + \lambda \sum_{t=1}^{T-1} \|v_t\|^2,$$

where $\lambda > 0$ is a parameter that trades off the two objectives.

(a) Explain how to find the sequences $u_1, \ldots, u_{T-1}$ and $v_1, \ldots, v_{T-1}$ that minimize $J$ while satisfying the rendezvous condition by solving a constrained least-squares problem.

(b) The problem data $A$, $B$, $C$, $x_{\text{start}}$, and $z_{\text{start}}$ are defined in `rendezvous.jl`. Use `LinearLeastSquares` to find $u_1, \ldots, u_{T-1}$ and $v_1, \ldots, v_{T-1}$ for $\lambda = 0.1$, $\lambda = 1$, and $\lambda = 10$. Plot the vehicle trajectories (*i.e.*, their positions) for each $\lambda$ using the plotting code in `rendezvous.jl`.

(c) Give a simple expression for $x_T$ in the limit where $\lambda \to \infty$ and for $z_T$ in the limit where $\lambda \to 0$. Assume that for any $w \in \mathbf{R}^n$ there exist sequences $u_1, \ldots, u_{T-1}$ and $v_1, \ldots, v_{T-1}$ such that the rendezvous constraints are satisfied with $w = z_T = x_T$.

**17.3** *A linear regulator for a linear dynamical system.* We consider a linear dynamical system with dynamics $x_{t+1} = Ax_t + Bu_t$, where the $n$-vector $x_t$ is the state at time $t$ and the $m$-vector $u_t$ is the input at time $t$. We assume that $x = 0$ represents the desired operating point; the goal is to find an input sequence $u_1, \ldots, u_{T-1}$ that results in $x_T = 0$, given the initial state $x_1$. Choosing an input sequence that takes the state to the desired operating point at time $T$ is called *regulation*.

(a) Find an explicit formula for the sequence of inputs that yields regulation, and minimizes $\|u_1\|^2 + \cdots + \|u_{T-1}\|^2$, in terms of $A$, $B$, $T$, and $x_1$. This control is called the *minimum energy regulator*.

*Hint.* Your formula may involve the *controllability matrix*

$$C = \left[\begin{array}{cccc} B & AB & \cdots & A^{T-2}B \end{array}\right],$$

and the vector $u = (u_{T-1}, u_{T-2}, \ldots, u_1)$ (which is the input sequence in reverse order). You do not need to expand expressions involving $C$, such as $CC^T$ or $C^\dagger$, in terms of $A$ and $B$; you are welcome to simply give your answers using $C$. You may assume that $C$ is wide and has independent rows.

(b) Show that $u_t$ (the $t$th input in the sequence found in part (a)) can be expressed as $u_t = K_t x_1$, where $K_t$ is an $m \times n$ matrix. Show how to find $K_t$ from $A$ and $B$. (But feel free to use the matrix $C$ in your answer.)

*Hint.* Your expression for $K_t$ can include submatrices of $C$ or $C^\dagger$.

(c) *A constant linear regulator.* A very common regulator strategy is to simply use $u_t = K_1 x_t$ for all $t$, $t = 1, 2, 3, \ldots$. This is called a (constant) *linear regulator*, and $K_1$ is called the *state feedback gain* (since it maps the current state into the control input). Using this control strategy can be interpreted as always carrying out the first step of minimum energy control, as if we were going to steer the state to zero $T$ steps in the future. This choice of input does not yield regulation in $T$ steps, but it typically achieves *asymptotic regulation*, which means that $x_t \to 0$ as $t \to \infty$.

Find the state feedback gain $K_1$ for the specific system with

$$A = \left[\begin{array}{ccc} 1.003 & 0 & -0.008 \\ 0.005 & .997 & 0 \\ 0 & 0.005 & 1.002 \end{array}\right], \quad B = \left[\begin{array}{cc} 1 & 0 \\ 4 & 5 \\ 6 & 2 \end{array}\right],$$

using $T = 10$. You may find the code in `regulation.jl` useful.

(d) Simulate the system given in part (c) from several choices of initial state $x_1$, under two conditions: *open-loop*, which means $u_t = 0$, and *closed-loop*, which means $u_t = K_1 x_t$, where $K_1$ is the state feedback gain found in part (c). Use `regulation.jl` to plot $x$.

## 18 Nonlinear least squares

**18.1** *Airplane steady flight.* Consider an airplane flying at constant speed $S$, along a straight line with *flight path angle* $\gamma$ (in radians, with respect to horizontal; $\gamma > 0$ means climbing). The other relevant quantities are $\alpha$, the *angle of attack*, which is the angle between the airplane body and the airplane flight path, the four forces *lift* $L$, *drag* $D$, engine *thrust* $T$, and airplane weight $W$. These are related by the *steady flight equations*,

$$T = W\gamma + D, \qquad L = W$$

which state that the horizontal and vertical forces on the airplane balance, *i.e.*, sum to zero. The lift and drag forces depend on the angle of attack and speed,

$$L = a_L C_L(\alpha) S^2, \qquad D = a_D C_D(\alpha) S^2,$$

where $a_L$ and $a_D$ are known constants, $C_L$ is the *coefficient of lift function*, and $C_D$ is the *coefficient of drag function*. These are well approximated as affine functions,

$$C_L(\alpha) = C_L^{(0)} + \alpha C_L^{(1)}, \qquad C_D(\alpha) = C_D^{(0)} + \alpha C_D^{(1)},$$

where the four constants $C_L^{(0)}, C_L^{(1)}, C_D^{(0)}, C_D^{(1)}$ are known. The airplane weight $W$ is known and constant; we can control the engine thrust $T$ and the angle of attack $\alpha$ (via the airplane control surfaces).

(a) Given a desired speed $S^{\mathrm{des}}$ and a desired flight path angle $\gamma^{\mathrm{des}}$, how would you determine the thrust $T$ and angle of attack $\alpha$. Explain how this can be done analytically. (You do not have to give formulas for $T$ and $\alpha$.)

(b) *Glide path.* Suppose that $T = 0$, *i.e.*, the engines are off. the flight path angle $\gamma$ and the speed $S$.

(c) Now suppose that we fix $T$ and $\alpha$. Use the Levenberg-Marquardt algorithm to compute the resulting speed $S$ and flight path angle $\gamma$.

**18.2** *Lambert $W$-function.* The *Lambert $W$-function*, denoted $W : [0, \infty) \to \mathbf{R}$, is defined as $z = W(u)$, where $z$ is the unique number $z \geq 0$ for which $ze^z = u$. (The notation just means that we restrict the argument $z$ to be nonnegative.) The Lambert function arises in a variety of applications. There is no analytical formula for $W(u)$; it must be computed numerically. In this exercise you will develop a solver to compute $W(u)$, given a nonnegative number $u$, using the Levenberg-Marquardt algorithm, by minimizing $(ze^z - u)^2$ over $z$.

(a) Suppose the current iterate is $z^{(k)}$ (which you can assume to be nonnegative). Give a formula for $z^{(k+1)}$ in terms of $z^{(k)}$ and $\lambda^{(k)}$, the current value of the regularization parameter in the Levenberg-Marquardt algorithm.

(b) Implement the Levenberg-Marquardt algorithm. You can start with $z^{(1)} = 1$ and $\lambda^{(1)} = 1$ (but it should work with other initializations). You can stop the algorithm when $\left| z^{(k)} e^{z^{(k)}} - u \right|$ is small (say, less than $10^{-6}$). Test your algorithm by evaluating $W(1)$, $W(2)$, and $W(3)$. Verify that these numbers (almost) satisfy $W(i)e^{W(i)} = i$, for $i = 1, 2, 3$. If you like, you can check the values you compute against those computed using the `LambertW` Julia package.

## 19 Constrained nonlinear least squares

**19.1**

## 20 Julia

**20.1** *Checking the MMA functions.* Check if the following MMA functions work by checking them on values. rms, avg, std.

**20.2** *Angle and distance.* Generate two random 10-vectors $a$ and $b$ and calculate the angle and distance between $a$ and $b$.

**20.3** *Triangle inequality.* Check the triangle inequality by generating three random 10-vectors $a$, $b$, and $c$, and comparing $\mathbf{dist}(a, c)$ with $\mathbf{dist}(a, b) + \mathbf{dist}(b, c)$.

**20.4** *Chebyshev inequality.* Generate a random 200-vector $x$. Verify that no more than 8 of the entries of $x$ satisfy $|x_i| \geq 5\,\mathbf{rms}(x)$.

**20.5** *Verifying Cauchy-Schwarz in Julia.* Generate two 20-vectors $x$ and $y$. Compute $|x^T y|$ and $\|x\|\|y\|$ and verify that the Cauchy-Schwarz inequality holds.

**20.6** *Linear independence.* Are the following sets of vectors linearly dependent or linearly independent? Give a simple argument showing dependence or independence, and also check your answers using the Julia function `gram_schmidt()` in `gramschmidt.jl`.

The `gram_schmidt()` function accepts a list of vectors ($a_i$ in our notation in the slides and book), and returns a list of vectors ($q_i$ in our notation). So,
`length(gram_schmidt(vectors))==length(vectors)` is TRUE when the vectors are linearly independent.

For example, the following code checks if the vectors $(1, 0), (1, 0), (0, 1)$ are linearly independent:

```
include("gramschmidt.jl")
using GramSchmidt

vectors = {[1,0] [1,0] [0,1]}
gram_schmidt(vectors)

Out[19]:
 1-element Array{Any,1}:
  [1.0,0.0]
```

The output was only the vector $(1, 0)$, so the vectors are not linearly independent.

(a) $(1, -1.1)$, $(-2.8, -0.3)$, $(-0.4, 1.5)$.

(b) $(1, 0, 1, 0, 1)$, $(0, 1, 0, 0, 1)$, $(0, 1, 0, 1, 0)$.

(c) $(1, 2, 0)$, $(-2, 0, 3)$, $(1, 0, 2)$.

**20.7** *Matrices.* In each part below, use Julia to create the matrix $A$, described by its effect on an arbitrary vector $x$ as $y = Ax$. Check that the matrix you create has the correct behavior when it multiplies a randomly generated vector $x$.

(a) $x$ is a 5-vector; $y$ is a 3-vector containing the first 3 components of $x$.

(b) $x$ and $y$ are 7-vectors; $y$ has the entries of $x$ in reverse order.

(c) $x$ is a 10-vector, and $y$ is the 9-vector of differences of consecutive entries of $x$:

$$y = (x_2 - x_1, x_3 - x_2), \ldots, x_{10} - x_9).$$

(d) $x$ and $y$ are 8-vectors, and $y_i$ is the difference between $x_i$ and the average of $x_1, \ldots, x_i$.

**20.8** *Block matrix.* Generate four 3 by 5 matrices $A$, $B$, $C$, and $D$.

(a) Verify that

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^T$$

is equal to

$$\begin{bmatrix} A^T & B^T \\ C^T & D^T \end{bmatrix}.$$

(b) Verify that

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

is equal to

$$\begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}.$$

**20.9** *Column-major order.*

`reshape` changes the dimension of a matrix by moving its entries. It can be used, for example, to vectorize matrices. There are multiple ways this could occur, but here we examine Julia's implementation.

(a) Reshape the matrix

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

into a (column) vector. Note the order of the entries, and briefly explain what `reshape` is doing.

(b) Now reshape the same matrix into a $3 \times 2$ matrix. Are the entries where you expect?

(c) In a single line of code, express the matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

using only range construction (*i.e.*, `1:10`), reshape, and transpose.

**20.10** *Taylor approximations.* For each of the following functions below, plot the function, $f$ over the interval $[-10, 10]$. Then, on the same graph, plot the function's Taylor approximation near 0.

(a) $f(x) = x^2 + x$.

(b) $f(x) = 0.5x^10 + 0.3x^2 + x$.

(c) $f(x) = \sin(x)$.

**20.11** *KKT and least squares.* Solve the following least-norm problem

$$\begin{array}{ll} \text{minimize} & \|x\|^2 \\ \text{subject to} & Ax = b \end{array}$$

using LLS, and then again using KKT and backslash. Confirm that your solutions are nearly identical, and that both satisfy the constraint (up to numerical error). You may generate $A$ and $b$ using the following code.

```
srand(0); A = randn(3,5); b = randn(3,1);
```

**20.12** *Checking a classifier.* Write a Julia script that computes the false positive and false negative rates for a given data set, with a given linear classifier.

**20.13** *Plotting in Julia.* Create the 100-vectors $x$ and $y$ in Julia where

$$x_i = \sin(i/8), \quad y_i = \begin{cases} 0 & i \leq 25 \\ (i-25)/50 & 25 < i < 75 \\ 1 & i \geq 75 \end{cases}.$$

Using JuliaBox and PyPlot, plot $x$. On a second figure, plot $x$, $y$ and $x+y$. Use different colors to distinguish the three lines, and add labels to the lines.

# 21 Miscellaneous

**21.1** *Appropriate response.* For each of the situations (a), (b), and (c) described below, circle the most appropriate response from among the five choices. You can circle only one for each situation.

(a) An intern working for you develops several different models to predict the daily demand for a product. How should you choose which model is the best one, the one to put into production?

*Laplacian* *Magic* *Regularization* *Validation* *k-means*

(b) As an intern you develop an auto-regressive model to predict tomorrow's sales volume. It works very well, making predictions that are typically within 5% of the actual sales volume. Your boss, who has an MBA and is not particularly interested in mathematical details, asks how your predictor works. How do you respond?

*Laplacian* *Magic* *Regularization* *Validation* *k-means*

(c) A colleague needs a quantitative measure of how rough an image is, *i.e.*, how much adjacent pixel values differ. What do you suggest?

*Laplacian* *Magic* *Regularization* *Validation* *k-means*