Exercises for EE133A (Spring Quarter 2016)

L. Vandenberghe

Contents

1 Vectors

1.1 Which of the following scalar valued functions on $\mathbf{R}^n$ are linear? Which are affine? If a function is linear, give its inner product representation, *i.e.*, an n-vector a such that $f(x) = a^T x$ for all x. If it is affine, give a and b such that $f(x) = a^T x + b$ holds for all x. If it is neither, give specific x, y, α, and β for which superposition fails, *i.e.*,

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

(Provided $\alpha + \beta = 1$, this shows the function is neither linear nor affine.)

(a) The spread of values of the vector, defined as $f(x) = \max_k x_k - \min_k x_k$.

(b) The difference of the last element and the first, $f(x) = x_n - x_1$.

(c) The difference of the squared distances to two fixed vectors c and d, defined as

$$f(x) = \|x - c\|^2 - \|x - d\|^2.$$

(d) The median of an n-vector, defined as the middle value of the sorted vector, when n is odd, and the average of the two middle values in the sorted vector, when n is even.

1.2 Let $f : \mathbf{R}^n \to \mathbf{R}$ be an affine function. Show that f can be expressed as $f(x) = a^T x + b$ for some $a \in \mathbf{R}^n$ and $b \in \mathbf{R}$.

1.3 A unit mass moves on a straight line (in one dimension). The position of the mass at time t is denoted by $s(t)$, and its derivatives (the velocity and acceleration) by $s'(t)$ and $s''(t)$. The position as a function of time can be determined from Newton's second law

$$s''(t) = F(t),$$

where $F(t)$ is the force applied at time t, and the initial conditions $s(0)$, $s'(0)$. We assume $F(t)$ is piecewise-constant, and is kept constant in intervals of one second. The sequence of forces $F(t)$, for $0 \leq t < 10$s, can then be represented by a 10-vector x, with

$$F(t) = x_k, \quad k - 1 \leq t < k.$$

(a) Suppose the initial position and velocity are zero ($s(0) = s'(0) = 0$). Derive expressions for the velocity $s'(10)$ and position $s(10)$ at time $t = 10$. Show that $s(10)$ and $s'(10)$ are linear functions of x.

(b) How does the answer change if we start from nonzero initial position or velocity?

1.4 *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 $a \in \mathbf{R}^n$.

1.5 The temperature T of an electronic device containing three processors is an affine function of the power dissipated by the three processors, $P = (P_1, P_2, P_3)$. When all three processors are idling, we have $P = (10, 10, 10)$, which results in a temperature $T = 30$. When the first processor operates at full power and the other two are idling, we have $P = (100, 10, 10)$, and the temperature rises to $T = 60$. When the second processor operates at full power and the other two are idling, we have $P = (10, 100, 10)$ and $T = 70$. When the third processor operates at full power and the other two are idling, we have $P = (10, 10, 100)$ and $T = 65$. Now suppose that all three processors are operated at the same power P. How large can P be, if we require that $T \leq 85$?

1.6 Verify that the following identities hold for any two vectors a and b of the same size.

(a) $(a+b)^T(a-b) = \|a\|^2 - \|b\|^2$.

(b) $\|a+b\|^2 + \|a-b\|^2 = 2(\|a\|^2 + \|b\|^2)$.

1.7 When does the triangle inequality hold with equality, *i.e.*, what are the conditions on a and b to have $\|a+b\| = \|a\| + \|b\|$?

1.8 Show that $\|a+b\| \geq \big|\|a\| - \|b\|\big|$.

1.9 *Norm of linear combination of orthonormal vectors.* Suppose $\{a_1, \ldots, a_k\}$ is an orthonormal set of n-vectors, and $x = \beta_1 a_1 + \cdots + \beta_k a_k$. Express $\|x\|$ in terms of β_1, $\ldots$, β_k.

1.10 *Relative deviation between vectors.* Suppose a and b are nonzero vectors of the same size. The relative deviation of b from a is defined as the distance between a and b, divided by the norm of a,

$$\eta_{ab} = \frac{\|a-b\|}{\|a\|}.$$

This is often expressed as a percentage.

The relative deviation is not a symmetric function of a and b; in general, $\eta_{ab} \neq \eta_{ba}$. Suppose $\eta_{ab} = 0.1$ (*i.e.*, 10%). How big and how small can be η_{ba} be? Explain your reasoning.

1.11 *Average and norm.* Use the Cauchy-Schwarz inequality to prove that

$$-\frac{1}{\sqrt{n}}\|x\| \leq \frac{1}{n}\sum_{i=1}^{n} x_i \leq \frac{1}{\sqrt{n}}\|x\|$$

for all n-vectors x. In other words, the average of the elements of a vector lies between $\pm 1/\sqrt{n}$ times its norm. What are the conditions on x to have equality in the upper bound? When do we have equality in the lower bound?

1.12 Use the Cauchy-Schwarz inequality to prove that

$$\frac{1}{n}\sum_{k=1}^{n} x_k \geq \left(\frac{1}{n}\sum_{k=1}^{n}\frac{1}{x_k}\right)^{-1}$$

for all n-vectors x with positive elements x_k.

The left-hand side of the inequality is the arithmetic mean (average) of the numbers x_k; the right-hand side is called the harmonic mean.

1.13 *Cauchy-Schwarz inequality for complex vectors.* We generalize the proof of the Cauchy-Schwarz inequality in lecture 2 to complex vectors. We show that the inequality

$$|b^H a| \leq \|a\|\|b\|$$

holds for all complex n-vectors a, b, *i.e.*,

$$\left|\bar{b}_1 a_1 + \cdots + \bar{b}_n a_n\right| \leq \left(|a_1|^2 + \cdots + |a_n|^2\right)^{1/2}\left(|b_1|^2 + \cdots + |b_n|^2\right)^{1/2}.$$

This is obviously true if $a = 0$ or $b = 0$, so we will assume that $a \neq 0$ and $b \neq 0$.

(a) First assume that $\|a\| = \|b\| = 1$. Define $\alpha = \arg(b^H a)$ (where arg denotes phase angle), so that $b^H a = |b^H a| e^{\mathrm{j}\alpha}$. Show that

$$\|a - e^{\mathrm{j}\alpha} b\|^2 = 2(1 - |b^H a|).$$

Since the left-hand side is nonnegative, this shows that $|b^H a| \leq 1$. What are the conditions on a and b to have equality $|b^H a| = 1$?

(b) Use the result in part (a) to show that $|b^H a| \leq \|a\|\|b\|$ when $a \neq 0$, $b \neq 0$. What are the conditions on a and b to have equality $|a^H b| = \|a\|\|b\|$?

1.14 *Euclidean norm of sum.* Derive a formula for $\|a+b\|$ in terms of $\|a\|$, $\|b\|$, and $\theta = \angle(a,b)$. Use this formula to show the following:

(a) $a \perp b$ if and only if $\|a+b\| = \sqrt{\|a\|^2 + \|b\|^2}$.

(b) a and b make an acute angle if and only if $\|a+b\| \geq \sqrt{\|a\|^2 + \|b\|^2}$.

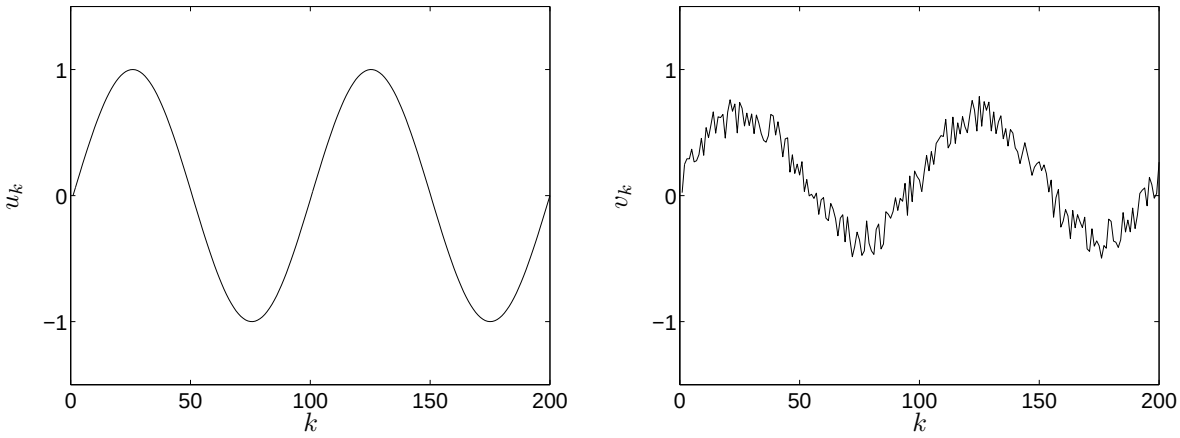
(c) a and b make an obtuse angle if and only if $\|a+b\| \leq \sqrt{\|a\|^2 + \|b\|^2}$.

Draw a picture illustrating each case (in $\mathbf{R}^2$).

1.15 *Decoding using inner products.* An input signal is sent over a noisy communication channel. The channel adds a small noise to the input signal, and attenuates it by an unknown factor α. We represent the input signal as an n-vector u, the output signal as an n-vector v, and the noise signal as an n-vector w. The elements u_k, v_k, w_k give the values of the signals at time k. The relation between u and v is

$$v = \alpha u + w.$$

The two plots below show an example with $\alpha = 0.5$.



Now suppose we know that the input signal u was chosen from a set of four possible signals

$$x^{(1)},\ x^{(2)},\ x^{(3)},\ x^{(4)} \in \mathbf{R}^n.$$

We know these signals $x^{(i)}$, but we don't know which one was used as input signal u. Our task is to find a simple, automated way to estimate the input signal, based on the received signal v. There are many ways to do this. One possibility is to calculate the angle θ_k between v and $x^{(k)}$, via the formula

$$\cos\theta_k = \frac{v^T x^{(k)}}{\|v\|\,\|x^{(k)}\|}$$

and pick the signal $x^{(k)}$ that makes the smallest angle with v.

Download the file `decoding.m` from the class webpage, save it in your working directory, and execute it in MATLAB using the command `[x1, x2, x3, x4, v] = decoding`. The first four output arguments are the possible input signals $x^{(k)}$, $k = 1, 2, 3, 4$. The fifth output argument is the received (output) signal v. The length of the signals is $n = 200$.

(a) Plot the vectors v, $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, $x^{(4)}$. Visually, it should be obvious which input signal was used to generate v.

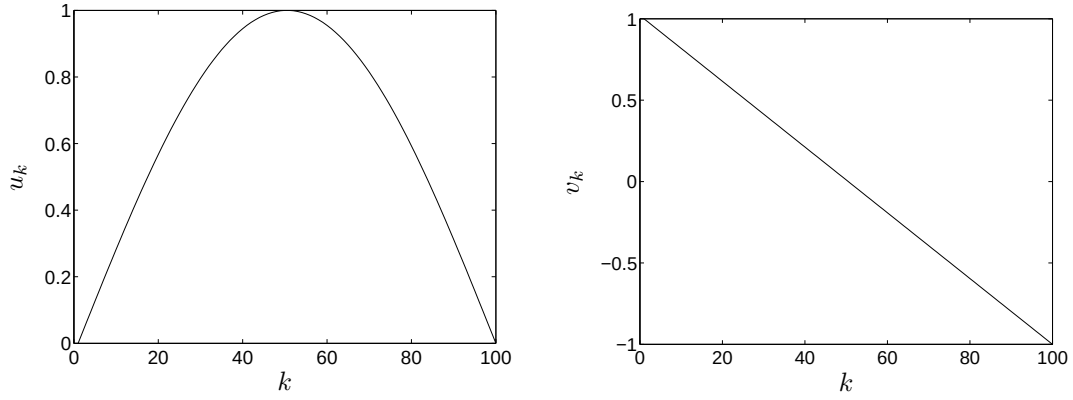
(b) Calculate the angles of v with each of the four signals $x^{(k)}$. This can be done in MATLAB using the command

```
acos( (v'*x) / ( norm(x) * norm(v) ))
```

(which returns the angle between x and v in radians). Which signal $x^{(k)}$ makes the smallest angle with v? Does this confirm your conclusion in part (a)?

1.16 *Multiaccess communication.* A communication channel is shared by several users (transmitters), who use it to send binary sequences (sequences with values $+1$ and -1) to a receiver. The following technique allows the receiver to separate the sequences transmitted by each transmitter. We explain the idea for the case with two transmitters.

We assign to each transmitter a different signal or *code*. The codes are represented as n-vectors u and v: u is the code for user 1, v is the code for user 2. The codes are chosen to be orthogonal ($u^T v = 0$). The figure shows a simple example of two orthogonal codes of length $n = 100$.



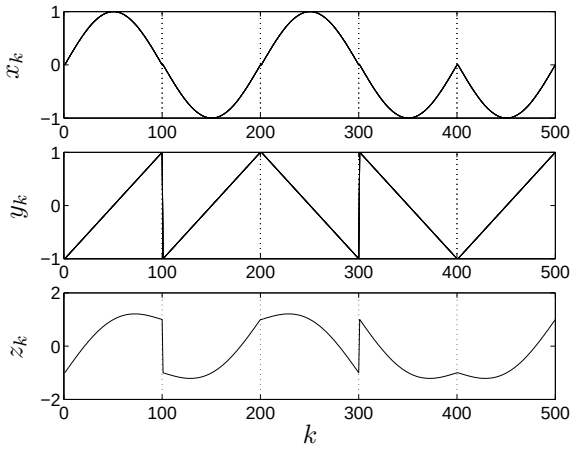
Suppose user 1 wants to transmit a binary sequence b_1, b_2, …, b_m (with values $b_i = 1$ or $b_i = -1$), and user 2 wants to transmit a sequence c_1, c_2, …, c_m (with values $c_i = 1$ or $c_i = -1$). From these sequences and the user codes, we construct two signals x and y, both of length mn, as follows:

$$x = \begin{bmatrix} b_1 u \\ b_2 u \\ \vdots \\ b_m u \end{bmatrix}, \qquad y = \begin{bmatrix} c_1 v \\ c_2 v \\ \vdots \\ c_m v \end{bmatrix}.$$

(Note that here we use block vector notation: x and y consist of m blocks, each of size n. The first block of x is $b_1 u$, the code vector u multiplied with the scalar b_1, etc.) User 1 sends the signal x over the channel, and user 2 sends the signal y. The receiver receives the sum of the two signals. We write the received signal as z:

$$z = x + y = \begin{bmatrix} b_1 u \\ b_2 u \\ \vdots \\ b_m u \end{bmatrix} + \begin{bmatrix} c_1 v \\ c_2 v \\ \vdots \\ c_m v \end{bmatrix}.$$

The figure shows an example where we use the two code vectors u and v shown before. In this example $m = 5$, and the two transmitted sequences are $b = (1, -1, 1, -1, -1)$ and $c = (-1, -1, 1, 1, -1)$.



How can the receiver recover the two sequences b_k and c_k from the received signal z? Let us denote the first block (consisting of the first n values) of the received signal z as $z^{(1)}$: $z^{(1)} = b_1 u + c_1 v$. If we make the inner product of $z^{(1)}$ with u, and use the fact that $u^T v = 0$, we get

$$u^T z^{(1)} = u^T (b_1 u + c_1 v) = b_1 u^T u + c_1 u^T v = b_1 \|u\|^2.$$

Similarly, the inner product with v gives

$$v^T z^{(1)} = v^T (b_1 u + c_1 v) = b_1 v^T u + c_1 v^T v = c_1 \|v\|^2.$$

We see that b_1 and c_1 can be computed from the received signal as

$$b_1 = \frac{u^T z^{(1)}}{\|u\|^2}, \qquad c_1 = \frac{v^T z^{(1)}}{\|v\|^2}.$$

Repeating this for the other blocks of z allows us to recover the rest of the sequences b and c:

$$b_k = \frac{u^T z^{(k)}}{\|u\|^2}, \qquad c_k = \frac{v^T z^{(k)}}{\|v\|^2},$$

if $z^{(k)}$ is the kth block of z.

Download the file `multiaccess.m` from the class webpage, and run it in MATLAB as `[u,v,z] = multiaccess`. This generates two code vectors u and v of length $n = 100$, and a received signal z of length $mn = 500$. (The code vectors u and v are different from those used in the figures above.) In addition we added a small noise vector to the received signal z, *i.e.*, we have

$$z = \begin{bmatrix} b_1 u \\ b_2 u \\ \vdots \\ b_m u \end{bmatrix} + \begin{bmatrix} c_1 v \\ c_2 v \\ \vdots \\ c_m v \end{bmatrix} + w$$

where w is unknown but small compared to u and v.

(a) Calculate the angle between the code vectors u and v. Verify that they are nearly (but not quite) orthogonal. As a result, and because of the presence of noise, the formulas for b_k and c_k are not correct anymore. How does this affect the decoding scheme? Is it still possible to compute the binary sequences b and c from z?

(b) Compute $(b_1, b_2, b_3, b_4, b_5)$ and $(c_1, c_2, c_3, c_4, c_5)$.

1.17 *Approximating one vector with a scalar multiple of another.* Suppose we have a nonzero vector x and a vector y of the same size.

(a) How do you choose the scalar t so that $\|tx-y\|$ is minimized? Since $L = \{tx \mid t \in \mathbf{R}\}$ describes the line passing through the origin and the point x, the problem is to find the point in L closest to the point y. *Hint.* Work with $\|tx-y\|^2$.

(b) Let $t^\star$ be the value found in part (a), and let $z = t^\star x$. Show that $(y-z) \perp x$.

(c) Draw a picture illustrating this exercise, for 2-vectors. Show the points x and y, the line L, the point z, and the line segment between y and z.

(d) Express $d = \|z-y\|$, which is the closest distance of a point on the line L and the point y, in terms of $\|y\|$ and $\theta = \angle(x,y)$.

1.18 *Minimum distance between two lines.* Find the minimum distance between the lines

$$L = \{u + tv \mid t \in \mathbf{R}\}, \qquad \tilde{L} = \{\tilde{u} + \tilde{t}\tilde{v} \mid \tilde{t} \in \mathbf{R}\}.$$

The vectors u, v, $\tilde{u}$, and $\tilde{v}$ are given n-vectors with $v \neq 0$, $\tilde{v} \neq 0$.

1.19 *Regression line.* Let a, b be two real n-vectors. To simplify notation we write the vector averages as

$$m_a = \mathbf{avg}(a) = \frac{\mathbf{1}^T a}{n}, \qquad m_b = \mathbf{avg}(b) = \frac{\mathbf{1}^T b}{n},$$

and their standard deviations as

$$s_a = \mathbf{std}(a) = \frac{1}{\sqrt{n}}\|a - m_a\mathbf{1}\|, \qquad s_b = \mathbf{std}(b) = \frac{1}{\sqrt{n}}\|b - m_b\mathbf{1}\|.$$

We assume the vectors are not constant ($s_a \neq 0$ and $s_b \neq 0$) and write the correlation coefficient as

$$\rho = \frac{1}{n}\frac{(a - m_a\mathbf{1})^T(b - m_b\mathbf{1})}{s_a s_b}.$$

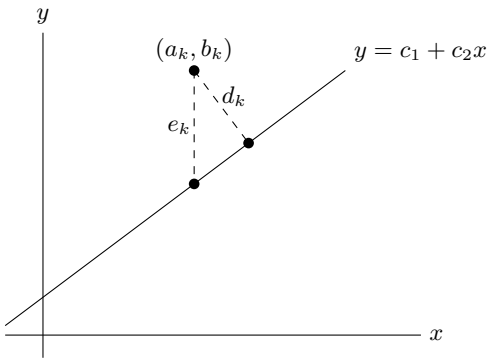
In lecture 2, we considered the problem of fitting a straight line to the points (a_k, b_k), by minimizing

$$J = \frac{1}{n}\sum_{k=1}^{n}(c_1 + c_2 a_k - b_k)^2 = \frac{1}{n}\|c_1\mathbf{1} + c_2 a - b\|^2.$$

We found that the optimal coefficients are $c_2 = \rho s_b/s_a$ and $c_1 = m_b - m_a c_2$. Show that for those values of c_1 and c_2, we have

$$J = (1-\rho^2)s_b^2.$$

1.20 *Orthogonal distance regression.* We use the same notation as in exercise 1.19: a, b are non-constant n-vectors, with means m_a, m_b, standard deviations s_a, s_b, and correlation coefficient ρ.



For each point (a_k, b_k), the vertical deviation from the straight line defined by $y = c_1 + c_2 x$ is given by

$$e_k = |c_1 + c_2 a_k - b_k|.$$

The least squares regression method of the lecture minimizes the sum $\sum_k e_k^2$ of the squared vertical deviations. The orthogonal (shortest) distance of (a_k, b_k) to the line is

$$d_k = \frac{|c_1 + c_2 a_k - b_k|}{\sqrt{1 + c_2^2}}.$$

As an alternative to the least squares method, we can find the straight line that minimizes the sum of the squared orthogonal distances $\sum_k d_k^2$. Define

$$J = \frac{1}{n} \sum_{k=1}^{n} d_k^2 = \frac{\|c_1 \mathbf{1} + c_2 a - b\|^2}{n(1 + c_2^2)}.$$

(a) Show that the optimal value of c_1 is $c_1 = m_b - m_a c_2$, as for the least squares fit.

(b) If we plug in the optimal value of c_1, we obtain

$$J = \frac{\|c_2(a - m_a \mathbf{1}) - (b - m_b \mathbf{1})\|^2}{n(1 + c_2^2)}$$

Simplify this expression and show that it is equal to

$$J = \frac{s_a^2 c_2^2 + s_b^2 - 2\rho s_a s_b c_2}{1 + c_2^2}.$$

Set the derivative of J with respect to c_2 to zero, to derive a quadratic equation for c_2:

$$\rho c_2^2 + \left(\frac{s_a}{s_b} - \frac{s_b}{s_a} \right) c_2 - \rho = 0.$$

If $\rho = 0$ and $s_a = s_b$, any value of c_2 is optimal. If $\rho = 0$ and $s_a \neq s_b$ the quadratic equation has a unique solution $c_2 = 0$.

If $\rho \neq 0$, the quadratic equation has a positive and a negative root. Show that the solution that minimizes J is the root c_2 with the same sign as ρ.

(c) Download the file `orthregdata.m` and execute it in MATLAB to create two arrays `a`, `b` of length 100. Fit a straight line to the data points (a_k, b_k) using orthogonal distance regression and compare with the least squares solution. Make a MATLAB plot of the two lines and the data points.

1.21 *Distance between parallel hyperplanes.* Consider two hyperplanes

$$H_1 = \{x \in \mathbf{R}^n \mid a^T x = b\}, \qquad H_2 = \{y \in \mathbf{R}^n \mid a^T y = c\},$$

where a is a nonzero n-vector, and b and c are scalars. The hyperplanes are parallel because they have the same coefficient vectors a. The distance between the hyperplanes is defined as

$$d = \min_{\substack{x \in H_1 \\ y \in H_2}} \|x - y\|.$$

Which of the following three expressions for d is correct? Explain your answer.

$$\text{(a)} \quad d = |b - c|, \qquad \text{(b)} \quad d = \frac{|b - c|}{\|a\|}, \qquad \text{(c)} \quad d = \frac{|b - c|}{\|a\|^2}.$$

1.22 *The k-means algorithm.* In this exercise we implement the clustering example described on page 58 of the textbook.

Download the file `mnist_train.mat` from the course website and load it in MATLAB or Octave using the command `load mnist_train`. This creates two variables: a 784×60000 matrix `digits` and a 1×60000 matrix `labels`. We will not need `labels`. Each column of `digits` is a 28×28 grayscale image, stored as a vector of length $28^2 = 784$ with elements between 0 and 1 (0 denotes a black pixel and 1 a white pixel). Figure 4.6 in the book shows the first 25 images. To display the image in the ith column of `digits` you can use the commands

```
X = reshape(digits(:,i), 28, 28);
imshow(X);
```

The first line converts column i of `digits` to a 28×28 matrix. The second command displays the matrix as an image. To speed up the computations we will use only the first 10000 digits:

```
digits = digits(:, 1:10000);
```

You are asked to apply the k-means algorithm to this set of $N = 10000$ vectors, with $k = 20$ groups, and starting from a random initial group assignment (as opposed to starting from 20 randomly generated group representatives, as in Algorithm 4.1 (page 54).

In the following list of MATLAB hints and comments we assume that the 20 group representatives are stored as columns of a 784×20 matrix `Z`, and that the group assignment is represented by a 1×10000 matrix (*i.e.*, row vector) `group`. The ith element `group(i)` is an integer between 1 and 20, with the index of the group that column i of `digits` is assigned to.

- You can create an initial group assignment using the `randi` function:

  ```
  group = randi(20, 1, 10000);
  ```

  This generates a pseudorandom 1×10000 matrix of integers between 1 and 20.

- Since we start from a random partition, the order of the two steps in algorithm 4.1 is switched. The first step in each cycle is to compute the group representatives for the current assignment. We can find the columns of `digits` that are assigned to group i using the function `find`. The command

  ```
  I = find(group == i);
  ```

  defines an index vector I such that `digits(:,I)` is the submatrix of `digits` with the columns assigned to class i. To find the average of the columns in this matrix, you can use for-loops, or the MATLAB functions `sum` or `mean`. (Be sure to look up what `sum` or `mean` do when applied to a matrix; see `help sum` and `help mean`.)

- We evaluate the quality of the clustering using the clustering objective

  $$J = \frac{1}{N} \sum_{i=1}^{N} \min_{j=1,\ldots,k} \|x_i - z_j\|^2.$$

  The algorithm is terminated when J is nearly equal in two successive iterations (*e.g.*, we terminate when $|J - J_{\text{prev}}| \leq 10^{-5} J$, where J_{prev} is the value of J after the previous iteration).

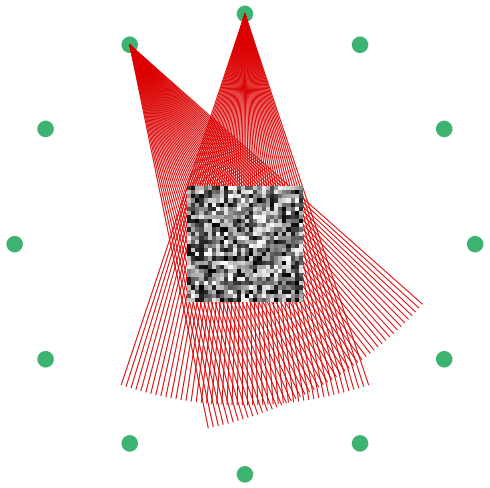
- After running the k-means algorithm you can display the representative vectors of the 20 groups as follows:

  ```
  for k=1:20
      subplot(4,5,k)
      imshow(reshape(Z(:,k), 28, 28));
  end
  ```

This produces a figure similar to figure 4.7 in the textbook. Your results will be different because figure 4.7 was computed using the full set of 60000 digits and, moreover, the result of the k-means algorithm depends on the starting point.

- Include the MATLAB code and a figure of a typical set of group representatives with your solution.

1.23 *Tomography.* Download the file `tomography.mat` from the class webpage and load it in MATLAB using the command `load tomography`. This creates a matrix A of size 576×784 and a vector b of length 576. The matrix A and the vector b describe a toy tomography example and were constructed using the MATLAB AIR Tools package that can be found at `www2.compute.dtu.dk/~pcha/AIRtools`. (You do not need this package for the exercise.) The geometry is shown in the figure below.



The image at the center is a black-and-white image of size 28×28. The figure shows a random image but in the actual problem we used one of the images of handwritten digits of exercise 1.22. The green dots are twelve source locations. For each source location we generate 48 rays emanating from the source. (The figure shows the rays for two sources only.) For each ray, we calculate the line integral of the pixel intensities along the ray. This gives $12 \cdot 48 = 576$ linear equations

$$\sum_{j=1}^{784} A_{ij} x_j = b_i, \quad i = 1, \ldots, 576.$$

Here x_j denotes the intensity in pixel j of the image (images are stored as vectors of length 784, as in homework 1), A_{ij} is the length of the intersection of ray i with pixel j, and b_i is the value of the line integral along ray i. These equations can be written in matrix form as $Ax = b$ where A and b are the data in `tomography.mat`, or as

$$a_i^T x = b_i, \quad i = 1, \ldots, 576,$$

where a_i^T is row i of A.

The purpose is to reconstruct the image x from the line integral measurements. We will use Kaczmarz's iterative algorithm for this purpose. Note however that this is a small problem and very easy to solve using the standard non-iterative methods that are discussed later in the course.

Kaczmarz's algorithm starts at an arbitrary point x (for example, a zero vector) and then cycles through the equations. At each iteration we take a new equation, and update x by replacing it with its projection on the hyperplane defined by the equation. If K is the number of cycles and $m = 576$ is the number of equations, the algorithm can be summarized as follows.

Initialize x.
For $k = 1, \ldots, K$:
    For $i = 1, \ldots, m$:
        Project x on the ith hyperplane:
            $$x := x - \frac{a_i^T x - b_i}{\|a_i\|^2} a_i.$$
    end
end

Run the algorithm for $K = 10$ cycles, starting at $x = 0$ (a black image). Compute the error $\|Ax - b\|/\|b\|$ after each cycle. Also display the reconstructed image x (using the command `imshow(reshape(x, 28, 28))`).

2 Matrices

2.1 *Block matrix notation.* Consider the block matrix

$$A = \begin{bmatrix} I & B & 0 \\ B^T & 0 & 0 \\ 0 & 0 & BB^T \end{bmatrix}$$

where B is 10×5. What are the dimensions of the zero matrices and the identity matrix in the definition of A, and of A itself?

2.2 *Shift matrices.*

(a) Give a simple description in words of the function $f(x) = Ax$, where A is the 5×5-matrix

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

(b) Same question for $f(x) = A^k x$, for $k = 2, 3, 4, 5$. (A^k is the kth power of A: $A^2 = AA$, $A^3 = A^2A = AAA$, etc.)

(c) Same question for $f(x) = A^T x$.

2.3 *Permutation matrices.* A square matrix A is called a *permutation matrix* if it satisfies the following three properties (lecture 3).

- All elements of A are either zero or one.
- Each column of A contains exactly one element equal to one.
- Each row of A contains exactly one element equal to one.

Suppose A is a permutation matrix. We define two linear functions $f(x) = Ax$ and $g(x) = A^T x$. What is the relation between f and g?

2.4 The *cross-product* $a \times x$ of two 3-vectors $a = (a_1, a_2, a_3)$ and $x = (x_1, x_2, x_3)$ is defined as the vector

$$a \times x = \begin{bmatrix} a_2x_3 - a_3x_2 \\ a_3x_1 - a_1x_3 \\ a_1x_2 - a_2x_1 \end{bmatrix}.$$

(a) Assume a is fixed and nonzero. Show that the function $f(x) = a \times x$ is a linear function of x, by giving a matrix A that satisfies $f(x) = Ax$ for all x.

(b) Verify that $A^T A = (a^T a)I - aa^T$.

(c) Show that for nonzero x,

$$\|a \times x\| = \|a\| \, \|x\| \, |\sin\theta|$$

where θ is the angle between a and x.

2.5 *Projection of a vector x on a given vector y.* Let y be a given n-vector, and consider the function $f : \mathbf{R}^n \to \mathbf{R}^n$, defined as

$$f(x) = \frac{x^T y}{\|y\|^2} y.$$

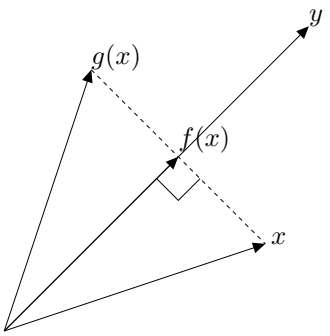
In exercise 1.17 it is shown that that $f(x)$ is the projection of x on the (fixed, given) vector y.

Is f a linear function of x? If your answer is yes, give an $n \times n$-matrix A such that $f(x) = Ax$ for all x. If your answer is no, show with an example that f does not satisfy the definition of linearity $f(\alpha u + \beta v) = \alpha f(u) + \beta f(v)$.

2.6 *Reflection of a vector about a given vector y.* As in exercise 2.5, let $f(x)$ be the projection of x on a given n-vector y. Define $g : \mathbf{R}^n \to \mathbf{R}^n$ as the reflection of x with respect to the line defined by y. $g(x)$ can be expressed as

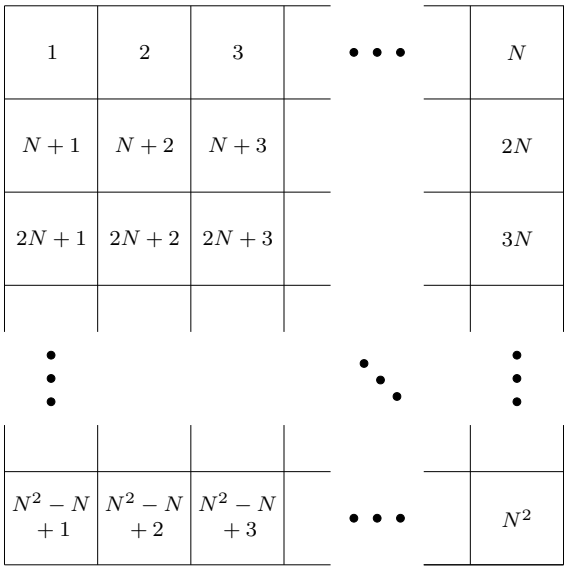
$$g(x) = x + 2(f(x) - x)$$

The definition is illustrated in the figure below.



Show that g is linear, and that it can be expressed as $g(x) = Ax$ with A an orthogonal matrix.

2.7 *Linear functions of grayscale images.* The figure shows an image divided into $N \times N$ pixels. We represent the image as a vector x of size N^2, where x_i denotes the grayscale level of pixel i (a number between 0 and 1, where 0 represents black, and 1 represents white).



Each of the following operations defines a function $y = f(x)$, where the N^2-vector x represents the original image, and the N^2-vector y represents the resulting image. For each of these operations, show that f is linear, and give an $N^2 \times N^2$-matrix A such that

$$y = Ax$$

for all x. You can assume that $N = 3$ for simplicity.

(a) Turn the image represented by x upside-down.

(b) Rotate the image clockwise over 90°.

(c) Translate the image up by 1 pixel, and to the right by 1 pixel. In the translated image, assign a value $y_i = 0$ to the pixels in the first column and the last row.

(d) Replace each pixel grayscale level by the average of the grayscale levels in a 3×3-neighborhood.

2.8 Represent each of the following three functions $f : \mathbf{R}^2 \to \mathbf{R}^2$ as a matrix-vector product $f(x) = Ax$.

(a) $f(x)$ is obtained by reflecting x about the x_1 axis.

(b) $f(x)$ is x reflected about the x_1 axis, followed by a counterclockwise rotation of 30 degrees.

(c) $f(x)$ is x rotated counterclockwise over 30 degrees, followed by a reflection about the x_1 axis.

2.9 We consider n-vectors x that represent signals, with x_k the value of the signal at time k for $k = 1, \ldots, n$. Below we describe two linear functions of x that produce new signals $f(x)$. For each function, give a matrix A such that $f(x) = Ax$ for all x.

(a) 2× *downsampling*. We assume n is even, and define $f(x)$ as the $n/2$-vector y with elements $y_k = x_{2k}$. To simplify your notation you can assume that $n = 8$, *i.e.*,

$$f(x) = (x_2,\, x_4,\, x_6,\, x_8).$$

(b) 2× *up-conversion with linear interpolation*. We define $f(x)$ as the $(2n-1)$-vector y with elements $y_k = x_{(k+1)/2}$ if k is odd and $y_k = (x_{k/2} + x_{k/2+1})/2$ if k is even. To simplify your notation you can assume that $n = 5$, *i.e.*,

$$f(x) = (x_1,\, \frac{x_1 + x_2}{2},\, x_2,\, \frac{x_2 + x_3}{2},\, x_3,\, \frac{x_3 + x_4}{2},\, x_4,\, \frac{x_4 + x_5}{2},\, x_5).$$

2.10 *Linear dynamical system.* A linear dynamical system is described by the recurrence

$$x(t+1) = Ax(t) + Bu(t), \quad t = 0, 1, 2, \ldots$$

The vector $x(t) \in \mathbf{R}^n$ is the *state* at time t, and $u(t) \in \mathbf{R}^m$ is the *input* at time t. The parameters in the model are the $n \times n$-matrix A and the $n \times m$-matrix B.

Find a formula for $x(N)$ (for a given $N \geq 0$) in terms of $x(0)$, $u(0)$, …, $u(N-1)$. Is $x(t)$ a linear function of $x(0)$, $u(0), \ldots, u(N-1)$? If not, give a specific counterexample. If it is linear, find a specific matrix G of size n by $n + mN$ such that

$$x(N) = G \begin{bmatrix} x(0) \\ u(0) \\ \vdots \\ u(N-1) \end{bmatrix}$$

for all $x(0)$, $u(0)$, …, $u(N-1)$.

2.11 Give the number of flops required to evaluate a product of three matrices

$$X = ABC,$$

where A is $n \times n$, B is $n \times 10$, and C is $10 \times n$. You can evaluate the product in two possible ways:

(a) from left to right, as $X = (AB)C$

(b) from right to left, as $X = A(BC)$

Which method is faster for large n? (You should assume that A, B, and C are dense, *i.e.*, they do not possess any special structure that you can take advantage of.)

2.12 A matrix C is defined as

$$C = Auv^T B$$

where A and B are $n \times n$-matrices, and u and v are n-vectors. The product on the right-hand side can be evaluated in many different ways, *e.g.*, as $A(u(v^T B))$ or as $A((uv^T)B)$, etc. What is the fastest method (requiring the least number of flops) when n is large?

2.13 The following MATLAB code generates three random vectors u, v, x of dimension $n = 1000$, and calculates the CPU time required to evaluate $y = (I + uv^T)x$ using two equivalent expressions.

```
n = 1000;
u = randn(n,1);  v = randn(n,1);  x = randn(n,1);
t1 = cputime;  y = (eye(n) + u*v') * x;  t1 = cputime - t1
t2 = cputime;  y = x + (v'*x) * u;  t2 = cputime - t2
```

Run this code, and compare t_1 and t_2. Repeat for $n = 2000$. Are your observations consistent with what you expect based on a flop count?

2.14 The *trace* of a square matrix is the sum of its diagonal elements. What is the complexity (number of flops for large n) of computing the trace of AB, where A and B are $n \times n$ matrices?

3 Linear equations

3.1 *Polynomial interpolation.* In this problem we construct polynomials

$$p(t) = x_1 + x_2 t + \cdots + x_{n-1} t^{n-2} + x_n t^{n-1}$$

of degree 5, 10, and 15 (*i.e.*, for $n = 6$, 11, 16), that interpolate points on the graph of the function $f(t) = 1/(1 + 25t^2)$ in the interval $[-1, 1]$. For each value of n, we compute the interpolating polynomial as follows. We first generate n pairs (t_i, y_i), using the MATLAB commands

```
t = linspace(-1, 1, n)';
y = 1 ./ (1 + 25*t.^2);
```

This produces two n-vectors: a vector `t` with elements t_i, equally spaced in $[-1, 1]$, and a vector `y` with elements $y_i = f(t_i)$. (See '`help rdivide`' and '`help power`' for the meaning of the operations `./` and `.^`.) We then solve a set of linear equations

$$\begin{bmatrix} 1 & t_1 & \cdots & t_1^{n-2} & t_1^{n-1} \\ 1 & t_2 & \cdots & t_2^{n-2} & t_2^{n-1} \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & t_{n-1} & \cdots & t_{n-1}^{n-2} & t_{n-1}^{n-1} \\ 1 & t_n & \cdots & t_n^{n-2} & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \tag{1}$$

to find the coefficients x_i.

Calculate the three polynomials (for $n = 6$, $n = 11$, $n = 16$). Plot the three polynomials and the function f on the interval $[-1, 1]$, and attach a printout of the plots to your solutions. What do you conclude about the effect of increasing the degree of the interpolating polynomial?

MATLAB hints.

- Use `x = A \ b` to solve a set of n linear equations in n variables $Ax = b$.
- To construct the coefficient matrix in (1), you can write a double for-loop, or use the built-in MATLAB function `vander`, which constructs a matrix of the form

$$\begin{bmatrix} t_1^{n-1} & t_1^{n-2} & \cdots & t_1^2 & t_1 & 1 \\ t_2^{n-1} & t_2^{n-2} & \cdots & t_2^2 & t_2 & 1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ t_n^{n-1} & t_n^{n-2} & \cdots & t_n^2 & t_n & 1 \end{bmatrix}.$$

  Type '`help vander`' for details. This is almost what we need, but you have to 'flip' this matrix from left to right. This operation is also built in in MATLAB (type `help fliplr`).
- We are interested in the behavior of the interpolating polynomials between the points t_i that you used in the construction. Therefore, when you plot the three polynomials, you should use a much denser grid of points (*e.g.*, a few hundred points equally spaced in interval $[-1, 1]$) than the n points that you used to generate the polynomials.

3.2 Formulate the following problem as a set of linear equations $Ax = b$. Find two cubic polynomials

$$p(t) = x_1 + x_2 t + x_3 t^2 + x_4 t^3, \qquad q(t) = x_5 + x_6 t + x_7 t^2 + x_8 t^3$$

that satisfy the following eight conditions:

- $p(t_1) = y_1$, $p(t_2) = y_2$, $p(t_3) = y_3$.

- $q(t_5) = y_5$, $q(t_6) = y_6$, $q(t_7) = y_7$.
- $p(t_4) = q(t_4)$, $p'(t_4) = q'(t_4)$. This specifies that at $t = t_4$ the polynomials should have the same value and the same derivative.

The variables in the problem are the coefficients $x_1, \ldots, x_8$. The numbers t_i, y_i are given, with $t_1 < t_2 < t_3 < t_4 < t_5 < t_6 < t_7$.

Test the method in MATLAB on the following problem. We take the 7 points t_i equally spaced in the interval $[-0.75, 0.25]$ (using `t = linspace(-0.75, 0.25, 7)'`), and

$$y_1 = 0, \qquad y_2 = -0.1, \qquad y_3 = 0.5, \qquad y_5 = 1, \qquad y_6 = 0.8, \qquad y_7 = 0.5.$$

Calculate the two polynomials $p(t)$ and $q(t)$ (using the command `x = A \ b` to solve the equation $Ax = b$), and plot them on the interval $[-0.75, 0.25]$.

3.3 Express the following problem as a set of linear equations. Find a cubic polynomial

$$f(t) = c_1 + c_2(t - t_1) + c_3(t - t_1)^2 + c_4(t - t_1)^2(t - t_2)$$

that satisfies

$$f(t_1) = y_1, \qquad f(t_2) = y_2, \qquad f'(t_1) = s_1, \qquad f'(t_2) = s_2.$$

The numbers t_1, t_2, y_1, y_2, s_1, s_2 are given, with $t_1 \neq t_2$. The unknowns are the coefficients c_1, c_2, c_3, c_4. Write the equations in matrix-vector form $Ax = b$, and solve them.

3.4 Express the following problem as a set of linear equations. Find a rational function

$$f(t) = \frac{c_0 + c_1 t + c_2 t^2}{1 + d_1 t + d_2 t^2}$$

that satisfies the following conditions

$$f(1) = 2.3, \qquad f(2) = 4.8, \qquad f(3) = 8.9, \qquad f(4) = 16.9, \qquad f(5) = 41.0.$$

The variables in the problem are the coefficients c_0, c_1, c_2, d_1 and d_2. Write the equations in matrix-vector form $Ax = b$ and solve the equations with MATLAB.

3.5 Express the following problem as a set of linear equations. Find a quadratic function

$$f(u_1, u_2) = \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} q_1 & q_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + r$$

that satisfies the following six conditions:

$$f(0,1) = 6, \qquad f(1,0) = 6, \qquad f(1,1) = 3,$$

$$f(-1,-1) = 7, \qquad f(1,2) = 2, \qquad f(2,1) = 6.$$

The variables in the problem are the parameters p_{11}, p_{12}, p_{22}, q_1, q_2 and r. Write the equations in matrix-vector form $Ax = b$, and solve the equations with MATLAB.

3.6 Let a, b, c, d be four given points in $\mathbf{R}^3$. The points do not lie in one plane. Algebraically, this can be expressed by saying that the three vectors $b - a$, $c - a$, $d - a$ are linearly independent, *i.e.*, the equality $y_1(b - a) + y_2(c - a) + y_3(d - a) = 0$ holds only if $y_1 = y_2 = y_3 = 0$.

Suppose we are given the distances of a point $x = (x_1, x_2, x_3)$ to the four points:

$$\|x - a\| = r_a, \qquad \|x - b\| = r_b, \qquad \|x - c\| = r_c, \qquad \|x - d\| = r_d.$$

Write a set of linear equations $Ax = f$, with A nonsingular, from which the coordinates x_1, x_2, x_3 can be computed. Explain why the matrix A is nonsingular. Apply your method to solve the problem with

$$a = \begin{bmatrix} -10 \\ -10 \\ 10 \end{bmatrix}, \qquad b = \begin{bmatrix} 10 \\ 0 \\ 0 \end{bmatrix}, \qquad c = \begin{bmatrix} -10 \\ 10 \\ 0 \end{bmatrix}, \qquad d = \begin{bmatrix} -20 \\ -10 \\ -10 \end{bmatrix},$$

and

$$r_a = 18.187, \qquad r_b = 9.4218, \qquad r_c = 14.310, \qquad r_d = 24.955.$$

3.7 *Numerical integration.* Numerical integration rules (also called *quadrature rules*) are numerical methods for approximating integrals

$$\int_a^b f(t)\,dt.$$

The general idea is to approximate the integral by a weighted sum of function values $f(t_i)$, at n points t_i in the interval $[a, b]$:

$$\int_a^b f(t)\,dt \approx \sum_{i=1}^n w_i f(t_i),$$

where $a \leq t_1 < t_2 < \cdots < t_n \leq b$. The points t_i are called the *abscissas* or *nodes* of the integration rule, and the coefficients w_i are the *weights*. Many different integration rules exist, which use different choices of n, w_i, and t_i.

For simplicity we will take $a = 0$, $b = 1$.

(a) Suppose you are given n distinct points t_i in the interval $[0, 1]$, and you are asked to determine the weights w_i in such a way that the integration rule is exact for the functions $f(t) = t^k$, $k = 0, 1, \ldots, n-1$. In other words, we require that

$$\int_0^1 f(t)\,dt = \sum_{i=1}^n w_i f(t_i)$$

if f is one of the functions

$$f(t) = 1, \qquad f(t) = t, \qquad f(t) = t^2, \qquad \ldots, \qquad f(t) = t^{n-1}.$$

Formulate this problem as a set of linear equations with variables $w_1, \ldots, w_n$.

(b) Suppose you use the weights computed in (a). Show that the integration rule is exact for all polynomials of degree $n - 1$ or less. In other words, show that

$$\int_0^1 f(t)\,dt = \sum_{i=1}^n w_i f(t_i)$$

if $f(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_{n-1} t^{n-1}$.

(c) Apply your method for part (a) to compute the weights in the following integration rules.

(i) $n = 2$, $t_1 = 0$, $t_2 = 1$. This gives the *trapezoid rule*.

(ii) $n = 3$, $t_1 = 0$, $t_2 = 0.5$, $t_3 = 1$. This gives *Simpson's rule*.

(iii) $n = 4$, $t_1 = 0$, $t_2 = 1/3$, $t_3 = 2/3$, $t_4 = 1$.

Compare the accuracy of these three integration rules when applied to the function $f(t) = \exp(t)$.

4 Matrix inverses

4.1 Do the following matrices have linearly independent columns?

(a) $A = \begin{bmatrix} -1 & 2 \\ 3 & -6 \\ 2 & -1 \end{bmatrix}$.

(b) $A = \begin{bmatrix} -1 & 3 & 2 \\ 2 & -6 & -1 \end{bmatrix}$.

(c) $A = \begin{bmatrix} -9 & 0 & 7 \\ 4 & 0 & -5 \\ -1 & 0 & 6 \end{bmatrix}$.

(d) $A = \begin{bmatrix} D \\ B \end{bmatrix}$, where B is $m \times n$ and D is a diagonal $n \times n$-matrix with nonzero diagonal elements.

(e) $A = ab^T$ where a and b are n-vectors and $n > 1$.

(f) $A = I - ab^T$ where a and b are n-vectors with $\|a\|\|b\| < 1$.

4.2 Prove or disprove each the following statements. To prove a statement, show that it is true for all matrices with the stated properties (you don't need to consider special cases). To disprove a statement, give a small counterexample.

In each problem A is a $p \times m$ matrix, B is an $m \times n$ matrix, and $C = AB$.

(a) If A and B have linearly independent columns, then C has linearly independent columns.

(b) If A and B have orthonormal columns, then C has orthonormal columns.

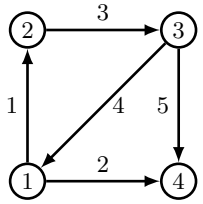
(c) If A and B are left invertible, X is a left inverse of A, and Y is a left inverse of B, then C is left invertible and YX is a left inverse of C.

(d) If A and B are left invertible, X is the pseudo-inverse of A, and Y is the pseudo-inverse of B, then C is left invertible and YX is the pseudo-inverse of C.

4.3 The node-arc incidence matrix of a directed graph with n nodes and m arcs is the $n \times m$ matrix A with elements

$$A_{ij} = \begin{cases} -1 & \text{if arc } j \text{ leaves node } i \\ 1 & \text{if arc } j \text{ enters node } i \\ 0 & \text{otherwise.} \end{cases}$$

The figure shows an example.



$$A = \begin{bmatrix} -1 & -1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Show that a node-arc incidence matrix is not right invertible.

4.4 Suppose A is a nonsingular matrix of order n, u and v are n-vectors, and $v^T A^{-1} u \neq -1$. Show that $A + uv^T$ is nonsingular with inverse

$$(A + uv^T)^{-1} = A^{-1} - \frac{1}{1 + v^T A^{-1} u} A^{-1} u v^T A^{-1}.$$

4.5 Suppose A is a nonsingular matrix of order n. Consider the matrix $2n \times 2n$-matrix M defined as

$$M = \begin{bmatrix} A & A + A^{-T} \\ A & A \end{bmatrix}.$$

(a) Show that M is nonsingular, by showing that $Mx = 0$ implies $x = 0$.

(b) Find the inverse of M. To find M^{-1}, express it as a block matrix

$$M^{-1} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix}$$

with blocks of dimension $n \times n$, and determine the matrices W, X, Y, Z from the condition $MM^{-1} = I$.

4.6 Define a sequence of matrices A_k for $k = 0, 1, 2 \ldots$, as follows: $A_0 = 1$ and for $k \geq 1$,

$$A_k = \begin{bmatrix} A_{k-1} & A_{k-1} \\ -A_{k-1} & A_{k-1} \end{bmatrix}.$$

Therefore A_k is a matrix of size $2^k \times 2^k$. The first four matrices in the sequence are

$$A_0 = 1, \qquad A_1 = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix}.$$

Show that all matrices A_k in the sequence are nonsingular. What is the inverse of A_k?

4.7 *Lagrange interpolation.* In exercise 3.1 we considered the problem of finding a polynomial

$$p(t) = x_1 + x_2 t + \cdots + x_n t^{n-1} \tag{2}$$

with specified values

$$p(t_1) = y_1, \qquad p(t_2) = y_2, \qquad , \ldots , \qquad p(t_n) = y_n. \tag{3}$$

The polynomial p is called the *interpolating* polynomial through the points (t_1, y_1), $\ldots$, (t_n, y_n). Its coefficients can be computed by solving the set of linear equations

$$\begin{bmatrix} 1 & t_1 & \cdots & t_1^{n-2} & t_1^{n-1} \\ 1 & t_2 & \cdots & t_2^{n-2} & t_2^{n-1} \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & t_{n-1} & \cdots & t_{n-1}^{n-2} & t_{n-1}^{n-1} \\ 1 & t_n & \cdots & t_n^{n-2} & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}. \tag{4}$$

The coefficient matrix is called a *Vandermonde* matrix. As we have seen in the lecture, a Vandermonde matrix is nonsingular if the points t_i are distinct ($t_i \neq t_j$ for $i \neq j$). As a consequence, the interpolating polynomial is unique: if the points t_i are distinct, then there exists exactly one polynomial of degree less than or equal to $n - 1$ that satisfies (3), and its coefficients are the solution of the equations (4).

In this problem we describe another method for finding p, known as *Lagrange interpolation*.

(a) We define n polynomials l_i:

$$l_i(t) = \frac{\prod_{j \neq i}(t - t_j)}{\prod_{j \neq i}(t_i - t_j)}, \quad i = 1, \ldots, n.$$

Verify that l_i is a polynomial of degree $n - 1$, and that

$$l_i(t_k) = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{if } k \neq i. \end{cases}$$

For example, for $n = 3$, we have the three polynomials

$$l_1(t) = \frac{(t - t_2)(t - t_3)}{(t_1 - t_2)(t_1 - t_3)}, \quad l_2(t) = \frac{(t - t_1)(t - t_3)}{(t_2 - t_1)(t_2 - t_3)}, \quad l_3(t) = \frac{(t - t_1)(t - t_2)}{(t_3 - t_1)(t_3 - t_2)}.$$

(b) Show that the polynomial

$$p(t) = y_1 l_1(t) + y_2 l_2(t) + \cdots + y_n l_n(t) \tag{5}$$

has degree $n-1$ or less, and satisfies the interpolation conditions (3). It is therefore equal to the unique interpolating polynomial through those points.

(c) This provides another method for polynomial interpolation. To find the coefficients x_i we express the polynomial (5) in the form (2) by expanding the polynomials l_i as weighted sums of powers of t.
As an example, for $n = 3$, the polynomial (5) is given by

$$p(t) = y_1 \frac{(t - t_2)(t - t_3)}{(t_1 - t_2)(t_1 - t_3)} + y_2 \frac{(t - t_1)(t - t_3)}{(t_2 - t_1)(t_2 - t_3)} + y_3 \frac{(t - t_1)(t - t_2)}{(t_3 - t_1)(t_3 - t_2)}.$$

Express this as $p(t) = x_1 + x_2 t + x_3 t^2$, *i.e.*, give expressions for x_1, x_2, x_3 in terms of t_1, t_2, t_3, y_1, y_2, y_3. Use the result to prove the following expression for the inverse of a 3×3 Vandermonde matrix:

$$\begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \end{bmatrix}^{-1} = \begin{bmatrix} \dfrac{t_2 t_3}{(t_1 - t_2)(t_1 - t_3)} & \dfrac{t_1 t_3}{(t_2 - t_1)(t_2 - t_3)} & \dfrac{t_1 t_2}{(t_3 - t_1)(t_3 - t_2)} \\ \dfrac{-t_2 - t_3}{(t_1 - t_2)(t_1 - t_3)} & \dfrac{-t_1 - t_3}{(t_2 - t_1)(t_2 - t_3)} & \dfrac{-t_1 - t_2}{(t_3 - t_1)(t_3 - t_2)} \\ \dfrac{1}{(t_1 - t_2)(t_1 - t_3)} & \dfrac{1}{(t_2 - t_1)(t_2 - t_3)} & \dfrac{1}{(t_3 - t_1)(t_3 - t_2)} \end{bmatrix}.$$

5 Triangular matrices

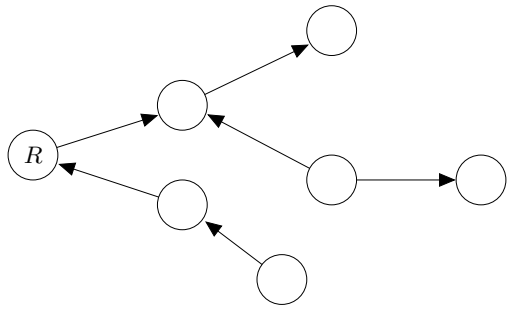
5.1 Let A be a lower triangular matrix of order n. Verify the following properties.

(a) If B is lower triangular of order n, then the product AB is lower triangular.

(b) The matrix A^k is lower triangular for all positive integers k.

(c) If A is nonsingular, then A^k is lower triangular for all integers k (positive or negative).

5.2 A *directed tree* is a connected graph with $n+1$ nodes and n directed arcs. The figure shows an example with $n = 6$ (7 nodes and 6 arcs).



A directed tree can be represented by a *reduced node-arc incidence matrix* defined as follows. We label one node as the *root* node. The other nodes are labeled with integers from 1 to n. The arcs are also given integer labels from 1 to n. The node-arc incidence matrix A is then defined as the $n \times n$ matrix with elements

$$A_{ij} = \begin{cases} +1 & \text{if arc } j \text{ ends at node } i \\ -1 & \text{if arc } j \text{ begins at node } i \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, n$.

(a) Suppose we use the node labeled R as the root node in the example. Number the non-root nodes and the arcs in such a way that the node-arc incidence matrix is lower triangular.

(b) Show that the node-arc incidence matrix obtained in part (a) is nonsingular.

(c) Show that the elements of the inverse of the node-arc incidence matrix in part (a) have values 0, +1 or -1.

5.3 Let A be a nonsingular triangular matrix of order n.

(a) What is the cost of computing A^{-1}?

(b) What is the cost of solving $Ax = b$ by first computing A^{-1} and then forming the matrix-vector product $x = A^{-1}b$? Compare with the cost of forward and backward substitution.

5.4 The *trace* of a square matrix is the sum of its diagonal elements. What is the complexity (number of flops for large n) of computing the trace of A^{-1}, where the $n \times n$ matrix A is lower triangular and nonsingular?

5.5 A lower triangular matrix A is *bidiagonal* if $A_{ij} = 0$ for $i > j + 1$:

$$A = \begin{bmatrix} A_{11} & 0 & 0 & \cdots & 0 & 0 & 0 \\ A_{21} & A_{22} & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_{32} & A_{33} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & A_{n-2,n-2} & 0 & 0 \\ 0 & 0 & 0 & \cdots & A_{n-1,n-2} & A_{n-1,n-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & A_{n,n-1} & A_{nn} \end{bmatrix}.$$

Assume A is a nonsingular bidiagonal and lower triangular matrix of size $n \times n$.

(a) What is the complexity of solving $Ax = b$?

(b) What is the complexity of computing the inverse of A?

State the algorithm you use in each subproblem, and give the dominant term (exponent and coefficient) of the flop count. If you know several methods, consider the most efficient one.

5.6 A lower triangular matrix A is called a lower triangular band matrix with k subdiagonals if $A_{ij} = 0$ for $i > j + k$. The matrix

$$A = \begin{bmatrix} -0.9 & 0 & 0 & 0 & 0 & 0 \\ 0.7 & -0.7 & 0 & 0 & 0 & 0 \\ 1.4 & -2.7 & 3.7 & 0 & 0 & 0 \\ 0 & 0.6 & 0.3 & -1.2 & 0 & 0 \\ 0 & 0 & -2.2 & 1.1 & -0.6 & 0 \\ 0 & 0 & 0 & 2.4 & 2.4 & 0.7 \end{bmatrix}$$

is a 6×6 lower triangular band matrix with 2 subdiagonals.

What is the cost of solving a set of linear equations $Ax = b$ if A is an $n \times n$ lower triangular band matrix with k subdiagonals and nonzero diagonal elements? Express the cost as the number of flops as a function of n and k. You only have to give the dominant term in the flop count and you can assume that $k \ll n$.

5.7 Describe an efficient method for each of the following two problems, and give a complexity estimate (number of flops for large n).

(a) Solve

$$DX + XD = B$$

where D is $n \times n$ and diagonal. The diagonal elements of D satisfy $d_{ii} + d_{jj} \neq 0$ for all i and j. The matrices D and B are given. The variable is the $n \times n$-matrix X.

(b) Solve

$$LX + XL^T = B$$

where L is lower triangular. The diagonal elements of L satisfy $L_{ii} + L_{jj} \neq 0$ for all i and j. The matrices L and B are given. The variable is the $n \times n$-matrix X. (*Hint:* Solve for X column by column.)

If you know several methods, choose the fastest one (least number of flops for large n).

5.8 *Inverse of a block-triangular matrix.* Express the inverse of the 2×2 block matrix

$$A = \begin{bmatrix} B & 0 \\ C & D \end{bmatrix}$$

in terms of B, C, D, B^{-1} and D^{-1}. We assume B is $n \times n$, C is $p \times n$, and D is $p \times p$, with B and D nonsingular.

6 Orthogonal matrices

6.1 A square matrix A is called *normal* if $AA^T = A^TA$. Show that if A is normal and nonsingular, then the matrix $Q = A^{-1}A^T$ is orthogonal.

6.2 Let S be a square matrix that satisfies $S^T = -S$. (This is called a *skew-symmetric* matrix.)

(a) Show that $I - S$ is nonsingular. (Hint: first show that $x^TSx = 0$ for all x.)

(b) Show that $(I + S)(I - S)^{-1} = (I - S)^{-1}(I + S)$. (This property does not rely on the skew-symmetric property; it is true for any matrix S for which $I - S$ is nonsingular.)

(c) Show that the matrix

$$A = (I + S)(I - S)^{-1}$$

is orthogonal.

6.3 Let Q be an $n \times n$ orthogonal matrix, partitioned as

$$Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$$

where Q_1 has size $n \times m$ and Q_2 has size $n \times (n - m)$, with $0 < m < n$. Consider the $n \times n$ matrix

$$A = Q_1Q_1^T - Q_2Q_2^T.$$

(a) Show that A can also be written in the following two forms:

$$A = 2Q_1Q_1^T - I, \qquad A = I - 2Q_2Q_2^T.$$

(b) Show that A is orthogonal.

(c) Describe an efficient method for solving $Ax = b$ and give the complexity of the method (the dominant term(s) in the flop count, including the coefficient). If you know several methods, give the method that has the lowest complexity when $m < n/2$.

6.4 Let A be a tall $m \times n$ matrix with linearly independent columns. Define

$$P = A(A^TA)^{-1}A^T.$$

(a) Show that the matrix $2P - I$ is orthogonal.

(b) Use the Cauchy-Schwarz inequality to show that the inequalities

$$-\|x\|\|y\| \leq x^T(2P - I)y \leq \|x\|\|y\|$$

hold for all m-vectors x and y.

(c) Take $x = y$ in part (b). Show that the right-hand inequality implies that $\|Px\| \leq \|x\|$ for all m-vectors x.

6.5 Let B be an $m \times n$ matrix.

(a) Prove that the matrix $I + B^TB$ is nonsingular. Since we do not impose any conditions on B, this also shows that the matrix $I + BB^T$ is nonsingular.

(b) Show that the matrix

$$A = \begin{bmatrix} I & B^T \\ -B & I \end{bmatrix}$$

is nonsingular and that the following two expressions for its inverse are correct:

$$A^{-1} = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -B^T \\ I \end{bmatrix} (I + BB^T)^{-1} \begin{bmatrix} B & I \end{bmatrix},$$

$$A^{-1} = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} I \\ B \end{bmatrix} (I + B^TB)^{-1} \begin{bmatrix} I & -B^T \end{bmatrix}.$$

(c) Now assume B has orthonormal columns. Use the result in part (b) to formulate a simple method for solving $Ax = b$. What is the complexity of your method? If you know several methods, give the most efficient one.

6.6 (a) For what property of the matrix B is a matrix of the form

$$A = \frac{1}{\sqrt{2}} \begin{bmatrix} I & B^T \\ -B & I \end{bmatrix}$$

orthogonal? Give the necessary and sufficient conditions on B.

(b) What are the properties of B needed to make the matrix A nonsingular?

6.7 *Circulant Toeplitz matrices and discrete Fourier transform.* A *circulant Toeplitz matrix* is a square matrix of the form

$$T(a) = \begin{bmatrix} a_1 & a_n & a_{n-1} & \cdots & a_3 & a_2 \\ a_2 & a_1 & a_n & \cdots & a_4 & a_3 \\ a_3 & a_2 & a_1 & \cdots & a_5 & a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_1 & a_n \\ a_n & a_{n-1} & a_{n-2} & \cdots & a_2 & a_1 \end{bmatrix}. \tag{6}$$

We use the notation $T(a)$ for this matrix, where $a = (a_1, a_2, \ldots, a_n)$ is the n-vector in the first column. Each of the other columns is obtained by a circular downward shift of the previous column. In matrix notation,

$$T(a) = \begin{bmatrix} a & Sa & S^2a & \cdots & S^{n-1}a \end{bmatrix}$$

where S is the $n \times n$ *circular shift* matrix

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

(a) Let W be the $n \times n$ DFT matrix:

$$W = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \cdots & \omega^{-2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \ldots & \omega^{-(n-1)(n-1)} \end{bmatrix} \tag{7}$$

where $\omega = e^{2\pi \mathrm{j}/n}$. Verify that

$$WS^{k-1} = \mathbf{diag}(We_k)W, \quad k = 1, \ldots, n,$$

where S^i is the ith power of S (with $S^0 = I$), e_k is the kth unit vector (hence, We_k is column k of W), and $\mathbf{diag}(We_k)$ is the $n \times n$ diagonal matrix with We_k on its diagonal.

(b) The inverse of W is $W^{-1} = (1/n)W^H$. The expression in part (a) can therefore be written as

$$S^{k-1} = \frac{1}{n} W^H \, \mathbf{diag}(We_k)W, \quad k = 1, \ldots, n.$$

Use this to show that $T(a)$ can be factored as a product of three matrices:

$$T(a) = \frac{1}{n} W^H \, \mathbf{diag}(Wa)W. \tag{8}$$

(c) The matrix-vector product $y = Wx$ is the discrete Fourier transform of x. The matrix-vector product $y = W^{-1}x = (1/n)W^H x$ is the inverse discrete Fourier transform. The DFT and its inverse can be computed in order $n \log n$ operations using the Fast Fourier Transform algorithm.

Use the factorization (8) to formulate a fast algorithm, with a complexity of order $n \log n$, for computing matrix-vector products $T(a)x$. The product $T(a)x$ is known as the *circular convolution* of the vectors a and x.

(d) Use the factorization in part (b) to formulate a fast algorithm, with an order $n \log n$ complexity, for solving a set of linear equations $Ax = b$ with variable x and coefficient matrix $A = T(a)$ (assuming $T(a)$ is nonsingular). Compare the speed of your algorithm with the standard method (`A \ b`), for randomly generated a and b. You can use the following code to generate a and b, and construct the circulant Toeplitz matrix $A = T(a)$.

```
a = randn(n, 1);
b = randn(n, 1);
A = toeplitz(a, [a(1), flipud(a(2:n))']);
```

Use the MATLAB functions `fft` and `ifft` to implement the fast algorithm. `y = fft(x)` evaluates $y = Wx$ using the Fast Fourier Transform algorithm; `y = ifft(x)` evaluates $y = W^{-1}x$.

6.8 Refer to the factorization (8) of a circulant Toeplitz matrix (6) with the n-vector a as its first column. In (8), W is the $n \times n$ discrete Fourier Transform matrix and $\mathbf{diag}(Wa)$ is the diagonal matrix with the vector Wa (the discrete Fourier transform of a) on its diagonal.

(a) Suppose $T(a)$ is nonsingular. Show that its inverse $T(a)^{-1}$ is a circulant Toeplitz matrix. Give a fast method for computing the vector b that satisfies $T(b) = T(a)^{-1}$.

(b) Let a and b be two n-vectors. Show that the product $T(a)T(b)$ is a circulant Toeplitz matrix. Give a fast method for computing the vector c that satisfies $T(c) = T(a)T(b)$.

6.9 What is the QR factorization of the matrix

$$A = \begin{bmatrix} 2 & 8 & 13 \\ 4 & 7 & -7 \\ 4 & -2 & -13 \end{bmatrix} \; ?$$

You can use MATLAB to check your answer, but you must provide the details of all intermediate steps on paper.

6.10 A diagonal matrix with diagonal elements $+1$ or -1 is called a *signature matrix*. The matrix

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

is an example of a 3×3 signature matrix. If S is a signature matrix, and A is a square matrix that satisfies

$$A^T S A = S, \tag{9}$$

then we say that A is *pseudo-orthogonal* with respect to S.

(a) Suppose S is an $n \times n$ signature matrix, and u is an n-vector with $u^T S u \neq 0$. Show that the matrix

$$A = S - \frac{2}{u^T S u} u u^T$$

is pseudo-orthogonal with respect to S.

(b) Show that pseudo-orthogonal matrices are nonsingular. In other words, show that any square matrix A that satisfies (9) for some signature matrix S is nonsingular.

(c) Describe an efficient method for solving $Ax = b$ when A is pseudo-orthogonal. 'Efficient' here means that the complexity is at least an order of magnitude less than the $(2/3)n^3$ complexity of the standard method for a general set of linear equations. Give the complexity of your method (number of flops for large n).

(d) Show that if A satisfies (9) then $ASA^T = S$. In other words, if A is pseudo-orthogonal with respect to S, then A^T is also pseudo-orthogonal with respect to S.

6.11 The *Kronecker product* of two $n \times n$ matrices A and B is the $n^2 \times n^2$ matrix

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{21}B & A_{22}B & \cdots & A_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1}B & A_{n2}B & \cdots & A_{nn}B \end{bmatrix}.$$

For example,

$$\begin{bmatrix} 1 & 0 \\ 2 & -1 \end{bmatrix} \otimes \begin{bmatrix} 3 & 4 \\ -5 & 6 \end{bmatrix} = \begin{bmatrix} 3 & 4 & 0 & 0 \\ -5 & 6 & 0 & 0 \\ 6 & 8 & -3 & -4 \\ -10 & 12 & 5 & -6 \end{bmatrix}.$$

Suppose A and B are orthogonal. Is $A \otimes B$ orthogonal? Explain your answer.

6.12 Let A be an $m \times n$ matrix with linearly independent columns. The Householder algorithm for the QR factorization of A computes an orthogonal $m \times m$ matrix Q such that

$$Q^T A = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

where R is upper triangular with nonzero diagonal elements. The matrix Q is computed as a product $Q = Q_1 Q_2 \cdots Q_{n-1}$ of orthogonal matrices. In this problem we discuss the first step, the calculation of Q_1. This matrix has the property that

$$Q_1^T A = \begin{bmatrix} R_{11} & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix}.$$

The '$\times$' symbols denote elements that may or may not be zero.

Let $a = (A_{11}, A_{21}, \ldots, A_{m1})$ be the first column of A. Define an m-vector

$$v = \frac{1}{\sqrt{1 + |A_{11}|/\|a\|}} \left(\frac{1}{\|a\|} a + s e_1 \right)$$

where $s = 1$ if $A_{11} \geq 0$ and $s = -1$ if $A_{11} < 0$. The vector e_1 is the first unit vector $(1, 0, \ldots, 0)$.

(a) Show that v has norm $\sqrt{2}$.

(b) Define $Q_1 = I - vv^T$. Show that Q_1 is orthogonal.

(c) Show that $Q_1^T a = R_{11} e_1$, where $R_{11} = -s\|a\|$.

(d) Give the complexity (dominant term in the flop count for large m, n) of computing the matrix-matrix product

$$Q_1^T A = (I - vv^T)A.$$

7 LU factorization

7.1 (a) For what values of a_1, a_2, ..., a_n is the $n \times n$ matrix

$$A = \begin{bmatrix} a_1 & 1 & 0 & \cdots & 0 & 0 \\ a_2 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & 0 & 0 & \cdots & 1 & 0 \\ a_{n-1} & 0 & 0 & \cdots & 0 & 1 \\ a_n & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

nonsingular?

(b) Assuming A is nonsingular, how many floating-point operations (flops) do you need to solve $Ax = b$?

(c) Assuming A is nonsingular, what is the inverse A^{-1}?

7.2 Consider the set of linear equations

$$(D + uv^T)x = b$$

where u, v, and b are given n-vectors, and D is a given diagonal matrix. The diagonal elements of D are nonzero and $u^T D^{-1} v \neq -1$. (This implies that the matrix $D + uv^T$ is nonsingular.)

(a) What is the cost of solving these equations using the following method?

- First calculate $A = D + uv^T$.
- Then solve $Ax = b$ using the standard method (via LU factorization, which costs $(2/3)n^3$ flops).

(b) In exercise 4.4, the following expression for the inverse of $D + uv^T$ is derived:

$$(D + uv^T)^{-1} = D^{-1} - \frac{1}{1 + v^T D^{-1} u} D^{-1} u v^T D^{-1}.$$

This means we can also solve the equations by evaluating $x = (D + uv^T)^{-1} b$, using the expression for the inverse. What is the cost of this method?

7.3 For each subproblem, we give a naive but correct algorithm, in MATLAB notation. Derive a flop count, assuming that matrix inverses and solutions of linear equations are computed using the LU factorization. Use the values $f = (2/3)n^3$, $s = 2n^2$ for the flop counts of the factorization and solve steps (for a set of n linear equations in n variables). Assume the cost of computing the inverse of an $n \times n$ matrix is $f + ns = (8/3)n^3$ flops.

If possible, give a more efficient method. You do not have to provide any MATLAB code, as long as the description of your method is clear. If you know several methods, give the most efficient one.

(a) Calculate $c^T A^{-1} b$ where $c \in \mathbf{R}^n$, $A \in \mathbf{R}^{n \times n}$, and $b \in \mathbf{R}^n$ are given. The matrix A is nonsingular.

```
val = c' * (inv(A) * b)
```

(b) Calculate $c^T A^{-1} B$ where $c \in \mathbf{R}^n$, $A \in \mathbf{R}^{n \times n}$, and $B \in \mathbf{R}^{n \times m}$ are given. The matrix A is nonsingular.

```
val = c' * (inv(A) * B)
```

(c) Solve the set of equations

$$\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} x = \begin{bmatrix} b \\ c \end{bmatrix},$$

where $A \in \mathbf{R}^{n \times n}$, $B \in \mathbf{R}^{n \times n}$, $b \in \mathbf{R}^n$, and $c \in \mathbf{R}^n$ are given, and 0 is the zero matrix of dimension $n \times n$. The matrices A and B are nonsingular.

```
x = [A zeros(n,n); zeros(n,n) B ] \ [b; c]
```

(d) Solve the set of equations

$$\begin{bmatrix} A & B \\ C & I \end{bmatrix} x = \begin{bmatrix} b \\ c \end{bmatrix},$$

where $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^{n\times 10n}$, $C \in \mathbf{R}^{10n\times n}$, $b \in \mathbf{R}^n$, and $c \in \mathbf{R}^{10n}$ are given, and I is the identity matrix of dimension $10n \times 10n$. The matrix

$$\begin{bmatrix} A & B \\ C & I \end{bmatrix}$$

is nonsingular.

```
x = [A, B; C, eye(10*n)] \ [b; c]
```

(e) Solve the set of equations

$$\begin{bmatrix} I & B \\ C & I \end{bmatrix} x = \begin{bmatrix} b \\ c \end{bmatrix},$$

where B is $m \times n$, C is $n \times m$, and $n > m$. The matrix

$$\begin{bmatrix} I & B \\ C & I \end{bmatrix} x = \begin{bmatrix} b \\ c \end{bmatrix}$$

is nonsingular.

```
x = [eye(m), B; C, eye(n)] \ [b; c]
```

7.4 Calculate the LU factorization without pivoting of the matrix

$$A = \begin{bmatrix} -3 & 2 & 0 & 3 \\ 6 & -6 & 0 & -12 \\ -3 & 6 & -1 & 16 \\ 12 & -14 & -2 & -25 \end{bmatrix}.$$

You can check your result in MATLAB, but you have to provide the details of your calculation.

7.5 You are given a nonsingular $n \times n$ matrix A and an n-vector b. You are asked to evaluate

$$x = (I + A^{-1} + A^{-2} + A^{-3})\, b$$

where $A^{-2} = (A^2)^{-1}$ and $A^{-3} = (A^3)^{-1}$.

Describe in detail how you would compute x, and give the flop counts of the different steps in your algorithm. If you know several methods, give the most efficient one (least number of flops for large n).

7.6 Suppose you are asked to solve K sets of linear equations

$$\begin{aligned} AD_1Bx_1 &= b_1 \\ AD_2Bx_2 &= b_2 \\ &\vdots \\ AD_KBx_K &= b_K. \end{aligned}$$

The $n \times n$ matrices A and B are nonsingular and given. The matrices D_k are diagonal with nonzero diagonal elements. The right-hand sides $b_k \in \mathbf{R}^n$ are also given. The variables in the problem are the K n-vectors x_k, $k = 1, \ldots, K$.

Describe an efficient method for computing the vectors x_k. Compare with the cost of solving K sets of linear equations of size $n \times n$, using a standard method.

7.7 What is the most efficient way to compute the following $n \times n$ matrices X? Justify your answer by giving the number of flops, assuming n is large. The vectors u and v have size n, and the matrix A is $n \times n$. A is nonsingular.

(a) $X = vu^T Avu^T$.

(b) $X = vu^T A^{-1} vu^T$.

(c) $X = vu^T (A + A^{-1}) vu^T$.

7.8 Suppose you have to solve two sets of linear equations

$$Ax_1 = b_1, \qquad A^T x_2 = b_2$$

where A is $n \times n$, b_1 and b_2 are n-vectors, and A is nonsingular. The unknowns are the n-vectors x_1 and x_2.

What is the most efficient way to solve these two problems (for general A, *i.e.*, when A has no special structure that you can take advantage of)? You do not have to give any code, but you must say clearly what the different steps in your algorithm are, and how many flops they cost.

7.9 Suppose you have to solve two sets of linear equations

$$Ax = b, \qquad (A + uv^T)y = b,$$

where A is $n \times n$ and given, and u, v, and b are given n-vectors. The variables are x and y. We assume that A and $A + uv^T$ are nonsingular.

The cost of solving the two systems from scratch is $(4/3)n^3$ flops. Give a more efficient method, based on the expression

$$(A + uv^T)^{-1} = A^{-1} - \frac{1}{1 + v^T A^{-1} u} A^{-1} uv^T A^{-1}.$$

Clearly state the different steps in your algorithm and give the flop count of each step (for large n). What is the total flop count (for large n)?

7.10 Consider the equation

$$AXA^T = B$$

where A and B are given $n \times n$-matrices, with A nonsingular. The variables are the n^2 elements of the $n \times n$-matrix X.

(a) Prove that there is a unique solution X.

(b) Can you give an efficient algorithm for computing X, based on factoring A and/or A^T? What is the cost of your algorithm (number of flops for large n)?

7.11 Assume A is a nonsingular $n \times n$ matrix. Show that thee inverse of the matrix

$$M = \begin{bmatrix} A & A + A^{-T} \\ A & A \end{bmatrix} \tag{10}$$

is given by

$$M^{-1} = \begin{bmatrix} -A^T & A^{-1} + A^T \\ A^T & -A^T \end{bmatrix}. \tag{11}$$

(a) Compare the cost (number of flops for large n) of the following two methods for solving a set of linear equations $Mx = b$, given A and b.

(i) Calculate A^{-1}, build the matrix M as defined in equation (10), and solve $Mx = b$ using the standard method This method would correspond to the MATLAB code

`x = [A,  A+inv(A)'; A, A] \ b`.

(ii) Calculate A^{-1}, build the matrix M^{-1} as defined in equation (11), and form the matrix vector product $x = M^{-1}b$. This method would correspond to the MATLAB code

`x = [-A' inv(A)+A'; A'  -A']*b`.

(b) Can you improve the fastest of the two methods described in part (a)? (You can state your answer in the form of an improved version of the MATLAB code given in part (a), but that is not necessary, as long as the steps in your method are clear.)

7.12 Consider the set of linear equations with a 3×3 block coefficient matrix

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & A_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}. \tag{12}$$

The $m \times m$ matrices A_{ij} and the m-vectors b_1, b_2, b_3 are given. The variables are the three m-vectors x_1, x_2, x_3. In other words we have $n = 3m$ equations in $3m$ variables. We assume that the matrices A_{11}, A_{22}, A_{33} are nonsingular.

(a) Describe an efficient method for solving (12). You do not have to write any MATLAB code, but you should state clearly what the different steps in your method are, and how many flops they cost. If you know several methods, you should select the most efficient one.

(b) Same question, assuming that $A_{11} = A_{22} = A_{33}$.

(c) Can you extend the algorithm of parts (a) and (b) to equations of the form

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1,K-2} & A_{1,K-1} & A_{1K} \\ 0 & A_{22} & \cdots & A_{2,K-2} & A_{2,K-1} & A_{2K} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & A_{K-2,K-2} & A_{K-2,K-1} & A_{K-2,K} \\ 0 & 0 & \cdots & 0 & A_{K-1,K-1} & A_{K-1,K} \\ 0 & 0 & \cdots & 0 & 0 & A_{KK} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{K-2} \\ x_{K-1} \\ x_K \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{K-2} \\ b_{K-1} \\ b_K \end{bmatrix}$$

where A_{ij} is $m \times m$ and b_i is an m-vector. The variables are the m-vectors x_i. The diagonal blocks A_{ii} are nonsingular. Compare the cost of your algorithm with the cost of a standard method for solving Km equations in Km variables.

7.13 Describe an efficient method for solving the equation

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ I & 0 & D \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} b \\ c \\ d \end{bmatrix}.$$

The nine blocks in the coefficient matrix have size $n \times n$. The matrix A is nonsingular, and the matrix D is diagonal with nonzero diagonal elements. The vectors b, c, and d in the right-hand side are n-vectors. The variables are the n-vectors x, y, z.

If you know several methods, give the most efficient one. Clearly state the different steps in your algorithm, give the complexity (number of flops) of each step, and the total number of flops.

7.14 We define a $2m \times 2m$ matrix

$$B = \begin{bmatrix} -2A & 4A \\ 3A & -5A \end{bmatrix},$$

where A is a nonsingular $m \times m$ matrix.

(a) Express the inverse of B in terms of A^{-1}.

(b) The cost of solving the linear equations

$$\begin{bmatrix} -2A & 4A \\ 3A & -5A \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

with variables $x_1 \in \mathbf{R}^m$, $x_2 \in \mathbf{R}^m$, using the standard method (*i.e.*, using the command

```
[-2*A  4*A; 3*A  -5*A] \ [b1; b2]
```

in MATLAB), is $(2/3)(2m)^3 = (16/3)m^3$ flops for large m.

Formulate a more efficient method. Clearly state the different steps in your algorithm and give the cost (number of flops for large m) of each step, as well as the total flop count. If you know several methods, give the most efficient one.

7.15 For each subproblem, describe an efficient method to evaluate the expression, and give a flop count. A is a nonsingular $n \times n$ matrix, and v_i, w_i, $i = 1, \ldots, m$, are n-vectors.

(a) $\sum_{i=1}^{m} v_i^T A^{-1} w_i$

(b) $\sum_{i=1}^{m} v_i^T (A + A^{-1}) w_i$

(c) $\sum_{i=1}^{m} \sum_{j=1}^{m} v_i^T A^{-1} w_j$

(d) $\sum_{i=1}^{m} \sum_{j=1}^{m} v_i w_j^T A^{-1} w_i v_j^T$

If you know several methods, choose the most efficient one. Include only the dominant terms in the flop counts, assuming m and n are large.

7.16 Let A be a nonsingular $n \times n$ matrix and b an n-vector. In each subproblem, describe an efficient method for computing the vector x and give a flop count of your algorithm, including terms of order two (n^2) and higher. If you know several methods, give the most efficient one (least number of flops for large n).

(a) $x = (A^{-1} + A^{-2})b$.

(b) $x = (A^{-1} + A^{-T})b$.

(c) $x = (A^{-1} + JA^{-1}J)b$ where J is the $n \times n$ matrix

$$J = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

(J is the identity matrix with its columns reversed: $J_{ij} = 1$ if $i + j = n + 1$ and $J_{ij} = 0$ otherwise.)

7.17 Suppose A and B are $n \times n$ matrices with A nonsingular, and b, c and d are n-vectors. Describe an efficient algorithm for solving the set of linear equations

$$\begin{bmatrix} A & B & 0 \\ 0 & A^T & B \\ 0 & 0 & A \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b \\ c \\ d \end{bmatrix}$$

with variables $x_1 \in \mathbf{R}^n$, $x_2 \in \mathbf{R}^n$, $x_3 \in \mathbf{R}^n$. Give a flop count for your algorithm, including all terms that are cubic or quadratic in n. If you know several methods, give the most efficient one (least number of flops for large n).

7.18 Consider the linear equation

$$(A + \epsilon B)x = b,$$

where A and B are given $n \times n$ matrices, b is a given n-vector, and ϵ is a scalar parameter. We assume that A is nonsingular, and therefore $A + \epsilon B$ is nonsingular for sufficiently small ϵ. The solution of the equation is

$$x(\epsilon) = (A + \epsilon B)^{-1}b,$$

a complicated nonlinear function of ϵ. In order to find a simple approximation of $x(\epsilon)$, valid for small ϵ, we can expand $x(\epsilon) = (A + \epsilon B)^{-1}b$ in a series

$$x(\epsilon) = x_0 + \epsilon x_1 + \epsilon^2 x_2 + \epsilon^3 x_3 + \cdots,$$

where x_0, x_1, x_2, x_3, ... are n-vectors, and then truncate the series after a few terms. To determine the coefficients x_i in the series, we examine the equation

$$(A + \epsilon B)(x_0 + \epsilon x_1 + \epsilon^2 x_2 + \epsilon^3 x_3 + \cdots) = b.$$

Expanding the product on the left-hand side gives

$$Ax_0 + \epsilon(Ax_1 + Bx_0) + \epsilon^2(Ax_2 + Bx_1) + \epsilon^3(Ax_3 + Bx_2) + \cdots = b.$$

We see that if this holds for all ϵ in a neighborhood of zero, the coefficients x_i must satisfy

$$Ax_0 = b, \qquad Ax_1 + Bx_0 = 0, \qquad Ax_2 + Bx_1 = 0, \qquad Ax_3 + Bx_2 = 0, \qquad \ldots. \tag{13}$$

Describe an efficient method for computing the first $k + 1$ coefficients x_0, ..., x_k from (13). What is the complexity of your method (number of flops for large n, assuming $k \ll n$)? If you know several methods, give the most efficient one.

7.19 Suppose A is a nonsingular $n \times n$ matrix.

(a) Show that the matrix

$$\begin{bmatrix} A & b \\ a^T & 1 \end{bmatrix}$$

is nonsingular if a and b are n-vectors that satisfy $a^T A^{-1} b \neq 1$.

(b) Suppose a_1, a_2, b_1, b_2 are n-vectors with $a_1^T A^{-1} b_1 \neq 1$ and $a_2^T A^{-1} b_2 \neq 1$. From part 1, this means that the coefficient matrices in the two equations

$$\begin{bmatrix} A & b_1 \\ a_1^T & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} c_1 \\ d_1 \end{bmatrix}, \qquad \begin{bmatrix} A & b_2 \\ a_2^T & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} c_2 \\ d_2 \end{bmatrix}$$

are nonsingular. Describe an efficient method for solving the two equations. The variables are the two n-vectors x_1 and x_2, and the two scalars y_1 and y_2.

If you know several methods, give the most efficient one. Take advantage of the fact that the 1,1 blocks of the two coefficient matrices are the same. What is the complexity of your method (number of flops for large n)?

7.20 Let A be a nonsingular $n \times n$ matrix and let u, v be two n-vectors that satisfy $v^T A^{-1} u \neq 1$.

(a) Show that

$$\begin{bmatrix} A & u \\ v^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{1 - v^T A^{-1} u} \begin{bmatrix} A^{-1}u \\ -1 \end{bmatrix} \begin{bmatrix} v^T A^{-1} & -1 \end{bmatrix}.$$

(b) Describe an efficient method for solving the two equations

$$Ax = b, \qquad \begin{bmatrix} A & u \\ v^T & 1 \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}.$$

The variables are the n-vectors x and y, and the scalar z.

Describe in detail the different steps in your algorihm and give a flop count of each step. If you know several methods, choose the most efficient one (least number of flops for large n).

7.21 Explain how you can solve the following problems using a single LU factorization and without computing matrix inverses. The matrix A is a given nonsingular $n \times n$ matrix. For each problem, carefully explain the different steps in your algorithm, give the cost of each step, and the total cost (number of flops for large n, excluding the LU factorization itself). If you know several methods, give the most efficient one.

(a) Compute $\left(A^{-1} + A^{-T}\right)^2 b$, where b is a given n-vector.

(b) Solve the equation $AXA = B$ for the unknown X, where B is a given $n \times n$-matrix.

(c) Compute $\sum_{i=1}^{n} \sum_{j=1}^{n} (A^{-1})_{ij}$, the sum of all the entries of A^{-1}.

7.22 Explain how you can solve the following problem using an LU factorization of A. Given a nonsingular $n \times n$ matrix A and two n-vectors b and c, find an n-vector x and a scalar y such that

$$Ax + yb = c \quad \text{and} \quad \|x\|^2 = 1.$$

We assume that $b \neq 0$ and $\|A^{-1}c\| < 1$. Clearly state every step in your algorithm. How many solutions (x, y) are there?

7.23 The pseudo-inverse of a right invertible matrix B is the matrix

$$B^\dagger = B^T(BB^T)^{-1}.$$

Note that $B^\dagger B$ is a symmetric matrix. It can be shown that $B^\dagger$ is the only right inverse X of B with the property that XB is symmetric.

(a) Assume A is a nonsingular $n \times n$ matrix and b is an n-vector. Show that the $n \times (n+1)$ matrix

$$B = \begin{bmatrix} A & b \end{bmatrix}$$

is right invertible and that

$$X = \begin{bmatrix} A^{-1} - A^{-1}by^T \\ y^T \end{bmatrix}$$

is a right inverse of B, for any value of the n-vector y.

(b) Show that XB is symmetric (hence, $X = B^\dagger$) if

$$y = \frac{1}{1 + \|A^{-1}b\|^2} A^{-T}A^{-1}b.$$

(c) What is the complexity of computing the vector y in part (b) using an LU factorization of A? Give a flop count, including all cubic and quadratic terms. If you know several methods, consider the most efficient one.

7.24 Let A be an $n \times m$ matrix and B an $m \times n$ matrix. We compare the complexity of two methods for solving

$$(I + AB)x = b.$$

We assume the matrix $I + AB$ is nonsingular.

(a) In the first method we compute the matrix $C = I + AB$ and then solve $Cx = b$ using the standard method (LU factorization). Give the complexity of this method.

(b) Suppose the matrix $I + BA$ is nonsingular. Show that

$$(I + AB)^{-1} = I - A(I + BA)^{-1}B.$$

(c) This suggests a second method for solving the equation: compute the solution via the formula

$$x = \big(I - A(I + BA)^{-1}B\big)\, b.$$

Describe an efficient method for evaluating this formula and give the complexity. Which of the two methods is faster when $m \ll n$? Explain your answer.

7.25 Let A be an $n \times n$ matrix that has an LU factorization

$$A = LU$$

(*i.e.*, the general LU factorization $A = PLU$ with $P = I$). Suppose you are given L and U. Show that the ith diagonal element of the inverse,

$$(A^{-1})_{ii},$$

can be computed in $2(n - i)^2$ flops if we ignore terms that are linear in n or constant.

8 Least squares

8.1 Formulate the following problems as least squares problems. For each problem, give a matrix A and a vector b such that the problem can be expressed as

$$\text{minimize} \quad \|Ax - b\|^2.$$

(You are not asked to solve the problems.)

(a) Minimize $x_1^2 + 2x_2^2 + 3x_3^2 + (x_1 - x_2 + x_3 - 1)^2 + (-x_1 - 4x_2 + 2)^2$.

(b) Minimize $(-6x_2 + 4)^2 + (-4x_1 + 3x_2 - 1)^2 + (x_1 + 8x_2 - 3)^2$.

(c) Minimize $2(-6x_2 + 4)^2 + 3(-4x_1 + 3x_2 - 1)^2 + 4(x_1 + 8x_2 - 3)^2$.

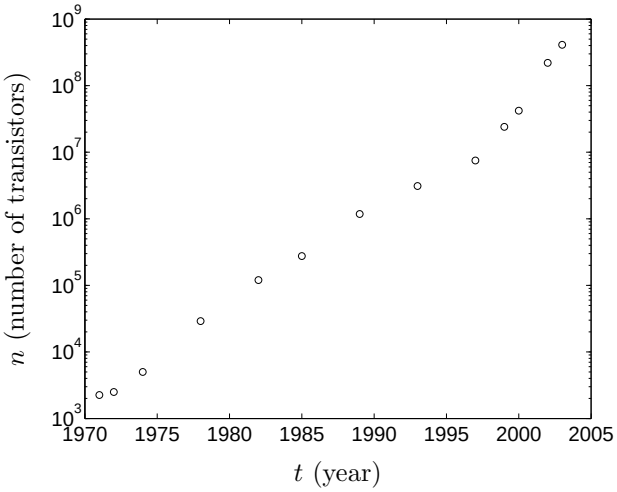
(d) Minimize $x^T x + \|Bx - d\|^2$ where the $p \times n$ matrix B and the p-vector d are given.

(e) Minimize $\|Bx - d\|^2 + 2\|Fx - g\|^2$. The $p \times n$ matrix B, the $l \times n$ matrix F, the p-vector d and the l-vector g are given.

(f) Minimize $x^T Dx + \|Bx - d\|^2$. D is a $n \times n$ diagonal matrix with positive diagonal elements, B is $p \times n$, and d is a p-vector. D, B and D are given.

8.2 *Moore's law.* The figure and the table show the number of transistors in 13 microprocessors, and the year of their introduction.

year	transistors
1971	2,250
1972	2,500
1974	5,000
1978	29,000
1982	120,000
1985	275,000
1989	1,180,000
1993	3,100,000
1997	7,500,000
1999	24,000,000
2000	42,000,000
2002	220,000,000
2003	410,000,000



These numbers in the table are available in the MATLAB file `mooreslaw.m`. The command `[t, n] = mooreslaw` creates arrays `t` with the first column of the table (introduction year) and `n` with the second column (number of transistors). The right-hand plot was produced by the MATLAB command `semilogy(t, n, 'o')`). The plot suggests that we can obtain a good fit with a function of the form

$$n(t) = \alpha^{t - t_0},$$

where t is the year, and n is the number of transistors. This is a straight line if we plot $n(t)$ on a logarithmic scale versus t on a linear scale. In this problem we use least squares to estimate the parameters α and t_0.

Explain how you would use least squares to find α and t_0 such that

$$n_i \approx \alpha^{t_i - t_0}, \quad i = 1, \ldots, 13,$$

and solve the least squares problem in MATLAB. Compare your result with Moore's law, which states that the number of transistors per integrated circuit roughly doubles every two years.

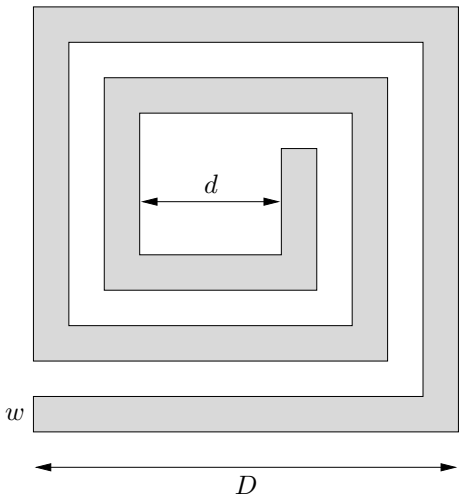
Remark. The MATLAB command to solve a least squares problem

$$\text{minimize} \quad \|Ax - b\|^2$$

is `x = A \ b`, *i.e.*, the same command as for solving a set of linear equations. The meaning of the backslash operator therefore depends on the context. If A is a square matrix, then `A \ b` solves the linear set of equations $Ax = b$; if A is tall, it solves the least squares problem.

8.3 The figure shows a planar spiral inductor, implemented in CMOS, for use in RF circuits. The inductor is characterized by four key parameters:

- n, the number of turns (which is a multiple of 1/4, but that needn't concern us)
- w, the width of the wire
- d, the inner diameter
- D, the outer diameter



The inductance L of such an inductor is a complicated function of the parameters n, w, d, and D. It can be found by solving Maxwell's equations, which takes considerable computer time, or by fabricating the inductor and measuring the inductance. In this problem you will develop a simple approximate inductance model of the form

$$\hat{L} = \alpha n^{\beta_1} w^{\beta_2} d^{\beta_3} D^{\beta_4},$$

where $\alpha, \beta_1, \beta_2, \beta_3, \beta_4 \in \mathbf{R}$ are constants that characterize the approximate model. (Since L is positive, we have $\alpha > 0$, but the constants $\beta_2, \ldots, \beta_4$ can be negative.) This simple approximate model, if accurate enough, can be used for design of planar spiral inductors.

The file `inductordata.m` contains data for 50 inductors, obtained from measurements. Download the file, and execute it in MATLAB using `[n, w, d, D, L] = inductordata`. This generates 5 vectors n, w, d, D, L of length 50. The ith elements of these vectors are the parameters n_i, w_i (in μm), d_i (in μm), D_i (in μm) and the inductance L_i (in nH) for inductor i.. Thus, for example, w_{13} gives the wire width of inductor 13.

Your task is to find α, $\beta_1, \ldots, \beta_4$ so that

$$\widehat{L}_i = \alpha n_i^{\beta_1} w_i^{\beta_2} d_i^{\beta_3} D_i^{\beta_4} \approx L_i \qquad \text{for } i = 1, \ldots, 50.$$

Your solution must include a clear description of how you found your parameters, as well as their actual numerical values.

Note that we have not specified the criterion that you use to judge the approximate model (*i.e.*, the fit between $\widehat{L}_i$ and L_i); we leave that to your judgment.

We can define the *percentage error* between $\widehat{L}_i$ and L_i as

$$e_i = 100\frac{|\widehat{L}_i - L_i|}{L_i}.$$

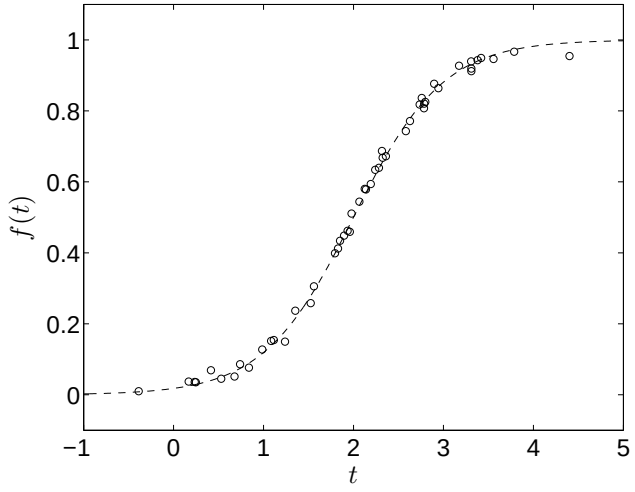
Find the average percentage error for the 50 inductors, *i.e.*, $(e_1 + \cdots + e_{50})/50$, for your model. (We are only asking you to find the average percentage error for your model; we do not require that your model minimize the average percentage error.)

Remark. For details on solving least squares problems in MATLAB, see the remark at the end of exercise 8.2.

8.4 The figure shows $m = 50$ points (t_i, y_i) as circles. These points are well approximated by a function of the form

$$f(t) = \frac{e^{\alpha t+\beta}}{1 + e^{\alpha t+\beta}}.$$

(An example, for two specific values of α and β, is shown in dashed line).



Formulate the following problem as a linear least squares problem. Find values of the parameters α, β such that

$$\frac{e^{\alpha t_i+\beta}}{1 + e^{\alpha t_i+\beta}} \approx y_i, \qquad i = 1, \ldots, m, \tag{14}$$

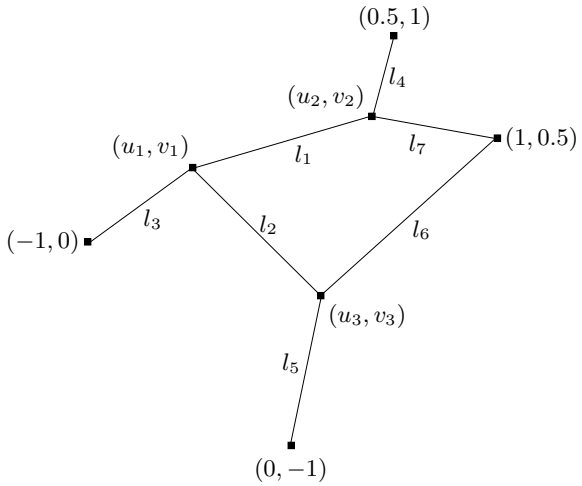
You can assume that $0 < y_i < 1$ for $i = 1, \ldots, m$.

Clearly state the error function you choose to measure the quality of the fit in (14), and the matrix A and the vector b of the least squares problem. Test your method on the example data in the file `logistic_fit.m`. (The command `[t, y] = logistic_fit;` creates arrays with the points t_i, y_i.)

8.5 We have N points in $\mathbf{R}^2$, and a list of pairs of points that must be connected by links. The positions of some of the N points are fixed; our task is to determine the positions of the remaining points. The objective is to place the points so that some measure of the total interconnection length of the links is minimized. As an example application, we can think of the points as locations of plants or warehouses of a company, and the links as the routes over which goods must be shipped. The goal is to find locations that minimize the total transportation cost. In another application, the points represent the position of modules or cells on an integrated circuit, and the links represent wires that connect pairs of cells. Here the goal might be to place the cells in such a way that the the total length of wire used to interconnect the cells is minimized.

The problem can be described in terms of a graph with N nodes, representing the N points. With each free node we associate a variable $(u_i, v_i) \in \mathbf{R}^2$, which represents its location or position.

In this problem we will consider the example shown in the figure below.



Here we have 3 free points with coordinates (u_1, v_1), (u_2, v_2), (u_3, v_3). We have 4 fixed points, with coordinates $(-1, 0)$, $(0.5, 1)$, $(0, -1)$, and $(1, 0.5)$. There are 7 links, with lengths l_1, l_2, ..., l_7. We are interested in finding the coordinates (u_1, v_1), (u_2, v_2) and (u_3, v_3) that minimize the total squared length

$$l_1^2 + l_2^2 + l_3^2 + l_4^2 + l_5^2 + l_6^2 + l_7^2.$$

(a) Formulate this problem as a least squares problem

$$\text{minimize} \quad \|Ax - b\|^2$$

where the 6-vector x contains the six variables u_1, u_2, u_3, v_1, v_2, v_3. Give the coefficient matrix A and the vector b.

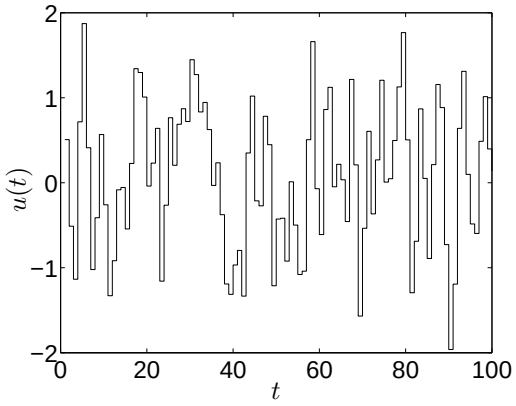
(b) Show that you can also obtain the optimal coordinates by solving two smaller least squares problems

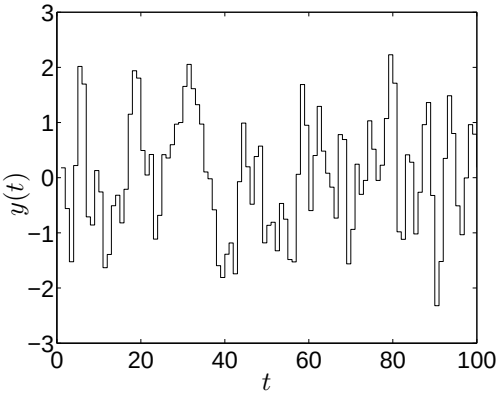
$$\text{minimize} \quad \|\bar{A}u - \bar{b}\|^2, \qquad \text{minimize} \quad \|\hat{A}v - \hat{b}\|^2,$$

where $u = (u_1, u_2, u_3)$ and $v = (v_1, v_2, v_3)$. Give the coefficient matrices $\bar{A}$, $\hat{A}$ and the vectors $\bar{b}$ and $\hat{b}$. What is the relation between $\bar{A}$ and $\hat{A}$?

(c) Solve the least squares problems derived in part (a) or (b) using MATLAB.

8.6 *Least squares model fitting.* In this problem we use least squares to fit several different types of models to a given set of input-output data. The data set consists of a scalar input sequence $u(1)$, $u(2)$, ..., $u(N)$, and a scalar output sequence $y(1)$, $y(2)$, ..., $y(N)$, with $N = 100$. The signals are shown in the following plots.





We will develop and compare seven different models that relate the signals u and y. The models range in complexity from a simple constant to a nonlinear dynamic model:

(a) constant model: $y(t) = \alpha$

(b) static linear: $y(t) = \beta u(t)$

(c) static affine: $y(t) = \alpha + \beta u(t)$

(d) static quadratic: $y(t) = \alpha + \beta u(t) + \gamma u(t)^2$

(e) linear, 2-tap: $y(t) = \beta_1 u(t) + \beta_2 u(t-1)$

(f) affine, 2-tap: $y(t) = \alpha + \beta_1 u(t) + \beta_2 u(t-1)$

(g) quadratic, 2-tap: $y(t) = \alpha + \beta_1 u(t) + \gamma_1 u(t)^2 + \beta_2 u(t-1) + \gamma_2 u(t-1)^2 + \delta u(t)u(t-1)$.

The first four models are *memoryless*. In a memoryless model the output at time t, *i.e.*, $y(t)$, depends only the input at time t, *i.e.*, $u(t)$. Another common term for such a model is *static*.

In a *dynamic model*, $y(t)$ depends on $u(s)$ for some $s \neq t$. Models (e), (f), and (g) are dynamic models, in which the current output depends on the current input and the previous input. Such models are said to have a *finite memory* of length one. Another term is 2-tap system (the taps refer to taps on a delay line).

Each of the models is specified by a number of parameters, *i.e.*, the scalars α, β, etc. You are asked to find least squares estimates ($\hat{\alpha}$, $\hat{\beta}$, ...) for the parameters, *i.e.*, the values that minimize the sum-of-squares of the errors between predicted outputs and actual outputs. Your solutions should include:

- a clear description of the least squares problems that you solve
- the computed values of the least squares estimates of the parameters
- a plot of the predicted output $\hat{y}(t)$
- a plot of the residual $\hat{y}(t) - y(t)$
- the root-mean-square (RMS) residual, *i.e.*, the squareroot of the mean of the squared residuals.

For example, the affine 2-tap model (part (f)) depends on three parameters α, β_1, and β_2. The least squares estimates $\hat{\alpha}$, $\hat{\beta}_1$, $\hat{\beta}_2$ are found by minimizing

$$\sum_{t=2}^{N} \left(y(t) - \alpha - \beta_1 u(t) - \beta_2 u(t-1)\right)^2 .$$

(Note that we start at $t = 2$ so $u(t-1)$ is defined). You are asked to formulate this as a least squares problem, solve it to find $\hat{\alpha}$, $\hat{\beta}_1$, and $\hat{\beta}_2$, plot the predicted output

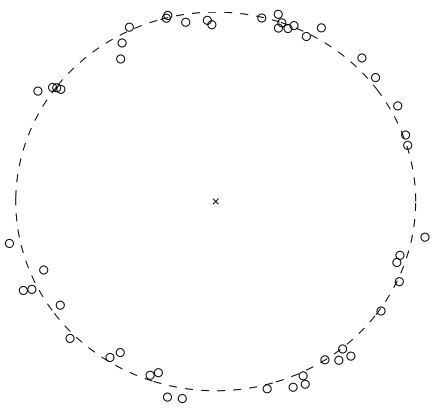
$$\hat{y}(t) = \hat{\alpha} + \hat{\beta}_1 u(t) + \hat{\beta}_2 u(t-1),$$

and the residual $r(t) = \hat{y}(t) - y(t)$, for $t = 2, \ldots, N$, and give the value of the RMS residual

$$R_{\rm rms} = \left(\frac{1}{N-1} \sum_{t=2}^{N} (y(t) - \hat{y}(t))^2 \right)^{1/2}.$$

The data for the problem are available from the class webpage in the m-file `systemid.m`. The command is `[u, y] = systemid`.

8.7 In this problem we use least squares to fit a circle to given points (u_i, v_i) in a plane, as shown in the figure.



We use $(u_{\rm c}, v_{\rm c})$ to denote the center of the circle and R for its radius. A point (u, v) is on the circle if $(u - u_{\rm c})^2 + (v - v_{\rm c})^2 = R^2$. We can therefore formulate the fitting problem as

$$\text{minimize} \quad \sum_{i=1}^{m} \left((u_i - u_{\rm c})^2 + (v_i - v_{\rm c})^2 - R^2 \right)^2$$

with variables $u_{\rm c}$, $v_{\rm c}$, R.

Show that this can be written as a linear least squares problem if we make a change of variables and use as variables $u_{\rm c}$, $v_{\rm c}$, and $w = u_{\rm c}^2 + v_{\rm c}^2 - R^2$.

(a) Define A, b, and x in the equivalent linear least squares formulation.

(b) Show that the optimal solution $u_{\rm c}$, $v_{\rm c}$, w of the least squares problem satisfies $u_{\rm c}^2 + v_{\rm c}^2 - w \geq 0$. (This is necessary to compute $R = \sqrt{u_{\rm c}^2 + v_{\rm c}^2 - w}$ from the result $u_{\rm c}$, $v_{\rm c}$, w.)

Test your formulation on the problem data in the file `circlefit.m` on the course website. The commands

```
[u,v] = circlefit;
plot(u, v, 'o');
axis equal
```

will create a plot of the $m = 50$ points (u_i, v_i) in the figure. The following code plots the 50 points and the computed circle.

```
t = linspace(0, 2*pi, 1000);
plot(u, v, 'o',  R * cos(t) + uc, R * sin(t) + vc, '-');
axis equal
```

(assuming your MATLAB variables are called `uc`, `vc`, and `R`).

8.8 Suppose you are given m ($m \geq 2$) straight lines

$$L_i = \{p_i + t_i q_i \mid t_i \in \mathbf{R}\}, \quad i = 1, \ldots, m$$

in $\mathbf{R}^n$. Each line is defined by two n-vectors p_i, q_i. The vector p_i is a point on the line; the vector q_i specifies the direction. We assume that the vectors q_i are normalized ($\|q_i\| = 1$) and that at least two of them are linearly independent. (In other words, the vectors q_i are not all scalar multiples of the same vector, so the lines are not all parallel.) We denote by

$$d_i(y) = \min_{u_i \in L_i} \|y - u_i\| = \min_{t_i} \|y - p_i - t_i q_i\|$$

the distance of a point y to the line L_i.

Express the following problem as a linear least squares problem. Find the point $y \in \mathbf{R}^n$ that minimizes the sum of its squared distances to the m lines, *i.e.*, find the solution of the optimization problem

$$\text{minimize} \quad \sum_{i=1}^{m} d_i(y)^2$$

with variable y. Express the least squares problem in the standard form

$$\text{minimize} \quad \|Ax - b\|^2$$

where A has linearly independent columns.

(a) Clearly state what the variables x in the least squares problem are and how A and b are defined.

(b) Explain why A has linearly independent columns.

8.9 Let $\hat{x}$ be the solution of the least squares problem

$$\text{minimize } \|Ax - b\|^2$$

and let $r = b - A\hat{x}$ be the residual. Suppose the elements in the first column of A are all equal to one.

(a) Show that $\mathbf{avg}(r) = 0$, *i.e.*,

$$\mathbf{avg}(A\hat{x}) = \mathbf{avg}(b).$$

(b) Show that

$$\mathbf{std}(A\hat{x})^2 + \mathbf{std}(r)^2 = \mathbf{std}(b)^2.$$

We use the notation $\mathbf{avg}(y) = (\mathbf{1}^T y)/m$ for the average of an m-vector y and $\mathbf{std}(y) = \|a - \mathbf{avg}(a)\mathbf{1}\|/\sqrt{m}$ for its standard deviation.

8.10 Let A be an $m \times n$ matrix with linearly independent columns.

(a) Show that the $(m+n) \times (m+n)$ matrix

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}$$

is nonsingular.

(b) Show that the solution $\bar{x}$, $\bar{y}$ of the set of linear equations

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

is given by $\bar{x} = b - Ax_{\text{ls}}$ and $\bar{y} = x_{\text{ls}}$, where x_{ls} is the solution of the least squares problem

$$\text{minimize} \quad \|Ax - b\|^2.$$

8.11 Consider the set of $p+q$ linear equations in $p+q$ variables

$$\begin{bmatrix} I & A \\ A^T & -I \end{bmatrix} \begin{bmatrix} \bar{y} \\ \bar{x} \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}.$$

The $p \times q$ matrix A, the p-vector b, and the q-vector c are given. The variables are the q-vector $\bar{x}$ and the p-vector $\bar{y}$.

(a) Show that the coefficient matrix

$$\begin{bmatrix} I & A \\ A^T & -I \end{bmatrix}$$

is nonsingular, regardless of the dimensions of A.

(b) From part (a) we know that the solution $\bar{x}$, $\bar{y}$ is unique. Show that $\bar{x}$ minimizes

$$\|Ax - b\|^2 + \|x + c\|^2.$$

8.12 Explain how you can solve the following problems using the QR factorization.

(a) Find the vector x that minimizes

$$\|Ax - b_1\|^2 + \|Ax - b_2\|^2.$$

The problem data are the $m \times n$ matrix A and two m-vectors b_1 and b_2. The matrix A has linearly independent columns. If you know several methods, give the most efficient one.

(b) Find x_1 and x_2 that minimize

$$\|Ax_1 - b_1\|^2 + \|Ax_2 - b_2\|^2.$$

The problem data are the $m \times n$ matrix A, and the m-vectors b_1 and b_2. The matrix A has linearly independent columns.

8.13 *Solving normal equations versus QR factorization.* In this problem we compare the accuracy of the two methods for solving a least squares problem

$$\text{minimize} \quad \|Ax - b\|^2.$$

We take

$$A = \begin{bmatrix} 1 & 1 \\ 10^{-k} & 0 \\ 0 & 10^{-k} \end{bmatrix}, \qquad b = \begin{bmatrix} -10^{-k} \\ 1 + 10^{-k} \\ 1 - 10^{-k} \end{bmatrix},$$

for $k = 6$, $k = 7$ and $k = 8$.

(a) Write the normal equations, and solve them analytically (*i.e.*, on paper, without using MATLAB).

(b) Solve the least squares problem in MATLAB, for $k = 6$, $k = 7$ and $k = 8$, using the recommended method `x = A \ b`. This method is based on the QR factorization.

(c) Repeat part (b), using `x = (A'*A) \ (A'*b)`. Compare the results of this method with the results of parts (a) and (b).

Remark. Type `format long` to make MATLAB display more than five digits.

8.14 *Least squares updating.* Suppose $\hat{x}$ is the solution of the least squares problem

$$\text{minimize} \quad \|Ax - b\|^2$$

where A is an $m \times n$ matrix with linearly independent columns and b is an m-vector.

(a) Show that the solution of the problem

$$\text{minimize} \quad \|Ay - b\|^2 + (c^T y - d)^2$$

with variable y (where c is an n-vector, and d is a scalar) is given by

$$\hat{y} = \hat{x} + \frac{d - c^T \hat{x}}{1 + c^T (A^T A)^{-1} c} (A^T A)^{-1} c.$$

(b) Describe an efficient method for computing $\hat{x}$ and $\hat{y}$, given A, b, c and d, using the QR factorization of A. Clearly describe the different steps in your algorithm. Give a flop count for each step and a total flop count. In your total flop count, include all terms that are cubic (n^3, mn^2, m^2n, m^3) and quadratic (m^2, mn, n^2).

If you know several methods, give the most efficient one.

8.15 *Least squares downdating.* Let A be an $m \times n$ matrix and b and m-vector. We assume that A has linearly independent columns and define $\hat{x}$ as the solution of the least squares problem

$$\text{minimize} \quad \|Ax - b\|^2 = \sum_{i=1}^{m} (a_i^T x - b_i)^2 \tag{15}$$

(a_i^T is the ith row of A). We also define $\hat{y}_k$ as the solution of the least squares problem (15) with the kth term in the sum, corresponding to row k of A, removed:

$$\text{minimize} \quad \sum_{i \neq k} (a_i^T y - b_i)^2 = \sum_{i=1}^{k-1} (a_i^T y - b_i)^2 + \sum_{i=k+1}^{m} (a_i^T y - b_i)^2. \tag{16}$$

We assume that for each $k = 1, \ldots, m$, the matrix formed by removing row k from A has linearly independent columns, so the solution $\hat{y}_k$ of (16) is unique.

(a) Consider the least squares problem

$$\text{minimize} \quad \sum_{i=1}^{k-1} (a_i^T y - b_i)^2 + (a_k^T y + z - b_k)^2 + \sum_{i=k+1}^{m} (a_i^T y - b_i)^2, \tag{17}$$

with variables $y \in \mathbf{R}^n$ and $z \in \mathbf{R}$. Give a simple argument why the solution is equal to $y = \hat{y}_k$ and $z = b_k - a_k^T \hat{y}_k$. In other words, explain why the y-component of the solution of (17) is also the solution of (16).

(b) Use the normal equations for (17) to show that

$$\hat{y}_k = \hat{x} - \frac{b_k - a_k^T \hat{x}}{1 - a_k^T (A^T A)^{-1} a_k} (A^T A)^{-1} a_k.$$

(c) Give an efficient algorithm, based on the QR factorization of A and the expression in part (b), for computing $\hat{x}$ and the m vectors $\hat{y}_1, \ldots, \hat{y}_m$. Clearly explain the steps in your algorithm and give the overall complexity (dominant term in the flop count as a function of m and n). If you know several algorithms, give the most efficient one.

8.16 Let $\hat{x}$ be the solution of the least squares problem

$$\text{minimize} \quad \|Ax - b\|^2$$

where A is an $m \times n$ matrix with linearly independent columns. Suppose a is an m-vector, not in the range of A. Let $(\hat{y}, \hat{z})$ be the solution of the least squares problem

$$\text{minimize} \quad \left\| \begin{bmatrix} A & a \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} - b \right\|^2.$$

The variables in this problem are the n-vector y and the scalar z.

(a) Show that

$$\begin{bmatrix} \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} \hat{x} \\ 0 \end{bmatrix} - \frac{a^T(b - A\hat{x})}{a^T(I - A(A^TA)^{-1}A^T)a} \begin{bmatrix} (A^TA)^{-1}A^Ta \\ -1 \end{bmatrix}.$$

(b) Formulate an efficient algorithm for computing $\hat{x}$, $\hat{y}$, $\hat{z}$, using the QR factorization of A. Give the complexity of the algorithm (including all cubic and quadratic terms in the flop count).

8.17 Let A be an $m \times n$ matrix with linearly independent columns, and b an m-vector not in the range of A.

(a) Explain why the QR factorization

$$\begin{bmatrix} A & b \end{bmatrix} = QR$$

exists.

(b) Suppose we partition the matrices in the QR factorization of part (a) as

$$Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}, \qquad R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix},$$

where Q_1 is $m \times n$, Q_2 is $m \times 1$, R_{11} is $n \times n$, R_{12} is $n \times 1$ and R_{22} is a scalar. Show that $x_{\text{ls}} = R_{11}^{-1}R_{12}$ is the solution of the least squares problem

$$\text{minimize} \quad \|Ax - b\|^2$$

and that $R_{22} = \|Ax_{\text{ls}} - b\|$.

8.18 Let $\hat{x}$ and $\hat{y}$ be the solutions of the least squares problems

$$\text{minimize} \quad \|Ax - b\|^2, \qquad \text{minimize} \quad \|Ay - c\|^2$$

where A is an $m \times n$ matrix with linearly independent columns, and b and c are m-vectors. We assume that $A\hat{x} \neq b$.

(a) Show that the $m \times (n+1)$ matrix $\begin{bmatrix} A & b \end{bmatrix}$ has linearly independent columns.

(b) Show that the solution of the least squares problem

$$\text{minimize} \quad \left\| \begin{bmatrix} A & b \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} - c \right\|^2,$$

with variables $u \in \mathbf{R}^n$, $v \in \mathbf{R}$, is given by

$$\hat{u} = \hat{y} - \frac{b^Tc - b^TA\hat{y}}{b^Tb - b^TA\hat{x}}\hat{x}, \qquad \hat{v} = \frac{b^Tc - b^TA\hat{y}}{b^Tb - b^TA\hat{x}}.$$

(c) Describe an efficient method for computing $\hat{x}$, $\hat{y}$, $\hat{u}$, $\hat{v}$, given A, b, c, using the QR factorization of A. Clearly describe the different steps in your algorithm. Give a flop count for each step and a total flop count. In the total flop count, include all terms that are cubic (n^3, mn^2, m^2n, m^3) and quadratic (m^2, mn, n^2). If you know several methods, give the most efficient one (least number of flops for large m and n).

8.19 An $m \times n$ matrix A is given in factored form

$$A = UDV^T$$

where U is $m \times n$ with orthonormal columns, D is $n \times n$ and diagonal with nonzero diagonal elements, and V is $n \times n$ and orthogonal. Describe an efficient method for solving the least squares problem

$$\text{minimize} \quad \|Ax - b\|^2.$$

'Efficient' here means that the complexity is substantially less than the complexity of the standard method based on the QR factorization. What is the cost of your algorithm (number of flops for large m and n)? (Note: we assume that U, D, V are given; you are not asked to include the cost of computing these matrices in the complexity analysis.)

9 Multi-objective least squares

9.1 Formulate the following problem as a least squares problem. Find a polynomial

$$p(t) = x_1 + x_2 t + x_3 t^2 + x_4 t^3$$

that satisfies the following conditions.

- The values $p(t_i)$ at 4 given points t_i in the interval $[0, 1]$ is approximately equal to given values y_i:

  $$p(t_i) \approx y_i, \quad i = 1, \ldots, 4.$$

  The points t_i are given and distinct ($t_i \neq t_j$ for $i \neq j$). The values y_i are also given.
- The derivatives of p at $t = 0$ and $t = 1$ are small:

  $$p'(0) \approx 0, \qquad p'(1) \approx 0.$$

- The average value of p over the interval $[0, 1]$ is approximately equal to the value at $t = 1/2$:

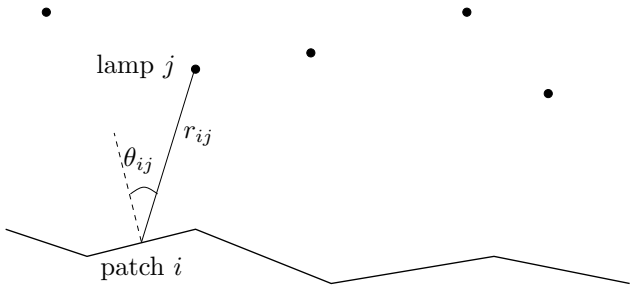
  $$\int_0^1 p(t)\, dt \approx p(1/2).$$

To determine coefficients x_i that satisfy these conditions, we minimize

$$E(x) = \frac{1}{4}\sum_{i=1}^{4}(p(t_i) - y_i)^2 + p'(0)^2 + p'(1)^2 + \left(\int_0^1 p(t)\, dt - p(1/2)\right)^2.$$

Give A and b such that $E(x) = \|Ax - b\|^2$. Clearly state the dimensions of A and b, and what their elements are.

9.2 The figure shows an illumination system of n lamps illuminating m flat patches. The variables in the problem are the lamp powers $x_1, \ldots, x_n$, which can vary between 0 and 1.



The illumination intensity at (the midpoint of) patch i is denoted I_i. We will use a simple linear model for the illumination intensities I_i as a function of the lamp powers x_j: for $i = 1, \ldots, m$,

$$I_i = \sum_{j=1}^{n} A_{ij} x_j.$$

The matrix A (with coefficients A_{ij}) is available from the class webpage (see below), and was constructed as follows. We take

$$A_{ij} = r_{ij}^{-2} \max\{\cos\theta_{ij}, 0\},$$

where r_{ij} denotes the distance between lamp j and the midpoint of patch i, and θ_{ij} denotes the angle between the upward normal of patch i and the vector from the midpoint of patch i to lamp j, as shown in the figure. This model takes into account "self-shading" (*i.e.*, the fact that a patch is illuminated only by lamps in the halfspace it faces) but not shading of one patch caused by another. Of course we could use a more complex illumination model, including shading and even reflections. This just changes the matrix relating the lamp powers to the patch illumination levels.

The problem is to determine lamp powers that make the illumination levels I_i close to a given desired level $I_{\rm des}$. In other words, we want to choose the n-vector x such that

$$\sum_{j=1}^{n} A_{ij}x_j \approx I_{\rm des}, \quad i = 1, \ldots, m,$$

but we also have to observe the power limits $0 \leq x_j \leq 1$. This is an example of a *constrained optimization problem*. The objective is to achieve an illumination level that is as uniform as possible; the constraint is that the components of x must satisfy $0 \leq x_j \leq 1$. Finding the exact solution of this minimization problem requires specialized numerical techniques for constrained optimization. However, we can solve it *approximately* using least squares.

In this problem we consider two approximate methods that are based on least squares, and compare them for the data generated using `[A, Ides] = illumdata`, with the MATLAB file `illumdata.m` for the course website. The elements of A are the coefficients A_{ij}. In this example we have $m = 11$, $n = 7$ so A is 11×7, and $I_{\rm des} = 2$.

(a) *Saturate the least squares solution.* The first method is simply to ignore the bounds on the lamp powers. We solve the least squares problem

$$\text{minimize} \quad \sum_{i=1}^{m}(\sum_{j=1}^{n} A_{ij}x_j - I_{\rm des})^2$$

ignoring the constraints $0 \leq x_j \leq 1$. If we are lucky, the solution will satisfy the bounds $0 \leq x_j \leq 1$, for $j = 1, \ldots, n$. If not, we replace x_j with zero if $x_j < 0$ and with one if $x_j > 1$.

Apply this method to the problem data generated by `illumdata.m`, and calculate the resulting value of the cost function $\sum_{i=1}^{m}(I_i - I_{\rm des})^2$.

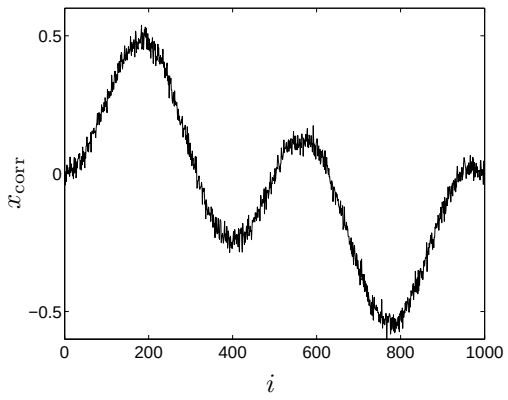
(b) *Weighted least squares.* The second method is to solve the problem

$$\text{minimize} \quad \sum_{i=1}^{m}(\sum_{j=1}^{n} A_{ij}x_j - I_{\rm des})^2 + \mu\sum_{j=1}^{n}(x_j - 0.5)^2,$$

where the constant $\mu \geq 0$ is used to attach a cost to the deviation of the powers from the value 0.5, which lies in the middle of the power limits. For $\mu = 0$, this is the same least squares problem as in part (a). If we take μ large enough, the solution of this problem will satisfy $0 \leq x_j \leq 1$.

Formulate this problem as a least squares problem in the variables x, and solve it for $\mu = 1$, $\mu = 2$, $\mu = 3$, etc., until you find a value of μ such that all components of the solution x satisfy $0 \leq x_j \leq 1$. For that solution x, calculate the cost function $\sum_{i=1}^{m}(I_i - I_{\rm des})^2$ and compare with the value you obtained in part (a).

9.3 *De-noising using least squares.* The figure shows a signal of length 1000, corrupted with noise. We are asked to estimate the original signal. This is called signal reconstruction, or de-noising, or smoothing. In this problem we apply a smoothing method based on least squares.



We will represent the corrupted signal as a vector $x_{\rm cor}$ of size 1000. (The values can be obtained as `xcor = lsdenoising` using the file `lsdenoising.m`.) The estimated signal (*i.e.*, the variable in the problem) will be represented as a vector $\hat{x}$ of size 1000.

The idea of the method is as follows. We assume that the noise in the signal is the small and rapidly varying component. To reconstruct the signal, we decompose $x_{\rm cor}$ in two parts

$$x_{\rm cor} = \hat{x} + v$$

where v is small and rapidly varying, and $\hat{x}$ is close to $x_{\rm cor}$ ($\hat{x} \approx x_{\rm cor}$) and slowly varying ($\hat{x}_{i+1} \approx \hat{x}_i$). We can achieve such a decomposition by choosing $\hat{x}$ as the solution of the least squares problem

$$\text{minimize} \quad \|x - x_{\rm cor}\|^2 + \mu \sum_{i=1}^{999} (x_{i+1} - x_i)^2, \tag{18}$$

where μ is a positive constant. The first term $\|x - x_{\rm cor}\|^2$ measures how much x deviates from $x_{\rm cor}$. The second term, $\sum_{i=1}^{999}(x_{i+1} - x_i)^2$, penalizes rapid changes of the signal between two samples. By minimizing a weighted sum of both terms, we obtain an estimate $\hat{x}$ that is close to $x_{\rm cor}$ (*i.e.*, has a small value of $\|\hat{x} - x_{\rm cor}\|^2$) and varies slowly (*i.e.*, has a small value of $\sum_{i=1}^{999}(\hat{x}_{i+1} - \hat{x}_i)^2$). The parameter μ is used to adjust the relative weight of both terms.

Problem (18) is a least squares problem, because it can be expressed as

$$\text{minimize} \quad \|Ax - b\|^2$$

where

$$A = \begin{bmatrix} I \\ \sqrt{\mu}\, D \end{bmatrix}, \qquad b = \begin{bmatrix} x_{\rm cor} \\ 0 \end{bmatrix},$$

and D is a 999×1000-matrix defined as

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

The matrix A is quite large (1999×1000), but also very sparse, so we will solve the least squares problem by solving the normal equations

$$(I + \mu D^T D)x = x_{\rm cor}. \tag{19}$$

MATLAB provides special routines for solving sparse linear equations, and they are used as follows. There are two types of matrices: full (or dense) and sparse. If you define a matrix, it is considered full by default, unless you specify that it is sparse. You can convert a full matrix to sparse format using the command `A = sparse(A)`, and a sparse matrix to full format using the command `A = full(A)`.

When you type `x = A \ b` where A is $n \times n$, MATLAB chooses different algorithms depending on the type of A. If A is full it uses the standard methods for general matrices. If A is sparse, it uses an algorithm that takes advantage of sparsity. In our application, the matrix $I + \mu D^T D$ is sparse (in fact tridiagonal), so if we make sure to define it as a sparse matrix, the normal equations will be solved much more quickly than if we ignore the sparsity.

The command to create a sparse zero matrix of dimension $m \times n$ is `A = sparse(m,n)`. The command `A = speye(n)` creates a sparse $n \times n$-identity matrix. If you add or multiply sparse matrices, the result is automatically considered sparse.

This means you can solve the normal equations (19) by the following MATLAB code (assuming μ and x_{cor} are defined):

```
D = sparse(999,1000);
D(:,1:999) =  -speye(999);
D(:,2:1000) =  D(:,2:1000) + speye(999);
xhat = (speye(1000) + mu*D'*D) \ xcor;
```

Solve the least squares problem (18) with the vector x_{cor} defined in `lsdenoising.m`, for three values of μ: $\mu = 1$, $\mu = 100$, and $\mu = 10000$. Plot the three reconstructed signals $\hat{x}$. Discuss the effect of μ on the quality of the estimate $\hat{x}$.

9.4 *Regularized least squares image deblurring.* This exercise is on the image deblurring problem on page 202 of the textbook. The purpose is to develop a fast method for solving the regularized least-squares problem on page 202:

$$\text{minimize } \|Ax - y\|^2 + \lambda(\|D_{\mathrm{v}}x\|^2 + \|D_{\mathrm{h}}x\|^2). \tag{20}$$

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

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

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

In (20), the vectors x and y have length n^2. The vector y is given and represents an observed, noisy and blurred image Y of size $n \times n$. The variable x is the reconstructed $n \times n$ image X in vector form.

The following notation will be used to express the matrices A, D_{v}, D_{h} in (20). As in exercise 6.7, we use $T(v)$, with v an n-vector, to denote the $n \times n$ circulant matrix with v as its first column:

$$T(v) = \begin{bmatrix} v_1 & v_n & v_{n-1} & \cdots & v_3 & v_2 \\ v_2 & v_1 & v_n & \cdots & v_4 & v_3 \\ v_3 & v_2 & v_1 & \cdots & v_5 & v_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ v_{n-1} & v_{n-2} & v_{n-3} & \cdots & v_1 & v_n \\ v_n & v_{n-1} & v_{n-2} & \cdots & v_2 & v_1 \end{bmatrix}.$$

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

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

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

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

$$Y_{ij} = \sum_{k=-r}^{r} \sum_{l=-r}^{r} P_{kl} X_{i-k,j-l}, \quad i = 1, \ldots, n, \; j = 1, \ldots, n. \tag{21}$$

Note that the sum in this definition references some values of $X_{i-k,j-l}$ with indices outside the interval $[1, n]$. To fully specify the blurring operation we therefore need to make assumptions about the image X outside the frame. These assumptions are called *boundary conditions*. In this exercise we will use *periodic* boundary conditions. Periodic boundary conditions assume that the image X is repeated periodically outside the frame. One way to write this is to replace the convolution (21) with

$$Y_{ij} = \sum_{k=-r}^{r} \sum_{l=-r}^{r} P_{kl} \bar{X}_{i-k,j-l}, \quad i = 1, \ldots, n, \quad j = 1 \ldots, n, \tag{22}$$

where $\bar{X}$ denotes the larger, bordered image

$$\bar{X} = \begin{array}{|c|c|c|} \hline X & X & X \\ \hline X & X & X \\ \hline X & X & X \\ \hline \end{array}$$

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

$$\left[\begin{array}{c} Y_{11} \\ Y_{21} \\ Y_{31} \\ \hline Y_{12} \\ Y_{22} \\ Y_{32} \\ \hline Y_{13} \\ Y_{23} \\ Y_{33} \end{array}\right] = \left[\begin{array}{ccc|ccc|ccc} P_{00} & P_{-1,0} & P_{1,0} & P_{0,-1} & P_{-1,-1} & P_{1,-1} & P_{01} & P_{-1,1} & P_{11} \\ P_{10} & P_{00} & P_{-1,0} & P_{1,-1} & P_{0,-1} & P_{-1,-1} & P_{11} & P_{01} & P_{-1,1} \\ P_{-1,0} & P_{10} & P_{00} & P_{-1,-1} & P_{1,-1} & P_{0,-1} & P_{-1,1} & P_{11} & P_{01} \\ \hline P_{01} & P_{-1,1} & P_{11} & P_{00} & P_{-10} & P_{10} & P_{0,-1} & P_{-1,-1} & P_{1,-1} \\ P_{11} & P_{01} & P_{-1,1} & P_{10} & P_{00} & P_{-1,0} & P_{1,-1} & P_{0,-1} & P_{-1,-1} \\ P_{-1,1} & P_{11} & P_{01} & P_{-1,0} & P_{10} & P_{00} & P_{-1,-1} & P_{1,-1} & P_{0,-1} \\ \hline P_{0,-1} & P_{-1,-1} & P_{1,-1} & P_{01} & P_{-1,1} & P_{11} & P_{00} & P_{-1,0} & P_{10} \\ P_{1,-1} & P_{0,-1} & P_{-1,-1} & P_{11} & P_{01} & P_{-1,1} & P_{10} & P_{00} & P_{-1,0} \\ P_{-1,-1} & P_{1,-1} & P_{0,-1} & P_{-1,1} & P_{11} & P_{01} & P_{-1,0} & P_{10} & P_{00} \end{array}\right] \left[\begin{array}{c} X_{11} \\ X_{21} \\ X_{31} \\ \hline X_{12} \\ X_{22} \\ X_{32} \\ \hline X_{13} \\ X_{23} \\ X_{33} \end{array}\right].$$

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

We note that the matrix on the right-hand side is a BCCB matrix. More generally, if we write the convolution with periodic boundary conditions (22) in matrix form as $y = Ax$, then A is the $n^2 \times n^2$ matrix $A = T(B)$,

where B is the following $n \times n$ matrix constructed from the PSF coefficients.

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

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

Vertical and horizontal differencing. Next we discuss the matrices D_{v} and D_{h} in (20). The vector $D_{\mathrm{v}}x$ is the image obtained by subtracting from X the image X shifted up over one pixel, so that

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

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

$$D_{\mathrm{v}} = \begin{bmatrix} D & 0 & \cdots & 0 \\ 0 & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D \end{bmatrix},$$

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

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

Note that D_{v} is a BCCB matrix: $D_{\mathrm{v}} = T(E)$ where E is the $n \times n$ matrix

$$E = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \\ -1 & 0 & \cdots & 0 \end{bmatrix}. \tag{23}$$

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

The last term $\|D_{\rm h}x\|^2$ in (20) is a penalty on the horizontal differences in the image: $D_{\rm h}$ is defined by

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

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

$$D_{\rm h} = \begin{bmatrix} I & -I & 0 & \cdots & 0 & 0 & 0 \\ 0 & I & -I & \cdots & 0 & 0 & 0 \\ 0 & 0 & I & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & I & -I & 0 \\ 0 & 0 & 0 & \cdots & 0 & I & -I \\ -I & 0 & 0 & \cdots & 0 & 0 & I \end{bmatrix}.$$

This is a BCCB matrix: $D_{\rm h} = T(E^T)$ with E the matrix defined in (23).

Two-dimensional discrete Fourier transform. The two-dimensional DFT of a complex $n \times n$ matrix U is the complex $n \times n$ matrix

$$V = WUW, \tag{24}$$

where W is the DFT matrix (7). The inverse two-dimensional DFT maps an $n \times n$-matrix V to the matrix

$$U = \frac{1}{n^2}W^H V W^H. \tag{25}$$

In MATLAB the two-dimensional DFT and its inverse are implemented in the functions `fft2` and `ifft2`. We use `V = fft2(U)` to evaluate (24) and `U = ifft2(V)` to evaluate (25). The complexity is roughly $n^2 \log n$.

The two-dimensional DFT can also be interpreted as a matrix-vector product. Suppose u and v are the matrices U and V converted to n^2-vectors (in column-major order). Then the relation $V = WUW$ can be written as

$$v = \widetilde{W}u, \qquad \widetilde{W} = \begin{bmatrix} W_{11}W & W_{12}W & \cdots & W_{1n}W \\ W_{21}W & W_{22}W & \cdots & W_{2n}W \\ \vdots & \vdots & & \vdots \\ W_{n1}W & W_{n2}W & \cdots & W_{nn}W \end{bmatrix}. \tag{26}$$

The matrix $\widetilde{W}$ is $n^2 \times n^2$ and has n block rows and columns. The i,j block of $\widetilde{W}$ is $W_{ij}W$ where W_{ij} is the i,j element of W. (The operation that constructs $\widetilde{W}$ from W is known as a *Kronecker product* and written $\widetilde{W} = W \otimes W$.) Using (26) and the property $W^H W = (1/n)I$ one can show that

$$\widetilde{W}^H\widetilde{W} = n^2 I.$$

Therefore $\widetilde{W}^{-1} = (1/n^2)\widetilde{W}^H$ and $\widetilde{W}\widetilde{W}^H = n^2 I$.

It is important to keep in mind that we never need the matrix $\widetilde{W}$ explicitly, since we use the `fft2` and `ifft2` functions for the matrix-vector multiplications with $\widetilde{W}$. The product $v = \widetilde{W}u$ can be evaluated by combining `fft2` and the `reshape` function:

```
v = reshape( fft2( reshape( u, n, n ) ), n^2, 1);
```

Similarly, the matrix-vector product $u = \widetilde{W}^{-1}v = (1/n^2)\widetilde{W}^H v$, can be computed as

```
u = reshape( ifft2( reshape( v, n, n ) ), n^2, 1);
```

Finally, we need the two-dimensional counterpart of the factorization result for circulant matrices proved in exercise 6.7. The result is as follows. Suppose U is an $n \times n$ matrix and u is the corresponding n^2-vector (with the elements of U in column-major order). Then the $n^2 \times n^2$ BCCB matrix $T(U)$ can be factored as

$$T(U) = \frac{1}{n^2} \widetilde{W}^H \, \mathbf{diag}(\widetilde{W}u) \widetilde{W}. \tag{27}$$

Assignment.

(a) The normal equations for (20) are

$$(A^T A + \lambda D_{\mathrm{v}}^T D_{\mathrm{v}} + \lambda D_{\mathrm{h}}^T D_{\mathrm{h}})x = A^T y.$$

Assume A, D_{v}, D_{h} are BCCB matrices

$$A = T(B), \qquad D_{\mathrm{v}} = T(E), \qquad D_{\mathrm{h}} = T(E^T),$$

where B is a given $n \times n$ matrix and E is the $n \times n$ matrix defined in (23). Use the factorization (27) to derive a fast algorithm (with order $n^2 \log n$ complexity) for solving the normal equations. Implement the fast algorithm in MATLAB or Octave.

(b) Download the file `deblur.mat` on the course website and load it in MATLAB or Octave (`load deblur`). The file contains two matrices Y and B. The matrix Y is the blurred image and can be displayed using the command `imshow(Y)`. The matrix B defines the blurring matrix $A = T(B)$. (The image is from the USC-SIPI Image Database at `http://sipi.usc.edu/database`.)

Test your deblurring code with several values of λ, plot the reconstructed image for each value (using `imshow(X)`), and determine the value of λ that gives the best result (visually, in your judgment). It is best to search for a good value of λ over a large range, by using a series of values evenly spaced on a logarithmic scale, for example, $\lambda = 10^{-6}, 10^{-5}, 10^{-4}, \ldots$.

9.5 *Least squares fit of piecewise-polynomial function.* This exercise is on the piecewise-polynomial fitting problem of page 214 of the textbook. Download the file `splinefit.m` and run it in MATLAB/Octave to create two arrays `t` and `y` of length 100. The vector t contains 100 points in the interval $[-1, 1]$. The first 50 points are in $[-1, 0]$, the last 50 in $[0, 1]$. You are asked to find two cubic polynomials

$$p(x) = \theta_1 + \theta_2 x + \theta_3 x^2 + \theta_4 x^3, \qquad q(x) = \theta_5 + \theta_6 x + \theta_7 x^2 + \theta_8 x^3,$$

that give the best least squares fit on the intervals $[-1, 0]$ and $[0, 1]$, respectively, and satisfy the continuity constraints $p(0) = q(0)$, $p'(0) = q'(0)$:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{50} (p(t_i) - y_i)^2 + \sum_{i=51}^{100} (q(t_i) - y_i)^2 \\ \text{subject to} & p(0) = q(0), \quad p'(0) = q'(0). \end{array}$$

The variables are the coefficients $\theta_1, \ldots, \theta_8$. Formulate this as a constrained least squares problem

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

and compute the solution by solving the optimality conditions

$$\begin{bmatrix} A^T A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ z \end{bmatrix} = \begin{bmatrix} A^T b \\ d \end{bmatrix}.$$

In MATLAB/Octave:

```
sol = [ A'*A, C'; C, zeros(p,p) ] \ [ A'*b; d ];
x = sol(1:n);
```

10 Constrained least squares

10.1 *Minimum-energy optimal control.* A simple model of a vehicle moving in one dimension is given by

$$\begin{bmatrix} s_1(t+1) \\ s_2(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0.95 \end{bmatrix} \begin{bmatrix} s_1(t) \\ s_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u(t), \qquad t = 0, 1, 2, \ldots .$$

$s_1(t)$ is the position at time t, $s_2(t)$ is the velocity at time t, and $u(t)$ is the actuator input. Roughly speaking, the equations state that the actuator input affects the velocity, which in turn affects the position. The coefficient 0.95 means that the velocity decays by 5% in one sample period (for example, because of friction), if no actuator signal is applied. We assume that the vehicle is initially at rest at position 0: $s_1(0) = s_2(0) = 0$.

We will solve the *minimum energy optimal control problem:* for a given time horizon N, choose inputs $u(0), \ldots, u(N-1)$ so as to minimize the total energy consumed, which we assume is given by

$$E = \sum_{t=0}^{N-1} u(t)^2.$$

In addition, the input sequence must satisfy the constraint $s_1(N) = 10$, $s_2(N) = 0$. Our task therefore is to bring the vehicle to the final position $s_1(N) = 10$ with final velocity $s_2(N) = 0$, as efficiently as possible.

(a) Formulate the minimum energy optimal control problem as a least-norm problem

$$\begin{array}{ll} \text{minimize} & \|x\|^2 \\ \text{subject to} & Cx = d. \end{array}$$

Clearly state what the variables x, and the problem data C and d are.

(b) Solve the problem for $N = 30$. Plot the optimal $u(t)$, the resulting position $s_1(t)$, and velocity $s_2(t)$.

(c) Solve the problem for $N = 2, 3, \ldots, 29$. For each N calculate the energy E consumed by the optimal input sequence. Plot E versus N. (The plot looks best if you use a logarithmic scale for E, *i.e.*, `semilogy` instead of `plot`.)

(d) Suppose we allow the final position to deviate from 10. However, if $s_1(N) \neq 10$, we have to pay a penalty, equal to $(s_1(N) - 10)^2$. The problem is to find the input sequence that minimizes the sum of the energy E consumed by the input and the terminal position penalty,

$$\sum_{t=0}^{N-1} u(t)^2 + (s_1(N) - 10)^2,$$

subject to the constraint $s_2(N) = 0$.

Formulate this problem as a least-norm problem, and solve it for $N = 30$. Plot the optimal input signals $u(t)$, the resulting position $s_1(t)$ and the resulting velocity $s_2(t)$.

Remark. If C has linearly independent rows, then the MATLAB command `x = C \ d` computes a solution to $Cx = d$, but it is *not* the least-norm solution. We can use the command `x = C' * ((C*C') \ d)` to compute the least-norm solution. We can also use the QR factorization method, using the code

```
[Q, R] = qr(C', 0);
x = Q * (R' \ d);
```

10.2 Two vehicles are moving along a straight line. For the first vehicle we use the same model as in exercise 10.1:

$$\begin{bmatrix} s_1(t+1) \\ s_2(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0.95 \end{bmatrix} \begin{bmatrix} s_1(t) \\ s_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u(t), \qquad t = 0, 1, 2, \ldots ,$$

$s_1(t)$ is the position at time t, $s_2(t)$ is the velocity at time t, and $u(t)$ is the actuator input. We assume that the vehicle is initially at rest at position 0: $s_1(0) = s_2(0) = 0$.

The model for the second vehicle is

$$\begin{bmatrix} p_1(t+1) \\ p_2(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0.8 \end{bmatrix} \begin{bmatrix} p_1(t) \\ p_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.2 \end{bmatrix} v(t), \qquad t = 0, 1, 2, \ldots,$$

$p_1(t)$ is the position at time t, $p_2(t)$ is the velocity at time t, and $v(t)$ is the actuator input. We assume that the second vehicle is initially at rest at position 1: $p_1(0) = 1$, $p_2(0) = 0$.

Formulate the following problem as a least-norm problem, and solve it in MATLAB (see the remark at the end of exercise 10.1). Find the control inputs $u(0)$, $u(1)$, ..., $u(19)$ and $v(0)$, $v(1)$, ..., $v(19)$ that minimize the total energy

$$\sum_{t=0}^{19} u(t)^2 + \sum_{t=0}^{19} v(t)^2$$

and satisfy the following three conditions:

$$s_1(20) = p_1(20), \qquad s_2(20) = 0, \qquad p_2(20) = 0. \tag{28}$$

In other words, at time $t = 20$ the two vehicles must have velocity zero, and be at the same position. (The final position itself is not specified, *i.e.*, you are free to choose any value as long as $s_1(20) = p_1(20)$.)

Plot the positions $s_1(t)$ and $p_1(t)$ of the two vehicles, for $t = 1, 2, \ldots, 20$.

10.3 Explain how you would solve the following problems using the QR factorization.

(a) Find the solution of $Cx = d$ with the smallest value of $\sum_{i=1}^n w_i x_i^2$:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n w_i x_i^2 \\ \text{subject to} & Cx = d. \end{array}$$

The problem data are the $p \times n$ matrix C, the p-vector d, and the n vector w. We assume that A has linearly independent rows, and $w_i > 0$ for all i.

(b) Find the solution of $Cx = d$ with the smallest value of $\|x\|^2 - c^T x$:

$$\begin{array}{ll} \text{minimize} & \|x\|^2 - c^T x \\ \text{subject to} & Cx = d. \end{array}$$

The problem data are the n-vector c, the $p \times n$ matrix C, and the p-vector d. We assume that C has linearly independent rows.

10.4 Show how to solve the following problems using the QR factorization of A. In each problem A is an $m \times n$ matrix with linearly independent columns. Clearly state the different steps in your method. Also give a flop count, including all the terms that are quadratic (order m^2, mn, or n^2), or cubic (order m^3, m^2n, mn^2, n^3).

If you know several methods, give the most efficient one.

(a) Solve the set of linear equations

$$\begin{bmatrix} 0 & A^T \\ A & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}.$$

The variables are the n-vector x and the m-vector y.

(b) Solve the least squares problem

$$\text{minimize} \quad 2\|Ax - b\|^2 + 3\|Ax - c\|^2.$$

The variable is the n-vector x.

(c) Solve the least-norm problem

$$\begin{array}{ll} \text{minimize} & \|x\|^2 + \|y\|^2 \\ \text{subject to} & A^T x - 2A^T y = b. \end{array}$$

The variables are the m-vectors x and y.

(d) Solve the quadratic minimization problem

$$\text{minimize} \quad x^T A^T A x + b^T x + c.$$

The variable is the n-vector x.

10.5 If A is an $m \times n$ matrix with linearly independent columns, and D is an $m \times m$ diagonal matrix with positive diagonal elements, then the coefficient matrix of the equation

$$\begin{bmatrix} D^2 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}$$

is nonsingular. Therefore the equation has a unique solution $\hat{x}$, $\hat{y}$.

(a) Show that $\hat{x}$ is the solution of the optimization problem

$$\begin{array}{ll} \text{minimize} & \|Dx - D^{-1}b\|^2 \\ \text{subject to} & A^T x = c. \end{array}$$

(b) Show that $\hat{y}$ is the solution of the optimization problem

$$\text{minimize} \quad \|D^{-1}(Ay - b)\|^2 + 2c^T y.$$

(Hint: set the gradient of the cost function to zero.)

(c) Describe an efficient method, based on the QR factorization of $D^{-1}A$, for computing $\hat{x}$ and $\hat{y}$. Clearly state the different steps in your algorithm, the complexity of each step (number of flops for large m, n), and the total complexity.

10.6 Let A be an $m \times n$ matrix, b an n-vector, and suppose the QR factorization

$$\begin{bmatrix} A^T & b \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}$$

exists. The matrix Q_1 has size $n \times m$, Q_2 has size $n \times 1$, R_{11} has size $m \times m$, R_{12} is $m \times 1$, and R_{22} is a scalar. Show that $\hat{x} = Q_2 R_{22}$ solves the optimization problem

$$\begin{array}{ll} \text{minimize} & \|x - b\|^2 \\ \text{subject to} & Ax = 0. \end{array}$$

10.7 Suppose A is an $m \times n$ matrix with linearly independent columns. Let $\hat{x}$ be the solution of the optimization problem

$$\text{minimize} \quad \|Ax - b\|^2 + 2c^T x$$

with $b \in \mathbf{R}^m$ and $c \in \mathbf{R}^n$, and let $\hat{y}$ be the solution of

$$\begin{array}{ll} \text{minimize} & \|y - b\|^2 \\ \text{subject to} & A^T y = c. \end{array}$$

(a) Show that $\hat{y} = b - A\hat{x}$. (*Hint.* To find an expression for $\hat{x}$, set the gradient of $\|Ax - b\|^2 + 2c^T x$ equal to zero.)

(b) Describe an efficient method for calculating $\hat{x}$ and $\hat{y}$ using the QR factorization of A. Clearly state the different steps in your algorithm and give a flop count, including all terms that are quadratic (m^2, mn, n^2) or cubic (m^3, m^2n, mn^2, n^3) in m and n.

10.8 Consider the underdetermined set of linear equations

$$Ax + By = b$$

where the p-vector b, the $p \times p$ matrix A, and the $p \times q$ matrix B are given. The variables are the p-vector x and the q-vector y. We assume that A is nonsingular, and that B has linearly independent columns (which implies $q \leq p$). The equations are underdetermined, so there are infinitely many solutions. For example, we can pick any y, and solve the set of linear equations $Ax = b - By$ to find x.

Below we define four solutions that minimize some measure of the magnitude of x, or y, or both. For each of these solutions, describe a method for computing x and y using a QR or LU factorization. Clearly specify the matrices that you factor, and the type of factorization. If you know several methods, give the most efficient one.

(a) The solution x, y with the smallest value of $\|x\|^2 + \|y\|^2$

(b) The solution x, y with the smallest value of $\|x\|^2 + 2\|y\|^2$.

(c) The solution x, y with the smallest value of $\|y\|^2$.

(d) The solution x, y with the smallest value of $\|x\|^2$.

10.9 Let A be a real $m \times n$ matrix with linearly independent columns, and let b be a real m-vector. We consider two least-squares problems. The first problem is the standard

$$\text{minimize} \quad \|Ax - b\|^2. \tag{29}$$

In the second problem we remove column i of A or, equivalently, set $x_i = 0$:

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|^2 \\ \text{subject to} & e_i^T x = 0. \end{array} \tag{30}$$

Here e_i denotes the ith unit vector of length n (an n-vector with all its elements zero, except the ith element, which is one).

(a) Let $\hat{x}$ be the solution of (29). Show that the solution of (30) is

$$x = \hat{x} - \frac{\hat{x}_i}{((A^TA)^{-1})_{ii}}(A^TA)^{-1}e_i. \tag{31}$$

The denominator in the second term is the ith diagonal element of the inverse $(A^TA)^{-1}$.

(b) Describe an efficient algorithm, based on the QR factorization of A, to calculate $\hat{x}$ and the vector x in (31). Carefully state the different steps in your algorithm, and give the complexity of each step (number of flops for large m, n).

10.10 Let A be a positive definite $n \times n$ matrix, and c a nonzero n-vector.

(a) Show that the solution of the optimization problem

$$\begin{array}{ll} \text{minimize} & x^TAx \\ \text{subject to} & c^Tx = 1 \end{array}$$

is given by

$$x = \frac{1}{c^TA^{-1}c}A^{-1}c.$$

Hint. By writing $x^TAx = x^TR^TRx = \|Rx\|^2$, where R is the Cholesky factor of A, we can interpret the problem as a constrained least squares problem.

(b) Suppose $\lambda > 0$. Show that the solution of the optimization problem

$$\text{minimize} \quad x^T A x + \lambda (c^T x - 1)^2$$

is given by

$$x = \frac{\lambda}{1 + \lambda c^T A^{-1} c} A^{-1} c.$$

11 Cholesky factorization

11.1 Are the following matrices positive definite?

(a) $A = \begin{bmatrix} -1 & 2 & 3 \\ 2 & 5 & -3 \\ 3 & -3 & 2 \end{bmatrix}$.

(b) $A = I - uu^T$ where u is an n-vector with $\|u\| < 1$.

(c) $A = \begin{bmatrix} I & B \\ B^T & I + B^T B \end{bmatrix}$ where B is an $m \times n$-matrix.

(d) $A = \begin{bmatrix} 1 & u^T \\ u & I \end{bmatrix}$ where u is an n-vector with $\|u\| < 1$.

11.2 Show that the inverse of a positive definite matrix is positive definite.

11.3 Suppose the matrix

$$\begin{bmatrix} 0 & A \\ A^T & B \end{bmatrix}$$

is positive semidefinite. Show that $A = 0$ and B is positive semidefinite.

11.4 A square matrix P is called a *symmetric projection matrix* if $P = P^T$ and $P^2 = P$. Show that a symmetric projection matrix P satisfies the following properties.

(a) $I - P$ is also a symmetric projection matrix.

(b) $\|x\|^2 = \|Px\|^2 + \|(I - P)x\|^2$ for all x.

(c) P is positive semidefinite.

Carefully explain each step in your arguments.

11.5 Let a be a nonzero n-vector with $n \geq 2$. We define two $n \times n$-matrices

$$A = \frac{1}{\|a\|^2} aa^T, \qquad B = I - \frac{1}{\|a\|^2} aa^T.$$

The mapping $f(x) = Ax$ is the orthogonal projection of x on the line through a. The mapping $g(x) = Bx = x - Ax$ is the difference between x and its projection on the line through a.

(a) Are A and B positive semidefinite?

(b) Are A and B positive definite?

Explain your answers.

11.6 The elementwise product $C = A \circ B$ of two $n \times n$ matrices A, B is defined as the $n \times n$ matrix C with elements $C_{ij} = A_{ij}B_{ij}$. (In MATLAB notation: `C = A .* B` .)

(a) Suppose A is $n \times n$ and d is an n-vector. Verify that

$$A \circ (dd^T) = \mathbf{diag}(d) A \, \mathbf{diag}(d)$$

where $\mathbf{diag}(d)$ is the diagonal matrix with the vector d on its diagonal. Use this observation to show that the matrix $A \circ (dd^T)$ is positive semidefinite if A is positive semidefinite.

(b) Suppose A is $n \times n$ and positive semidefinite, and D is an $n \times m$ matrix. Show that $A \circ (DD^T)$ is positive semidefinite.
(*Hint.* Write DD^T as $DD^T = \sum_{k=1}^m d_k d_k^T$ where d_k is the kth column of D.)

(c) Suppose A is $n \times n$ and positive definite, and D is an $n \times m$ matrix with at least one nonzero element in every row. Show that $A \circ (DD^T)$ is positive definite.

11.7 Compute the Cholesky factorization of

$$A = \begin{bmatrix} 4 & 6 & 2 & -6 \\ 6 & 34 & 3 & -9 \\ 2 & 3 & 2 & -1 \\ -6 & -9 & -1 & 38 \end{bmatrix}$$

You can use MATLAB to verify the result, but you have to provide the details of your calculations.

11.8 For what values of the scalar a are the following matrices positive definite? To derive the conditions, factor A using a Cholesky factorization and collect the conditions on a needed for the factorization to exist.

(a) $A = \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}$.

(b) $A = \begin{bmatrix} 1 & a & 0 \\ a & 1 & a \\ 0 & a & 1 \end{bmatrix}$.

(c) $A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & a \end{bmatrix}$.

(d) $A = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & 0 \\ a & 0 & 1 \end{bmatrix}$.

(e) $A = \begin{bmatrix} a & 1 & 0 \\ 1 & -a & 1 \\ 0 & 1 & a \end{bmatrix}$.

(f) $A = \begin{bmatrix} I & aI \\ aI & I \end{bmatrix}$. I is the $n \times n$ identity matrix.

(g) $A = \begin{bmatrix} I & au \\ au^T & 1 \end{bmatrix}$. I is the $n \times n$ identity matrix and $u = (1, 1, \ldots, 1)$, the n-vector with all its elements equal to one.

(h) $A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & a & a \\ 1 & a & 2 \end{bmatrix}$.

11.9 Suppose A is an $n \times n$ positive definite matrix. For what values of the scalar β is the matrix

$$\begin{bmatrix} A & -A \\ -A & \beta A \end{bmatrix}$$

positive definite?

11.10 Let A be a positive definite matrix of size $n \times n$. For what values of the scalar a are the following matrices positive definite?

$$\text{(a)} \quad \begin{bmatrix} A & ae_1 \\ ae_1^T & 1 \end{bmatrix} \qquad \text{(b)} \quad \begin{bmatrix} A & e_1 \\ e_1^T & a \end{bmatrix} \qquad \text{(c)} \quad \begin{bmatrix} A & ae_1 \\ ae_1^T & a \end{bmatrix}.$$

($e_1 = (1, 0, \ldots, 0)$ denotes the first unit vector of length n.) Give your answer for each of the three problems in the form of upper and/or lower bounds ($a_{\min} < a < a_{\max}$, $a > a_{\min}$, or $a < a_{\max}$). Explain how you can compute the limits $a_{\min}$ and $a_{\max}$ using the Cholesky factorization $A = R^T R$. (You don't need to discuss the complexity of the computation.)

11.11 Suppose u is a nonzero n-vector. For what values of the scalar a are the following matrices positive definite? Express your answer as upper and/or lower bounds on a that depend on the elements of u.

(a) The matrix $aI - uu^T$.

(b) The matrix $aI + uu^T$.

(c) The matrix

$$\begin{bmatrix} \mathbf{diag}(u)^2 & \mathbf{1} \\ \mathbf{1}^T & a \end{bmatrix} = \begin{bmatrix} u_1^2 & 0 & \cdots & 0 & 1 \\ 0 & u_2^2 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & u_n^2 & 1 \\ 1 & 1 & \cdots & 1 & a \end{bmatrix}.$$

(Here we assume that u has nonzero elements.)

11.12 Let A be defined as

$$A = I + BB^T$$

where B is a given $n \times m$ matrix with orthonormal columns (not necessarily square).

(a) Show that A is positive definite.

(b) What is the cost (number of flops for large m and n) of solving $Ax = b$ by first computing $A = I + BB^T$ and then applying the standard method for linear equations $Ax = b$ with positive definite A?

(c) Show that $A^{-1} = I - (1/2)BB^T$.

(d) Use the expression in part (c) to derive an efficient method for solving $Ax = b$ (*i.e.*, a method that is much more efficient than the method in part (b).) Give the cost of your method (number of flops for large m and n).

11.13 You are given the Cholesky factorization $A = R^T R$ of a positive definite matrix A of size $n \times n$, and an n-vector u.

(a) What is the Cholesky factorization of the $(n+1) \times (n+1)$-matrix

$$B = \begin{bmatrix} A & u \\ u^T & 1 \end{bmatrix}?$$

You can assume that B is positive definite.

(b) What is the cost of computing the Cholesky factorization of B, if the factorization of A (*i.e.*, the matrix R) is given?

11.14 A matrix A is *tridiagonal* if $A_{ij} = 0$ for $|i - j| > 1$, *i.e.*, A has the form

$$A = \begin{bmatrix} A_{11} & A_{12} & 0 & \cdots & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & \cdots & 0 & 0 & 0 \\ 0 & A_{32} & A_{33} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{n-2,n-2} & A_{n-2,n-1} & 0 \\ 0 & 0 & 0 & \cdots & A_{n-1,n-2} & A_{n-1,n-1} & A_{n-1,n} \\ 0 & 0 & 0 & \cdots & 0 & A_{n,n-1} & A_{nn} \end{bmatrix}.$$

What is the cost of computing the Cholesky factorization of a tridiagonal positive definite matrix of size $n \times n$? Count square roots as one flop and keep only the leading term in the total number of flops. Explain your answer.

11.15 Let A be a positive definite 5×5-matrix with the nonzero pattern

$$A = \begin{bmatrix} \bullet & & \bullet & \bullet & \\ & \bullet & & \bullet & \\ \bullet & & \bullet & & \bullet \\ \bullet & \bullet & & \bullet & \\ & & \bullet & & \bullet \end{bmatrix}.$$

The dots indicate the positions of the nonzero elements; all the other elements are zero.

The Cholesky factor R of A has one of the following upper-triangular nonzero patterns. Which one is correct? For each R, explain why R is or is not the Cholesky factor of A.

(a) $R =$
$$\begin{bmatrix} \bullet & & \bullet & & \bullet \\ & \bullet & & \bullet & \\ & & \bullet & & \bullet \\ & & & \bullet & \\ & & & & \bullet \end{bmatrix}$$

(b) $R =$
$$\begin{bmatrix} \bullet & & \bullet & \bullet & \\ & \bullet & \bullet & \bullet & \\ & & \bullet & & \bullet \\ & & & \bullet & \\ & & & & \bullet \end{bmatrix}$$

(c) $R =$
$$\begin{bmatrix} \bullet & & \bullet & \bullet & \\ & \bullet & & \bullet & \bullet \\ & & \bullet & \bullet & \\ & & & \bullet & \bullet \\ & & & & \bullet \end{bmatrix}$$

(d) $R =$
$$\begin{bmatrix} \bullet & & \bullet & \bullet & \\ & \bullet & & \bullet & \\ & & \bullet & \bullet & \bullet \\ & & & \bullet & \bullet \\ & & & & \bullet \end{bmatrix}$$

11.16 Define a block matrix

$$K = \begin{bmatrix} A & B \\ B^T & -C \end{bmatrix},$$

where the three matrices A, B, C have size $n \times n$. The matrices A and C are symmetric and positive definite. Show that K can be factored as

$$K = \begin{bmatrix} R_{11}^T & 0 \\ R_{12}^T & R_{22}^T \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix},$$

where R_{11} and R_{22} are upper triangular matrices with positive diagonal elements. The blocks R_{11}, R_{12}, R_{22}, and the two identity matrices on the right-hand side, all have size $n \times n$.

What is the cost of computing this factorization (number of flops for large n)? Carefully explain your answers.

11.17 Let u be a nonzero n-vector.

(a) Show that $I + auu^T$ is positive definite if $a > -1/\|u\|^2$.

(b) Suppose $a > -1/\|u\|^2$. It can be shown that the triangular factor in the Cholesky factorization $I + auu^T = R^T R$ can be written as

$$R = \begin{bmatrix} \beta_1 & \gamma_1 u_1 & \gamma_1 u_3 & \cdots & \gamma_1 u_{n-1} & \gamma_1 u_n \\ 0 & \beta_2 & \gamma_2 u_3 & \cdots & \gamma_2 u_{n-1} & \gamma_2 u_n \\ 0 & 0 & \beta_3 & \cdots & \gamma_3 u_{n-1} & \gamma_3 u_n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \beta_{n-1} & \gamma_1 u_n \\ 0 & 0 & 0 & \cdots & & \beta_n \end{bmatrix}.$$

The entries to the right of the diagonal in row k are the entries $u_{k+1}, \ldots, u_n$ of the vector u, multiplied with γ_k.

Give an algorithm, with a complexity linear in n, for calculating the parameters $\beta_1, \ldots, \beta_n$ and $\gamma_1, \ldots, \gamma_{n-1}$, given the vector u and the scalar a.

11.18 Consider the symmetric $n \times n$ matrix

$$A = \begin{bmatrix} 1 & a & a^2 & \cdots & a^{n-2} & a^{n-1} \\ a & 1 & a & \cdots & a^{n-3} & a^{n-2} \\ a^2 & a & 1 & \cdots & a^{n-4} & a^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a^{n-2} & a^{n-3} & a^{n-4} & \cdots & 1 & a \\ a^{n-1} & a^{n-2} & a^{n-3} & \cdots & a & 1 \end{bmatrix}$$

where a is a real scalar. The i, j element is $A_{ij} = a^{|i-j|}$. The matrix A is a Toeplitz matrix, *i.e.*, constant along the diagonals. (Note, however, that A is not circulant; the problem does not require any of the properties of circulant matrices used in exercise 6.7).

We will use the following property (which you are not asked to show): if A is positive definite, then its Cholesky factor has the form

$$R = \begin{bmatrix} D_{11} & 0 & 0 & \cdots & 0 \\ 0 & D_{22} & 0 & \cdots & 0 \\ 0 & 0 & D_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D_{nn} \end{bmatrix} \begin{bmatrix} 1 & a & a^2 & \cdots & a^{n-1} \\ 0 & 1 & a & \cdots & a^{n-2} \\ 0 & 0 & 1 & \cdots & a^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{32}$$

i.e., the product of a diagonal matrix with the upper-triangular part of A.

(a) Suppose A is positive definite, so its Cholesky factor can be written as (32). Give an expression for the diagonal elements D_{ii}. (*Hint.* Consider the diagonal elements in the equality $A = R^T R$.)

(b) For what values of a is A positive definite?

(c) Give a simple explicit formula for the inverse of R in (32).

(d) Use part (c) to formulate a fast algorithm, with order n complexity, for solving $Ax = b$, when A is positive definite.

11.19 *Multi-class classification.* In this exercise we implement the handwritten digit classification method of the lecture on Cholesky factorization on a smaller data set (of 5000 examples) than used in the lecture. The data are available in the file `mnist.mat`. The file contains four variables: `Xtrain`, `Xtest`, `labels_train`, `labels_test`.

The variable `Xtrain` is a 5000×784 matrix X_{train} containing 5000 images. Each row is a 28×28 image stored as a vector of length 784. (To display the image in row i, use `imshow(reshape(Xtrain(i,:), 28, 28)')`.) The array `labels_train` is a vector of length 5000, with elements from $\{0, 1, \ldots, 9\}$. The ith element is the digit shown in row i of X_{train}. The 5000 images in X_{train} will be used to compute the classifier.

The matrix `Xtest` and vector `labels_test` are defined similarly, and give 5000 examples that will be used to test the classifier.

- *Binary classifiers.* The multi-class classification method is based on 10 binary classifiers. Each of the binary classifiers is designed to distinguish one of the digits versus the rest. We will use the polynomial kernel of degree 3, as in the lecture.

  To compute the classifier for digit k versus the rest, we solve a linear equation

  $$(Q + \lambda I)w = y,$$

  where Q is an $N \times N$ matrix ($N = 5000$) with elements

  $$Q_{ij} = (1 + x_i^T x_j)^3, \quad i, j = 1, \ldots, N,$$

and x_i^T is the ith row of the matrix X_{train}. The coefficient λ is a positive regularization parameter. The right-hand side y is a vector of length N, with $y_i = 1$ if the image in row i of X_{train} is an example of digit k, and $y_i = -1$ otherwise.

In MATLAB the coefficients w for all ten binary classifiers can be computed as follows.

```
load mnist;
[N, n] = size(Xtrain);
Y = -ones(N, 10);
for j = 1:10
    I = find(labels_train == j-1);
    Y(I, j) = 1;
end;
W = ( (1 + Xtrain * Xtrain').^3 + lambda * eye(N) ) \ Y;
```

Here we first create an $N \times 10$ matrix Y with $Y_{ij} = 1$ if image i is an example of digit $j-1$ and $Y_{ij} = -1$ otherwise. The computed matrix W has size $N \times 10$ and contains in its jth column the coefficients w of the classifier for digit $j-1$ versus the rest. The binary classifier for digit $j-1$ can be evaluated at a new image z by computing

$$\tilde{f}^{(j)}(z) = \sum_{i=1}^{N} W_{ij}(1 + z^T x_i)^3$$

and assigning z to class j if $\tilde{f}^{(j)}(x)$ is greater than or equal to zero.

- *Multi-class classifier.* We combine the ten binary classifiers into a multi-class classifier by taking the maximum of the ten functions $\tilde{f}^{(j)}(z)$:

$$\hat{f}(z) = \underset{j=1,\ldots,10}{\operatorname{argmax}} \tilde{f}^{(j)}(z) = \underset{j=1,\ldots,10}{\operatorname{argmax}} \left(\sum_{i=1}^{N} W_{ij}(1 + z^T x_i)^3 \right).$$

  In MATLAB, if z is an image stored as a column vector of length 784, the prediction $\hat{f}(z)$ can be computed as

  ```
  [val, prediction] = max(( 1 + z' * Xtrain' ).^3) * W);
  ```

  On the right-hand side we compute the maximum of a row vector of length 10. The first output argument on the left-hand side is the maximum value; the second output argument is the column index of the maximum (an integer between 1 and 10).

  The predictions for all examples in the training set can be computed using

  ```
  [val, prediction] = max(( 1 + Xtrain * Xtrain').^3) * W, [], 2);
  ```

  On the right-hand side we have a matrix of size 5000×10 as the first argument of `max`. The second and third arguments are needed to indicate that we are taking the maximum over the elements in each row. The first output argument on the left-hand side is a column vector of length N with the maximum value in each row; the second output argument is a column vector of length N with the column indices of the maximum elements in each row. This vector therefore contains the class predictions for the N rows of X_{train}. The command

  ```
  I = find( prediction - 1 ~= labels_train )
  ```

  returns the indices of the rows of X_{train} that are misclassified by the multi-class classifier.

  Similarly, we can compute the predictions for all examples in the test set using

  ```
  [val, prediction] = max(( 1 + Xtest * Xtrain').^3) * W, [], 2);
  ```

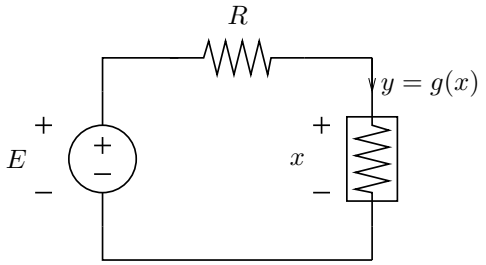
Implement this method for a range of regularization parameters $\lambda = 1, 10, \ldots, 10^7$. Plot the error rate for training set and test set as a function of λ. Choose the λ that (approximately) gives the smallest error on the test set, and give the confusion matrix of the classifier for this value of λ.

12 Nonlinear equations

12.1 The nonlinear resistive circuit shown below is described by the nonlinear equation

$$f(x) = g(x) - (E - x)/R = 0.$$

The function $g(x)$ gives the current through the nonlinear resistor as a function of the voltage x across its terminals.



Use Newton's method to find all the solutions of this nonlinear equation, assuming that

$$g(x) = x^3 - 6x^2 + 10x.$$

Consider three cases:

(a) $E = 5$, $R = 1$.

(b) $E = 15$, $R = 3$.

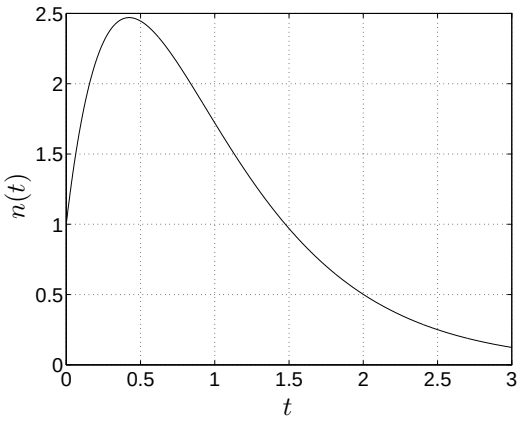
(c) $E = 4$, $R = 0.5$.

Select suitable starting points by plotting f over the interval $[0, 4]$, and visually selecting a good starting point. You can terminate the Newton iteration when $|f(x^{(k)})| < 10^{-8}$. Compare the speed of convergence for the three problems, by plotting $|f(x^{(k)})|$ versus k, or by printing out $|f(x^{(k)})|$ at each iteration.

12.2 The concentration of a certain chemical in a reaction is given by the following function of time:

$$n(t) = 10te^{-2t} + e^{-t},$$

which is shown below.



Use Newton's method to answer the following questions.

(a) Find the two values of t at which $n(t) = 2$.

(b) Find the value of t at which $n(t)$ reaches its maximum.

12.3 Find all values of x for which

$$\|(A + xI)^{-1}b\| = 1$$

where

$$A = \begin{bmatrix} -3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Use Newton's method, applied to the equation $g(x) = 1$, where

$$g(x) = \|(A + xI)^{-1}b\|^2 = \frac{1}{(-3+x)^2} + \frac{1}{(1+x)^2} + \frac{1}{(2+x)^2}.$$

First plot $g(x)$ versus x to determine the number of solutions and to select good starting points.

12.4 Use Newton's method to find all solutions of the two equations

$$(x_1 - 1)^2 + 2(x_2 - 1)^2 = 1, \qquad 3(x_1 + x_2 - 2)^2 + (x_1 - x_2 - 1)^2 = 2$$

in the two variables x_1, x_2. Each equation defines an ellipse in the (x_1, x_2)-plane. You are asked to find the points where the ellipses intersect.

12.5 Explain how you would solve the following problem using Newton's method. We are given three distinct points $a = (a_1, a_2)$, $b = (b_1, b_2)$, $c = (c_1, c_2)$ in a plane, and two positive numbers α and β. Find a point $x = (x_1, x_2)$ that satisfies

$$\|x - a\| = \alpha\|x - b\|, \qquad \|x - a\| = \beta\|x - c\|. \tag{33}$$

Clearly state the equations $f(x) = 0$ to which you apply Newton's method (these equations can be the two equations (33) or an equivalent set of equations), and the linear equations you solve at each iteration of the algorithm. You do not have to discuss the selection of the starting point, the stopping criterion, or the convergence of the method.

13 Unconstrained optimization

13.1 Derive the gradient and Hessian of the following functions g. Show that $\nabla^2 g(x)$ is positive definite.

(a) $g(x) = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2 + 1} = \sqrt{\|x\|^2 + 1}$.
Hint. Express $\nabla^2 g(x)$ as

$$\nabla^2 g(x) = \frac{1}{g(x)}(I - uu^T)$$

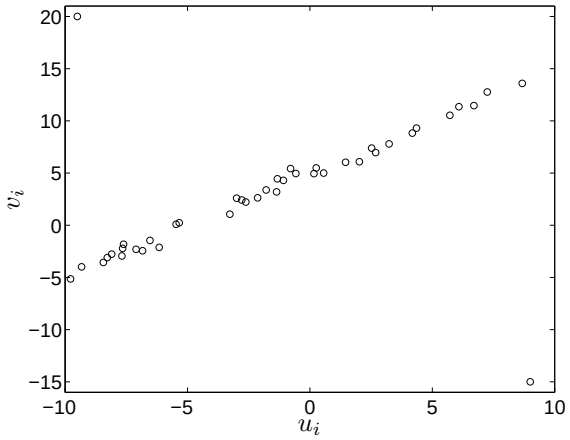
where u is an n-vector with norm less than one. Then prove that $I - uu^T$ is positive definite.

(b) $g(x) = \sqrt{\|Cx\|^2 + 1}$ where A is an $m \times n$ matrix with linearly independent columns.
Hint. Use the result of part (a) and the expression

$$\nabla^2 g(x) = C^T \nabla^2 h(Cx + d)C$$

for the Hessian of the function $g(x) = h(Cx + d)$.

13.2 The figure shows 42 data points (u_i, v_i). The points are approximately on a straight line, except for the first and the last point.



The data are available in the file `robappr.m` as `[u,v] = robappr;`.

(a) Fit a straight line $v = \alpha + \beta u$ to the points by solving a least squares problem

$$\text{minimize} \quad g(\alpha, \beta) = \sum_{i=1}^{42} (\alpha + \beta u_i - v_i)^2,$$

with variables α, β.

(b) Fit a straight line $v = \alpha + \beta u$ to the points by solving the unconstrained minimization problem

$$\text{minimize} \quad g(\alpha, \beta) = \sum_{i=1}^{42} \sqrt{(\alpha + \beta u_i - v_i)^2 + 25}$$

using Newton's method. Use as initial points the values of α and β computed in part (a). (With this starting point, no line search should be necessary; for other starting points, a line search might be needed.) Terminate the iteration when $\|\nabla g(\alpha, \beta)\| \leq 10^{-6}$.

(c) Fit a straight line $v = \alpha + \beta u$ to the points by solving the unconstrained minimization problem

$$\text{minimize} \quad g(\alpha, \beta) = \sum_{i=1}^{42} (\alpha + \beta u_i - v_i)^4$$

using Newton's method. Use as initial points the values of α and β computed in part (a). (With this starting point, no line search should be necessary.) Terminate the iteration when $\|\nabla g(\alpha, \beta)\| \leq 10^{-6}$.

(d) Plot the three functions $\alpha + \beta u$ computed in parts (a), (b) and (c) versus u, and compare the results.

13.3 Consider the problem of fitting a quadratic function

$$f(t) = \alpha + \beta t + \gamma t^2$$

to m given points (t_i, y_i), $i = 1, \ldots, m$. Suppose we choose to estimate the parameters α, β, γ by minimizing the function

$$g(\alpha, \beta, \gamma) = -\sum_{i=1}^{m} \log\left(1 - (\alpha + \beta t_i + \gamma t_i^2 - y_i)^2\right)$$

(where log denotes the natural logarithm).

(a) Give expressions for the gradient $\nabla g(\alpha, \beta, \gamma)$ and Hessian $\nabla^2 g(\alpha, \beta, \gamma)$ of g, at a point (α, β, γ) that satisfies

$$|\alpha + \beta t_i + \gamma t_i^2 - y_i| < 1, \quad i = 1, \ldots, m.$$

(This condition guarantees that g and its derivatives are defined at (α, β, γ).)

(b) Show that the Hessian is positive definite.

Assume that the values t_i are distinct and that $m \geq 3$.

13.4 For the data and with the same notation as in exercise 8.5, compute the coordinates (u_1, v_1), (u_2, v_2), (u_3, v_3) that minimize

$$l_1^4 + l_2^4 + l_3^4 + l_5^4 + l_6^4 + l_7^4.$$

Use Newton's method, with $u = v = 0$ as starting point. With this starting point no line search should be necessary.

13.5 Consider the problem of minimizing the function

$$\begin{aligned} g(x) &= \sum_{i=1}^{n} \exp(x_i + a_i) + \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 \\ &= \sum_{i=1}^{n} \exp(x_i + a_i) + (x_2 - x_1)^2 + (x_3 - x_2)^2 + \cdots + (x_n - x_{n-1})^2. \end{aligned}$$

The n-vector a is given.

(a) Give the gradient and Hessian of g. Show that the Hessian is positive definite everywhere.

(b) Describe an efficient method for computing the Newton step

$$v = -\nabla^2 g(x)^{-1} \nabla g(x).$$

What is the cost of your method (number of flops for large n)? It is sufficient to give the exponent of the dominant term in the flop count.

13.6 Define

$$g(x) = \|x - a\|^2 + \sum_{k=1}^{n-1} \sqrt{(x_{k+1} - x_k)^2 + \rho},$$

where a is a given n-vector and ρ is a given positive scalar. The variable is the n-vector x.

(a) Give expressions for the gradient and Hessian of g. Show that the Hessian is positive definite at all x.

(b) Describe an efficient method for computing the Newton step

$$v = -\nabla^2 g(x)^{-1} \nabla g(x).$$

Is the complexity of your method linear, quadratic, or cubic in n?

14 Nonlinear least squares

14.1 A cell phone at location (x_1, x_2) in a plane (we assume that the elevation is zero for simplicity) transmits an emergency signal at time x_3. This signal is received at m base stations, located at positions (p_1, q_1), (p_2, q_2), $\ldots$, (p_m, q_m). Each base station can measure the time of arrival of the emergency signal, within a few tens of nanoseconds. The measured times of arrival are

$$\tau_i = \frac{1}{c}\sqrt{(x_1 - p_i)^2 + (x_2 - q_i)^2} + x_3 + v_i, \quad i = 1, \ldots, m,$$

where c is the speed of light (0.3 meters/nanosecond), and v_i is the noise or error in the measured time of arrival. The problem is to estimate the cell phone position $x = (x_1, x_2)$, as well as the time of transmission x_3, based on the time of arrival measurements $\tau_1, \ldots, \tau_m$.

The m-file `e911.m`, available on the course web site, defines the data for this problem. It is executed as `[p,q,tau] = e911`. The 9×1-arrays `p` and `q` give the positions of the 9 base stations. The 9×1-array `tau` contains the measured times of arrival. Distances are given in meters and times in nanoseconds.

Determine an estimate $\hat{x}_1$, $\hat{x}_2$, $\hat{x}_3$ of the unknown coordinates and the time of transmission, by solving the nonlinear least squares problem

$$\text{minimize} \quad \sum_{i=1}^{9} r_i(x)^2$$

where

$$r_i(x_1, x_2, x_3) = \frac{1}{c}\sqrt{(x_1 - p_i)^2 + (x_2 - q_i)^2} + x_3 - \tau_i.$$

Use the Gauss-Newton method, with $x_1 = x_2 = x_3 = 0$ as starting point. From this starting point, the method should converge without backtracking, and you do not have to include backtracking in your code.

Your solution should include:

- a description of the least squares problems that you solve at each iteration
- the computed estimates $\hat{x}_1$, $\hat{x}_2$, $\hat{x}_3$
- a plot of $g(x^{(k)}) - g(\hat{x})$ versus k, where $g(x) = \sum_{i=1}^{9} r_i(x)^2$.

14.2 In exercise 8.3 we developed a simple approximate model for the inductance of a planar CMOS inductor. The model had the form

$$\hat{L} = \alpha n^{\beta_1} w^{\beta_2} d^{\beta_3} D^{\beta_4},$$

and depended on five model parameters: α, β_1, β_2, β_3, β_4. We calculated the five parameters by minimizing the error function

$$\sum_{i=1}^{50} (\log L_i - \log \hat{L}_i)^2,$$

which is a linear least squares problem in the variables

$$x_1 = \log \alpha, \qquad x_2 = \beta_1, \qquad x_3 = \beta_2, \qquad x_4 = \beta_3, \qquad x_5 = \beta_4.$$

In this problem you are asked to estimate the model parameters by minimizing the function

$$g(x) = \sum_{i=1}^{50} (L_i - \hat{L}_i)^2.$$

This is a nonlinear least squares problem

$$\text{minimize} \quad \sum_{i=1}^{50} r_i(x)^2$$

with x defined as before, and

$$r_i(x) = e^{x_1} n_i^{x_2} w_i^{x_3} d_i^{x_4} D_i^{x_5} - L_i.$$

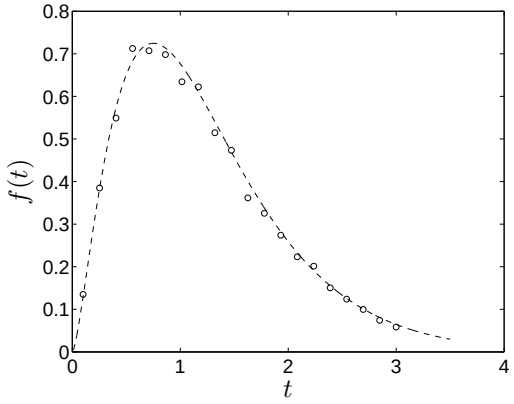
Solve this nonlinear least squares problem using the Gauss-Newton method with backtracking line search, and the data from `inductordata.m`.

Consider two starting points: (1) the answer of exercise 8.3, *i.e.*, the values of the model parameters calculated by solving the linear least squares problem; (2) $x = 0$. For both starting points, plot the step sizes $t^{(k)}$ and the error $g(x^{(k)}) - g(x^\star)$ versus the iteration number k.

14.3 The figure shows $m = 20$ points (t_i, y_i) as circles. These points are well approximated by a function of the form

$$f(t) = \alpha t^\beta e^{\gamma t}.$$

(An example is shown in dashed line.)



Explain how you would compute values of the parameters α, β, γ such that

$$\alpha t_i^\beta e^{\gamma t_i} \approx y_i, \quad i = 1, \ldots, m, \tag{34}$$

using the following two methods.

(a) The Gauss-Newton method applied to the nonlinear least squares problem

$$\text{minimize} \quad \sum_{i=1}^{m} \left(\alpha t_i^\beta e^{\gamma t_i} - y_i \right)^2$$

with variables α, β, γ. Your description should include a clear statement of the linear least squares problems you solve at each iteration. You do not have to include a line search.

(b) Solving a single linear least squares problem, obtained by selecting a suitable error function for (34) and/or making a change of variables. Clearly state the least squares problem, the relation between its variables and the parameters α, β, γ, and the error function you choose to measure the quality of fit in (34).

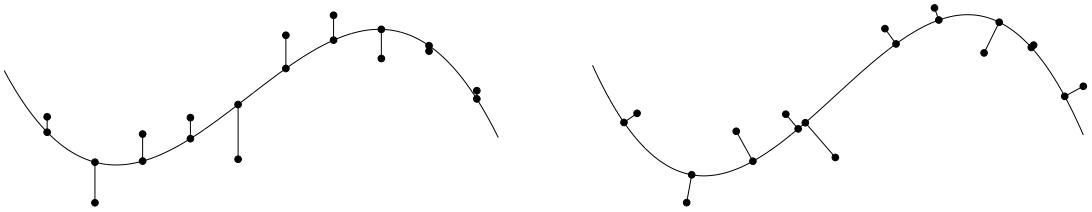
14.4 We have discussed the problem of fitting a polynomial

$$p(t) = c_1 + c_2 t + c_3 t^2 + \cdots + c_n t^{n-1}$$

to observations (t_1, y_1), $\ldots$, (t_N, y_N). The simplest method is to solve the least-squares problem

$$\text{minimize} \quad \sum_{i=1}^{N} (c_1 + c_2 t_i + c_3 t_i^2 + \cdots + c_n t_i^{n-1} - y_i)^2 \tag{35}$$

with variables $c_1, \ldots, c_n$. An example is shown in the figure on the left.



As a variation, one can seek to minimize the sum of the squares of the distances of the points (t_i, y_i) to the graph of the polynomial, as in the figure on the right. To formulate this as an optimization problem, we first note that the squared distance between the point (t_i, y_i) and the graph of the polynomial is given by

$$\min_{u_i} \left((p(u_i) - y_i)^2 + (u_i - t_i)^2 \right).$$

The problem of finding a polynomial that minimizes the sum of the squared distances can therefore be written as an optimization problem

$$\text{minimize} \quad \sum_{i=1}^{N} \left(c_1 + c_2 u_i + c_3 u_i^2 + \cdots + c_n u_i^{n-1} - y_i \right)^2 + \sum_{i=1}^{N} (u_i - t_i)^2, \tag{36}$$

with variables $c_1, \ldots, c_n$, and $u_1, \ldots, u_N$. Problem (36) is a nonlinear least-squares problem

$$\text{minimize} \quad \sum_{i=1}^{m} r_i(x)^2$$

with $m = 2N$, a variable $x = (c_1, \ldots, c_n, u_1, \ldots, u_N)$ of length $n + N$, and functions r_i defined as

$$r_i(x) = c_1 + c_2 u_i + c_3 u_i^2 + \cdots + c_n u_i^{n-1} - y_i, \qquad r_{N+i}(x) = u_i - t_i$$

for $i = 1, \ldots, N$.

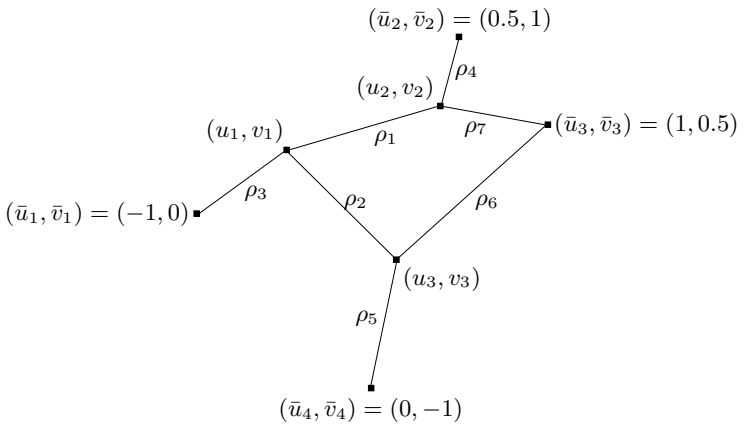
Use the Gauss-Newton method to fit a polynomial of degree 3 ($n = 4$) to the data in the file `polyfit.m`. (The command `[t, y] = polyfit` returns vectors t, y of length 25 containing the data points t_i, y_i.) A natural choice for the starting point in the Gauss-Newton method is to use the solution of the standard least-squares problem (35) to initialize the coefficients c_i, and to use t_i as initial value for u_i. With this choice, no line search is necessary for the data in `polyfit.m`.

14.5 In this exercise we use the Gauss-Newton method to solve the following problem: determine the coordinates of N points in a plane, given the coordinates of M other points with known positions, and (noisy) measurements of the distances between certain pairs of the $M + N$ points. We call the N points with unknown coordinates the *free points*, and the M points with known coordinates the *anchor points*.

In a practical application, the points correspond to nodes in a wireless sensor network. Some nodes (the anchor nodes) have been carefully placed or are equipped with GPS receivers, so their coordinates are known. The coordinates of the other nodes (the free nodes) have to be determined from measurements of the distances to neighboring nodes.

The problem can be represented as a graph. The $M + N$ nodes of the graph represent the N free points and the M anchor points. The coordinates of the free nodes are denoted (u_1, v_1), $\ldots$, (u_N, v_N). They are the variables in the problem. The coordinates of the anchor points are denoted $(\bar{u}_1, \bar{v}_1)$, $\ldots$, $(\bar{u}_M, \bar{v}_M)$ and are given. The K edges in the graph represent the measurements: if there is an edge between two nodes in the graph, a measurement is made of the distance between the corresponding points.

An example with 4 anchor points, 3 free points, and 7 distance measurements is shown below.



We will formulate the problem in terms of a $K \times N$-matrix B and three K-vectors c, d, ρ, defined as follows. The rows in B and the elements of b, c, and ρ correspond to the different edges in the graph.

- If row k corresponds to an edge between free points i and j, we take

  $$b_{ki} = 1, \qquad b_{kj} = -1, \qquad c_k = d_k = 0.$$

  The other elements of the kth row of B are zero. ρ_k is a (noisy) measurement of the distance $((u_i - u_j)^2 + (v_i - v_j)^2)^{1/2}$.
- If row k corresponds to an edge between free point i and anchor point j, we take

  $$b_{ki} = 1, \qquad c_k = -\hat{u}_j, \qquad d_k = -\hat{v}_j.$$

  The other elements in row k of B are zero. ρ_k is a measurement of the distance $((u_i - \hat{u}_j)^2 + (v_i - \hat{v}_j)^2)^{1/2}$.

With this notation, the length of edge k is

$$l_k(u, v) = \sqrt{(b_k^T u + c_k)^2 + (b_k^T v + d_k)^2}$$

where b_k^T is the kth row of B and $u = (u_1, \ldots, u_N)$, $v = (v_1, \ldots, v_N)$.
For the example of the figure, we can define B, c and d as

$$B = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad c = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -0.5 \\ 0 \\ -1 \\ -1 \end{bmatrix}, \qquad d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ -0.5 \\ -0.5 \end{bmatrix}.$$

The problem is to find values of u, v that satisfy $l_k(u, v) \approx \rho_k$ for $k = 1, \ldots, K$. This can be posed as a nonlinear least squares problem

$$g(u, v) = \sum_{k=1}^{K} r_k(u, v)^2 \tag{37}$$

with variables u and v, where

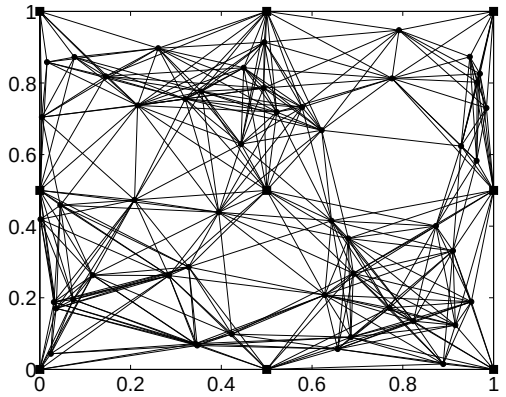
$$\begin{aligned} r_k(u, v) &= l_k(u, v)^2 - \rho_k^2 \\ &= (b_k^T u + c_k)^2 + (b_k^T v + d_k)^2 - \rho_k^2. \end{aligned}$$

(Here we define r_k as $l_k(u,v)^2 - \rho_k^2$ rather than $l_k(u,v) - \rho_k$ to simplify the calculation of the derivates.)

The file `networkloc.m` on the class webpage contains the data that we will use in the problem. It can be executed in MATLAB using the command

```
[B, c, d, rho] = networkloc;
```

This creates the problem data B, c, d, ρ for the network shown in the figure below.



There are 50 free nodes ($N = 50$), 9 anchor nodes shown as squares ($M = 9$), and 389 edges ($K = 389$).

Find estimates of u and v by solving (37) using the Gauss-Newton method. When testing your code, try several randomly generated starting points in the square $[0,1] \times [0,1]$ (using the MATLAB commands `u = rand(N,1)`, `v = rand(N,1)`). Terminate the iteration when $\|\nabla g(u,v)\|_2 \leq 10^{-6}$.

14.6 As in exercise 8.7 we consider the problem of fitting a circle to m points in a plane, but with a different error function. In this exercise we formulate the problem as a nonlinear least squares problem

$$\text{minimize} \quad g(u_c, v_c, R) = \sum_{i=1}^{m} \left(\sqrt{(u_i - u_c)^2 + (v_i - v_c)^2} - R \right)^2$$

with variables u_c, v_c, R. Note that $|\sqrt{(u_i - u_c)^2 + (v_i - v_c)^2} - R|$ is the distance of the point (u_i, v_i) to the circle, so in this formulation we minimize the sum of the squared distances of the points to the circle.

Apply the Gauss-Newton method to this nonlinear least squares problem with the data in `circlefit.m`. Terminate the iteration when $\|\nabla g(u_c, v_c, R)\| \leq 10^{-5}$. You can select a good starting point from the plot of the 50 points (u_i, v_i) or from the solution of exercise 8.7. With a good starting point, no backtracking line search will be needed. Include in your solution:

- a description of the least squares problem you solve at each iteration (the matrix A and the vector b)
- the MATLAB code
- the computed solution u_c, v_c, R
- a plot of the computed circle, created with the commands

```
t = linspace(0, 2*pi, 1000);
plot(u, v, 'o',  R * cos(t) + uc, R * sin(t) + vc, '-');
axis square
```

(assuming your MATLAB variables are called `uc`, `vc`, and `R`).

14.7 Suppose you are asked to fit the following three functions $f(v,w)$ to experimental data:

(a) $f(v,w) = \alpha v^\beta w^\gamma (v+w)^\delta$

(b) $f(v, w) = \alpha v^\beta + \gamma w^\delta$

(c) $f(v, w) = \alpha v^\beta w^\gamma / (1 + \alpha v^\beta w^\gamma)$.

Each function has two variables (v, w), and depends on several parameters (α, β, γ, δ). Your task is to calculate values of these parameters such that

$$f(v_i, w_i) \approx t_i, \quad i = 1, \ldots, N.$$

The problem data t_i, v_i, w_i are given. We assume that $N > 4$, and $v_i > 0$, $w_i > 0$, $t_i > 0$. In the third subproblem, we also assume that $t_i < 1$. The exponents of v, w, and $v + w$ in the definitions of f are allowed to be positive, negative, or zero, and do not have to be integers.

For each of the three functions, explain how you would formulate the model fitting problem as a linear least squares problem

$$\text{minimize} \quad \|Ax - b\|^2$$

or as a nonlinear least squares problem

$$\text{minimize} \quad \sum_{i=1}^{m} r_i(x)^2.$$

You are free to use any reasonable error criterion to judge the quality of the fit between the experimental data and the model. Your solution must include a clear statement of the following:

- the variables x in the linear or nonlinear least squares problem
- the error criterion that you minimize
- if you use linear least squares: the matrix A and the vector b
- if you use nonlinear least squares: the functions $r_i(x)$, and the matrix $A^{(k)}$ and the vector $b^{(k)}$ in the linear least squares problem

  $$\text{minimize} \quad \|A^{(k)}x - b^{(k)}\|^2$$

  that you solve at iteration k of the Gauss-Newton method.

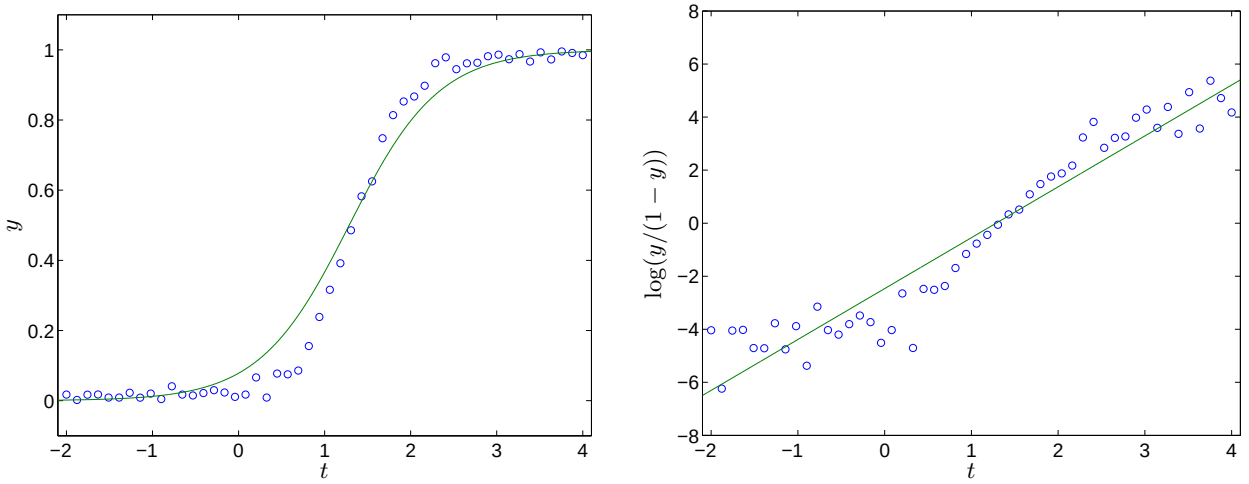
14.8 We revisit the data fitting problem of exercise 8.4. In that problem we used (linear) least squares to fit a function

$$f(t) = \frac{e^{\alpha t + \beta}}{1 + e^{\alpha t + \beta}}$$

to 50 points (t_i, y_i). To formulate the problem as a least squares problem we applied a nonlinear transformation $\log(y/(1-y))$ (the inverse of the function f) to the points y_i, and minimized the function

$$\sum_{i=1}^{m} \left(\alpha t_i + \beta - \log(\frac{y_i}{1 - y_i}) \right)^2.$$

An example is shown in the following figure (for a different set of points than in exercise 8.4.



As can be seen in the left-hand figure, the quality of the fit is not uniform (better for y_i near 1 or 0 than at the other points). A second problem with this approach is that it requires that $0 < y_i < 1$ for all data points.

In this exercise we compute the true least squares fit on the original scale, by solving the optimization problem

$$\text{minimize} \quad \sum_{i=1}^{m} \left(\frac{e^{\alpha t_i + \beta}}{1 + e^{\alpha t_i + \beta}} - y_i \right)^2. \tag{38}$$

This is a nonlinear least squares problem with two variables α, β.

Download the file `logistic_gn.m`, and execute in MATLAB as `[t, y] = logistic_gn;`. This is the set of 50 points used in the figures above. Solve the nonlinear least squares problem (38) using the Gauss-Newton method. You can use as starting point the solution of the linear least squares method of exercise 8.4 (which applies because $0 < y_i < 1$ at all points), or simply take $\alpha = \beta = 0$. With these starting points, no line search is necessary. Terminate the iteration when $\|\nabla g(\alpha, \beta)\| \leq 10^{-6}$, where $g(\alpha, \beta)$ is the cost function in (38). Compare the solution with the result of the linear least squares method.

14.9 The image taken by a camera can be described by a projective transformation F from $\mathbf{R}^3$ to $\mathbf{R}^2$, *i.e.*, a transformation

$$F(x) = \frac{1}{f^T x + g}(Cx + d)$$

where C is a 2×3 matrix, d is a 2-vector, f is a 3-vector, and g is a scalar. The 2-vector $F(x)$ is the projection of the point $x \in \mathbf{R}^3$ on the image plane of the camera.

Suppose a small (point) object at unknown location $x \in \mathbf{R}^3$ is viewed by l cameras, each described by a projective transformation $F_i(x) = (C_i x + d_i)/(f_i^T x + g_i)$. This gives l measurements

$$y_i = \frac{1}{f_i^T x + g_i}(C_i x + d_i) + v_i, \quad i = 1, \ldots, l,$$

where v_i is unknown measurement error. To estimate the position x from the l camera views, we solve the nonlinear least squares problem

$$\text{minimize} \quad \sum_{i=1}^{l} \left\| \frac{1}{f_i^T x + g_i}(C_i x + d_i) - y_i \right\|^2. \tag{39}$$

The variable is the vector $x \in \mathbf{R}^3$. The vectors $y_i \in \mathbf{R}^2$, and the camera parameters $C_i \in \mathbf{R}^{2\times 3}$, $d_i \in \mathbf{R}^2$, $f_i \in \mathbf{R}^3$, $g_i \in \mathbf{R}$ are given.

Suppose we solve problem (39) using the Gauss-Newton method. Describe in detail the linear least squares problem that is solved at the iteration that updates $x^{(k)}$ to $x^{(k+1)}$. You can assume that $f_i^T x^{(k)} + g_i > 0$ for $i = 1, \ldots, l$.

15 Matrix norm and condition number

15.1 What are the norms of the following matrices A?

(a) A matrix with one row:

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \end{bmatrix}.$$

(b) A matrix of the form $A = uu^T$ where u is a given n-vector.

(c) A matrix of the form $A = uv^T$ where u and v are given n-vectors.

15.2 Compute the matrix norm of each of the following matrices, without using MATLAB.

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & -3/2 \end{bmatrix}, \qquad \begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & -3/2 \end{bmatrix}.$$

15.3 In this exercise we show that $\|A\| = \|A^T\|$.

(a) Let u be a vector. Show that

$$\|u\| = \max_{v \neq 0} \frac{u^T v}{\|v\|}.$$

(b) Let A be a matrix. Show that

$$\|A\| = \max_{y \neq 0, x \neq 0} \frac{y^T A x}{\|x\| \|y\|}.$$

(c) Use the result of part (b) to show that $\|A\| = \|A^T\|$.

15.4 Let U and V be tall $m \times n$ matrices (*i.e.*, $m > n$) with orthonormal columns. Define $A = UV^T$. For each of the following three statements, either show that it is true, or give a small example (i.e., a specific U, V) for which it is false.

(a) A is nonsingular.

(b) A is orthogonal.

(c) $\|A\| = 1$.

15.5 Let P be a nonzero symmetric projection matrix (exercise 11.4). Show that $\|P\| = 1$.

15.6 Let A be an $m \times n$ matrix with $\|A\| < 1$.

(a) Show that the matrix $I - A^T A$ is positive definite.

(b) Show that the matrix

$$\begin{bmatrix} I & A \\ A^T & I \end{bmatrix}$$

is positive definite.

15.7

$$A = \begin{bmatrix} 0 & 0 & -10^4 & 0 \\ 0 & 0 & 0 & -10 \\ 0 & 10^{-3} & 0 & 0 \\ 10^{-2} & 0 & 0 & 0 \end{bmatrix}.$$

(a) What is the norm of A?

(b) What is the inverse of A?

(c) What is the norm of the inverse of A?

(d) What is the condition number of A?

Explain your answers, without referring to any MATLAB results. (Of course, you are free to check the answers in MATLAB.)

15.8 Give the matrix norm $\|A\|$ of each of the following matrices A, without using MATLAB. If A is nonsingular, also give $\|A^{-1}\|$ and $\kappa(A)$.

(a) $A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

(b) $A = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$

(c) $A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & -3/2 \end{bmatrix}$

(d) $A = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & -3/2 \end{bmatrix}$

(e) $A = \begin{bmatrix} 0 & -1 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & -3 \end{bmatrix}$

(f) $A = \begin{bmatrix} 2 & -1 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & -3 \end{bmatrix}$

(g) $A = \begin{bmatrix} 2 & -1 & -3 \\ -2 & 1 & 3 \\ 2 & -1 & -3 \end{bmatrix}$

15.9 Suppose Q is orthogonal. For each of the following matrices A, give $\|A\|$ and, if A is invertible, also $\|A^{-1}\|$. Explain your answers.

(a) $A = \begin{bmatrix} Q & -Q \\ Q & Q \end{bmatrix}$.

(b) $A = \begin{bmatrix} Q & Q \\ Q & Q \end{bmatrix}$.

15.10 The table shows $\|Ax^{(i)}\|$ and $\|x^{(i)}\|$ for four vectors $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, $x^{(4)}$, where A is a nonsingular $n \times n$ matrix.

x	$\|x\|$	$\|Ax\|$
$x^{(1)}$	1	100
$x^{(2)}$	100	1
$x^{(3)}$	10^3	10^4
$x^{(4)}$	10^{-3}	10^2

What are the best (*i.e.*, greatest) lower bounds on $\|A\|$, $\|A^{-1}\|$ and $\kappa(A)$ that you can derive based on this information?

15.11 Suppose A is a nonsingular 4×4 matrix with columns a_1, a_2, a_3, a_4:

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix}.$$

We are given the norms of the column vectors:

$$\|a_1\| = 1, \qquad \|a_2\| = 10^4, \qquad \|a_3\| = 10^{-3}, \qquad \|a_4\| = 10^{-3}.$$

Based on this information, what can you say about

(a) the norm of A,

(b) the norm of A^{-1},

(c) the condition number of A?

Be as precise as possible. If you can derive the exact value (for example, $\|A\| = 2.3$), give the exact value. Otherwise give an upper bound (for example, $\|A\| \leq 2.3$) or a lower bound (for example, $\|A\| \geq 2.3$). Upper bounds are more precise if they are lower ($\|A\| \leq 2.3$ is a more interesting bound than $\|A\| \leq 5$). Lower bounds are more precise if they are higher ($\|A\| \geq 2.3$ is a more interesting bound then $\|A\| \geq 1.5$).

15.12 Consider a set of linear equations $Ax = b$ with

$$A = \begin{bmatrix} 1 + 10^{-8} & 1 \\ 1 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

It is easily verified that A is nonsingular with inverse

$$A^{-1} = 10^8 \begin{bmatrix} 1 & -1 \\ -1 & 1 + 10^{-8} \end{bmatrix}.$$

(a) Prove, without using MATLAB, that $\kappa(A) \geq 10^8$.

(b) Find the solution of $Ax = b$. Construct a perturbation Δb of the right-hand side for which we have

$$\frac{\|\Delta x\|}{\|x\|} \geq 10^8 \frac{\|\Delta b\|}{\|b\|},$$

where $x + \Delta x$ is the solution of the equations $A(x + \Delta x) = b + \Delta b$.

15.13 Sort the following matrices in order of decreasing condition number (without using MATLAB):

$$A_1 = \begin{bmatrix} 10^5 & 1 \\ 1 & -10^5 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 10^5 & 1 \\ 1 & -10^{-5} \end{bmatrix},$$

$$A_3 = \begin{bmatrix} 10^{-5} & 1 \\ 1 & -10^{-5} \end{bmatrix}, \qquad A_4 = \begin{bmatrix} 10^5 & 1 \\ 1 & 10^{-5} \end{bmatrix}.$$

If any of the matrices is singular, take ∞ as its condition number. Explain your answer.

15.14 *Condition numbers and diagonal scaling.* The matrix A that represents a linear function $y = Ax$ depends on the units we choose for x and y. Suppose for example that the components of x are currents and the components of y are voltages, and that we have $y = Ax$ with x in amperes and y in volts. Now suppose $\tilde{x}_1$ is x_1 expressed in milliamperes, *i.e.*, $\tilde{x}_1 = 1000 x_1$. Then we can write

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} A_{11}/1000 & A_{12} & \cdots & A_{1n} \\ A_{21}/1000 & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1}/1000 & A_{n2} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = AD \begin{bmatrix} \widetilde{x}_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

where D is a diagonal matrix with diagonal elements $D_{11} = 1/1000$, $D_{22} = \cdots = D_{nn} = 1$.

Likewise, if $\tilde{y}_1$ is y_1 expressed in millivolts, we have

$$\begin{bmatrix} \tilde{y}_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1000\,A_{11} & 1000\,A_{12} & \cdots & 1000\,A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = DAx$$

where D is a diagonal matrix with diagonal elements $D_{11} = 1000$, $D_{22} = \cdots = D_{nn} = 1$.

In general, changing the units for x corresponds to replacing A with AD where D is positive diagonal matrix; changing the units for y corresponds to replacing A with DA where D is positive and diagonal.

In this problem we examine the effect of scaling columns or rows of a matrix on its condition number.

(a) Prove the following properties of the matrix norm. We assume A is $n \times n$ with columns a_i:

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}.$$

(i) $\|A\| \geq \max_{i=1,\ldots,n} \|a_i\|$ (*i.e.*, the norm of A is greater than or equal to the norm of each column a_i)

(ii) $\|A^{-1}\| \geq 1/\min_{i=1,\ldots,n} \|a_i\|$ (*i.e.*, $1/\|A^{-1}\|$ is less than or equal to the norm of each column a_i)

(iii) The condition number satisfies the lower bound

$$\kappa(A) \geq \frac{\max_{i=1,\ldots,n} \|a_i\|}{\min_{i=1,\ldots,n} \|a_i\|}.$$

In other words, if the columns of A are very different in norm, (*i.e.*, $\max_i \|a_i\| \gg \min_i \|a_i\|$), then A will certainly have a large condition number. In practice, it is therefore recommended to scale the columns of a matrix so that they are approximately equal in norm. Similar comments apply to row scaling.

(b) As an example, consider the matrix

$$A = \begin{bmatrix} 1 & 3 \cdot 10^{-3} & 11 \\ -2 \cdot 10^{-2} & 10^5 & -4 \cdot 10^2 \\ 1 & 10^4 & 10^8 \end{bmatrix}.$$

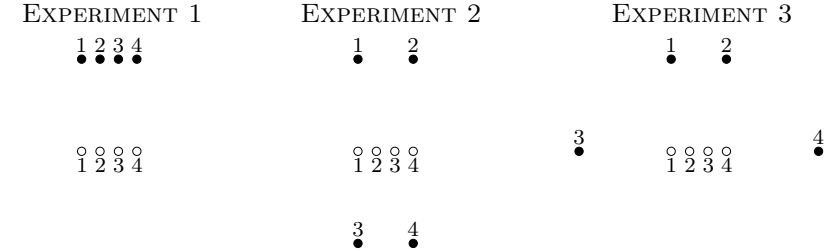
(i) Determine the condition number of A (using MATLAB's `cond` function).

(ii) Find a diagonal matrix D such that all columns of $\widetilde{A} = AD$ have the same norm. Determine the condition number of $\widetilde{A}$.

15.15 The figure below shows three possible experiments designed to estimate the magnitudes of signals emitted by four sources. The location of the sources is indicated by the empty circles. The solid circles show the location of the sensors. The output y_i of sensor i is given by

$$y_i = x_1/r_{i1}^2 + x_2/r_{i2}^2 + x_3/r_{i3}^2 + x_4/r_{i4}^2$$

where x_j is the (unknown) magnitude of the signal emitted by source j and r_{ij} is the (given) distance from source j to sensor i.



For each of the three configurations, we can determine the distances r_{ij} and write these equations as

$$Ax = y$$

where

$$A = \begin{bmatrix} 1/r_{11}^2 & 1/r_{12}^2 & 1/r_{13}^2 & 1/r_{14}^2 \\ 1/r_{21}^2 & 1/r_{22}^2 & 1/r_{23}^2 & 1/r_{24}^2 \\ 1/r_{31}^2 & 1/r_{32}^2 & 1/r_{33}^2 & 1/r_{34}^2 \\ 1/r_{41}^2 & 1/r_{42}^2 & 1/r_{43}^2 & 1/r_{44}^2 \end{bmatrix}, \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}. \tag{40}$$

From the measured sensor outputs y we can then determine x by solving the equations $Ax = y$.

There is a measurement error Δy in the sensor readings, which can be as large as 0.01%, *i.e.*, $\|\Delta y\|/\|y\| \leq 10^{-4}$. From the analysis in section 7.3 of the *EE133A Lecture Notes*, we have the following bound on the relative error in x:

$$\frac{\|\Delta x\|}{\|x\|} \leq \kappa(A) \frac{\|\Delta y\|}{\|y\|} \leq \kappa(A) \cdot 10^{-4}$$

where $\kappa(A)$ is the condition number of A.

(a) Download the MATLAB file `expdesign.m` from the class webpage and execute it in MATLAB as `[A1, A2, A3] = expdesign`. The three matrices A_1, A_2, A_3 are the values of the matrix A in (40) for each of the three experiments.

Compute the condition numbers of A_1, A_2, A_3 using the MATLAB command `cond(A)`.

(b) Based on the results of part (a), which configuration would you prefer?

(c) Can you give an intuitive argument for your conclusion in part (b)?

15.16 Vandermonde matrices

$$A = \begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{n-2} & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{n-2} & t_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & t_{n-1} & t_{n-1}^2 & \cdots & t_{n-1}^{n-2} & t_{n-1}^{n-1} \\ 1 & t_n & t_n^2 & \cdots & t_n^{n-2} & t_n^{n-1} \end{bmatrix}$$

are often badly conditioned. As an example, suppose

$$t_1 = 1, \qquad t_2 = 2, \qquad t_3 = 3, \qquad \ldots, \qquad t_{n-1} = n-1, \qquad t_n = n.$$

Show that $\kappa(A) \geq n^{n-3/2}$.

15.17 Consider the matrix

$$A = \begin{bmatrix} 1+\epsilon & 1 & 2 \\ 1 & -1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

(a) Show that A is singular for $\epsilon = 0$. Verify that for $\epsilon \neq 0$, the inverse is given by

$$A^{-1} = \frac{1}{\epsilon} \begin{bmatrix} 1 & 1 & -2 \\ 1 & 1-\epsilon & -2 \\ -1 & -1 & 2+\epsilon \end{bmatrix}.$$

(b) Prove that $\kappa(A) \geq 1/|\epsilon|$ if $\epsilon \neq 0$.

(c) Show with an example that the set of linear equations

$$Ax = b, \qquad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

is badly conditioned when ϵ is small (and nonzero). More specifically, give a $\Delta b \neq 0$ such that

$$\frac{\|\Delta x\|}{\|x\|} \geq \frac{1}{|\epsilon|} \frac{\|\Delta b\|}{\|b\|}$$

where x is the solution of $Ax = b$ and $x + \Delta x$ is the solution of $A(x + \Delta x) = b + \Delta b$.

15.18 Suppose A is a nonsingular $n \times n$ matrix. We denote by α_k the (Euclidean) norm of the kth column of A and by β_i the (Euclidean) norm of the ith row of A:

$$\alpha_k = \sqrt{\sum_{i=1}^{n} a_{ik}^2}, \qquad \beta_i = \sqrt{\sum_{k=1}^{n} a_{ik}^2}.$$

Show that

$$\kappa(A) \geq \frac{\max\{\alpha_{\max}, \beta_{\max}\}}{\min\{\alpha_{\min}, \beta_{\min}\}},$$

where

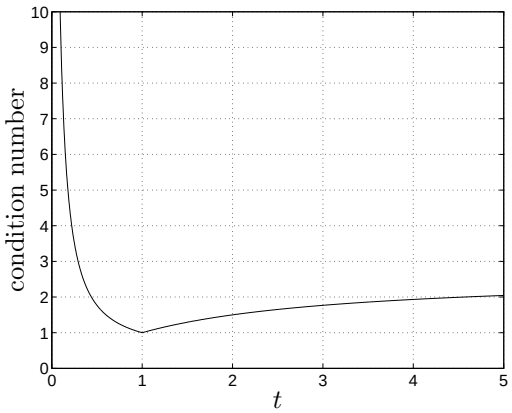
$$\alpha_{\max} = \max_{k=1,\ldots,n} \alpha_k, \qquad \alpha_{\min} = \min_{k=1,\ldots,n} \alpha_k, \qquad \beta_{\max} = \max_{i=1,\ldots,n} \beta_i, \qquad \beta_{\min} = \min_{i=1,\ldots,n} \beta_i.$$

15.19 Let L be a nonsingular $n \times n$ lower triangular matrix with elements L_{ij}. Show that

$$\kappa(L) \geq \frac{\max_{i=1,\ldots,n} |L_{ii}|}{\min_{j=1,\ldots,n} |L_{jj}|}.$$

15.20 The graph shows the condition number of one of the following matrices as a function of t for $t \geq 0$.

$$A_1 = \begin{bmatrix} t & 1 \\ 1 & -t \end{bmatrix}, \quad A_2 = \begin{bmatrix} t & t \\ -t & 1 \end{bmatrix}, \quad A_3 = \begin{bmatrix} t & 0 \\ 0 & 1+t \end{bmatrix}, \quad A_4 = \begin{bmatrix} t & -t \\ -t & 1 \end{bmatrix}.$$



Which of the four matrices was used in the figure? Carefully explain your answer.

15.21 We define A as the $n \times n$ lower triangular matrix with diagonal elements 1, and elements -1 below the diagonal:

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

(a) What is A^{-1}?

(b) Show that $\kappa(A) \geq 2^{n-2}$. This means that the matrix is ill-conditioned for large n.

(c) Show with an example that small errors in the right-hand side of $Ax = b$ can produce very large errors in x. Take $b = (0, 0, \ldots, 0, 1)$ (the n-vector with all its elements zero, except the last element, which is one), and find a nonzero Δb for which

$$\frac{\|\Delta x\|}{\|x\|} \geq 2^{n-2} \frac{\|\Delta b\|}{\|b\|},$$

where x is the solution of $Ax = b$ and $x + \Delta x$ is the solution of $A(x + \Delta x) = b + \Delta b$.

15.22 For an $n \times n$ matrix A, define

$$\gamma(A) = \frac{\max_{i=1,\ldots,n} |A_{ii}|}{\min_{j=1,\ldots,n} |A_{jj}|}.$$

This is the ratio of the absolute values of the largest and the smallest diagonal elements. If A has some zero diagonal elements, we define $\gamma(A) = \infty$.

In each of the two subproblems, select the correct statement (a, b, or c) and prove it.

(a) If A is diagonal and nonsingular, then

(i) $\kappa(A) = \gamma(A)$.

(ii) $\kappa(A) \leq \gamma(A)$, but equality does not always hold (*i.e.*, $\kappa(A) < \gamma(A)$ is possible).

(iii) $\kappa(A) \geq \gamma(A)$, but equality does not always hold.

(b) Same question for a triangular matrix: if A is triangular and nonsingular, then

(i) $\kappa(A) = \gamma(A)$.

(ii) $\kappa(A) \leq \gamma(A)$, but equality does not always hold.

(iii) $\kappa(A) \geq \gamma(A)$, but equality does not always hold.

15.23 Let A and E be two $n \times n$ matrices, with A nonsingular and $\|A^{-1}E\| < 1$.

(a) Show that $A + E$ is nonsingular.

(b) Let b be a nonzero n-vector. Define x and y as the solutions of the linear equations

$$Ax = b, \qquad (A + E)y = b.$$

Prove the inequalities

$$\|x - y\| \leq \left\|A^{-1}E\right\| \|y\| \leq \frac{\left\|A^{-1}E\right\|}{1 - \|A^{-1}E\|} \|x\|.$$

16 Algorithm stability

16.1 If we evaluate

$$\frac{1-\cos x}{\sin x} \tag{41}$$

at $x = 10^{-2}$, rounding the cosine and sine to 4 significant digits using the `chop` command[1], we obtain

```
>> (1 - chop(cos(1e-2), 4)) / chop(sin(1e-2), 4)
ans =
    0
```

The first four digits of the correct answer are $5.000\ 10^{-3}$. The large error is due to cancellation in the numerator $1-\cos(x)$: $\cos 10^{-2} = 0.99995000\ldots$, so rounding to four digits yields one, and subtracting from one yields zero.

Rewrite the expression (41) in a form that is mathematically equivalent, but avoids cancellation. Evaluate the stable formula (still rounding cosines and sines to 4 significant digits) and compare with the result above.

16.2 Recall the definition of average and standard deviation of a vector: if x is an n-vector, then

$$\mathbf{avg}(x) = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \mathbf{std}(x) = \frac{1}{\sqrt{n}}\|x - \mathbf{avg}(x)\mathbf{1}\|. \tag{42}$$

The square of the standard deviation can also be written as

$$\begin{aligned}
\mathbf{std}(x)^2 &= \frac{1}{n}\sum_{i=1}^{n}(x_i^2 - 2x_i\,\mathbf{avg}(x) + \mathbf{avg}(x)^2) \\
&= \frac{1}{n}\left(\sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i\,\mathbf{avg}(x) + n\,\mathbf{avg}(x)^2\right) \\
&= \frac{1}{n}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right).
\end{aligned} \tag{43}$$

The following MATLAB code evaluates the expressions (42) and (43), using the `chop` function to simulate a machine with a precision of 6 decimal digits.

```
>> n = length(x);
>> sum1 = 0;
>> sum2 = 0;
>> for i=1:n
     sum1 = chop(sum1 + x(i)^2, 6);
     sum2 = chop(sum2 + x(i), 6);
   end;
>> a = chop(sum2/n, 6)
>> s = chop((sum1 - sum2^2/n) / n, 6)
```

If we run this code with input vector

```
x =  [1002; 1000; 1003; 1001; 1002; 1002; 1001; 1004; 1002; 1001];
```

it returns

[1]The MATLAB command `chop(x,n)` rounds the number x to n decimal digits. We use it to artificially introduce rounding errors. For example, `chop(pi,4)` returns the number 3.14200000000000.

```
a =
   1.0018e+03
s =
   -3.2400
```

This is clearly wrong, because the variance must be a nonnegative number. (The correct answer is $\mathbf{avg}(x) = 1001.8$, $\mathbf{std}(x)^2 = 1.1600$.)

(a) Explain why the result is wrong.

(b) How would you compute the sample variance more accurately, without increasing the number of correct digits in the calculation (*i.e.*, if you still round all the intermediate results to 6 decimal digits)?

16.3 It can be shown that

$$\sum_{k=1}^{\infty} k^{-2} = \pi^2/6 = 1.644934\ldots.$$

Suppose we evaluate the first 3000 terms of the sum in MATLAB, rounding the result of each addition to four digits:

```
>> sum = 0;
>> for i = 1:3000
     sum = chop(sum + 1/i^2, 4);
   end
>> sum

sum =

   1.6240
```

The result has only 2 correct significant digits.

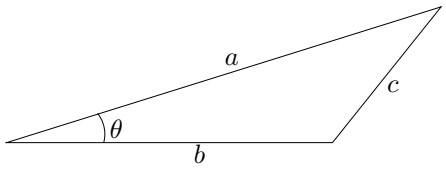
We can also evaluate the sum in the reverse order.

```
>> sum = 0;
>> for i = 3000:-1:1
     sum = chop(sum + 1/i^2, 4);
   end
>> sum

sum =

    1.6450
```

The result has four correct significant digits. Explain the difference. (Note that the calculation does not involve any subtractions, so this is an example of a large error that is *not* caused by cancellation.)

16.4 The length of one side of a triangle (c in the figure) can be calculated from the lengths of the two other sides (a and b) and the opposing angle (θ) by the formula

$$c = \sqrt{a^2 + b^2 - 2ab\cos\theta}. \tag{44}$$



Two equivalent expressions are

$$c = \sqrt{(a-b)^2 + 4ab\,(\sin(\theta/2))^2} \tag{45}$$

and

$$c = \sqrt{(a+b)^2 - 4ab\,(\cos(\theta/2))^2}. \tag{46}$$

(The equivalence of the three formulas follows from the identities $\cos\theta = 1 - 2\,(\sin(\theta/2))^2$ and $\cos\theta = -1 + 2\,(\cos(\theta/2))^2$.)

Which of the three formulas gives the most stable method for computing c if $a \approx b$ and θ is small? For simplicity you can assume that the calculations are exact, except for a small error in the evaluation of the cosine and sine functions. Explain your answer.

16.5 Consider the two nonlinear equations in the two (scalar) variables x, y

$$x^2 - y^2 = a, \qquad 2xy = b.$$

The right-hand sides a and b can be positive, negative, or zero. It is easily verified that

$$\hat{x} = \sqrt{\frac{a + \sqrt{a^2+b^2}}{2}}, \qquad \hat{y} = \mathrm{sign}(b)\sqrt{\frac{-a + \sqrt{a^2+b^2}}{2}}$$

is a solution, where $\mathrm{sign}(b) = 1$ if $b \geq 0$ and $\mathrm{sign}(b) = -1$ if $b < 0$.

Give a numerically stable algorithm for computing $\hat{x}$ and $\hat{y}$ when $|a| \gg |b|$.

17 Floating-point numbers

17.1 Evaluate the following expressions in MATLAB and explain the results. MATLAB uses IEEE double precision arithmetic.

(a) `(1 + 1e-16) - 1`

(b) `1 + (1e-16 - 1)`

(c) `(1 - 1e-16) - 1`

(d) `(2 + 2e-16) - 2`

(e) `(2 - 2e-16) - 2`

(f) `(2 + 3e-16) - 2`

(g) `(2 - 3e-16) - 2`

17.2 Answer the following questions, assuming IEEE double precision arithmetic.

(a) What is the largest floating-point number less than $1/2$?

(b) What is the smallest floating-point number greater than 4 ?

(c) How many floating-point numbers are there in the interval $[1/2, 4)$?

17.3 How many IEEE double precision floating-point numbers are contained in the following intervals?

(a) The interval $[1/2, 3/2)$.

(b) The interval $[3/2, 5/2)$.

Explain your answer.

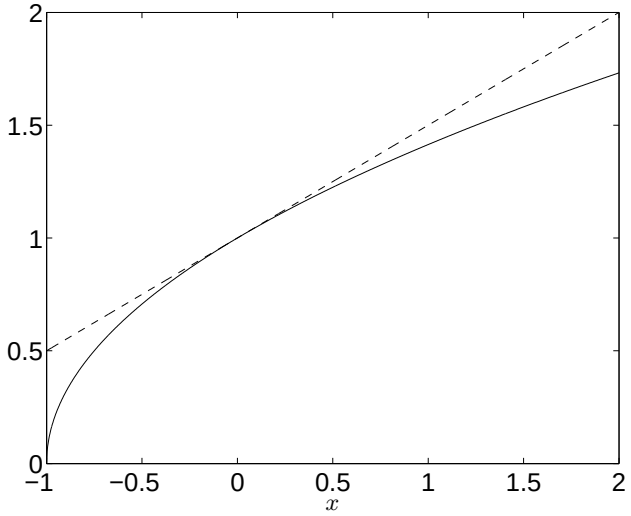
17.4 Run the following MATLAB code and explain the result. (*Hint.* If $x > 1$, then $1 < \sqrt{x} < 1 + \frac{1}{2}(x-1)$.)

```
>> x = 2;
>> for i=1:54
     x = sqrt(x);
   end;
>> for i=1:54
     x = x^2;
   end;
>> x
```

17.5 Explain the following results in MATLAB. (Note $\log(1+x)/x \approx 1$ for small x so the correct result is very close to 1.)

```
>>  log(1 + 3e-16) / 3e-16
ans =
    0.7401

>>  log(1 + 3e-16) / ((1 + 3e-16) - 1)
ans =
    1.0000
```

17.6 The figure shows $\sqrt{1+x}$ around $x = 0$ (solid line) and its first-order Taylor approximation $1 + x/2$ (dashed line).



It is clear that $\sqrt{1+x} \approx 1 + (1/2)x$ for small x. Therefore the function

$$f(x) = \frac{\sqrt{1+x} - 1}{x}$$

is approximately equal to $1/2$ for x around zero. We evaluated f in MATLAB, using the command

```
y = (sqrt(1 + x) - 1) / x
```

and obtained the following results:

x	y
$2 \cdot 10^{-16}$	0
$3 \cdot 10^{-16}$	0
$4 \cdot 10^{-16}$	0.5551
$5 \cdot 10^{-16}$	0.4441

(a) Explain the four values of y.

(b) Give a more accurate method for evaluating $f(x)$ for values of x near 0.

17.7 One possible definition of the number $e = 2.7182818\ldots$ is as the limit

$$e = \lim_{n\to\infty} (1 + 1/n)^n.$$

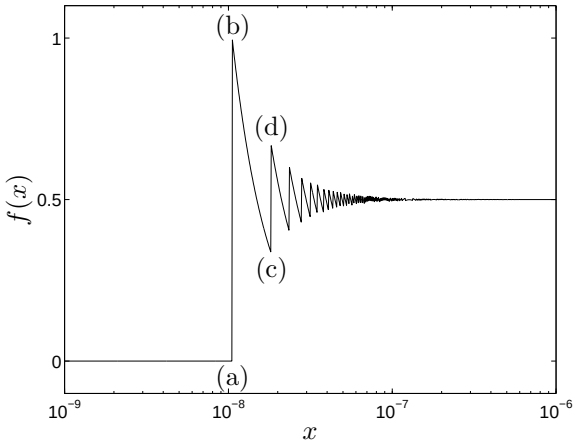
This suggests a method for evaluating e: we pick a large n, and evaluate $(1+1/n)^n$. One would expect that this yields better approximations as n increases.

Evaluate $(1 + 1/n)^n$ in MATLAB for $n = 10^4$, $n = 10^8$, $n = 10^{12}$, and $n = 10^{16}$. How many correct digits do you obtain? Explain briefly.

17.8 The plot shows the function

$$f(x) = \frac{1 - \cos(x)}{x^2},$$

evaluated in MATLAB using the command `(1 - cos(x)) / x^2`. We plot the function between $x = 10^{-9}$ and $x = 10^{-6}$, with a logarithmic scale for the x-axis.



We notice large errors: the correct value of $f(x)$ in this interval is very close to $1/2$, because $\cos x \approx 1 - x^2/2$ for small x. We also note that the computed function is not continuous.

(a) At what value of x does the first discontinuity (from point (a) to (b)) occur? Why is the computed value zero to the left of point (a)?

(b) At what point does the second discontinuity occur (from point (c) to (d))?

(c) What are the computed values at points (c) and (d)?

(d) Give a more stable method for computing $f(x)$ for small x.

Be as precise as possible in your answers for parts a, b and c. However, you can use the approximation $\cos x \approx 1 - x^2/2$.

17.9 The inequality

$$\|a - b\| \geq \|a\| - \|b\|$$

holds for all vectors a and b of the same length. The MATLAB code below evaluates the difference of the two sides of the inequality for

$$a = \begin{bmatrix} 1 \\ 2 \cdot 10^{-8} \end{bmatrix}, \qquad b = \begin{bmatrix} 10^{-16} \\ 0 \end{bmatrix}.$$

```
>> a = [1; 2e-8];
>> b = [1e-16; 0];
>> norm(a - b) - (norm(a) - norm(b))

ans =

  -2.2204e-16
```

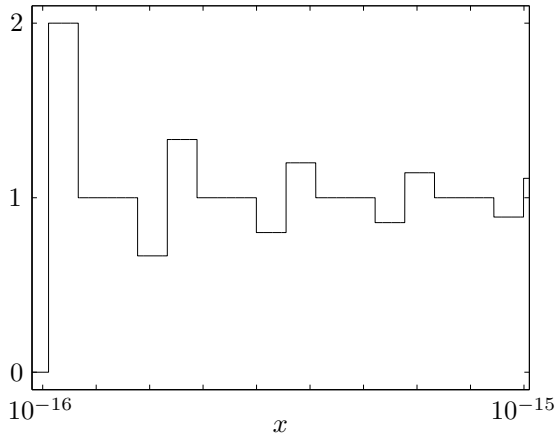
The result is negative, which contradicts the inequality. Explain the number returned by MATLAB, assuming IEEE double precision arithmetic was used.

(You can use the linear approximation $\sqrt{1+x} \approx 1 + x/2$ for small x. The linear function $1 + x/2$ is also an upper bound on $\sqrt{1+x}$ for all $x \geq -1$.)

17.10 The figure shows the function

$$f(x) = \frac{(1+x) - 1}{1 + (x-1)}$$

evaluated in IEEE double precision arithmetic in the interval $[10^{-16}, 10^{-15}]$, using the MATLAB command `((1 + x) - 1) / (1 + (x - 1))` to evaluate $f(x)$.



We notice that the computed function is piecewise-constant, instead of a constant 1.

(a) What are the endpoints of the intervals on which the computed values are constant?

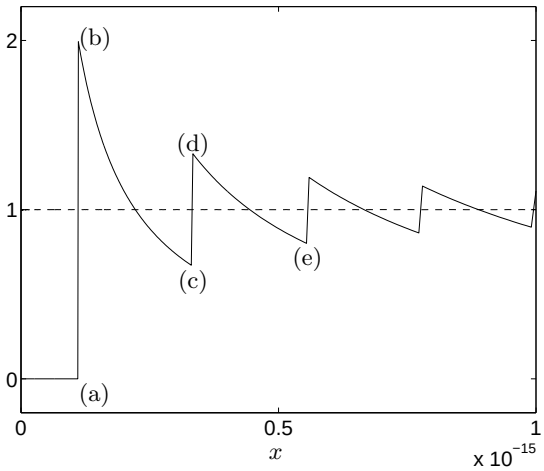
(b) What are the computed values on each interval?

Carefully explain your answers.

17.11 The graphs in the figure are the two functions

$$f(x) = \frac{\exp(\log x)}{x}, \qquad g(x) = \frac{\log(\exp x)}{x},$$

evaluated on the interval $(0, 10^{-15}]$ with IEEE double precision arithmetic, using the MATLAB commands `exp(log(x))/x` and `log(exp(x))/x`. One of the graphs is shown in dashed line and the other in solid line. Here, log denotes the natural logarithm, so the correct values are $f(x) = g(x) = 1$ for positive x. We see that the dashed line is quite accurate while the solid line is very inaccurate.



(a) Which of the two expressions (`exp(log(x))/x` or `log(exp(x))/x`) was used for the graph in solid line, and which one for the graph in dashed line? You can assume that the MATLAB functions `log(u)` and `exp(v)` return the exact values of $\log u$ and $\exp v$, rounded to the nearest floating-point number.

(b) Explain the graph in solid line up to point (e). What are (approximately) the horizontal and vertical values at the labeled points (a)–(e)? Why are the segments between (b) and (c), and between (d) and (e) nonlinear?

To analyze the effect of rounding error, you can use the first-order Taylor approximations $\log(a+b) \approx \log(a) + b/a$ for $a > 0$ and $|b| \ll a$, and $\exp(a+b) \approx \exp(a)(1+b)$ for $|b| \ll |a|$.

17.12 The derivative of a function f at a point $\hat{x}$ can be approximated as

$$f'(\hat{x}) \approx \frac{f(\hat{x}+h) - f(\hat{x}-h)}{2h}$$

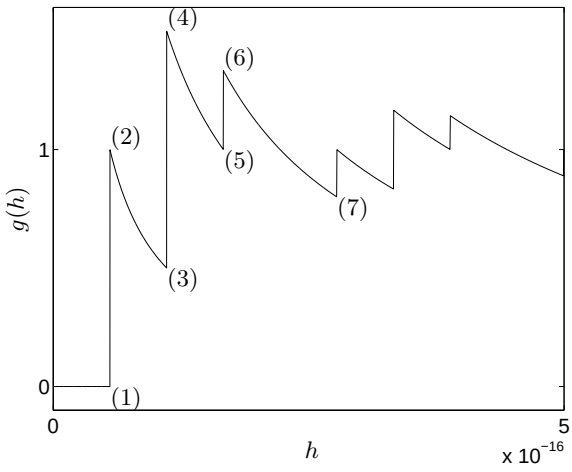
for small positive h. The right-hand side is known as a *finite-difference approximation* of the derivative.

The figure shows the finite-difference approximation of the derivative of $f(x) = \exp(x)$ at $\hat{x} = 0$, for values of h in the interval $(0, 5 \cdot 10^{-16}]$. The finite-difference approximation

$$g(h) = \frac{\exp(h) - \exp(-h)}{2h}$$

was computed using the MATLAB command

```
g = ( exp(h) - exp(-h) ) / ( 2*h )
```



Explain the first four segments of the graph, assuming IEEE double precision arithmetic was used in the calculation. You can make the approximation $\exp(t) \approx 1 + t$ for small t.

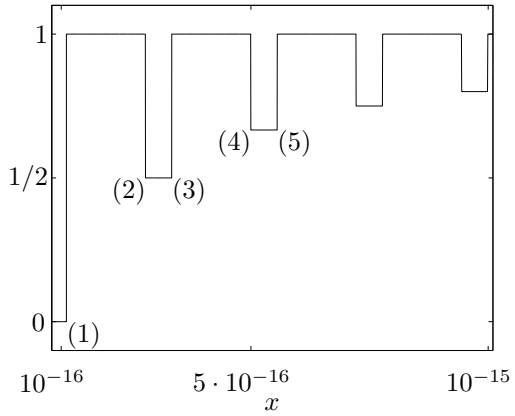
Your explanation should include the numerical values of the seven points marked on the graph, and an expression for the curve between points (2) and (3), (4) and (5), and (6) and (7).

17.13 For small positive x, the function

$$f(x) = \frac{1 - \dfrac{1}{1+x}}{-1 + \dfrac{1}{1-x}}$$

is very close to 1. The figure shows $f(x)$ in the interval $[10^{-16}, 10^{-15}]$, evaluated with IEEE double precision arithmetic, using the MATLAB command

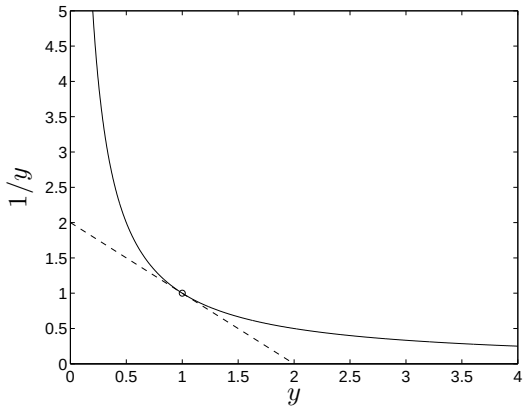
```
f = (1 - 1 / (1 + x)) / (-1 + 1 / (1 - x)).
```



Explain the graph up to the point labeled (5). What are the x-values of the five points? Why are the computed values on the first five segments 0, 1, 1/2, 1, and 2/3?

Assume that the result of each operation (addition, subtraction, division) is the exact value, rounded to the nearest floating-point number.

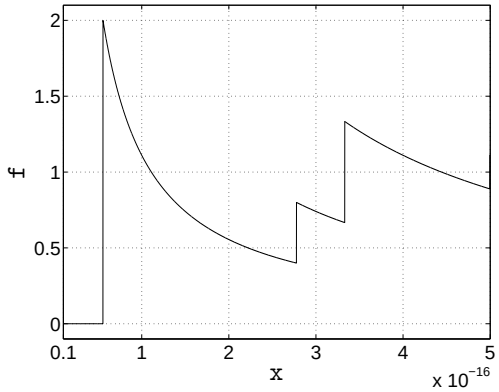
Hint. The linear approximation of $1/y$ around 1 is $1 - (y - 1)$ (the dashed line in the next figure).



17.14 The figure shows the result of evaluating the function

$$f(x) = \frac{\sqrt{1+x} - \sqrt{1-x}}{x}$$

in MATLAB (that is, using IEEE double precision arithmetic), in the interval $[10^{-17}, 5 \cdot 10^{-16}]$, using the command `f = ( sqrt(1 + x) - sqrt(1 - x) ) / x`.



(a) Carefully explain the following five values on the graph.

x	f
5e-17	0
1e-16	1.1102
2e-16	0.55511
3e-16	0.74015
4e-16	1.1102

You can use the approximation $\sqrt{1+x} \approx 1 + \frac{1}{2}x$ for $x \approx 0$, and the fact that $\sqrt{1+x} < 1 + \frac{1}{2}x$ for $x \neq 0$.

(b) Give a more stable method for evaluating $f(x)$.

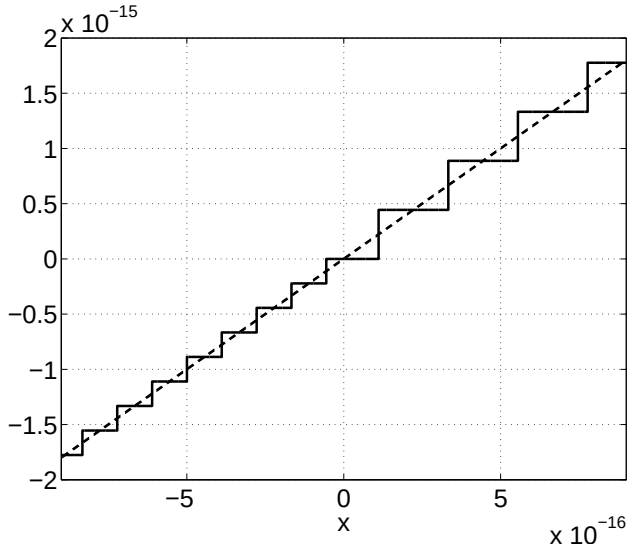
17.15 The two functions

$$f(x) = (1+x)^2 - 1, \qquad g(x) = x(2+x)$$

are obviously identical. We evaluated the two functions in MATLAB, for small x, using the commands

```
f = (1 + x)^2 - 1,      g = x * (2 + x),
```

and obtained the two graphs in the figure.



(a) Which expression (`f` or `g`) was used for the solid line and which one for the dashed line? Briefly explain your answer.

(b) Explain in detail the graph in solid line. At what values of x do the steps occur, and what is their height?