EE263 homework problems

Lecture 2 – Linear functions and examples

2.1 *A simple power control algorithm for a wireless network.* First some background. We consider a network of n transmitter/receiver pairs. Transmitter i transmits at power level p_i (which is positive). The path gain from transmitter j to receiver i is G_{ij} (which are all nonnegative, and G_{ii} are positive). The signal power at receiver i is given by $s_i = G_{ii}p_i$. The noise plus interference power at receiver i is given by

$$q_i = \sigma + \sum_{j \neq i} G_{ij} p_j$$

where $\sigma > 0$ is the self-noise power of the receivers (assumed to be the same for all receivers). The *signal to interference plus noise ratio* (SINR) at receiver i is defined as $S_i = s_i/q_i$. For signal reception to occur, the SINR must exceed some threshold value γ (which is often in the range $3 - 10$). Various *power control algorithms* are used to adjust the powers p_i to ensure that $S_i \geq \gamma$ (so that each receiver can receive the signal transmitted by its associated transmitter). In this problem, we consider a simple power control update algorithm. The powers are all updated synchronously at a fixed time interval, denoted by $t = 0, 1, 2, \ldots$. Thus the quantities p, q, and S are discrete-time signals, so for example $p_3(5)$ denotes the transmit power of transmitter 3 at time epoch $t = 5$. What we'd like is

$$S_i(t) = s_i(t)/q_i(t) = \alpha\gamma$$

where $\alpha > 1$ is an SINR safety margin (of, for example, one or two dB). Note that increasing $p_i(t)$ (power of the ith transmitter) increases S_i but decreases all other S_j. A very simple power update algorithm is given by

$$p_i(t+1) = p_i(t)(\alpha\gamma/S_i(t)). \tag{1}$$

This scales the power at the next time step to be the power that would achieve $S_i = \alpha\gamma$, if the interference plus noise term were to stay the same. But unfortunately, changing the transmit powers also changes the interference powers, so it's not that simple! Finally, we get to the problem.

(a) Show that the power control algorithm (1) can be expressed as a linear dynamical system with constant input, *i.e.*, in the form

$$p(t+1) = Ap(t) + b,$$

where $A \in \mathbf{R}^{n\times n}$ and $b \in \mathbf{R}^n$ are constant. Describe A and b explicitly in terms of σ, γ, α and the components of G.

(b) *Matlab simulation.* Use matlab to simulate the power control algorithm (1), starting from various initial (positive) power levels. Use the problem data

$$G = \begin{bmatrix} 1 & .2 & .1 \\ .1 & 2 & .1 \\ .3 & .1 & 3 \end{bmatrix}, \qquad \gamma = 3, \qquad \alpha = 1.2, \qquad \sigma = 0.01.$$

Plot S_i and p as a function of t, and compare it to the target value $\alpha\gamma$. Repeat for $\gamma = 5$. Comment briefly on what you observe. *Comment:* You'll soon understand what you see.

2.2 *State equations for a linear mechanical system.* The equations of motion of a lumped mechanical system undergoing small motions can be expressed as

$$M\ddot{q} + D\dot{q} + Kq = f$$

where $q(t) \in \mathbf{R}^k$ is the vector of deflections, M, D, and K are the *mass*, *damping*, and *stiffness* matrices, respectively, and $f(t) \in \mathbf{R}^k$ is the vector of externally applied forces. Assuming M is invertible, write linear system equations for the mechanical system, with state $x = [q^T \ \dot{q}^T]^T$, input $u = f$, and output $y = q$.

2.3 *Some standard time-series models.* A time series is just a discrete-time signal, *i.e.*, a function from $\mathbf{Z}_+$ into $\mathbf{R}$. We think of $u(k)$ as the value of the signal or quantity u at time (or *epoch*) k. The study of time series predates the extensive study of state-space linear systems, and is used in many fields (*e.g.*, econometrics). Let u and y be two time series (input and output, respectively). The relation (or *time series model*)

$$y(k) = a_0u(k) + a_1u(k-1) + \cdots + a_ru(k-r)$$

is called a *moving average (MA) model,* since the output at time k is a weighted average of the previous r inputs, and the set of variables over which we average 'slides along' with time. Another model is given by

$$y(k) = u(k) + b_1y(k-1) + \cdots + b_py(k-p).$$

This model is called an *autoregressive (AR) model*, since the current output is a linear combination of (*i.e.*, regression on) the current input and some previous values of the output. Another widely used model is the *autoregressive moving average (ARMA) model*, which combines the MA and AR models:

$$y(k) = b_1y(k-1) + \cdots + b_py(k-p) + a_0u(k) + \cdots + a_ru(k-r).$$

Finally, the problem: Express each of these models as a linear dynamical system with input u and output y. For the MA model, use state

$$x(k) = \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-r) \end{bmatrix},$$

and for the AR model, use state

$$x(k) = \begin{bmatrix} y(k-1) \\ \vdots \\ y(k-p) \end{bmatrix}.$$

You decide on an appropriate state vector for the ARMA model. (There are many possible choices for the state here, even with different dimensions. We recommend you choose a state for the ARMA model that makes it easy for you to derive the state equations.) **Remark:** multi-input, multi-output time-series models (*i.e.*, $u(k) \in \mathbf{R}^m$, $y(k) \in \mathbf{R}^p$) are readily handled by allowing the coefficients a_i, b_i to be matrices.

2.4 *Representing linear functions as matrix multiplication.* Suppose that $f : \mathbf{R}^n \longrightarrow \mathbf{R}^m$ is linear. Show that there is a matrix $A \in \mathbf{R}^{m \times n}$ such that for all $x \in \mathbf{R}^n$, $f(x) = Ax$. (Explicitly describe how you get the coefficients A_{ij} from f, and then verify that $f(x) = Ax$ for any $x \in \mathbf{R}^n$.) Is the matrix A that represents f unique? In other words, if $\tilde{A} \in \mathbf{R}^{m \times n}$ is another matrix such that $f(x) = \tilde{A}x$ for all $x \in \mathbf{R}^n$, then do we have $\tilde{A} = A$? Either show that this is so, or give an explicit counterexample.

2.5 *Some linear functions associated with a convolution system.* Suppose that u and y are scalar-valued discrete-time signals (*i.e.*, sequences) related via convolution:

$$y(k) = \sum_j h_j u(k-j), \quad k \in \mathbf{Z},$$

where $h_k \in \mathbf{R}$. You can assume that the convolution is *causal*, *i.e.*, $h_j = 0$ when $j < 0$.

(a) *The input/output (Toeplitz) matrix.* Assume that $u(k) = 0$ for $k < 0$, and define

$$U = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N) \end{bmatrix}, \quad Y = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N) \end{bmatrix}.$$

Thus U and Y are vectors that give the first $N+1$ values of the input and output signals, respectively. Find the matrix T such that $Y = TU$. The matrix T describes the linear mapping from (a chunk of) the input to (a chunk of) the output. T is called the input/output or Toeplitz matrix (of size $N+1$) associated with the convolution system.

(b) *The Hankel matrix.* Now assume that $u(k) = 0$ for $k > 0$ or $k < -N$ and let

$$U = \begin{bmatrix} u(0) \\ u(-1) \\ \vdots \\ u(-N) \end{bmatrix}, \quad Y = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N) \end{bmatrix}.$$

Here U gives the *past input* to the system, and Y gives (a chunk of) the resulting future output. Find the matrix H such that $Y = HU$. H is called the Hankel matrix (of size $N+1$) associated with the convolution system.

2.6 *Matrix representation of polynomial differentiation.* We can represent a polynomial of degree less than n,

$$p(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0,$$

as the vector $[a_0\ a_1\ \cdots\ a_{n-1}]^T \in \mathbf{R}^n$. Consider the linear transformation $\mathcal{D}$ that differentiates polynomials, *i.e.*, $\mathcal{D}p = dp/dx$. Find the matrix D that represents $\mathcal{D}$ (*i.e.*, if the coefficients of p are given by a, then the coefficients of dp/dx are given by Da).

2.7 Consider the (discrete-time) linear dynamical system

$$x(t+1) = A(t)x(t) + B(t)u(t), \quad y(t) = C(t)x(t) + D(t)u(t).$$

Find a matrix G such that

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

The matrix G shows how the output at $t = 0, \ldots, N$ depends on the initial state $x(0)$ and the sequence of inputs $u(0), \ldots, u(N)$.

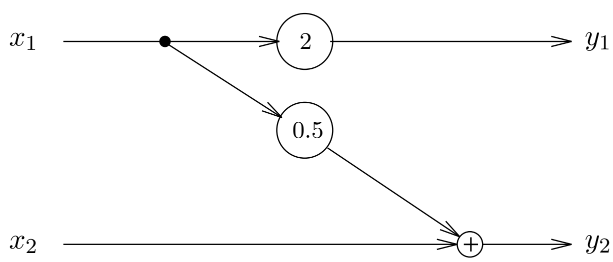
2.8 *Some sparsity patterns.*

(a) A matrix $A \in \mathbf{R}^{n\times n}$ is *tridiagonal* if $A_{ij} = 0$ for $|i-j| > 1$. Draw a block diagram of $y = Ax$ for A tridiagonal.

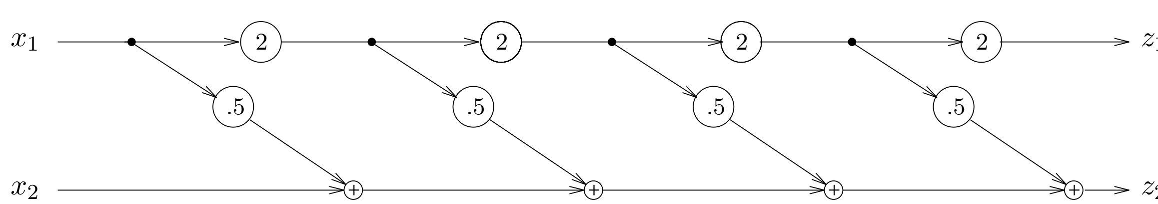
(b) Consider a certain linear mapping $y = Ax$ with $A \in \mathbf{R}^{m\times n}$. For i odd, y_i depends only on x_j for j even. Similarly, for i even, y_i depends only on x_j for j odd. Describe the sparsity structure of A. Give the structure a reasonable, suggestive name.

2.9 *Matrices and signal flow graphs.*

(a) Find $A \in \mathbf{R}^{2\times 2}$ such that $y = Ax$ in the system below:



(b) Find $B \in \mathbf{R}^{2\times 2}$ such that $z = Bx$ in the system below:



Do this two ways: first, by expressing the matrix B in terms of A from the previous part (explaining why they are related as you claim); and second, by directly evaluating all possible paths from each x_j to each z_i.

2.10 *Mass/force example.* Find the matrix A for the mass/force example in the lecture notes. For $n = 4$, find a specific input force sequence x that moves the mass to final position 1 and final velocity zero.

2.11 *Optimal force design.* We consider the mass/force example in the lecture notes, and in exercise 10, with $n = 4$, and the requirement that the final position is 1 and final velocity is 0. Roughly speaking, you have four variables and two equations, and therefore two extra degrees of freedom. In this problem you use the extra degrees of freedom to achieve other objectives, *i.e.*, minimize some cost functions that are described below.

(a) Find f that meets the specifications and minimizes the sum of squares of the forces, *i.e.*, $f_1^2 + f_2^2 + f_3^2 + f_4^2$.

(b) Find f that meets the specifications and minimizes the maximum force applied, *i.e.*, $\max\{|f_1|, |f_2|, |f_3|, |f_4|\}$.

(c) Find f that meets the specifications and minimizes the sum of absolute values of the forces applied, *i.e.*, $|f_1| + |f_2| + |f_3| + |f_4|$. (This would correspond to *minimum fuel usage* if the force comes from a thruster.)

There might be more than one minimizer; we just want one. If you can solve these optimization problems exactly, that's fine; but you are free to solve the problems numerically (and even approximately). We don't need an analysis verifying that your solution is indeed the best one. Make sure your solutions make sense to you. *Hints:*

- You can pick f_1 and f_2 to be anything; then f_3 and f_4 can be expressed in terms of f_1 and f_2.
- You can plot, or just evaluate, each cost function as a function of f_1 and f_2, over some reasonable range, to find the minimum.
- To evaluate the cost functions (which will be functions of two variables) for their minima, you might find one or more of the Matlab commands *norm*, *sum*, *abs*, *max*, *min* and *find* useful. (You can type *help norm*, for example, to see what *norm* does.) For example, if A is a matrix, $min(min(A))$ returns the smallest element of the matrix. The statement $[i, j] = find(A == min(min(A)))$ finds a pair of i, j indices in the matrix that correspond to the minimum value. Of course, there are many other ways to evaluate these functions; you don't have to use these commands.

2.12 *Undirected graph.* Consider an undirected graph with n nodes, and no self loops (*i.e.*, all branches connect two different nodes). Let $A \in \mathbf{R}^{n\times n}$ be the *node adjacency matrix*, defined as

$$A_{ij} = \begin{cases} 1 & \text{if there is a branch from node } i \text{ to node } j \\ 0 & \text{if there is no branch from node } i \text{ to node } j \end{cases}$$

Note that $A = A^T$, and $A_{ii} = 0$ since there are no self loops. We can intrepret A_{ij} (which is either zero or one) as the number of branches that connect node i to node j. Let $B = A^k$, where $k \in \mathbf{Z}$, $k \geq 1$. Give a simple interpretation of B_{ij} in terms of the original graph. (You might need to use the concept of a *path* of length m from node p to node q.)

2.13 *Counting sequences in a language or code.* We consider a language or code with an alphabet of n symbols $1, 2, \ldots, n$. A sentence is a finite sequence of symbols, $k_1, \ldots, k_m$ where $k_i \in \{1, \ldots, n\}$. A language or code consists of a set of sequences, which we will call the *allowable sequences*. A language is called *Markov* if the allowed sequences can be described by giving the allowable transitions between consecutive symbols. For each symbol we give a set of symbols which are allowed to follow the symbol. As a simple example, consider a Markov language with three symbols $1,2,3$. Symbol 1 can be followed by 1 or 3; symbol 2 must be followed by 3; and symbol 3 can be followed by 1 or 2. The sentence 1132313 is allowable (*i.e.*, in the language); the sentence 1132312 is not allowable (*i.e.*, not in the language). To describe the allowed symbol transitions we can define a matrix $A \in \mathbf{R}^{n\times n}$ by

$$A_{ij} = \begin{cases} 1 & \text{if symbol } i \text{ is allowed to follow symbol } j \\ 0 & \text{if symbol } i \text{ is not allowed to follow symbol } j \end{cases}.$$

(a) Let $B = A^k$. Give an interpretation of B_{ij} in terms of the language.

(b) Consider the Markov language with five symbols $1,2,3,4,5$, and the following transition rules:

- 1 must be followed by 2 or 3
- 2 must be followed by 2 or 5
- 3 must be followed by 1
- 4 must be followed by 4 or 2 or 5
- 5 must be followed by 1 or 3

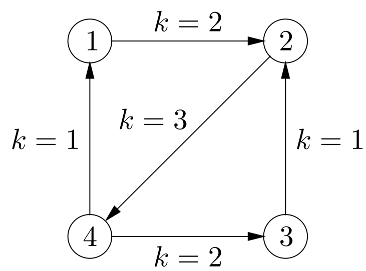
Find the total number of allowed sentences of length 10. Compare this number to the simple code that consists of all sequences from the alphabet (*i.e.*, all symbol transitions are allowed). In addition to giving the answer, you must explain how you solve the problem. Do not hesitate to use Matlab.

2.14 *Most common symbol in a given position.* Consider the Markov language from exercise 13, with five symbols $1,2,3,4,5$, and the following symbol transition rules:

- 1 must be followed by 2 or 3
- 2 must be followed by 2 or 5
- 3 must be followed by 1
- 4 must be followed by 4 or 2 or 5
- 5 must be followed by 1 or 3

Among all allowed sequences of length 10, find the most common value for the seventh symbol. In principle you could solve this problem by writing down all allowed sequences of length 10, and counting how many of these have symbol i as the seventh symbol, for $i = 1,\dots 5$. (We're interested in the symbol for which this count is largest.) But we'd like you to use a smarter approach. Explain clearly how you solve the problem, as well as giving the specific answer.

2.15 *Communication over a wireless network with time-slots.* We consider a network with n nodes, labeled $1,\dots,n$. A directed graph shows which nodes can send messages (directly) to which other nodes; specifically, an edge from node j to node i means that node j can transmit a message directly to node i. Each edge is assigned to one of K *time-slots*, which are labeled $1,\dots,K$. At time period $t = 1$, only the edges assigned to time-slot 1 can transmit a message; at time period $t = 2$, only the edges assigned to time-slot 2 can transmit a message, and so on. After time period $t = K$, the pattern repeats. At time period $t = K+1$, the edges assigned to time-slot 1 are again active; at $t = K+2$, the edges assigned to time-slot 2 are active, etc. This cycle repeats indefinitely: when $t = mK + k$, where m is an integer, and $k \in \{1, \ldots, K\}$, transmissions can occur only over edges assigned to time-slot k. Although it doesn't matter for the problem, we mention some reasons why the possible transmissions are assigned to time-slots. Two possible transmissions are assigned to different time-slots if they would interfere with each other, or if they would violate some limit (such as on the total power available at a node) if the transmissions occurred simultaneously. A message or packet can be sent from one node to another by a sequence of transmissions from node to node. At time period t, the message can be sent across any edge that is active at period t. It is also possible to store a message at a node during any time period, presumably for transmission during a later period. If a message is sent from node j to node i in period t, then in period $t+1$ the message is at node i, and can be stored there, or transmitted across any edge emanating from node i and active at time period $t+1$. To make sure the terminology is clear, we consider the very simple example shown below, with $n = 4$ nodes, and $K = 3$ time-slots.



In this example, we can send a message that starts in node 1 to node 3 as follows:

- During period $t = 1$ (time-slot $k = 1$), store it at node 1.
- During period $t = 2$ (time-slot $k = 2$), transmit it to node 2.
- During period $t = 3$ (time-slot $k = 3$), transmit it to node 4.
- During period $t = 4$ (time-slot $k = 1$), store it at node 4.
- During period $t = 5$ (time-slot $k = 2$), transmit it to node 3.

You can check that at each period, the transmission used is active, *i.e.*, assigned to the associated time-slot. The sequence of transmissions (and storing) described above gets the message from node 1 to node 3 in 5 periods. Finally, the problem. We consider a specific network with $n = 20$ nodes, and $K = 3$ time-slots, with edges and time-slot assignments given in `ts_data.m`. The labeled graph that specifies the possible transmissions and the associated time-slot assignments are given in a matrix $A \in \mathbf{R}^{n \times n}$, as follows:

$$A_{ij} = \begin{cases} k & \text{if transmission from node } j \text{ to node } i \text{ is allowed, and assigned to time-slot } k \\ 0 & \text{if transmission from node } j \text{ to node } i \text{ is never allowed} \\ 0 & i = j. \end{cases}$$

Note that we set $A_{ii} = 0$ for convenience. This choice has no significance; you can store a message at any node in any period. To illustrate this encoding of the graph, consider the simple example described above. For this example, we have

$$A_{\text{example}} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 3 & 0 & 0 \end{bmatrix}.$$

Very important: the problems below concern the network described in the mfile `ts_data.m`, and *not* the simple example given above.

(a) *Minimum-time point-to-point routing.* Find the fastest way to get a message that starts at node 5, to node 18. Give your solution as a prescription ordered in time from $t = 1$ to $t = T$ (the last transmission), as in the example above. At each time period, give the transmission (as in 'transmit from node 7 to node 9') or state that the message is to be stored (as in 'store at node 13'). Be sure that transmissions only occur during the associated time-slots. You only need to give *one* prescription for getting the message from node 5 to node 18 in minimum time.

(b) *Minimum time flooding.* In this part of the problem, we assume that once the message reaches a node, a copy is kept there, even when the message is transmitted to another node. Thus, the message is available at the node to be transmitted along any active edge emanating from that node, at any future period. Moreover, we allow multi-cast: if during a time period there are multiple active edges emanating from a node that has (a copy of) the message, then transmission can occur during that time period across *all* (or any subset) of the active edges. In this part of the problem, we are interested in getting a message that starts at a particular node, to all others, and we attach no cost to storage or transmission, so there is no harm is assuming that at each time period, every node that has the message forwards it to all nodes it is able to transmit to. What is the minimum time it takes before all nodes have a message that starts at node 7?

For both parts of the problem, you must give the specific solution, as well as a description of your approach and method.

2.16 *Solving triangular linear equations.* Consider the linear equations $y = Rx$, where $R \in \mathbf{R}^{n\times n}$ is upper triangular and invertible. Suggest a simple algorithm to solve for x given R and y. *Hint:* first find x_n; then find x_{n-1} (remembering that now you know x_n); then find x_{n-2} (remembering that now you know x_n and x_{n-1}); etc. **Remark:** the algorithm you will discover is called *back substitution.* It requires order n^2 floating point operations (flops); most methods for solving $y = Ax$ for general $A \in \mathbf{R}^{n\times n}$ require order n^3 flops.

2.17 *Gradient of some common functions.* Recall that the gradient of a differentiable function $f : \mathbf{R}^n \to \mathbf{R}$, at a point $x \in \mathbf{R}^n$, is defined as the vector

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix},$$

where the partial derivatives are evaluated at the point x. The first order Taylor approximation of f, near x, is given by

$$\hat{f}_{\text{tay}}(z) = f(x) + \nabla f(x)^T (z - x).$$

This function is affine, *i.e.*, a linear function plus a constant. For z near x, the Taylor approximation $\hat{f}_{\text{tay}}$ is very near f. Find the gradient of the following functions. Express the gradients using matrix notation.

(a) $f(x) = a^T x + b$, where $a \in \mathbf{R}^n$, $b \in \mathbf{R}$.

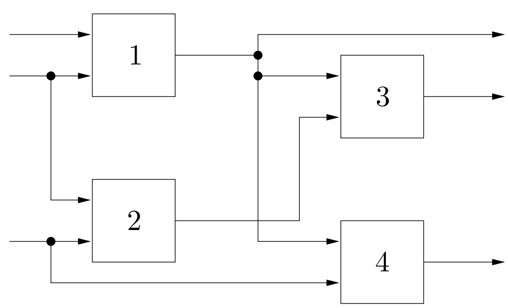
(b) $f(x) = x^T A x$, for $A \in \mathbf{R}^{n \times n}$.

(c) $f(x) = x^T A x$, where $A = A^T \in \mathbf{R}^{n \times n}$. (Yes, this is a special case of the previous one.)

2.18 *Digital circuit gate sizing.* A digital circuit consists of a set of n (logic) gates, interconnected by wires. Each gate has one or more inputs (typically between one and four), and one output, which is connected via the wires to other gate inputs and possibly to some external circuitry. When the output of gate i is connected to an input of gate j, we say that gate i *drives* gate j, or that gate j is in the *fan-out* of gate i. We describe the topology of the circuit by the *fan-out list* for each gate, which tells us which other gates the output of a gate connects to. We denote the fan-out list of gate i as $\mathrm{FO}(i) \subseteq \{1, \ldots, n\}$. We can have $\mathrm{FO}(i) = \emptyset$, which means that the output of gate i does not connect to the inputs of any of the gates $1, \ldots, n$ (presumably the output of gate i connects to some external circuitry). It's common to order the gates in such a way that each gate only drives gates with higher indices, *i.e.*, we have $\mathrm{FO}(i) \subseteq \{i+1, \ldots, n\}$. We'll assume that's the case here. (This means that the gate interconnections form a directed acyclic graph.)

To illustrate the notation, a simple digital circuit with $n = 4$ gates, each with 2 inputs, is shown below. For this circuit we have

$$\mathrm{FO}(1) = \{3, 4\}, \quad \mathrm{FO}(2) = \{3\}, \quad \mathrm{FO}(3) = \emptyset, \quad \mathrm{FO}(4) = \emptyset.$$



The 3 input signals arriving from the left are called *primary inputs*, and the 3 output signals emerging from the right are called *primary outputs* of the circuit. (You don't need to know this, however, to solve this problem.)

Each gate has a (real) *scale factor* or *size* x_i. These scale factors are the design variables in the gate sizing problem. They must satisfy $1 \leq x_i \leq x^{\max}$, where $x^{\max}$ is a given maximum allowed gate scale factor (typically on the order of 100). The total area of the circuit has the form

$$A = \sum_{i=1}^{n} a_i x_i,$$

where a_i are positive constants.

Each gate has an *input capacitance* C_i^{in}, which depends on the scale factor x_i as

$$C_i^{\mathrm{in}} = \alpha_i x_i,$$

where α_i are positive constants.

Each gate has a *delay* d_i, which is given by

$$d_i = \beta_i + \gamma_i C_i^{\text{load}}/x_i,$$

where β_i and γ_i are positive constants, and C_i^{load} is the *load capacitance* of gate i. Note that the gate delay d_i is always larger than β_i, which can be intepreted as the minimum possible delay of gate i, achieved only in the limit as the gate scale factor becomes large.

The load capacitance of gate i is given by

$$C_i^{\text{load}} = C_i^{\text{ext}} + \sum_{j \in \text{FO}(i)} C_j^{\text{in}},$$

where C_i^{ext} is a positive constant that accounts for the capacitance of the interconnect wires and external circuitry.

We will follow a simple design method, which assigns an equal delay T to all gates in the circuit, *i.e.*, we have $d_i = T$, where $T > 0$ is given. For a given value of T, there may or may not exist a feasible design (*i.e.*, a choice of the x_i, with $1 \leq x_i \leq x^{\max}$) that yields $d_i = T$ for $i = 1, \ldots, n$. We can assume, of course, that $T > \max_i \beta_i$, *i.e.*, T is larger than the largest minimum delay of the gates.

Finally, we get to the problem.

(a) Explain how to find a design $x^\star \in \mathbf{R}^n$ that minimizes T, subject to a given area constraint $A \leq A^{\max}$. You can assume the fanout lists, and all constants in the problem description are known; your job is to find the scale factors x_i. Be sure to explain how you determine if the design problem is feasible, *i.e.*, whether or not there is an x that gives $d_i = T$, with $1 \leq x_i \leq x^{\max}$, and $A \leq A^{\max}$.

Your method can involve any of the methods or concepts we have seen so far in the course. It can also involve a simple search procedure, *e.g.*, trying (many) different values of T over a range.

Note: this problem concerns the general case, and not the simple example shown above.

(b) Carry out your method on the particular circuit with data given in the file `gate_sizing_data.m`. The fan-out lists are given as an $n \times n$ matrix `F`, with i, j entry one if $j \in \text{FO}(i)$, and zero otherwise. In other words, the ith row of `F` gives the fanout of gate i. The jth entry in the ith row is 1 if gate j is in the fan-out of gate i, and 0 otherwise.

Comments and hints.

- You do not need to know anything about digital circuits; *everything* you need to know is stated above.
- Yes, this problem *does* belong on the EE263 midterm.

2.19 *Some matrices from signal processing.* We consider $x \in \mathbf{R}^n$ as a signal, with x_i the (scalar) value of the signal at (discrete) time period i, for $i = 1, \ldots, n$. Below we describe several transformations of the signal x, that produce a new signal y (whose dimension varies). For each one, find a matrix A for which $y = Ax$. formula for

(a) $2\times$ *up-conversion with linear interpolation.* We take $y \in \mathbf{R}^{2n-1}$. For i odd, $y_i = x_{(i+1)/2}$. For i even, $y_i = (x_{i/2} + x_{i/2+1})/2$. Roughly speaking, this operation doubles the sample rate, inserting new samples in between the original ones using linear interpolation.

(b) $2\times$ *down-sampling.* We assume here that n is even, and take $y \in \mathbf{R}^{n/2}$, with $y_i = x_{2i}$.

(c) $2\times$ *down-sampling with averaging.* We assume here that n is even, and take $y \in \mathbf{R}^{n/2}$, with $y_i = (x_{2i-1} + x_{2i})/2$.

2.20 *Smallest input that drives a system to a desired steady-state output.* We start with the discrete-time model of the system used in pages 16-19 of lecture 1:

$$x(t+1) = A_d x(t) + B_d u(t), \quad y(t) = C_d x(t), \quad t = 1, 2, \ldots,$$

where $A_d \in \mathbf{R}^{16\times 16}$, $B_d \in \mathbf{R}^{16\times 2}$, $C_d \in \mathbf{R}^{2\times 16}$. The system starts from the zero state, *i.e.*, $x(1) = 0$. (We start from initial time $t = 1$ rather than the more conventional $t = 0$ since Matlab indexes vectors starting from 1, not 0.) The data for this problem can be found in `ss_small_input_data.m`.

The goal is to find an input u that results in $y(t) \to y_{\text{des}} = (1, -2)$ as $t \to \infty$ (*i.e.*, asymptotic convergence to a desired output) or, even better, an input u that results in $y(t) = y_{\text{des}}$ for $t = T+1, \ldots$ (*i.e.*, exact convergence after T steps).

(a) *Steady-state analysis for desired constant output.* Suppose that the system is in steady-state, *i.e.*, $x(t) = x_{\text{ss}}$, $u(t) = u_{\text{ss}}$ and $y(t) = y_{\text{des}}$ are constant (do not depend on t). Find u_{ss} and x_{ss}.

(b) *Simple simulation.* Find $y(t)$, with initial state $x(1) = 0$, with $u(t) = u_{\text{ss}}$, for $t = 1, \ldots, 20000$. Plot u and y versus t. If you've done everything right, you should observe that $y(t)$ appears to be converging to y_{des}.

You can use the following Matlab code to obtain plots that look like the ones in lecture 1.

```
figure;
subplot(411); plot(u(1,:));
subplot(412); plot(u(2,:));
subplot(413); plot(y(1,:));
subplot(414); plot(y(2,:));
```

Here we assume that `u` and `y` are 2×20000 matrices. There will be two differences between these plots and those in lecture 1: These plots start from $t = 1$, and the plots in lecture 1 scale t by a factor of 0.1.

(c) *Smallest input.* Let $u^\star(t)$, for $t = 1, \ldots, T$, be the input with minimum RMS value

$$\left(\frac{1}{T}\sum_{t=1}^{T} \|u(t)\|^2\right)^{1/2}$$

that yields $x(T+1) = x_{\text{ss}}$ (the value found in part (a)). Note that if $u(t) = u^\star(t)$ for $t = 1, \ldots, T$, and then $u(t) = u_{\text{ss}}$ for $t = T+1, T+2, \ldots$, then $y(t) = y_{\text{des}}$ for $t \geq T+1$. In other words, we have exact convergence to the desired output in T steps.

For the three cases $T = 100$, $T = 200$, and $T = 500$, find $u^\star$ and its associated RMS value. For each of these three cases, plot u and y versus t.

(d) Plot the RMS value of $u^\star$ versus T for T between 100 and 1000 (for multiples of 10, if you like). The plot is probably better viewed on a log-log scale, which can be done using the command `loglog` instead of the command `plot`.

Lecture 3 – Linear algebra review

3.1 *Price elasticity of demand.* The demand for n different goods as a function of their prices is described by a function $f(\cdot)$ from $\mathbf{R}^n$ to $\mathbf{R}^n$:

$$q = f(p),$$

where p is the price vector, and q is the demand vector. Linear models of the demand function are often used to analyze the effects of small price changes. Denoting the current price and current demand vectors by p^* and q^*, we have that $q^* = f(p^*)$, and the linear approximation is:

$$q^* + \delta q \approx f(p^*) + \left.\frac{df}{dp}\right|_{p^*} \delta p.$$

This is usually rewritten in term of the *elasticity matrix* E, with entries

$$e_{ij} = \left.\frac{df_i}{dp_j}\right|_{p_j^*} \frac{1/q_i^*}{1/p_j^*}$$

(*i.e.*, relative change in demand per relative change in price.) Define the vector y of relative demand changes, and the vector x of relative price changes,

$$y_i = \frac{\delta q_i}{q_i^*}, \quad x_j = \frac{\delta p_j}{p_j^*},$$

and, finally, we have the linear model $y = Ex$. Here are the questions:

(a) What is a reasonable assumption about the diagonal elements e_{ii} of the elasticity matrix?

(b) Consider two goods. The off-diagonal elements of E describe how the demand for one good is affected by changes in the price of the other good. Assume $e_{11} = e_{22} = -1$ and $e_{12} = e_{21}$, that is,

$$E = \begin{bmatrix} -1 & e_{12} \\ e_{12} & -1 \end{bmatrix}.$$

Two goods are called *substitutes* if they provide a similar service or other satisfaction (for example: train tickets and bus tickets, cake and pie, etc.) Two goods are called *complements* if they tend to be used together (for example: automobiles and gasoline, left and right shoes, etc.) For each of these two generic situations, what can you say about e_{12}?

(c) Suppose the price elasticity of demand matrix is

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

Describe the *nullspace* of E, and give an interpretation (in one or two sentences.) What kind of goods could have such an elasticity matrix?

3.2 *Color perception.* Human color perception is based on the responses of three different types of color light receptors, called *cones*. The three types of cones have different spectral response characteristics and are called L, M, and, S because they respond mainly to long, medium, and short wavelengths, respectively. In this problem we will divide the visible spectrum into 20 bands, and model the cones' response as follows:

$$L_{\text{cone}} = \sum_{i=1}^{20} l_i p_i, \qquad M_{\text{cone}} = \sum_{i=1}^{20} m_i p_i, \qquad S_{\text{cone}} = \sum_{i=1}^{20} s_i p_i,$$

where p_i is the incident power in the ith wavelength band, and l_i, m_i and s_i are nonnegative constants that describe the spectral response of the different cones. The perceived color is a complex function of the three cone responses, *i.e.*, the vector $(L_{\text{cone}}, M_{\text{cone}}, S_{\text{cone}})$, with different cone response vectors perceived as different colors. (Actual color perception is a bit more complicated than this, but the basic idea is right.)

(a) *Metamers.* When are two light spectra, p and $\tilde{p}$, visually indistinguishable? (Visually identical lights with different spectral power compositions are called *metamers.*)

(b) *Visual color matching.* In a color matching problem, an observer is shown a test light and is asked to change the intensities of three primary lights until the sum of the primary lights looks like the test light. In other words, the observer is asked the find a spectrum of the form

$$p_{\text{match}} = a_1 u + a_2 v + a_3 w,$$

where u, v, w are the spectra of the primary lights, and a_i are the (nonnegative) intensities to be found, that is visually indistinguishable from a given test light spectrum p_{test}. Can this always be done? Discuss briefly.

(c) *Visual matching with phosphors.* A computer monitor has three phosphors, R, G, and B. It is desired to adjust the phosphor intensities to create a color that looks like a reference test light. Find weights that achieve the match or explain why no such weights exist. The data for this problem is in the an m-file `color_perception.m`. Running `color_perception` will define and plot the vectors `wavelength`, `B_phosphor`, `G_phosphor`, `R_phosphor`, `L_coefficients`, `M_coefficients`, `S_coefficients`, and `test_light`.

(d) *Effects of illumination.* An object's surface can be characterized by its reflectance (*i.e.*, the fraction of light it reflects) for each band of wavelengths. If the object is illuminated with a light spectrum characterized by I_i, and the reflectance of the object is r_i (which is between 0 and 1), then the reflected light spectrum is given by $I_i r_i$, where $i = 1, \ldots, 20$ denotes the wavelength band. Now consider two objects illuminated (at different times) by two different light sources, say an incandescent bulb and sunlight. Sally argues that if the two objects look identical when illuminated by a tungsten bulb, they will look identical when illuminated by sunlight. Beth disagrees: she says that two objects can appear identical when illuminated by a tungsten bulb, but look different when lit by sunlight. Who is right? If Sally is right, explain why. If Beth is right give an example of two objects that appear identical under one light source and different under another. You can use the vectors `sunlight` and `tungsten` defined in `color_perception.m` as the light sources.

3.3 *Halfspace.* Suppose $a, b \in \mathbf{R}^n$ are two given points. Show that the set of points in $\mathbf{R}^n$ that are closer to a than b is a halfspace, *i.e.*:

$$\{x \mid \|x-a\| \leq \|x-b\| \ \} = \{ \ x \mid c^T x \leq d\}$$

for appropriate $c \in \mathbf{R}^n$ and $d \in \mathbf{R}$. Give c and d explicitly, and draw a picture showing a, b, c, and the halfspace.

3.4 *Some properties of the product of two matrices.* For each of the following statements, either show that it is true, or give a (specific) counterexample.

- If AB is full rank then A and B are full rank.
- If A and B are full rank then AB is full rank.
- If A and B have zero nullspace, then so does AB.
- If A and B are onto, then so is AB.

You can assume that $A \in \mathbf{R}^{m \times n}$ and $B \in \mathbf{R}^{n \times p}$. Some of the false statements above become true under certain assumptions on the dimensions of A and B. As a trivial example, all of the statements above are true when A and B are scalars, *i.e.*, $n = m = p = 1$. For each of the statements above, find conditions on n, m, and p that make them true. Try to find the most general conditions you can. You can give your conditions as inequalities involving n, m, and p, or you can use more informal language such as "A and B are both skinny."

3.5 *Rank of a product.* Suppose that $A \in \mathbf{R}^{7 \times 5}$ has rank 4, and $B \in \mathbf{R}^{5 \times 7}$ has rank 3. What values can $\mathbf{Rank}(AB)$ possibly have? For each value r that is possible, give an example, *i.e.*, a specific A and B with the dimensions and ranks given above, for which $\mathbf{Rank}(AB) = r$. Please try to give simple examples, that make it easy for you to justify that the ranks of A, B, and AB are what you claim they are. We will deduct points for correct examples that are needlessly complicated. You can use Matlab to verify the ranks, but we don't recommend it: numerical roundoff errors in Matlab's calculations can sometimes give errors in computing the rank. (Matlab may still be useful; you just have to double check that the ranks it finds are correct.) Explain briefly why the rank of AB must be one of the values you give.

3.6 *Linearizing range measurements.* Consider a single (scalar) measurement y of the distance or range of $x \in \mathbf{R}^n$ to a fixed point or beacon at a, *i.e.*, $y = \|x - a\|$.

(a) Show that the linearized model near x_0 can be expressed as $\delta y = k^T \delta x$, where k is the unit vector (*i.e.*, with length one) pointing from a to x_0. Derive this analytically, and also draw a picture (for $n = 2$) to demonstrate it.

(b) Consider the error e of the linearized approximation, *i.e.*,

$$e = \|x_0 + \delta x - a\| - \|x_0 - a\| - k^T \delta x.$$

The relative error of the approximation is given by $\eta = e/\|x_0 - a\|$. We know, of course, that the absolute value of the relative error is very small provided δx is small. In many specific applications, it is possible and useful to make a stronger statement, for example, to derive a bound on how large the error can be. You will do that here. In fact you will prove that

$$0 \le \eta \le \frac{\alpha^2}{2}$$

where $\alpha = \|\delta x\|/\|x_0 - a\|$ is the relative size of δx. For example, for a relative displacement of $\alpha = 1\%$, we have $\eta \le 0.00005$, *i.e.*, the linearized model is accurate to about 0.005%. To prove this bound you can proceed as follows:

- Show that $\eta = -1 + \sqrt{1 + \alpha^2 + 2\beta} - \beta$ where $\beta = k^T \delta x/\|x_0 - a\|$.
- Verify that $|\beta| \le \alpha$.
- Consider the function $g(\beta) = -1 + \sqrt{1 + \alpha^2 + 2\beta} - \beta$ with $|\beta| \le \alpha$. By maximizing and minimizing g over the interval $-\alpha \le \beta \le \alpha$ show that

$$0 \le \eta \le \frac{\alpha^2}{2}.$$

3.7 *Orthogonal complement of a subspace.* If $\mathcal{V}$ is a subspace of $\mathbf{R}^n$ we define $\mathcal{V}^\perp$ as the set of vectors orthogonal to every element in $\mathcal{V}$, *i.e.*,

$$\mathcal{V}^\perp = \{\ x \mid \langle x, y \rangle = 0,\ \forall y \in \mathcal{V}\ \}.$$

(a) Verify that $\mathcal{V}^\perp$ is a subspace of $\mathbf{R}^n$.

(b) Suppose $\mathcal{V}$ is described as the span of some vectors $v_1, v_2, \ldots, v_r$. Express $\mathcal{V}$ and $\mathcal{V}^\perp$ in terms of the matrix $V = \begin{bmatrix} v_1 & v_2 & \cdots & v_r \end{bmatrix} \in \mathbf{R}^{n\times r}$ using common terms (range, nullspace, transpose, etc.)

(c) Show that every $x \in \mathbf{R}^n$ can be expressed uniquely as $x = v + v^\perp$ where $v \in \mathcal{V}$, $v^\perp \in \mathcal{V}^\perp$. *Hint:* let v be the projection of x on $\mathcal{V}$.

(d) Show that $\dim \mathcal{V}^\perp + \dim \mathcal{V} = n$.

(e) Show that $\mathcal{V} \subseteq \mathcal{U}$ implies $\mathcal{U}^\perp \subseteq \mathcal{V}^\perp$.

3.8 Consider the linearized navigation equations from the lecture notes. Find the conditions under which A has full rank. Describe the conditions geometrically (*i.e.*, in terms of the relative positions of the unknown coordinates and the beacons).

3.9 Suppose that $\angle(Ax, x) = 0$ for all $x \in \mathbf{R}^n$, *i.e.*, x and Ax always point in the same direction. What can you say about the matrix A? Be very specific.

3.10 *Proof of Cauchy-Schwarz inequality.* You will prove the Cauchy-Schwarz inequality.

(a) Suppose $a \ge 0$, $c \ge 0$, and for all $\lambda \in \mathbf{R}$, $a + 2b\lambda + c\lambda^2 \ge 0$. Show that $|b| \le \sqrt{ac}$.

(b) Given v, $w \in \mathbf{R}^n$ explain why $(v + \lambda w)^T(v + \lambda w) \ge 0$ for all $\lambda \in \mathbf{R}$.

(c) Apply (a) to the quadratic resulting when the expression in (b) is expanded, to get the Cauchy-Schwarz inequality:

$$|v^T w| \le \sqrt{v^T v}\sqrt{w^T w}.$$

(d) When does equality hold?

3.11 *Vector spaces over the Boolean field.* In this course the *scalar field*, *i.e.*, the components of vectors, will usually be the real numbers, and sometimes the complex numbers. It is also possible to consider vector spaces over other fields, for example $\mathbf{Z}_2$, which consists of the two numbers 0 and 1, with Boolean addition and multiplication (*i.e.*, $1+1=0$). Unlike $\mathbf{R}$ or $\mathbf{C}$, the field $\mathbf{Z}_2$ is finite, indeed, has only two elements. A vector in $\mathbf{Z}_2^n$ is called a *Boolean vector*. Much of the linear algebra for $\mathbf{R}^n$ and $\mathbf{C}^n$ carries over to $\mathbf{Z}_2^n$. For example, we define a function $f : \mathbf{Z}_2^n \to \mathbf{Z}_2^m$ to be linear (over $\mathbf{Z}_2$) if $f(x+y) = f(x) + f(y)$ and $f(\alpha x) = \alpha f(x)$ for every $x,\ y \in \mathbf{Z}_2^n$ and $\alpha \in \mathbf{Z}_2$. It is easy to show that every linear function can be expressed as matrix multiplication, *i.e.*, $f(x) = Ax$, where $A \in \mathbf{Z}_2^{m\times n}$ is a Boolean matrix, and all the operations in the matrix multiplication are Boolean, *i.e.*, in $\mathbf{Z}_2$. Concepts like nullspace, range, independence and rank are all defined in the obvious way for vector spaces over $\mathbf{Z}_2$. Although we won't consider them in this course, there are many important applications of vector spaces and linear dynamical systems over $\mathbf{Z}_2$. In this problem you will explore one simple example: block codes. *Linear block codes.* Suppose $x \in \mathbf{Z}_2^n$ is a Boolean vector we wish to transmit over an unreliable channel. In a linear block code, the vector $y = Gx$ is formed, where $G \in \mathbf{Z}_2^{m\times n}$ is the *coding matrix*, and $m > n$. Note that the vector y is 'redundant'; roughly speaking we have *coded* an n-bit vector as a (larger) m-bit vector. This is called an (n, m) code. The coded vector y is transmitted over the channel; the received signal $\hat{y}$ is given by

$$\hat{y} = y + v,$$

where v is a noise vector (which usually is zero). This means that when $v_i = 0$, the ith bit is transmitted correctly; when $v_i = 1$, the ith bit is changed during transmission. In a *linear decoder*, the received signal is multiplied by another matrix: $\hat{x} = H\hat{y}$, where $H \in \mathbf{Z}_2^{n\times m}$. One reasonable requirement is that if the transmission is perfect, *i.e.*, $v = 0$, then the decoding is perfect, *i.e.*, $\hat{x} = x$. This holds if and only if H is a left inverse of G, *i.e.*, $HG = I_n$, which we assume to be the case.

(a) What is the practical significance of $\mathcal{R}(G)$?

(b) What is the practical significance of $\mathcal{N}(H)$?

(c) A one-bit *error correcting code* has the property that for any noise v with one component equal to one, we still have $\hat{x} = x$. Consider $n = 3$. Either design a one-bit error correcting linear block code with the smallest possible m, or explain why it cannot be done. (By design we mean, give G and H explicitly and verify that they have the required properties.)

Remark: linear decoders are never used in practice; there are far better nonlinear ones.

3.12 *Quadratic extrapolation of a time series.* We are given a series z up to time n. Using a quadratic model, we want to extrapolate, or predict, $z(n+1)$ based on the three previous elements of the series: $z(n)$, $z(n-1)$, and $z(n-2)$. We'll denote the predicted value of $z(n+1)$ by y. Another way to describe this problem is: find a quadratic function $f(t) = a_2t^2 + a_1t + a_0$ which satisfies $f(n) = z(n)$, $f(n-1) = z(n-1)$, and $f(n-2) = z(n-2)$. The extrapolated value is then given by $y = f(n+1)$. Let the vector x denote the three previous elements of the series,

$$x = \begin{bmatrix} z(n) \\ z(n-1) \\ z(n-2) \end{bmatrix}.$$

You'll find a vector $c \in \mathbf{R}^3$ such that the extrapolated value is given by the linear transformation $y = c^T x$. You'll do this in two different ways.

(a) *With matrix inversion.* Without loss of generality, make the problem independent of n by writing the extrapolating function as

$$f(n+k) = u_1 k^2 + u_2 k + u_3.$$

With the condition that f fits the series for $k = 0$, $k = -1$, and $k = -2$, find the matrix $A \in \mathbf{R}^{3\times 3}$ that relates x to the vector u (of the coefficients of f),

$$x = Au.$$

Find the vector $b \in \mathbf{R}^3$ such that

$$y = b^T u.$$

Compute the inverse of A, and find c by eliminating u from the two previous equations.

(b) *With basis functions.* Assume first that $x = e_1 = (1, 0, 0)$, and find y (*i.e.*, fit a quadratic to x and find its value at $n + 1$.) Repeat this for $x = e_2 = (0, 1, 0)$, and for $x = e_3 = (0, 0, 1)$. Find the extrapolated value y for an arbitrary x as a linear combination of the three previous values.

3.13 *Right inverses.* This problem concerns the specific matrix

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

This matrix is full rank (*i.e.*, its rank is 3), so there exists at least one right inverse. In fact, there are many right inverses of A, which opens the possibility that we can seek right inverses that in addition have other properties. For each of the cases below, either find a specific matrix $B \in \mathbf{R}^{5\times 3}$ that satisfies $AB = I$ and the given property, or explain why there is no such B. In cases where there is a right inverse B with the required property, you must briefly explain how you found your B. You must also attach a printout of some Matlab scripts that show the verification that $AB = I$. (We'll be very angry if we have to type in your 5×3 matrix into Matlab to check it.) When there is no right inverse with the given property, briefly explain why there is no such B.

(a) The second row of B is zero.

(b) The nullspace of B has dimension one.

(c) The third column of B is zero.

(d) The second and third rows of B are the same.

(e) B is upper triangular, *i.e.*, $B_{ij} = 0$ for $i > j$.

(f) B is lower triangular, *i.e.*, $B_{ij} = 0$ for $i < j$.

3.14 *Nonlinear unbiased estimators.* We consider the standard measurement setup:

$$y = Ax + v,$$

where $A \in \mathbf{R}^{m\times n}$, $x \in \mathbf{R}^n$ is the vector of parameters we wish to estimate, $y \in \mathbf{R}^m$ is the vector of measurements we take, and $v \in \mathbf{R}^m$ is the vector of measurement errors and noise. You may not assume anything about the dimensions of A, its rank, nullspace, etc. If the function $f : \mathbf{R}^m \to \mathbf{R}^n$ satisfies $f(Ax) = x$ for all $x \in \mathbf{R}^n$, then we say that f is an *unbiased estimator* (of x, given y). What this means is that if f is applied to our measurement vector, and $v = 0$, then f returns the true parameter value x. In EE263 we have studied linear unbiased estimators, which are unbiased estimators that are also linear functions. Here, though, we allow the possibility that f is nonlinear (which we take to mean, f is not linear). One of the following statements is true. Pick the statement that is true, and justify it completely. You can quote any result given in the lecture notes.

A. *There is no such thing as a nonlinear unbiased estimator.* In other words, if f is any unbiased estimator, then f must be a linear function. (This statement is taken to be true if there are no unbiased estimators for a particular A.) If you believe this statement, explain why.

B. *Nonlinear unbiased estimators do exist, but you don't need them.* In other words: it's possible to have a nonlinear unbiased estimator. But whenever there is a nonlinear unbiased estimator, there is also a linear unbiased estimator. If you believe this statement, then give a specific example of a matrix A, and an unbiased nonlinear estimator. Explain in the general case why a linear unbiased estimator exists whenever there is a nonlinear one.

C. *There are cases for which nonlinear unbiased estimators exist, but no linear unbiased estimator exists.* If you believe this statement, give a specific example of a matrix A, and a nonlinear unbiased estimator, and also explain why no linear unbiased estimator exists.

3.15 *Channel equalizer with disturbance rejection.* A communication channel is described by $y = Ax + v$ where $x \in \mathbf{R}^n$ is the (unknown) transmitted signal, $y \in \mathbf{R}^m$ is the (known) received signal, $v \in \mathbf{R}^m$ is the (unknown) disturbance signal, and $A \in \mathbf{R}^{m\times n}$ describes the (known) channel. The disturbance v is known to be a linear combination of some (known) disturbance patterns,

$$d_1, \ldots, d_k \in \mathbf{R}^m.$$

We consider linear equalizers for the channel, which have the form $\hat{x} = By$, where $B \in \mathbf{R}^{n\times m}$. (We'll refer to the matrix B as the equalizer; more precisely, you might say that B_{ij} are the equalizer coefficients.) We say the equalizer B *rejects* the disturbance pattern d_i if $\hat{x} = x$, no matter what x is, when $v = d_i$. If the equalizer rejects a set of disturbance patterns, for example, disturbances d_1, d_3, and d_7 (say), then it can reconstruct the transmitted signal exactly, when the disturbance v is any linear combination of d_1, d_3, and d_7. Here is the problem. For the problem data given in `cedr_data.m`, find an equalizer B that rejects as many disturbance patterns as possible. (The disturbance patterns are given as an $m \times k$ matrix `D`, whose columns are the individual disturbance patterns.) Give the specific set of disturbance patterns that your equalizer rejects, as in 'My equalizer rejects three disturbance patterns: d_2, d_3, and d_6.' (We only need *one* set of disturbances of the maximum size.) Explain how you know that there is no equalizer that rejects more disturbance patterns than yours does. Show the Matlab verification that your B does indeed reconstruct x, and rejects the disturbance patterns you claim it does. Show any other calculations needed to verify that your equalizer rejects the maximum number of patterns possible.

3.16 *Identifying a point on the unit sphere from spherical distances.* In this problem we consider the *unit sphere* in $\mathbf{R}^n$, which is defined as the set of vectors with norm one: $S^n = \{x \in \mathbf{R}^n \mid \|x\| = 1\}$. We define the *spherical distance* between two vectors on the unit sphere as the distance between them, measured along the sphere, *i.e.*, as the angle between the vectors, measured in radians: If $x,\ y \in S^n$, the spherical distance between them is

$$\mathrm{sphdist}(x, y) = \angle(x, y),$$

where we take the angle as lying between 0 and π. (Thus, the maximum distance between two points in S^n is π, which occurs only when the two points x, y are *antipodal*, which means $x = -y$.) Now suppose $p_1, \ldots, p_k \in S^n$ are the (known) positions of some beacons on the unit sphere, and let $x \in S^n$ be an unknown point on the unit sphere. We have exact measurements of the (spherical) distances between each beacon and the unknown point x, *i.e.*, we are given the numbers

$$\rho_i = \mathrm{sphdist}(x, p_i), \quad i = 1, \ldots, k.$$

We would like to determine, without any ambiguity, the exact position of x, based on this information. Find the conditions on $p_1, \ldots, p_k$ under which we can unambiguously determine x, for any $x \in S^n$, given the distances ρ_i. You can give your solution algebraically, using any of the concepts used in class (*e.g.*, nullspace, range, rank), or you can give a geometric condition (involving the vectors p_i). You must justify your answer.

3.17 *Some true/false questions.* Determine if the following statements are true or false. No justification or discussion is needed for your answers. What we mean by "true" is that the statement is true for all values of the matrices and vectors given. You can't assume anything about the dimensions of the matrices (unless it's explicitly stated), but you can assume that the dimensions are such that all expressions make sense. For example, the statement "$A + B = B + A$" is true, because no matter what the dimensions of A and B (which must, however, be the same), and no matter what values A and B have, the statement holds. As another example, the statement $A^2 = A$ is false, because there are (square) matrices for which this doesn't hold. (There are also matrices for which it does hold, *e.g.*, an identity matrix. But that doesn't make the statement true.)

a. If all coefficients (*i.e.*, entries) of the matrix A are positive, then A is full rank.

b. If A and B are onto, then $A + B$ must be onto.

c. If A and B are onto, then so is the matrix $\begin{bmatrix} A & C \\ 0 & B \end{bmatrix}$.

d. If A and B are onto, then so is the matrix $\left[\begin{array}{c} A \\ B \end{array}\right]$.

e. If the matrix $\left[\begin{array}{c} A \\ B \end{array}\right]$ is onto, then so are the matrices A and B.

f. If A is full rank and skinny, then so is the matrix $\left[\begin{array}{c} A \\ B \end{array}\right]$.

3.18 *Some true/false questions.* Determine if the following statements are true or false. What we mean by "true" is that the statement is true for all values of the matrices and vectors given. (You can assume the entries of the matrices and vectors are all real.) You can't assume anything about the dimensions of the matrices (unless it's explicitly stated), but you can assume that the dimensions are such that all expressions make sense. For example, the statement "$A+B = B+A$" is true, because no matter what the dimensions of A and B (which must, however, be the same), and no matter what values A and B have, the statement holds. As another example, the statement $A^2 = A$ is false, because there are (square) matrices for which this doesn't hold. (There are also matrices for which it does hold, *e.g.*, an identity matrix. But that doesn't make the statement true.)

(a) If all coefficients (*i.e.*, entries) of the matrices A and B are nonnegative, and both A and B are onto, then $A + B$ is onto.

(b) $\mathcal{N}\left(\left[\begin{array}{c} A \\ A+B \\ A+B+C \end{array}\right]\right) = \mathcal{N}(A) \cap \mathcal{N}(B) \cap \mathcal{N}(C)$.

(c) $\mathcal{N}\left(\left[\begin{array}{c} A \\ AB \\ ABC \end{array}\right]\right) = \mathcal{N}(A) \cap \mathcal{N}(B) \cap \mathcal{N}(C)$.

(d) $\mathcal{N}(B^T A^T AB + B^T B) = \mathcal{N}(B)$.

(e) If $\left[\begin{array}{cc} A & 0 \\ 0 & B \end{array}\right]$ is full rank, then so are the matrices A and B.

(f) If $\left[\begin{array}{cc} A & 0 \end{array}\right]$ is onto, then A is full rank.

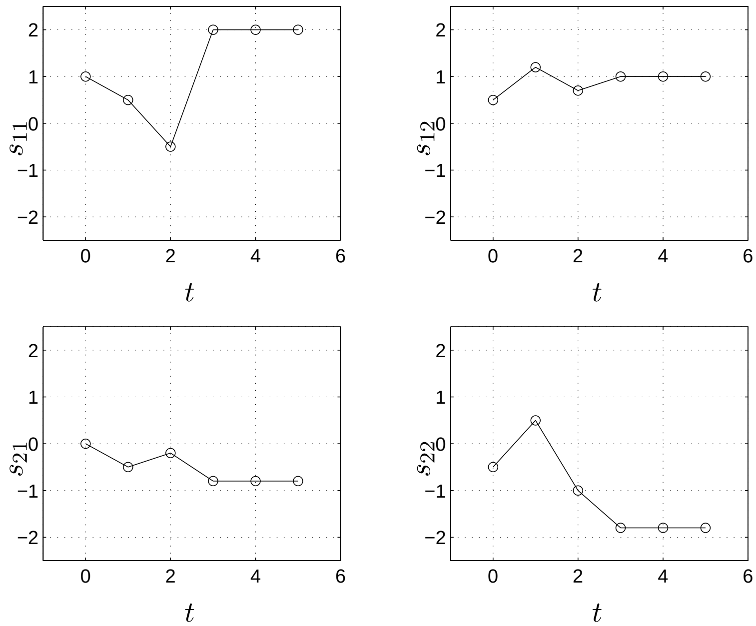
(g) If A^2 is onto, then A is onto.

(h) If $A^T A$ is onto, then A is onto.

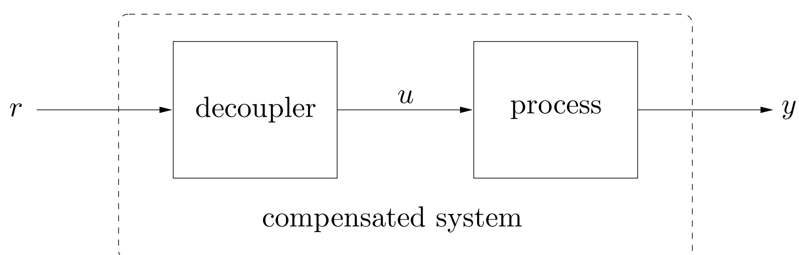
(i) Suppose $u_1, \ldots, u_k \in \mathbf{R}^n$ are nonzero vectors such that $u_i^T u_j \geq 0$ for all i, j. Then the vectors are *nonnegative independent*, which means if $\alpha_1, \ldots, \alpha_k \in \mathbf{R}$ are nonnegative scalars, and $\sum_{i=1}^k \alpha_i u_i = 0$, then $\alpha_i = 0$ for $i = 1, \ldots, k$.

(j) Suppose $A \in \mathbf{R}^{n \times k}$ and $B \in \mathbf{R}^{n \times m}$ are skinny, full rank matrices that satisfy $A^T B = 0$. Then $[A\ B]$ is skinny and full rank.

3.19 *Dynamic decoupling.* An industrial process is described by a 2-input 2-output discrete-time LDS with *finite impulse response* of length 4, which means that its impulse matrix h is nonzero only for $t = 0, 1, 2, 3$; $h(t) = 0$ for $t \geq 4$. This means that its step response matrix, defined as $s(t) = \sum_{\tau=0}^t h(\tau)$, converges to its final value by $t = 3$. If you want to think of this system in concrete terms, you can imagine it as a chemical process reactor, with u_1 a heater input, u_2 as a reactant flow rate, y_1 as the reactor temperature, and y_2 as the reactor pressure. The step response matrix of the system is shown below. The impulse response matrix of the system (for $t = 0, 1, 2, 3$) can be obtained from the class web page in `dynamic_dec_h.m`, where you will find the impulse response matrices `h0`, `h1`, `h2`, `h3` $\in \mathbf{R}^{2x2}$ (which represent $h(0), h(1), h(2), h(3)$).



The plots show that u_1 has a substantial effect on y_2, and that u_2 has a substantial effect on y_1, neither of which we want. To eliminate them, you will explore the design of a *dynamic decoupler* for this system, which is another 2-input, 2-output LDS with impulse matrix g. The decoupler is also FIR of length 4: $g(0), g(1), g(2), g(3) \in \mathbf{R}^{2\times 2}$ can be nonzero, but $g(t) = 0$ for $t \geq 4$. The decoupler is used as a prefilter for the process: the input r (which is called the reference or command input) is applied as the input to the decoupler, and the output of the decoupler is u, the input to the industrial process. This is shown below.



We refer to this cascaded system as the *compensated system*. Let $\tilde{s}$ denote the step response matrix of the compensated system, from input r to output y. The goal is to design the decoupler (*i.e.*, choose $g(0), g(1), g(2), g(3) \in \mathbf{R}^{2\times 2}$) so that the compensated system satisfies the following specifications.

- $\lim_{t\to\infty} \tilde{s}(t) = I$. This means if the reference input is constant, the process output converges to the reference input.
- The off-diagonal entries of $\tilde{s}(t)$ are all zero (for all t). This means the compensated system is *decoupled*: r_1 has no effect on y_2, and r_2 has no effect on y_1.

Find such a decoupler, and plot the compensated system step response matrix. If there is no such decoupler (*i.e.*, the problem specifications are not feasible), say so, and explain why. If there are many decouplers that satisfy the given specifications, say so, and do something sensible with any extra degrees of freedom you may have.

Lecture 4 – Orthonormal sets of vectors and QR factorization

4.1 *Bessel's inequality.* Suppose the columns of $U \in \mathbf{R}^{n\times k}$ are orthonormal. Show that $\|U^T x\| \leq \|x\|$. When do we have $\|U^T x\| = \|x\|$?

4.2 *Orthogonal matrices.*

(a) Show that if U and V are orthogonal, then so is UV.

(b) Show that if U is orthogonal, then so is U^{-1}.

(c) Suppose that $U \in \mathbf{R}^{2\times 2}$ is orthogonal. Show that U is either a rotation or a reflection. Make clear how you decide whether a given orthogonal U is a rotation or reflection.

4.3 *Projection matrices.* A matrix $P \in \mathbf{R}^{n\times n}$ is called a *projection matrix* if $P = P^T$ and $P^2 = P$.

(a) Show that if P is a projection matrix then so is $I - P$.

(b) Suppose that the columns of $U \in \mathbf{R}^{n\times k}$ are orthonormal. Show that UU^T is a projection matrix. (Later we will show that the converse is true: every projection matrix can be expressed as UU^T for some U with orthonormal columns.)

(c) Suppose $A \in \mathbf{R}^{n\times k}$ is full rank, with $k \leq n$. Show that $A(A^TA)^{-1}A^T$ is a projection matrix.

(d) If $S \subseteq \mathbf{R}^n$ and $x \in \mathbf{R}^n$, the point y in S closest to x is called the *projection of x on S*. Show that if P is a projection matrix, then $y = Px$ is the projection of x on $\mathcal{R}(P)$. (Which is why such matrices are called projection matrices . . .)

4.4 *Reflection through a hyperplane.* Find the matrix $R \in \mathbf{R}^{n\times n}$ such that reflection of x through the hyperplane $\{z | a^T z = 0\}$ is given by Rx. Verify that the matrix R is orthogonal. (To reflect x through the hyperplane means the following: find the point z on the hyperplane closest to x. Starting from x, go in the direction $z - x$ through the hyperplane to a point on the opposite side, which has the same distance to z as x does.)

4.5 *Sensor integrity monitor.* A suite of m sensors yields measurement $y \in \mathbf{R}^m$ of some vector of parameters $x \in \mathbf{R}^n$. When the system is operating normally (which we hope is almost always the case) we have $y = Ax$, where $m > n$. If the system or sensors fail, or become faulty, then we no longer have the relation $y = Ax$. We can exploit the redundancy in our measurements to help us identify whether such a fault has occured. We'll call a measurement y *consistent* if it has the form Ax for some $x \in \mathbf{R}^n$. If the system is operating normally then our measurement will, of course, be consistent. If the system becomes faulty, we hope that the resulting measurement y will become inconsistent, *i.e.*, not consistent. (If we are *really* unlucky, the system will fail in such a way that y is still consistent. Then we're out of luck.) A matrix $B \in \mathbf{R}^{k\times m}$ is called an *integrity monitor* if the following holds:

- $By = 0$ for any y which is consistent.
- $By \neq 0$ for any y which is inconsistent.

If we find such a matrix B, we can quickly check whether y is consistent; we can send an alarm if $By \neq 0$. Note that the first requirement says that every consistent y does not trip the alarm; the second requirement states that every inconsistent y does trip the alarm. Finally, the problem. Find an integrity monitor B for the matrix

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 1 & -1 & -2 \\ -2 & 1 & 3 \\ 1 & -1 & -2 \\ 1 & 1 & 0 \end{bmatrix}.$$

Your B should have the smallest k (*i.e.*, number of rows) as possible. As usual, you have to explain what you're doing, as well as giving us your explicit matrix B. You must also verify that the matrix you choose satisfies the requirements. *Hints:*

- You might find one or more of the Matlab commands `orth`, `null`, or `qr` useful. Then again, you might not; there are many ways to find such a B.
- When checking that your B works, don't expect to have By exactly zero for a consistent y; because of roundoff errors in computer arithmetic, it will be really, really small. That's OK.
- Be very careful typing in the matrix A. It's not just a random matrix.

4.6 *Householder reflections.* A *Householder matrix* is defined as

$$Q = I - 2uu^T,$$

where $u \in \mathbf{R}^n$ is normalized, that is, $u^T u = 1$.

(a) Show that Q is orthogonal.

(b) Show that $Qu = -u$. Show that $Qv = v$, for any v such that $u^T v = 0$. Thus, multiplication by Q gives reflection through the plane with normal vector u.

(c) Show that $\det Q = -1$.

(d) Given a vector $x \in \mathbf{R}^n$, find a unit-length vector u for which

$$Qx = (I - 2uu^T)x \in \operatorname{span}\{e_1\},$$

where $e_1 = [1\ 0\ \ldots\ 0]^T$. *Hint:* Try a u of the form $u = v/\|v\|$, with $v = x + \alpha e_1$ (find the appropriate α and show that such a u works ...) Compute such a u for $x = [3\ 2\ 4\ 1\ 5]^T$. Apply the corresponding Householder reflection to x: what is Qx?

Note: Multiplication by an orthogonal matrix has very good numerical properties, in the sense that it does not accumulate much roundoff error. For this reason, Householder reflections are used as building blocks for fast, numerically sound algorithms. They are used, for example, in algorithms for the QR and SVD decompositions that progressively introduce zeros in the non-diagonal entries of a matrix.

4.7 *Minimum distance and maximum correlation decoding.* We consider a simple communication system, in which a sender transmits one of N possible signals to a receiver, which receives a version of the signal sent that is corrupted by noise. Based on the corrupted received signal, the receiver has to estimate or guess which of the N signals was sent. We will represent the signals by vectors in $\mathbf{R}^n$. We will denote the possible signals as $a_1, \ldots, a_N \in \mathbf{R}^n$. These signals, which collectively are called the *signal constellation*, are known to both the transmitter and receiver. When the signal a_k is sent, the received signal is $a_{\text{recd}} = a_k + v$, where $v \in \mathbf{R}^n$ is (channel or transmission) noise. In a communications course, the noise v is described by a statistical model, but here we'll just assume that it is 'small' (and in any case, it does not matter for the problem). The receiver must make a guess or estimate as to which of the signals was sent, based on the received signal a_{recd}. There are many ways to do this, but in this problem we explore two methods.

- *Minimum distance decoding.* Choose as the estimate of the decoded signal the one in the constellation that is closest to what is received, *i.e.*, choose a_k that minimizes $\|a_{\text{recd}} - a_i\|$. For example, if we have $N = 3$ and

$$\|a_{\text{recd}} - a_1\| = 2.2, \qquad \|a_{\text{recd}} - a_2\| = 0.3, \qquad \|a_{\text{recd}} - a_3\| = 1.1,$$

then the minimum distance decoder would guess that the signal a_2 was sent.

- *Maximum correlation decoding.* Choose as the estimate of the decoded signal the one in the constellation that has the largest inner product with the received signal, *i.e.*, choose a_k that maximizes $a_{\text{recd}}^T a_i$. For example, if we have $N = 3$ and

$$a_{\text{recd}}^T a_1 = -1.1, \qquad a_{\text{recd}}^T a_2 = 0.2, \qquad a_{\text{recd}}^T a_3 = 1.0,$$

then the maximum correlation decoder would guess that the signal a_3 was sent.

For both methods, let's not worry about breaking ties. You can just assume that ties never occur; one of the signals is always closest to, or has maximum inner product with, the received signal. Give some general conditions on the constellation (*i.e.*, the set of vectors $a_1, \ldots, a_N$) under which these two decoding methods are the same. By 'same' we mean this: for any received signal a_{recd}, the decoded signal for the two methods is the same. Give the simplest condition you can. You can refer to any of the concepts from the course, *e.g.*, range, nullspace, independence, norms, QR factorization, etc. You must show how the decoding schemes always give the same answer, when your conditions hold. Also, give a specific counterexample, for which your conditions don't hold, and the methods differ. (We are *not* asking you to show that when your conditions don't hold, the two decoding schemes differ for some received signal.) You might want to check simple cases like $n = 1$ (scalar signals), $N = 2$ (only two messages in the constellation), or draw some pictures. But then again, you might not.

4.8 *Finding a basis for the intersection of ranges.*

(a) Suppose you are given two matrices, $A \in \mathbf{R}^{n \times p}$ and $B \in \mathbf{R}^{n \times q}$. Explain how you can find a matrix $C \in \mathbf{R}^{n \times r}$, with independent columns, for which

$$\text{range}(C) = \text{range}(A) \cap \text{range}(B).$$

This means that the columns of C are a basis for $\text{range}(A) \cap \text{range}(B)$. Your solution can involve any standard methods or decompositions we have studied, *e.g.*, compact and full SVD, QR factorization, pseudo-inverse, eigenvalue decomposition, and so on.

(b) Carry out the method described in part (a) for the particular matrices A and B defined in `intersect_range_data.m` on the course web site. Be sure to give us your matrix C, as well as the Matlab (or other) code that generated it. Verify that range$(C) \subseteq$ range(A) and range$(C) \subseteq$ range(B), by showing that each column of C is in the range of A, and also in the range of B.

Please carefully separate your answers to part (a) (the general case) and part (b) (the specific case). As always, we will deduct points in part (b) if your answer is correct, but we don't understand your method, or if your method is too complicated.

4.9 *Oh no. It's the dreaded theory problem.* In the list below there are 11 statements about two square matrices A and B in $\mathbf{R}^{n\times n}$.

(a) $\mathcal{R}(B) \subseteq \mathcal{R}(A)$.

(b) there exists a matrix $Y \in \mathbf{R}^{n\times n}$ such that $B = YA$.

(c) $AB = 0$.

(d) $BA = 0$.

(e) $\mathbf{rank}([\ A \quad B\ ]) = \mathbf{rank}(A)$.

(f) $\mathcal{R}(A) \perp \mathcal{N}(B^T)$.

(g) $\mathbf{rank}(\begin{bmatrix} A \\ B \end{bmatrix}) = \mathbf{rank}(A)$.

(h) $\mathcal{R}(A) \subseteq \mathcal{N}(B)$.

(i) there exists a matrix $Z \in \mathbf{R}^{n\times n}$ such that $B = AZ$.

(j) $\mathbf{rank}([\ A \quad B\ ]) = \mathbf{rank}(B)$.

(k) $\mathcal{N}(A) \subseteq \mathcal{N}(B)$.

Your job is to collect them into (the largest possible) groups of equivalent statements. Two statements are equivalent if each one implies the other. For example, the statement 'A is onto' is equivalent to '$\mathcal{N}(A) = \{0\}$' (when A is square, which we assume here), because every square matrix that is onto has zero nullspace, and vice versa. Two statements are not equivalent if there exist (real) square matrices A and B for which one holds, but the other does not. A group of statements is equivalent if any pair of statements in the group is equivalent.

We want *just* your answer, which will consist of lists of mutually equivalent statements. We will not read any justification. If you add any text to your answer, as in 'c and e are equivalent, provided A is nonsingular', we will mark your response as wrong.

Put your answer in the following specific form. List each group of equivalent statements on a line, in (alphabetic) order. Each new line should start with the first letter not listed above. For example, you might give your answer as

a, c, d, h
b, i
e
f, g, j, k.

This means you believe that statements a, c, d, and h are equivalent; statements b and i are equivalent; and statements f, g, j, and k are equivalent. You also believe that the first group of statements is not equivalent to the second, or the third, and so on. them per

We will take points off for false groupings (*i.e.*, listing statements in the same line when they are not equivalent) as well as for missed groupings (*i.e.*, when you list equivalent statements in different lines).

Lecture 5 – Least-squares

5.1 *Least-squares residuals.* Let the matrix A be skinny and full-rank. Let $x_{\rm ls}$ be the least-squares solution of $Ax = y$, and let $y_{\rm ls} = Ax_{\rm ls}$. Show that the residual vector $r = y - y_{\rm ls}$ satisfies

$$\|r\|^2 = \|y\|^2 - \|y_{\rm ls}\|^2.$$

Also, give a brief geometric interpretation of this equality (just a couple of sentences, and maybe a conceptual drawing).

5.2 *Complex linear algebra and least-squares.* Most of the linear algebra you have seen is unchanged when the scalars, matrices, and vectors are complex, *i.e.*, have complex entries. For example, we say a set of complex vectors $\{v_1, \ldots, v_n\}$ is dependent if there exist complex scalars $\alpha_1, \ldots, \alpha_n$, not all zero, such that $\alpha_1 v_1 + \cdots + \alpha_n v_n = 0$. There are some slight differences when it comes to the inner product and other expressions that, in the real case, involve the transpose operator. For complex matrices (or vectors) we define the *Hermitian conjugate* as the complex conjugate of the transpose. We denote this as A^*, which is equal to $(\overline{A})^T$. Thus, the ij entry of the matrix A^* is given by $\overline{(A_{ji})}$. The Hermitian conjugate of a matrix is sometimes called its *conjugate transpose* (which is a nice, explanatory name). Note that for a real matrix or vector, the Hermitian conjugate is the same as the transpose. We define the inner product of two complex vectors $u,\ v \in \mathbf{C}^n$ as

$$\langle u, v\rangle = u^* v,$$

which, in general, is a complex number. The norm of a complex vector is defined as

$$\|u\| = \sqrt{\langle u, u\rangle} = \left(|u_1|^2 + \cdots + |u_n|^2\right)^{1/2}.$$

Note that these two expressions agree with the definitions you already know when the vectors are real. The complex least-squares problem is to find the $x \in \mathbf{C}^n$ that minimizes $\|Ax - y\|^2$, where $A \in \mathbf{C}^{m\times n}$ and $y \in \mathbf{C}^m$ are given. Assuming A is full rank and skinny, the solution is $x_{\rm ls} = A^\dagger y$, where $A^\dagger$ is the (complex) pseudo-inverse of A, given by

$$A^\dagger = (A^* A)^{-1} A^*.$$

(Which also reduces to the pseudo-inverse you've already seen when A is real). There are two general approaches to dealing with complex linear algebra problems. In the first, you simply generalize all the results to work for complex matrices, vectors, and scalars. Another approach is to represent complex matrices and vectors using real matrices and vectors of twice the dimensions, and then you apply what you already know about real linear algebra. We'll explore that idea in this problem. We associate with a complex vector $u \in \mathbf{C}^n$ a *real* vector $\tilde{u} \in \mathbf{R}^{2n}$, given by

$$\tilde{u} = \begin{bmatrix} \Re u \\ \Im u \end{bmatrix}.$$

We associate with a complex matrix $A \in \mathbf{C}^{m\times n}$ the *real* matrix $\tilde{A} \in \mathbf{R}^{2m\times 2n}$ given by

$$\tilde{A} = \begin{bmatrix} \Re A & -\Im A \\ \Im A & \Re A \end{bmatrix}.$$

(a) What is the relation between $\langle u, v\rangle$ and $\langle \tilde{u}, \tilde{v}\rangle$? Note that the first inner product involves complex vectors and the second involves real vectors.

(b) What is the relation between $\|u\|$ and $\|\tilde{u}\|$?

(c) What is the relation between Au (complex matrix-vector multiplication) and $\tilde{A}\tilde{u}$ (real matrix-vector multiplication)?

(d) What is the relation between $\tilde{A}^T$ and A^*?

(e) Using the results above, verify that $A^\dagger y$ solves the complex least-squares problem of minimizing $\|Ax - y\|$ (where A, x, y are complex). Express $A^\dagger y$ in terms of the real and imaginary parts of A and y. (You don't need to simplify your expression; you can leave block matrices in it.)

Lecture 6 – Least-squares applications

6.1 *Design of a robust heating element power vector.* Consider a heating problem in which x_i denotes the change in power of the ith heating element from some nominal setting (so $x_i < 0$ makes sense), and $y \in \mathbf{R}$ denotes the temperature rise at some critical point. We have n heating elements, so $x \in \mathbf{R}^n$. The process is linear, so $y = a^T x$ for some $a \in \mathbf{R}^n$. In fact, the system is operated in any of k modes, for which the linear function mapping x into y is not quite the same. We model this as follows: in mode i, we have $y = a_i^T x$, where $a_i \in \mathbf{R}^n$, $i = 1, \ldots, k$. The vectors a_i are similar, but not quite the same.

(a) Suppose that $k > n$, *i.e.*, there are more operating modes than heating elements. Explain how to find a single heating element power vector x that results in $y \approx 1$ for each mode. Thus, this x yields about the same temperature rise for any of the operating modes. If some assumptions are necessary, state clearly what they are.

(b) Now suppose that $k < n$, *i.e.*, we have more heating elements than operating modes. Is it possible in this case to find an x that results in $y = 1$ for *every* operating mode? Suggest a reasonable choice for x, carefully stating any assumptions that are necessary.

6.2 *Optimal control of unit mass.* In this problem you will use Matlab to solve an optimal control problem for a force acting on a unit mass. Consider a unit mass at position $p(t)$ with velocity $\dot{p}(t)$, subjected to force $f(t)$, where $f(t) = x_i$ for $i - 1 < t \leq i$, for $i = 1, \ldots, 10$.

(a) Assume the mass has zero initial position and velocity: $p(0) = \dot{p}(0) = 0$. Find x that minimizes

$$\int_{t=0}^{10} f(t)^2 \, dt$$

subject to the following specifications: $p(10) = 1$, $\dot{p}(10) = 0$, and $p(5) = 0$. Plot the optimal force f, and the resulting p and $\dot{p}$. Make sure the specifications are satisfied. Give a short intuitive explanation for what you see.

(b) Assume the mass has initial position $p(0) = 0$ and velocity $\dot{p}(0) = 1$. Our goal is to bring the mass near or to the origin at $t = 10$, at or near rest, *i.e.*, we want

$$J_1 = p(10)^2 + \dot{p}(10)^2,$$

small, while keeping

$$J_2 = \int_{t=0}^{10} f(t)^2 \, dt$$

small, or at least not too large. Plot the optimal trade-off curve between J_2 and J_1. Check that the end points make sense to you. *Hint:* the parameter μ has to cover a very large range, so it usually works better in practice to give it a logarithmic spacing, using, *e.g.*, `logspace` in Matlab. You don't need more than 50 or so points on the trade-off curve.

Your solution to this problem should consist of a clear written narrative that explains what you are doing, and gives formulas symbolically; the Matlab source code you devise to find the numerical answers, along with comments explaining it all; and the final plots produced by Matlab.

6.3 *AR system identification.* In this problem you will use least-squares to develop and validate auto-regressive (AR) models of a system from some input/output (I/O) records. You are given I/O records

$$u(1), \ldots, u(N), \quad y(1), \ldots, y(N),$$

which are the measured input and output of an unknown system. You will use least-squares to find approximate models of the form

$$y(t) = a_0 u(t) + b_1 y(t-1) + \cdots + b_n y(t-n).$$

Specifically you will choose coefficients $a_0, b_1, \ldots, b_n$ that minimize

$$\sum_{t=n+1}^{N} (y(t) - a_0 u(t) - b_1 y(t-1) - \cdots - b_n y(t-n))^2$$

where u, y are the given data record. The squareroot of this quantity is the residual norm (on the model data). Dividing by $\sqrt{\sum_{t=n+1}^{N} y(t)^2}$ yields the relative error. You'll plot this as a function of n for $n = 1, \ldots, 35$. To validate or evaluate your models, you can try them on validation data records

$$\tilde{u}(1), \ldots, \tilde{u}(N), \quad \tilde{y}(1), \ldots, \tilde{y}(N).$$

To find the predictive ability of an AR model with coefficients $a_0, b_1, \ldots, b_n$, you can form the signal

$$\hat{y}(t) = a_0 \tilde{u}(t) + b_1 \tilde{y}(t-1) + \cdots + b_n \tilde{y}(t-n)$$

for $t = n+1, \ldots, N$, and compare it to the actual output signal, $\tilde{y}$. You will plot the squareroot of the sum of squares of the difference, divided by the squareroot of the sum of squares of $\tilde{y}$, for $n = 1, \ldots, 35$. Compare this to the plot above. Briefly discuss the results. To develop the models for different values of n, you can use inefficient code that just loops over n; you do not have to try to use an efficient method based on one QR factorization. The file `IOdata.m` contains the data for this problem and is available on the class web page. The `toeplitz()` command may be helpful.

6.4 *Least-squares model fitting.* In this problem you will use least-squares to fit several different types of models to a given set of input/output data. The data consists of a scalar input sequence u, and a scalar output sequence y, for $t = 1, \ldots, N$. You will develop several different models that relate the signals u and y. The models range in complexity from a simple constant to a nonlinear dynamic model.

- *Memoryless models.* 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.*
  - constant model: $y(t) = a$
  - static linear: $y(t) = bu(t)$
  - static affine: $y(t) = a + bu(t)$
  - static quadratic: $y(t) = a + bu(t) + cu(t)^2$

- *Dynamic models.* In a dynamic model, $y(t)$ depends on $u(s)$ for some $s \neq t$. We consider simple dynamic models in which $y(t)$ depends on $u(t)$ and $u(t-1)$; in other words the current output depends on the current input and the previous input. Such models are said to have a *finite memory* of one sample. Another term is 2-tap system (the taps refer to taps on a delay line).
  - linear, 2-tap: $y(t) = bu(t) + du(t-1)$
  - affine, 2-tap: $y(t) = a + bu(t) + du(t-1)$
  - quadratic, 2-tap: $y(t) = a + bu(t) + cu(t)^2 + du(t-1) + eu(t-1)^2 + fu(t)u(t-1)$

Each of these models is specified by its parameters, *i.e.*, the scalars $a, b, \ldots, f$. For each of these models, find the least-squares fit to the given data. In other words, find parameter values that minimize the sum-of-squares of the residuals. For example, for the affine 2-tap model, pick a, b, and d that minimize

$$\sum_{t=2}^{N} \left(y(t) - a - bu(t) - du(t-1)\right)^2 .$$

(Note that we start the sum at $t = 2$ so $u(t-1)$ is defined.) For each model, give the root-mean-square (RMS) residual, *i.e.*, the squareroot of the mean of the optimal residual squared. Plot the output $\hat{y}$ predicted by your model, and plot the residual (which is $y - \hat{y}$). The data for this problem are available in an m-file named `model_fitting_data.m`. Running `model_fitting_data` in Matlab will create the vectors u and y and the length N. Now you can plot u, y, etc. *Note:* the dataset u, y are *not* generated by any of the models above. It is generated by a nonlinear recursion, which has infinite (but rapidly fading) memory.

6.5 *Interpolation with rational functions.* In this problem we consider a function $f : \mathbf{R} \to \mathbf{R}$ of the form

$$f(x) = \frac{a_0 + a_1 x + \cdots + a_m x^m}{1 + b_1 x + \cdots + b_m x^m},$$

where $a_0, \ldots, a_m$, and $b_1, \ldots, b_m$ are parameters, with either $a_m \neq 0$ or $b_m \neq 0$. Such a function is called a *rational function of degree* m. We are given data points $x_1, \ldots, x_N \in \mathbf{R}$ and $y_1, \ldots, y_N \in \mathbf{R}$, where $y_i = f(x_i)$. The problem is to find a rational function of smallest degree that is consistent with this data. In other words, you are to find m, which should be as small as possible, and $a_0, \ldots, a_m$, $b_1, \ldots, b_m$, which satisfy $f(x_i) = y_i$. Explain how you will solve this problem, and then carry out your method on the problem data given in `ri_data.m`. (This contains two vectors, `x` and `y`, that give the values $x_1, \ldots, x_N$, and $y_1, \ldots, y_N$, respectively.) Give the value of m you find, and the coefficients $a_0, \ldots, a_m$, $b_1, \ldots, b_m$. Please show us your verification that $y_i = f(x_i)$ holds (possibly with some small numerical errors).

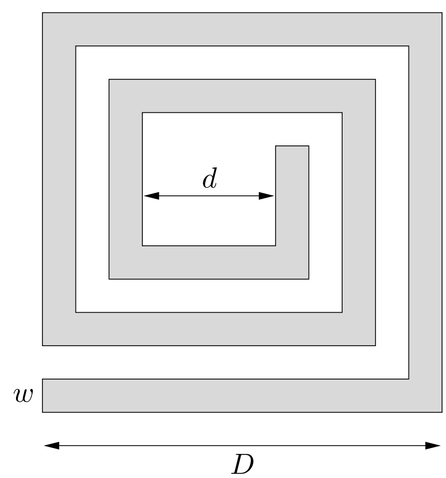
6.6 *The middle inverse.* In this problem we consider the matrix equation

$$AXB = I,$$

where $A \in \mathbf{R}^{n \times p}$, $B \in \mathbf{R}^{q \times n}$, and $X \in \mathbf{R}^{p \times q}$. The matrices A and B are given, and the goal is to find a matrix X that satisfies the equation, or to determine that no such matrix exists. (When such an X exists, we call it a *middle inverse* of the pair A, B. It generalizes the notions of left-inverse and right-inverse: When $A = I$, X is a left-inverse of B, and when $B = I$, X

is a right-inverse of A.) You will solve a specific instance of this problem, with data (*i.e.*, the matrices A and B) given in the mfile `axb_data.m`. If you think there is no X that satisfies $AXB = I$, explain why this is the case. Your explanation should be as concrete as possible. If you succeed in finding an X that satisfies $AXB = I$, please give it. You must explain how you found it, and you must submit the code that you used to generate your solution. You must also submit the Matlab code and output showing that you checked that $AXB = I$ holds (up to very small numerical errors). You can do this by typing `norm(A*X*B-eye(n))` in Matlab, and submitting a printout of what Matlab prints out. (We haven't yet studied the matrix norm, but it doesn't matter. Like the norm of a vector, it measures size, and here you are using it only to check that $AXB - I$ is small.) *The following interpretation is not needed to solve the problem.* We give it just to mention a concrete situation where a problem like this one might arise. One situation where this problem comes up is a nonstandard filtering or equalization problem. A vector signal $x \in \mathbf{R}^n$ is first processed by one channel, represented by B. At this point the signal is available for some filtering or processing by us, which is represented by the matrix X. After this processing, it is acted on by another channel, given by A. Our goal is to do the intermediate processing in such a way that it undoes the effect of the first and last channels.

6.7 *Approximate inductance formula.* The figure below 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. The inductance L 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 `inductor_data.m` on the course web site contains data for 50 inductors. (The data is real, not that it would affect how you solve the problem ...) For inductor i, we give parameters n_i, w_i, d_i, and D_i (all in μm), and also, the inductance L_i (in nH) obtained from measurements. (The data are organized as vectors of length 50. Thus, for example, w_{13} gives the wire width of inductor 13.) Your task, *i.e.*, the problem, is to find α, $\beta_1, \ldots, \beta_4$ so that

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

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 $\hat{L}_i$ and L_i); we leave that to your engineering judgment. But be sure to tell us what criterion you use. We define the *percentage error* between $\hat{L}_i$ and L_i as

$$e_i = 100|\hat{L}_i - L_i|/L_i.$$

Find the average percentage error for your model, *i.e.*, $(e_1 + \cdots + e_{50})/50$. (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.) *Hint:* you might find it easier to work with $\log L$.

6.8 *Quadratic extrapolation of a time series, using least-squares fit.* This is an extension of problem 12, which concerns quadratic extrapolation of a time series. We are given a series z up to time n. Using a quadratic model, we want to extrapolate, or predict, $z(n+1)$ based on a least-squares fit to the previous ten elements of the series: $z(n), z(n-1), \ldots, z(n-9)$. We'll denote the predicted value of $z(n+1)$ by y. Another way to describe this problem is: find a quadratic function $f(t) = a_2 t^2 + a_1 t + a_0$ for which $f(n) \approx z(n)$, $f(n-1) \approx z(n-1), \ldots, f(n-9) \approx z(n-9)$. Specifically, the approximation should be such that

$$\sum_{k=-9}^{0} \left(z(n+k) - f(n+k)\right)^2$$

is minimized. The extrapolated value is then given by $y = f(n+1)$. Let the vector x denote the ten previous elements of the series,

$$x = \begin{bmatrix} z(n) \\ z(n-1) \\ \vdots \\ z(n-9) \end{bmatrix}.$$

Find a vector $c \in \mathbf{R}^{10}$ such that the extrapolated value is given by the linear transformation $y = c^T x$.

6.9 *Image reconstruction from line integrals.* In this problem we explore a simple version of a tomography problem. We consider a square region, which we divide into an $n \times n$ array of square pixels, as shown in Figure 1. The pixels are indexed column first, by a single index

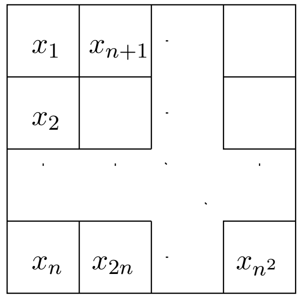


Figure 1: The square region and its division into pixels

i ranging from 1 to n^2, as shown in Figure 1. We are interested in some physical property such as density (say) which varies over the region. To simplify things, we'll assume that the density is constant inside each pixel, and we denote by x_i the density in pixel i, $i = 1, \dots, n^2$. Thus, $x \in \mathbf{R}^{n^2}$ is a vector that describes the density across the rectangular array of pixels. The problem is to estimate the vector of densities x, from a set of sensor measurements that we now describe. Each sensor measurement is a *line integral* of the density over a line L. In addition, each measurement is corrupted by a (small) noise term. In other words, the sensor measurement for line L is given by

$$\sum_{i=1}^{n^2} l_i x_i + v,$$

where l_i is the length of the intersection of line L with pixel i (or zero if they don't intersect), and v is a (small) measurement noise. Figure 2 gives an example, with a graphical explanation of l_i. Now suppose we have N line integral measurements, associated with lines $L_1, \dots, L_N$.

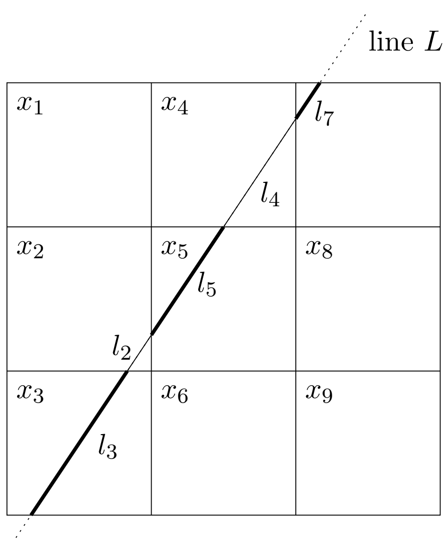


Figure 2: An example of a 3-by-3 pixel patch, with a line L and its intersections l_i with the pixels. Clearly, $l_1 = l_6 = l_8 = l_9 = 0$.

From these measurements, we want to estimate the vector of densities x. The lines are characterized by the intersection lengths

$$l_{ij}, \quad i = 1, \ldots, n^2, \quad j = 1, \ldots, N,$$

where l_{ij} gives the length of the intersection of line L_j with pixel i. Then, the whole set of measurements forms a vector $y \in \mathbf{R}^N$ whose elements are given by

$$y_j = \sum_{i=1}^{n^2} l_{ij} x_i + v_j, \quad j = 1, \ldots, N.$$

And now the problem: you will reconstruct the pixel densities x from the line integral measurements y. The class webpage contains the M-file `tomodata.m`, which you should download and run in Matlab. It creates the following variables:

- `N`, the number of measurements (N),
- `n_pixels`, the side length in pixels of the square region (n),
- `y`, a vector with the line integrals y_j, $j = 1, \ldots, N$,
- `lines_d`, a vector containing the displacement d_j, $j = 1, \ldots, N$, (distance from the center of the region in pixels lengths) of each line, and
- `lines_theta`, a vector containing the angles θ_j, $j = 1, \ldots, N$, of each line.

The file `tmeasure.m`, on the same webpage, shows how the measurements were computed, in case you're curious. You should take a look, but you don't need to understand it to solve the problem. We also provide the function `line_pixel_length.m` on the webpage, which you do need to use in order to solve the problem. This function computes the pixel intersection lengths for a given line. That is, given d_j and θ_j (and the side length n), `line_pixel_length.m` returns a $n \times n$ matrix, whose i, jth element corresponds to the intersection length for pixel i, j on the image. Use this information to find x, and display it as an image (of n by n pixels). You'll know you have it right when the image of x forms a familiar acronym... *Matlab hints:* Here are a few functions that you'll find useful to display an image:

- `A=reshape(v,n,m)`, converts the vector `v` (which must have `n*m` elements) into an $n \times m$ matrix (the first column of `A` is the first n elements of `v`, *etc.*),
- `imagesc(A)`, displays the matrix `A` as an image, scaled so that its lowest value is black and its highest value is white,
- `colormap gray`, changes Matlab's image display mode to grayscaled (you'll want to do this to view the pixel patch),
- `axis image`, redefines the axes of a plot to make the pixels square.

Note: While irrelevant to your solution, this is actually a simple version of *tomography*, best known for its application in medical imaging as the CAT scan. If an *x-ray* gets attenuated at rate x_i in pixel i (a little piece of a cross-section of your body), the j-th measurement is

$$z_j = \prod_{i=1}^{n^2} e^{-x_i l_{ij}},$$

with the l_{ij} as before. Now define $y_j = -\log z_j$, and we get

$$y_j = \sum_{i=1}^{n^2} x_i l_{ij}.$$

6.10 *Least-squares model fitting.* In this problem you will use least-squares to fit several different types of models to a given set of input/output data. The data consist of a scalar input sequence u, and a scalar output sequence y, for $t = 1, \ldots, N$. You will develop several different models that relate the signals u and y.

- *Memoryless models.* 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.*

constant model:	$y(t) = c_0$
static linear:	$y(t) = c_1 u(t)$
static affine:	$y(t) = c_0 + c_1 u(t)$
static quadratic:	$y(t) = c_0 + c_1 u(t) + c_2 u(t)^2$

- *Dynamic models.* In a dynamic model, $y(t)$ depends on $u(s)$ for some $s \neq t$. We consider some simple time-series models (see problem 2 in the reader), which are linear dynamic models.

moving average (MA):	$y(t) = a_0 u(t) + a_1 u(t-1) + a_2 u(t-2)$
autoregressive (AR):	$y(t) = a_0 u(t) + b_1 y(t-1) + b_2 y(t-2)$
autoregressive moving average (ARMA):	$y(t) = a_0 u(t) + a_1 u(t-1) + b_1 y(t-1)$

  Note that in the AR and ARMA models, $y(t)$ depends indirectly on all previous inputs, $u(s)$ for $s < t$, due to the recursive dependence on $y(t-1)$. For this reason, the AR and ARMA models are said to have *infinite memory.* The MA model, on the other hand, has a *finite memory*: $y(t)$ depends only on the current and two previous inputs. (Another term for this MA model is 3-tap system, where taps refer to taps on a delay line.)

Each of these models is specified by its parameters, *i.e.*, the scalars c_i, a_i, b_i. For each of these models, find the least-squares fit to the given data. In other words, find parameter values that minimize the sum-of-squares of the residuals. For example, for the ARMA model, pick a_0, a_1, and b_1 that minimize

$$\sum_{t=2}^{N} \left(y(t) - a_0 u(t) - a_1 u(t-1) - b_1 y(t-1)\right)^2.$$

(Note that we start the sum at $t = 2$ which ensures that $u(t-1)$ and $y(t-1)$ are defined.) For each model, give the root-mean-square (RMS) residual, *i.e.*, the squareroot of the mean of the optimal residual squared. Plot the output $\hat{y}$ predicted by your model, and plot the residual (which is $y - \hat{y}$). The data for this problem are available from the class web page, by following the link to Matlab datasets, in an m-file named `uy_data.m`. Copy this file to your working directory and type `uy_data` from within Matlab. This will create the vectors u and y and the scalar N (the length of the vectors). Now you can plot u, y, etc. *Note:* the dataset u, y is *not* generated by any of the models above. It is generated by a nonlinear recursion, which has infinite memory.

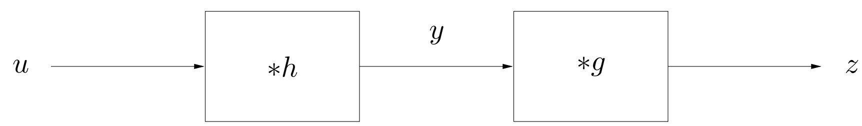
6.11 *Least-squares deconvolution.* A communications channel is modeled by a finite-impulse-response (FIR) filter:

$$y(t) = \sum_{\tau=0}^{n-1} u(t-\tau)h(\tau),$$

where $u : \mathbf{Z} \to \mathbf{R}$ is the channel input sequence, $y : \mathbf{Z} \to \mathbf{R}$ is the channel output, and $h(0), \ldots, h(n-1)$ is the impulse response of the channel. In terms of discrete-time convolution we write this as $y = h * u$. You will design a *deconvolution filter* or *equalizer* which also has FIR form:

$$z(t) = \sum_{\tau=0}^{m-1} y(t-\tau)g(\tau),$$

where $z : \mathbf{Z} \to \mathbf{R}$ is the filter output, y is the channel output, and $g(0), \ldots, g(m-1)$ is the impulse response of the filter, which we are to design. This is shown in the block diagram below.



The goal is to choose $g = (g(0), \ldots, g(m-1))$ so that the filter output is approximately the channel input delayed by D samples, *i.e.*, $z(t) \approx u(t-D)$. Since $z = g * h * u$ (discrete-time convolution), this means that we'd like

$$(g * h)(t) \approx \begin{cases} 0 & t \neq D, \\ 1 & t = D \end{cases}$$

We will refer to $g*h$ as the *equalized impulse response*; the goal is to make it as close as possible to a D-sample delay. Specifically, we want the *least-squares* equalizer is g that minimizes the sum-of-squares error

$$\sum_{t \neq D} (g * h)(t)^2,$$

subject to the constraint

$$(g * h)(D) = 1.$$

To solve the problem below you'll need to get the file `deconv_data.m` from the class web page in the *Matlab files* section. It will define the channel impulse response h as a Matlab vector `h`. (Indices in Matlab run from 1 to n, while the argument of the channel impulse response runs from $t = 0$ to $t = n-1$, so `h(3)` in Matlab corresponds to $h(2)$.)

(a) Find the least-squares equalizer g, of length $m = 20$, with delay $D = 12$. Plot the impulse responses of the channel (h) and the equalizer (g). Plot the equalized impulse response ($g * h$).

(b) The vector y (also defined in `deconv_data.m`) contains the channel output corresponding to a signal u passed through the channel (*i.e.*, $y = h * u$). The signal u is binary, *i.e.*, $u(t) \in \{-1, 1\}$, and starts at $t = 0$ (*i.e.*, $u(t) = 0$ for $t < 0$). Pass y through the least-squares equalizer found in part a, to form the signal z. Give a histogram plot of the amplitude distribution of both y and z. (You can remove the first and last D samples of z before making the histogram plot.) Comment on what you find.

Matlab hints: The command `conv` convolves two vectors; the command `hist` plots a histogram (of the amplitude distribution).

6.12 *Estimation with sensor offset and drift.* We consider the usual estimation setup:

$$y_i = a_i^T x + v_i, \qquad i = 1, \ldots, m,$$

where

- y_i is the ith (scalar) measurement
- $x \in \mathbf{R}^n$ is the vector of parameters we wish to estimate from the measurements
- v_i is the sensor or measurement error of the ith measurement

In this problem we assume the measurements y_i are taken at times evenly spaced, T seconds apart, starting at time $t = T$. Thus, y_i, the ith measurement, is taken at time $t = iT$. (This isn't really material; it just makes the interpretation simpler.) You can assume that $m \geq n$ and the measurement matrix

$$A = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_m^T \end{bmatrix}$$

is full rank (*i.e.*, has rank n). Usually we assume (often implicitly) that the measurement errors v_i are random, unpredictable, small, and centered around zero. (You don't need to worry about how to make this idea precise.) In such cases, least-squares estimation of x works well. In some cases, however, the measurement error includes some *predictable* terms. For example, each sensor measurement might include a (common) *offset* or *bias*, as well as a term that grows linearly with time (called a *drift*). We model this situation as

$$v_i = \alpha + \beta i T + w_i$$

where α is the sensor bias (which is unknown but the *same* for all sensor measurements), β is the drift term (again the same for all measurements), and w_i is part of the sensor error that is unpredictable, small, and centered around 0. If we knew the offset α and the drift term β we could just subtract the predictable part of the sensor signal, *i.e.*, $\alpha + \beta i T$ from the sensor signal. But we're interested in the case where we don't know the offset α or the drift coefficient β. Show how to use least-squares to *simultaneously* estimate the parameter vector $x \in \mathbf{R}^n$, the offset $\alpha \in \mathbf{R}$, and the drift coefficient $\beta \in \mathbf{R}$. Clearly explain your method. If your method always works, say so. Otherwise describe the conditions (on the matrix A) that must hold for your method to work, and give a simple example where the conditions don't hold.

6.13 *Estimating emissions from spot measurements.* There are n sources of a pollutant, at known locations $s_1, \ldots, s_n \in \mathbf{R}^2$. Each source emits the pollutant at some emission rate; we let x_j denote the emission rate for source j. (These are positive, but to simplify things we won't concern ourselves with that.) The emission rates are to be determined, or estimated. We measure the total pollutant level at m spots, located at $t_1, \ldots, t_m \in \mathbf{R}^2$, which are known. The total pollutant measured at spot i is the sum of the contributions from the n sources.

The contribution from source j to measurement i is given by $\alpha x_j / \|s_j - t_i\|^2$, where α is a known (positive) constant. In other words, the pollutant concentration from a source follows an inverse square law, and is proportional to the emission rate. We assume that measurement spots do not coincide with the source locations, *i.e.*, we do not have $s_j = t_i$ for any i or j. We also assume that none of the spot locations is repeated (*i.e.*, we have $t_i \neq t_j$ for $i \neq j$) and that none of the source locations is repeated (*i.e.*, we have $s_i \neq s_j$ for $i \neq j$).

(a) Give a *specific* example of source and spot measurement locations, with 4 sensors and 3 sources, for which it is *impossible* to find the emission rates given the spot measurements. In this part, we ignore the issue of noise or sensor errors; we assume the spot measurements are exactly as described above. To show that your configuration is a valid example, give two *specific* different sets of emission rates that yield identical spot measurements. You are free to (briefly) explain your example using concepts such as range, nullspace, rank, and so on; but remember, we want a *specific numerical example*, such as as $s_1 = [0\ 1]^T, \ldots, s_3 = [1\ 2]^T$, $t_1 = [1\ 1]^T, \ldots, t_4 = [3\ 2]^T$. (And similarly for the two emission rates that give the same spot measurements.)

(b) Get the data from the file `emissions_data.m` that is available on the class web site. This file defines three source locations (given as a 2×3 matrix; the columns give the locations), and ten spot measurement locations (given as a 2×10 matrix). It also gives two sets of spot measurements: one for part (b), and one for part (c). Be careful to use the right set of measurements for each problem! The spot measurements are not perfect (as we assumed in part (a)); they contain small noise and errors. Estimate the pollutant emission rates. Explain your method, and give your estimate for the emissions rates of the three sources.

(c) Now we suppose that *one* of the spot measurments is faulty, *i.e.*, its associated noise or error is far larger than the errors of the other spot measurements. Explain how you would identify or guess which one is malfunctioning, and then estimate the source emission rates. Carry out your method on the data given in the matlab file. Be sure to tell us which spot measurement you believe to be faulty, and what your guess of the emission rates is. (The emission rates are *not* the same as in part (b), but the source and spot measurement locations are.)

6.14 *Identifying a system from input/output data.* We consider the standard setup:

$$y = Ax + v,$$

where $A \in \mathbf{R}^{m \times n}$, $x \in \mathbf{R}^n$ is the input vector, $y \in \mathbf{R}^m$ is the output vector, and $v \in \mathbf{R}^m$ is the noise or disturance. We consider here the problem of estimating the matrix A, given some input/output data. Specifically, we are given the following:

$$x^{(1)}, \ldots, x^{(N)} \in \mathbf{R}^n, \qquad y^{(1)}, \ldots, y^{(N)} \in \mathbf{R}^m.$$

These represent N samples or observations of the input and output, respectively, possibly corrupted by noise. In other words, we have

$$y^{(k)} = Ax^{(k)} + v^{(k)}, \quad k = 1, \ldots, N,$$

where $v^{(k)}$ are assumed to be small. The problem is to estimate the (coefficients of the) matrix A, based on the given input/output data. You will use a least-squares criterion to form an estimate $\hat{A}$ of A. Specifically, you will choose as your estimate $\hat{A}$ the matrix that minimizes the quantity

$$J = \sum_{k=1}^{N} \|Ax^{(k)} - y^{(k)}\|^2$$

over A.

(a) Explain how to do this. If you need to make an assumption about the input/output data to make your method work, state it clearly. You may want to use the matrices $X \in \mathbf{R}^{n\times N}$ and $Y \in \mathbf{R}^{m\times N}$ given by

$$X = \left[\begin{array}{ccc} x^{(1)} & \cdots & x^{(N)} \end{array}\right], \qquad Y = \left[\begin{array}{ccc} y^{(1)} & \cdots & y^{(N)} \end{array}\right]$$

in your solution.

(b) On the course web site you will find some input/output data for an instance of this problem in the mfile `sysid_data.m`. Executing this mfile will assign values to m, n, and N, and create two matrices that contain the input and output data, respectively. The $n \times N$ matrix variable `X` contains the input data $x^{(1)}, \ldots, x^{(N)}$ (*i.e.*, the first column of `X` contains $x^{(1)}$, etc.). Similarly, the $m \times N$ matrix `Y` contains the output data $y^{(1)}, \ldots, y^{(N)}$. You must give your final estimate $\hat{A}$, your source code, and also give an explanation of what you did.

6.15 *Robust least-squares estimation methods.* We consider a standard measurement setup, with $y = Ax + v$, where $x \in \mathbf{R}^n$ is a vector we'd like to estimate, $y \in \mathbf{R}^m$ is the vector of measurements, $v \in \mathbf{R}^m$ is the vector of measurement errors, and $A \in \mathbf{R}^{m\times n}$. We assume that $m > n$, *i.e.*, there are more measurements than parameters to be estimated. The measurement error v is not known, but is assumed to be small. The goal is to estimate x, given y. Here is the twist: we do not know the matrix A exactly. In fact we calibrated our sensor system on $k > 1$ different days, and found the values

$$A^{(1)}, \ldots, A^{(k)}$$

for the matrix A, on the different days. These matrices are close to each other, but not exactly the same. There is no pattern to the (small) variations between the matrices; for example, there is no discernable drift; the variations appear to be small and random. You can assume that all of the matrices are full rank, *i.e.*, have rank n. Now suppose we have a measurement y taken on a day when we did not calibrate the sensor system. We want to form an estimate $\hat{x}$, based on this measurement. We don't know A exactly, but we can assume that it is close to the known values $A^{(1)}, \ldots, A^{(k)}$ found during calibration. A method for guessing x in this situtation is called a *robust estimation method*, since it attempts to take into account the uncertainty in the matrix A. Three very reasonable proposals for robust estimation are described below.

- *The average then estimate method.* First, we form the average of the calibration values,

$$A_{\text{avg}} = \frac{1}{k}\sum_{j=1}^{k} A^{(j)},$$

which is supposed to represent the most typical value of A. We then choose our estimate $\hat{x}$ to minimize the least squares residual using A_{avg}, *i.e.*, to minimize $\|A_{\text{avg}}\hat{x} - y\|$. We refer to this value of $\hat{x}$ as $\hat{x}_{\text{ae}}$, where the subscript stands for 'average (then) estimate'. (You can assume that A_{avg} is full rank.)

- *The estimate then average method.* First, we find the least-squares estimate of x for each of the calibration values, *i.e.*, we find $\hat{x}^{(j)}$ that minimizes $\|A^{(j)}\hat{x} - y\|$ over $\hat{x}$, for $j = 1, \ldots, k$. Since the matrices $A^{(j)}$ are close but not equal, we find that the estimates $\hat{x}^{(j)}$ are also close but not equal. We find our final estimate of x as the average of these estimates:
$$\hat{x}_{\text{ea}} = \frac{1}{k}\sum_{j=1}^{k} \hat{x}^{(j)}.$$
(Here the subscript 'ea' stands for 'estimate (then) average'.)
- *Minimum RMS residuals method.* If we make the guess $\hat{x}$, then the residual, using the jth calibrated value of A, is given by $r^{(j)} = A^{(j)}\hat{x} - y$. The RMS value of the collection of residuals is given by
$$\left(\frac{1}{k}\sum_{j=1}^{k} \|r^{(j)}\|^2\right)^{1/2}.$$
In the minimum RMS residual method, we choose $\hat{x}$ to minimize this quantity. We denote this estimate of x as $\hat{x}_{\text{rms}}$.

Here is the problem:

(a) For each of these three methods, say whether the estimate $\hat{x}$ is a linear function of y. If it is a linear function, give a formula for the matrix that gives $\hat{x}$ in terms of y. For example, if you believe that $\hat{x}_{\text{ea}}$ is a linear function of y, then you should give a formula for B_{ea} (in terms of $A^{(1)}, \ldots, A^{(k)}$), where $\hat{x}_{\text{ea}} = B_{\text{ea}}y$.

(b) Are the three methods described above different? If any two are the same (for all possible values of the data $A^{(1)}, \ldots, A^{(k)}$ and y), explain why. If they are different, give a specific example in which the estimates differ.

6.16 *Estimating a signal with interference.* This problem concerns three proposed methods for estimating a signal, based on a measurement that is corrupted by a small noise and also by an interference, that need not be small. We have
$$y = Ax + Bv + w,$$
where $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{m\times p}$ are known. Here $y \in \mathbf{R}^m$ is the measurement (which is known), $x \in \mathbf{R}^n$ is the signal that we want to estimate, $v \in \mathbf{R}^p$ is the interference, and w is a noise. The noise is unknown, and can be assumed to be small. The interference is unknown, but cannot be assumed to be small. You can assume that the matrices A and B are skinny and full rank (*i.e.*, $m > n$, $m > p$), and that the ranges of A and B intersect only at 0. (If this last condition does not hold, then there is no hope of finding x, even when $w = 0$, since a nonzero interference can masquerade as a signal.) Each of the EE263 TAs proposes a method for estimating x. These methods, along with some informal justification from their proposers, are given below. Nikola proposes the **ignore and estimate method.** He describes it as follows:

We don't know the interference, so we might as well treat it as noise, and just ignore it during the estimation process. We can use the usual least-squares method, for the model $y = Ax + z$ (with z a noise) to estimate x. (Here we have $z = Bv + w$, but that doesn't matter.)

Almir proposes the **estimate and ignore method.** He describes it as follows:

We should simultaneously estimate both the signal x *and* the interference v, based on y, using a standard least-squares method to estimate $[x^T \ v^T]^T$ given y. Once we've estimated x and v, we simply ignore our estimate of v, and use our estimate of x.

Miki proposes the **estimate and cancel method.** He describes it as follows:

Almir's method makes sense to me, but I can improve it. We should simultaneously estimate both the signal x and the interference v, based on y, using a standard least-squares method, exactly as in Almir's method. In Almir's method, we then throw away $\hat{v}$, our estimate of the interference, but I think we should use it. We can form the "pseudo-measurement" $\tilde{y} = y - B\hat{v}$, which is our measurement, with the effect of the estimated interference subtracted off. Then, we use standard least-squares to estimate x from $\tilde{y}$, from the simple model $\tilde{y} = Ax+z$. (This is exactly as in Nikola's method, but here we have subtracted off or cancelled the effect of the estimated interference.)

These descriptions are a little vague; part of the problem is to translate their descriptions into more precise algorithms.

(a) Give an explicit formula for each of the three estimates. (That is, for each method give a formula for the estimate $\hat{x}$ in terms of A, B, y, and the dimensions n, m, p.)

(b) Are the methods really different? Identify any pairs of the methods that coincide (*i.e.*, always give exactly the same results). If they are all three the same, or all three different, say so. Justify your answer. To show two methods are the same, show that the formulas given in part (a) are equal (even if they don't appear to be at first). To show two methods are different, give a specific numerical example in which the estimates differ.

(c) Which method or methods do you think work best? Give a very brief explanation. (If your answer to part (b) is "The methods are all the same" then you can simply repeat here, "The methods are all the same".)

6.17 *Vector time-series modeling.* This problem concerns a vector time-series, $y(1), \ldots, y(T) \in \mathbf{R}^n$. The n components of $y(t)$ might represent measurements of different quantities, or prices of different assets, at time period t. Our goal is to develop a model that allows us to *predict* the next element in the time series, *i.e.*, to predict $y(t+1)$, given $y(1), \ldots, y(t)$. A consultant proposes the following model for the time-series:

$$y(t) = Ay(t-1) + v(t), \quad t = 2, \ldots, T,$$

where the matrix $A \in \mathbf{R}^{n\times n}$ is the parameter of the model, and $v(t) \in \mathbf{R}^n$ is a signal that is small and unpredictable. (We keep the definition of the terms 'small' and 'unpredictable' vague, since the exact meaning won't matter.) This type of model has several names. It is called an VAR(1) model, which is short for *vector auto-regressive*, with one time lag. It is also called a Gauss-Markov model, which is a fancy name for a linear system driven by a noise. Once we have a model of this form, we can predict the next time-series sample using the formula

$$\hat{y}(t+1) = Ay(t), \quad t = 1, \dots, T.$$

The idea here is that $v(t)$ is unpredictable, so we simply replace it with zero when we estimate the next time-series sample. The prediction error is given by

$$e(t) = \hat{y}(t) - y(t), \quad t = 2, \dots, T.$$

The prediction error depends on the time-series data, and also A, the parameter in our model. There is one more twist. It is known that $y_1(t+1)$, the first component of the next time-series sample, does not depend on $y_2(t), \dots, y_n(t)$. The second component, $y_2(t+1)$, does not depend on $y_3(t), \dots, y_n(t)$. In general, the ith component, $y_i(t+1)$, does not depend on $y_{i+1}(t), \dots, y_n(t)$. Roughly speaking, this means that the current time-series component $y_i(t)$ only affects the next time-series components $y_1(t+1), \dots, y_i(t+1)$. This means that the matrix A is lower triangular, *i.e.*, $A_{ij} = 0$ for $i < j$. To find the parameter A that defines our model, you will use a least-squares criterion. You will pick A that minimizes the mean-square prediction error,

$$\frac{1}{T-1} \sum_{t=2}^{T} \|e(t)\|^2,$$

over all lower-triangular matrices. Carry out this method, using the data found in the `vts_data.m`, which contains an $n \times T$ matrix `Y`, whose columns are the vector time-series samples at $t = 1, \dots, T$. Explain your method, and submit the code that you use to solve the problem. Give your final estimated model parameter A, and the resulting mean-square error. Compare your mean-square prediction error to the mean-square value of y, *i.e.*,

$$\frac{1}{T} \sum_{t=1}^{T} \|y(t)\|^2.$$

Finally, predict what you think $y(T+1)$ is, based on the data given.

6.18 *Fitting a rational transfer function to frequency response data.* This problem concerns a rational function $H : \mathbf{C} \to \mathbf{C}$ of the form

$$H(s) = \frac{A(s)}{B(s)},$$

where A and B are the polynomials

$$A(s) = a_0 + a_1 s + \cdots + a_m s^m, \qquad B(s) = 1 + b_1 s + \cdots + b_m s^m.$$

Here $a_0, \dots, a_m \in \mathbf{R}$ and $b_1, \dots, b_m \in \mathbf{R}$ are *real* parameters, and $s \in \mathbf{C}$ is the *complex* independent variable. We define $a = (a_0, \dots, a_m) \in \mathbf{R}^{m+1}$ and $b = (b_1, \dots, b_m) \in \mathbf{R}^m$, *i.e.*, a and b are vectors containing the coefficients of A and B (not including the constant coefficient of B, which is fixed at one). We are given noisy measurements of H at some points on the imaginary axis, *i.e.*, some data

$$s_1 = j\omega_1, \ldots, s_N = j\omega_N \in \mathbf{C}, \qquad h_1, \ldots, h_N \in \mathbf{C},$$

and hope to choose a and b so that we have $H(s_i) \approx h_i$. Here $\omega_1, \ldots, \omega_N$ are real and nonnegative, and $h_1, \ldots, h_N$ are complex. To judge the quality of fit we use the mean-square error,

$$J = \frac{1}{N} \sum_{i=1}^{N} |H(s_i) - h_i|^2.$$

Interpretation. (Not needed to solve the problem.) You can think of H as a rational transfer function, with s the complex frequency variable. The data is a set of frequency response measurements, with some measurement errors. The goal is to find a rational transfer function that fits the measured frequency response. This problem explores a famous heuristic method, based on solving a sequence of (linear) least-squares problems, for finding coefficients a, b that approximately minimize J. We start by expressing J in the following (strange) way:

$$J = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{A(s_i) - h_i B(s_i)}{z_i} \right|^2, \qquad z_i = B(s_i), \quad i = 1, \ldots, N.$$

The method works by choosing a and b that minimize the lefthand expression (with z_i fixed), then updating the numbers z_i using the righthand formula, and then repeating. More precisely, let k denote the iteration number, with $a^{(k)}$, $b^{(k)}$, and $z_i^{(k)}$ denoting the values of these parameters at iteration k, and $A^{(k)}$, $B^{(k)}$ denoting the associated polynomials. To update these parameters from iteration k to iteration $k+1$, we proceed as follows. First, we set $z_i^{(k+1)} = B^{(k)}(s_i)$, for $i = 1, \ldots, N$. Then we choose $a^{(k+1)}$ and $b^{(k+1)}$ that minimize

$$\frac{1}{N} \sum_{i=1}^{N} \left| \frac{A^{(k+1)}(s_i) - h_i B^{(k+1)}(s_i)}{z_i^{(k+1)}} \right|^2.$$

(This can be done using ordinary linear least-squares.) We can start the iteration with $z_i^{(1)} = 1$, $i = 1, \ldots, N$ (which is what would happen if we set $B^{(0)}(s) = 1$). The iteration is stopped when (or more accurately, if) successive iterates are very close, *i.e.*, we have $a^{(k+1)} \approx a^{(k)}$, and $b^{(k+1)} \approx b^{(k)}$. Several pathologies can arise in this algorithm. For example, we can end up with $z_i^{(k)} = 0$, or a certain matrix can be less than full rank, which complicates solving the least-squares problem to find $a^{(k)}$ and $b^{(k)}$. You can ignore these pathologies, however.

(a) Explain how to obtain $a^{(k+1)}$ and $b^{(k+1)}$, given $z^{(k+1)}$. You must explain the math; you may not refer to any Matlab notation or operators (and especially, backslash) in your explanation. Please bear in mind that $a_0, \ldots, a_m$ and $b_1, \ldots, b_m$ are real, whereas many other variables and data in this problem are complex.

(b) Implement the method, and apply it to the data given in `rat_data.m`. This file contains the data $\omega_1, \ldots, \omega_N$, $h_1, \ldots, h_N$, as well as m and N. Give the final coefficient vectors a, b, and the associated final value of J. Terminate the algorithm when

$$\left\| \begin{bmatrix} a^{(k+1)} - a^{(k)} \\ b^{(k+1)} - b^{(k)} \end{bmatrix} \right\| \leq 10^{-6}.$$

Plot J versus iteration k, with J on a logarithmic scale, and k on a linear scale, using the command `semilogy`. Plot $|H(j\omega)|$ on a logarithmic scale versus ω on a linear scale (using `semilogy`), for the first iteration, last iteration, and the problem data. To evaluate a polynomial in Matlab, you can either write your own (short) code, or use the Matlab command `polyval`. This is a bit tricky, since `polyval` expects the polynomial coefficients to be listed in the reverse order than we use here. To evaluate $A(s)$ in Matlab you can use the command `polyval(a(m+1:-1:1),s)`. To evaluate $b(s)$ you can use `polyval([b(m:-1:1);1],s)`.

Note: no credit will be given for implementing any algorithm other than the one described in this problem.

6.19 *Quadratic placement.* We consider an integrated circuit (IC) that contains N cells or modules that are connected by K wires. We model a cell as a single point in $\mathbf{R}^2$ (which gives its location on the IC) and ignore the requirement that the cells must not overlap. The positions of the cells are

$$(x_1, y_1),\ (x_2, y_2), \ldots, (x_N, y_N),$$

i.e., x_i gives the x-coordinate of cell i, and y_i gives the y-coordinate of cell i. We have two types of cells: *fixed cells*, whose positions are fixed and given, and *free cells*, whose positions are to be determined. We will take the first n cells, at positions

$$(x_1, y_1), \ldots, (x_n, y_n),$$

to be the free ones, and the remaining $N - n$ cells, at positions

$$(x_{n+1}, y_{n+1}), \ldots, (x_N, y_N),$$

to be the fixed ones. The task of finding good positions for the free cells is called *placement.* (The fixed cells correspond to cells that are already placed, or external pins on the IC.) There are K wires that connect pairs of the cells. We will assign an orientation to each wire (even though wires are physically symmetric). Specifically, wire k goes *from* cell $I(k)$ *to* cell $J(k)$. Here I and J are functions that map wire number (*i.e.*, k) into the origination cell number (*i.e.*, $I(k)$), and the destination cell number (*i.e.*, $J(k)$), respectively. To describe the wire/cell topology and the functions I and J, we'll use the *node incidence matrix* A for the associated directed graph. The node incidence matrix $A \in \mathbf{R}^{K \times N}$ is defined as

$$A_{kj} = \begin{cases} 1 & \text{wire } k \text{ goes to cell } j,\ i.e.,\ j = J(k) \\ -1 & \text{wire } k \text{ goes from cell } j,\ i.e.,\ j = I(k) \\ 0 & \text{otherwise.} \end{cases}$$

Note that the kth row of A is associated with the kth wire, and the jth column of A is associated with the jth cell. (We know this isn't consistent with the definition used on the midterm; this is the more standard definition.) The goal in placing the free cells is to use the smallest amount of interconnect wire, assuming that the wires are run as straight lines between the cells. (In fact, the wires in an IC are *not* run on straight lines directly between the cells, but that's another story. Pretending that the wires do run on straight lines seems to give good placements.) One common method, called *quadratic placement*, is to place the free cells in order to minimize the the total square wire length, given by

$$J = \sum_{k=1}^{K} \left((x_{I(k)} - x_{J(k)})^2 + (y_{I(k)} - y_{J(k)})^2 \right).$$

(a) Explain how to find the positions of the free cells, *i.e.*,

$$(x_1, y_1), \ldots, (x_n, y_n),$$

that minimize the total square wire length. You may make an assumption about the rank of one or more matrices that arise.

(b) In this part you will determine the optimal quadratic placement for a specific set of cells and interconnect topology. The mfile `qplace_data.m` defines an instance of the quadratic placement problem. Specifically, it defines the dimensions n, N, and K, and $N-n$ vectors `xfixed` and `yfixed`, which give the x- and y-coordinates of the fixed cells. The mfile also defines the node incidence matrix `A`, which is $K \times N$. Be sure to explain how you solve this problem, and to explain the matlab source code that solves it (which you must submit). Give the optimal locations of the free cells. Check your placement against various others, such as placing all free cells at the origin. You will also find an mfile that plots a proposed placement in a nice way:
`view_layout(xfree,yfree,xfixed,yfixed,A)`.
This mfile takes as argument the x- and y-coordinates of the free and fixed cells, as well as the node incidence matrix that describes the wires. It plots the proposed placement. Plot your optimal placement using `view_layout`.

6.20 *Least-squares state tracking.* Consider the system $x(t+1) = Ax(t) + Bu(t) \in \mathbf{R}^n$, with $x(0) = 0$. We *do not* assume it is controllable. Suppose $x_{\mathrm{des}}(t) \in \mathbf{R}^n$ is given for $t = 1, \ldots, N$ (and is meant to be the desired or target state trajectory). For a given input u, we define the mean-square tracking error as

$$E(u) = \frac{1}{N} \sum_{t=1}^{N} \|x(t) - x_{\mathrm{des}}(t)\|^2.$$

(a) Explain how to find u_{opt} that minimizes E (in the general case). Your solution can involve a (general) pseudo-inverse.

(b) *True or false:* If the system is controllable, there is a unique u_{opt} that minimizes $E(u)$. Briefly justify your answer.

(c) Find $E(u_{\mathrm{opt}})$ for the specific system with

$$A = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},$$

$x_{\mathrm{des}}(t) = [t\ 0\ 0]^T$, and $N = 10$.

6.21 *Time series prediction.* We consider an autonomous discrete-time linear system of the form

$$x(t+1) = Ax(t), \quad y(t) = Cx(t) + v(t),$$

where $x(t) \in \mathbf{R}^n$, $y(t) \in \mathbf{R}$ is the measured output signal, and $v(t) \in \mathbf{R}$ represents an output noise signal. In this problem, you *do not know* the matrices $A \in \mathbf{R}^{n\times n}$ or $C \in \mathbf{R}^{1\times n}$, the state $x(t)$ (including the initial state $x(0)$), or even the system order n. You *do know* the measured output signal for $t = 1, \ldots, p$:

$$y(1), \ldots, y(p).$$

We give you two more pieces of information: the system order n is less than 20, and the RMS value of the noise, *i.e.*, $\left((1/p)\sum_{t=1}^{p} v(t)^2\right)^{1/2}$, is small (on the order of 0.001). The goal is to predict $y(t)$ for the next q time steps, *i.e.*, to predict what $y(p+1), \ldots, y(p+q)$ will be. Here is the problem: get the time series data from the class web site in the file `timeseriesdata.m`, which gives $y(1), \ldots, y(200)$. (We have $p = 200$ in this specific problem.) Then, predict what $y(201), \ldots, y(220)$ are. Plot your estimates $\hat{y}(201), \ldots, \hat{y}(220)$, and also, of course, give us the numbers. (You may also want to plot the whole set of data, from $t = 1$ to $t = 220$, just to make sure your prediction satisfies the 'eyeball test'.) It is extremely important that you explain very clearly how you come up with your prediction. What is your method? If you make choices during your procedure, how do you make them?

6.22 *Extracting RC values from delay data.* We consider a CMOS digital gate that drives a load consisting of interconnect wires that connect to the inputs of other gates. To find the delay of the gate plus its load, we have to solve a complex, nonlinear ordinary differential equation that takes into account circuit nonlinearities, parasitic capacitances, and so on. This can be done using a circuit simulator such as SPICE. A very simple approximation of the delay can be obtained by modeling the gate as a simple resistor with resistance R, and the load as a simple capacitor with capacitance C. In this simple model, the delay of the gate has the form ηRC, where η is a constant that depends on the precise definition of delay used. (One common choice is $\eta = 0.69$, which is based on the time it takes the voltage of a simple RC circuit to decay to 1/2 its original value.) This simple RC delay model can be used for design, or approximate (but very fast) analysis. We address the question of determining a value of R for each of a set of gates, and a value of C for each of a set of loads, given accurate delay data (obtained by simulation) for each combination of a gate driving a load. We have n digital gates labeled $1, \ldots, n$, and m loads labeled $1, \ldots, m$. By simulation, we obtain the (accurate) delay D_{ij} for gate j driving load i. (D is given as an $m \times n$ matrix.) The goal is to find positive numbers $R_1, \ldots, R_n$ and $C_1, \ldots, C_m$ so that $D_{ij} \approx R_j C_i$. (For simplicity we'll take $\eta = 1$ in our delay model.) Finding good values of parameters for a simple model, given accurate data, is sometimes called *parameter extraction*. In this problem, we are extracting the gate resistances R_j and the load capacitances C_i, given accurate delay data D_{ij} (obtained by simulation). If we scale all resistances by a constant $\alpha > 0$, and scale all capacitances by $1/\alpha$, the approximate delays $R_j C_i$ don't change. To remove this ambiguity, we will fix $C_1 = 1$, *i.e.*, we will take the first load as our 'unit load'. Finally we get to the problem.

(a) *Minimum mean-square absolute error.* Explain how to find $R_1^{\text{msa}}, \ldots, R_n^{\text{msa}}$ and $C_1^{\text{msa}}, \ldots, C_m^{\text{msa}}$

(positive, with $C_1^{\mathrm{msa}} = 1$) that minimize the mean-square absolute error,

$$E_{\mathrm{msa}} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (D_{ij} - R_j C_i)^2 .$$

(b) *Minimum mean-square logarithmic error.* Explain how to find $R_1^{\mathrm{msl}}, \ldots, R_n^{\mathrm{msl}}$ and $C_1^{\mathrm{msl}}, \ldots, C_m^{\mathrm{msl}}$ (positive, with $C_1^{\mathrm{msl}} = 1$) that minimize the mean-square logarithmic error,

$$E_{\mathrm{msl}} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (\log D_{ij} - \log(R_j C_i))^2 .$$

(The logarithm here is base e, but it doesn't really matter.)

(c) Find $R_1^{\mathrm{msa}}, \ldots, R_n^{\mathrm{msa}}$ and $C_1^{\mathrm{msa}}, \ldots, C_m^{\mathrm{msa}}$, as well as $R_1^{\mathrm{msl}}, \ldots, R_n^{\mathrm{msl}}$ and $C_1^{\mathrm{msl}}, \ldots, C_m^{\mathrm{msl}}$, for the particular delay data given in `rcextract_data.m` from the course web site. Submit the Matlab code used to calculate these values, as well as the values themselves. Also write down your minimum E_{msa} and E_{msl} values.

Please note the following:

- You do not need to know anything about digital circuits to solve this problem.
- The two criteria (absolute and logarithmic) are clearly close, so we expect the extracted Rs and Cs to be similar in the two cases. But they are not the same.
- In this problem we are more interested in your *approach* and *method* than the final numerical results. We will take points off if your method is not the best possible one, even if your answer is numerically close to the correct one.

6.23 *Reconstructing values from sums over subsets.* There are real numbers $u_1, \ldots, u_p$ that we do not know, but want to find. We do have information about sums of some subsets of the numbers. Specifically, we know $v_1, \ldots, v_q$, where

$$v_i = \sum_{j \in S_i} u_j .$$

Here, S_i denotes the subset of $\{1, \ldots, p\}$ that defines the partial sum used to form v_i. (We know both v_i and S_i, for $i = 1, \ldots, q$.) We call the collection of subsets $S_1, \ldots, S_q$ *informative* if we can determine, or reconstruct, $u_1, \ldots, u_p$ without ambiguity, from $v_1, \ldots, v_q$. If the set of subsets is not informative, we say it is *uninformative.* As an example with $p = 3$ and $q = 4$,

$$v_1 = u_2 + u_3, \quad v_2 = u_1 + u_2 + u_3, \quad v_3 = u_1 + u_3, \quad v_4 = u_1 + u_2.$$

This corresponds to the subsets

$$S_1 = \{2, 3\}, \quad S_2 = \{1, 2, 3\}, \quad S_3 = \{1, 3\}, \quad S_4 = \{1, 2\}.$$

This collection of subsets is informative. To see this, we show how to reconstruct u_1, u_2, u_3. First we note that $u_1 = v_2 - v_1$. Now that we know u_1 we can find u_2 from $u_2 = v_4 - u_1 = v_4 - v_2 + v_1$. In the same way we can get $u_3 = v_3 - u_1 = v_3 - v_2 + v_1$. **Note:** this is only an example to illustrate the notation.

(a) This subproblem concerns the following specific case, with $p = 6$ numbers and $q = 11$ subsets. The subsets are

$$S_1 = \{1,2,3\}, \quad S_2 = \{1,2,4\}, \quad S_3 = \{1,2,6\}, \quad S_4 = \{1,3,5\}, \quad S_5 = \{1,4,5\},$$
$$S_6 = \{2,3,6\}, \quad S_7 = \{2,4,6\}, \quad S_8 = \{3,4,5\}, \quad S_9 = \{3,5,6\}, \quad S_{10} = \{4,5,6\},$$
$$S_{11} = \{1,2,3,4,5,6\}.$$

The associated sums are

$$v_1 = -2, \quad v_2 = 14, \quad v_3 = 6, \quad v_4 = 4, \quad v_5 = 20, \quad v_6 = -5,$$
$$v_7 = 11, \quad v_8 = 9, \quad v_9 = 1, \quad v_{10} = 17, \quad v_{11} = 15.$$

Choose one of the following:

- *The collection of subsets $S_1, \ldots, S_{11}$ is informative.*
  Justify why you believe this is the case, and reconstruct $u_1, \ldots, u_6$.
- *The collection of subsets $S_1, \ldots, S_{11}$ is uninformative.*
  To justify this, give two different sets of values $u_1, \ldots, u_6$, and $\tilde{u}_1, \ldots, \tilde{u}_6$, whose given subset sums agree with the given $v_1, \ldots, v_{11}$.

(b) This subproblem concerns a general method for reconstructing $u = (u_1, \ldots, u_p)$ given $v = (v_1, \ldots, v_q)$ (and of course, the subsets $S_1, \ldots, S_q$). We define the *subset count matrix* $Z \in \mathbf{R}^{p \times p}$ as follows: Z_{ij} is the number of subsets containing both i and j. (Thus, Z_{ii} is the number of subsets that contain i.) For each i, we define f_i as the sum of all v_j, over subsets that contain i:

$$f_i = \sum_{i \in S_j} v_j, \qquad i = 1, \ldots, p.$$

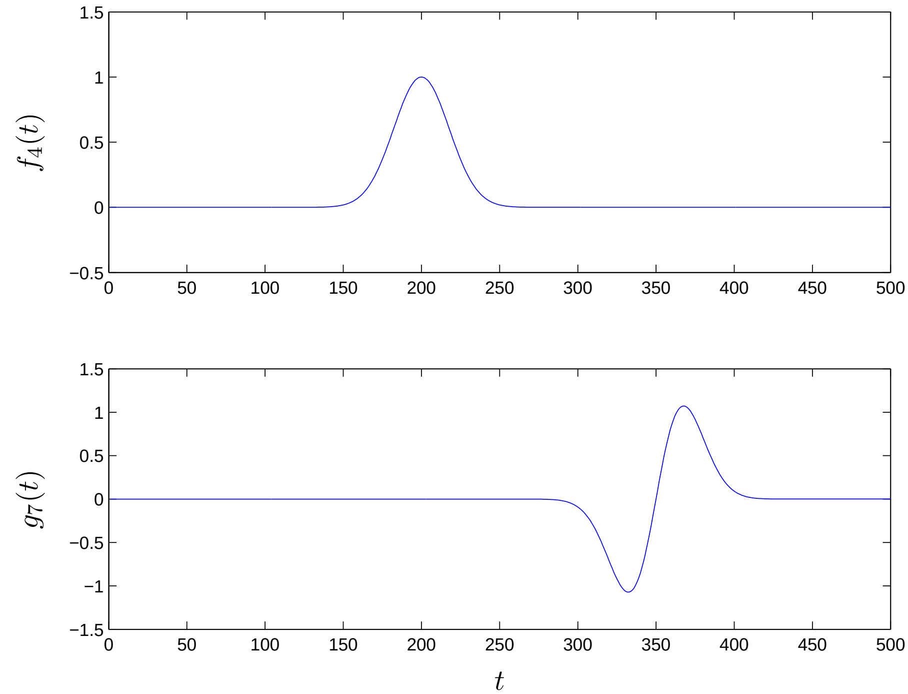
Then we reconstruct u as $u = Z^{-1} f$. (Of course, this requires that Z is invertible.) Choose one of the following:

- *The method works, whenever the collection of subsets is informative.*
  By 'works' we mean that Z is invertible, and that $Z^{-1} f$ is the unique u with subset sums v. If you believe this is the case, explain why.
- *The method can fail, even when the collection of subsets is informative.*
  To convince us of this, give a *specific example*, where the collection of subsets is informative, but the method above fails, *i.e.*, either Z is singular, or $Z^{-1} f$ does not have the required subset sums. (Please give us the simplest example you can think of.)

6.24 *Signal estimation using least-squares.* This problem concerns discrete-time signals defined for $t = 1, \ldots, 500$. We'll represent these signals by vectors in $\mathbf{R}^{500}$, with the index corresponding to the time. We are given a noisy measurement $y_{\text{meas}}(1), \ldots, y_{\text{meas}}(500)$, of a signal $y(1), \ldots, y(500)$ that is thought to be, at least approximately, a linear combination of the 22 signals

$$f_k(t) = e^{-(t-50k)^2/25^2}, \qquad g_k(t) = \left(\frac{t - 50k}{10}\right) e^{-(t-50k)^2/25^2},$$

where $t = 1, \ldots, 500$ and $k = 0, \ldots, 10$. Plots of f_4 and g_7 (as examples) are shown below.



As our estimate of the original signal, we will use the signal $\hat{y} = (\hat{y}(1), \ldots, \hat{y}(500))$ in the span of $f_0, \ldots, f_{10}, g_0, \ldots, g_{10}$, that is closest to $y_{\text{meas}} = (y_{\text{meas}}(1), \ldots, y_{\text{meas}}(500))$ in the RMS (root-mean-square) sense. Explain how to find $\hat{y}$, and carry out your method on the signal y_{meas} given in `sig_est_data.m` on the course web site. Plot y_{meas} and $\hat{y}$ on the same graph. Plot the residual (the difference between these two signals) on a different graph, and give its RMS value.

6.25 *Point of closest convergence of a set of lines.* We have m lines in $\mathbf{R}^n$, described as

$$\mathcal{L}_i = \{p_i + tv_i \mid t \in \mathbf{R}\}, \quad i = 1, \ldots, m,$$

where $p_i \in \mathbf{R}^n$, and $v_i \in \mathbf{R}^n$, with $\|v_i\| = 1$, for $i = 1, \ldots, m$. We define the distance of a point $z \in \mathbf{R}^n$ to a line $\mathcal{L}$ as

$$\mathbf{dist}(z, \mathcal{L}) = \min\{\|z - u\| \mid u \in \mathcal{L}\}.$$

(In other words, $\mathbf{dist}(z, \mathcal{L})$ gives the closest distance between the point z and the line $\mathcal{L}$.)

We seek a point $z^\star \in \mathbf{R}^n$ that minimizes the sum of the squares of the distances to the lines,

$$\sum_{i=1}^{m} \mathbf{dist}(z, \mathcal{L}_i)^2.$$

The point $z^\star$ that minimizes this quantity is called the *point of closest convergence*.

(a) Explain how to find the point of closest convergence, given the lines (*i.e.*, given $p_1, \ldots, p_m$ and $v_1, \ldots, v_m$). If your method works provided some condition holds (such as some matrix being full rank), say so. If you can relate this condition to a simple one involving the lines, please do so.

(b) Find the point $z^\star$ of closest convergence for the lines with data given in the Matlab file `line_conv_data.m`. This file contains $n \times m$ matrices `P` and `V` whose columns are the vectors $p_1, \ldots, p_m$, and $v_1, \ldots, v_m$, respectively. The file also contains commands to plot the lines and the point of closest convergence (once you have found it). Please include this plot with your solution.

6.26 *Estimating direction and amplitude of a light beam.* A light beam with (nonnegative) amplitude a comes from a direction $d \in \mathbf{R}^3$, where $\|d\| = 1$. (This means the beam travels in the direction $-d$.) The beam falls on $m \geq 3$ photodetectors, each of which generates a scalar signal that depends on the beam amplitude and direction, and the direction in which the photodetector is pointed. Specifically, photodetector i generates an output signal p_i, with

$$p_i = a\alpha \cos\theta_i + v_i,$$

where θ_i is the angle between the beam direction d and the outward normal vector q_i of the surface of the ith photodetector, and α is the photodetector sensitivity. You can interpret $q_i \in \mathbf{R}^3$, which we assume has norm one, as the direction the ith photodetector is pointed. We assume that $|\theta_i| < 90^\circ$, *i.e.*, the beam illuminates the top of the photodetectors. The numbers v_i are small measurement errors.

You are given the photodetector direction vectors $q_1, \ldots, q_m \in \mathbf{R}^3$, the photodetector sensitivity α, and the noisy photodetector outputs, $p_1, \ldots, p_m \in \mathbf{R}$. Your job is to estimate the beam direction $d \in \mathbf{R}^3$ (which is a unit vector), and a, the beam amplitude.

To describe unit vectors $q_1, \ldots, q_m$ and d in $\mathbf{R}^3$ we will use azimuth and elevation, defined as follows:

$$q = \begin{bmatrix} \cos\phi\cos\theta \\ \cos\phi\sin\theta \\ \sin\phi \end{bmatrix}.$$

Here ϕ is the elevation (which will be between 0° and 90°, since all unit vectors in this problem have positive 3rd component, *i.e.*, point upward). The azimuth angle θ, which varies from 0° to 360°, gives the direction in the plane spanned by the first and second coordinates. If $q = e_3$ (*i.e.*, the direction is directly up), the azimuth is undefined.

(a) Explain how to do this, using a method or methods from this class. The simpler the method the better. If some matrix (or matrices) needs to be full rank for your method to work, say so.

(b) Carry out your method on the data given in `beam_estim_data.m`. This mfile defines `p`, the vector of photodetector outputs, a vector `det_az`, which gives the azimuth angles of the photodetector directions, and a vector `det_el`, which gives the elevation angles of the photodetector directions. Note that both of these are given in *degrees*, not radians. Give your final estimate of the beam amplitude a and beam direction d (in azimuth and elevation, in degrees).

6.27 *Smooth interpolation on a 2D grid.* This problem concerns arrays of real numbers on an $m \times n$ grid. Such as array can represent an image, or a sampled description of a function defined on a rectangle. We can describe such an array by a matrix $U \in \mathbf{R}^{m\times n}$, where U_{ij}

gives the real number at location i, j, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. We will think of the index i as associated with the y axis, and the index j as associated with the x axis.

It will also be convenient to describe such an array by a vector $u = \mathbf{vec}(U) \in \mathbf{R}^{mn}$. Here **vec** is the function that stacks the columns of a matrix on top of each other:

$$\mathbf{vec}(U) = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix},$$

where $U = [u_1 \cdots u_n]$. To go back to the array representation, from the vector, we have $U = \mathbf{vec}^{-1}(u)$. (This looks complicated, but isn't; $\mathbf{vec}^{-1}$ just arranges the elements in a vector into an array.)

We will need two linear functions that operate on $m \times n$ arrays. These are simple approximations of partial differentiation with respect to the x and y axes, respectively. The first function takes as argument an $m \times n$ array U and returns an $m \times (n-1)$ array V of forward (rightward) differences:

$$V_{ij} = U_{i,j+1} - U_{ij}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n-1.$$

We can represent this linear mapping as multiplication by a matrix $D_x \in \mathbf{R}^{m(n-1) \times mn}$, which satisfies

$$\mathbf{vec}(V) = D_x \mathbf{vec}(U).$$

(This looks scarier than it is—each row of the matrix D_x has exactly one $+1$ and one -1 entry in it.)

The other linear function, which is a simple approximation of partial differentiation with respect to the y axis, maps an $m \times n$ array U into an $(m-1) \times n$ array W, is defined as

$$W_{ij} = U_{i+1,j} - U_{ij}, \quad i = 1, \ldots, m-1, \quad j = 1, \ldots, n.$$

We define the matrix $D_y \in \mathbf{R}^{(m-1)n \times mn}$, which satisfies $\mathbf{vec}(W) = D_y \mathbf{vec}(U)$.

We define the *roughness* of an array U as

$$R = \|D_x \mathbf{vec}(U)\|^2 + \|D_y \mathbf{vec}(U)\|^2.$$

The roughness measure R is the sum of the squares of the differences of each element in the array and its neighbors. Small R corresponds to smooth, or smoothly varying, U. The roughness measure R is zero precisely for constant arrays, *i.e.*, when U_{ij} are all equal.

Now we get to the problem, which is to interpolate some unknown values in an array in the smoothest possible way, given the known values in the array. To define this precisely, we partition the set of indices $\{1, \ldots, mn\}$ into two sets: I_{known} and I_{unknown}. We let $k \geq 1$ denote the number of known values (*i.e.*, the number of elements in I_{known}), and $mn - k$ the number of unknown values (the number of elements in I_{unknown}). We are given the values u_i for $i \in I_{\text{known}}$; the goal is to guess (or estimate or assign) values for u_i for $i \in I_{\text{unknown}}$. We'll choose the values for u_i, with $i \in I_{\text{unknown}}$, so that the resulting U is as smooth as possible, *i.e.*, so it minimizes R. Thus, the goal is to fill in or interpolate missing data in a 2D array (an image, say), so the reconstructed array is as smooth as possible.

We give the k known values in a vector $w_{\text{known}} \in \mathbf{R}^k$, and the $mn - k$ unknown values in a vector $w_{\text{unknown}} \in \mathbf{R}^{mn-k}$. The complete array is obtained by putting the entries of w_{known} and w_{unknown} into the correct positions of the array. We describe these operations using two matrices $Z_{\text{known}} \in \mathbf{R}^{mn \times k}$ and $Z_{\text{unknown}} \in \mathbf{R}^{mn \times (mn-k)}$, that satisfy

$$\mathbf{vec}(U) = Z_{\text{known}} w_{\text{known}} + Z_{\text{unknown}} w_{\text{unknown}}.$$

(This looks complicated, but isn't: Each row of these matrices is a unit vector, so multiplication with either matrix just stuffs the entries of the w vectors into particular locations in $\mathbf{vec}(U)$. In fact, the matrix $[Z_{\text{known}}\ Z_{\text{unknown}}]$ is an $mn \times mn$ permutation matrix.)

In summary, you are given the problem data w_{known} (which gives the known array values), Z_{known} (which gives the locations of the known values), and Z_{unknown} (which gives the locations of the unknown array values, in some specific order). Your job is to find w_{unknown} that minimizes R.

(a) Explain how to solve this problem. You are welcome to use any of the operations, matrices, and vectors defined above in your solution (*e.g.*, $\mathbf{vec}$, $\mathbf{vec}^{-1}$, D_x, D_y, Z_{known}, Z_{unknown}, w_{known}, $\ldots$). If your solution is valid provided some matrix is (or some matrices are) full rank, say so.

(b) Carry out your method using the data created by `smooth_interpolation.m`. The file gives m, n, w_{known}, Z_{known} and Z_{unknown}. This file also creates the matrices D_x and D_y, which you are welcome to use. (This was *very* nice of us, by the way.) You are welcome to look at the code that generates these matrices, but you do not need to understand it. For this problem instance, around 50% of the array elements are known, and around 50% are unknown.

The mfile also includes the original array `Uorig` from which we removed elements to create the problem. This is just so you can see how well your smooth reconstruction method does in reconstructing the original array. Of course, you cannot use `Uorig` to create your interpolated array `U`.

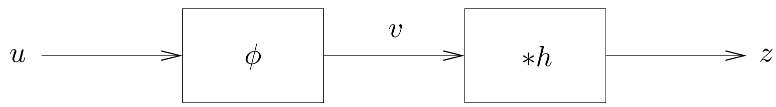
To visualize the arrays use the Matlab command `imagesc()`, with matrix argument. If you prefer a grayscale image, or don't have a color printer, you can issue the command `colormap gray`. The mfile that gives the problem data will plot the original image `Uorig`, as well as an image containing the known values, with zeros substituted for the unknown locations. This will allow you to see the pattern of known and unknown array values.

Compare `Uorig` (the original array) and `U` (the interpolated array found by your method), using `imagesc()`. Hand in complete source code, as well as the plots. Be sure to give the value of roughness R of U.

Hints:

- In Matlab, $\mathbf{vec}(U)$ can be computed as `U(:)`;
- $\mathbf{vec}^{-1}(u)$ can be computed as `reshape(u,m,n)`.

6.28 *Designing a nonlinear equalizer from I/O data.* This problem concerns the discrete-time system shown below, which consists of a memoryless nonlinearity ϕ, followed by a convolution filter with finite impulse response h. The scalar signal u is the input, and the scalar signal z is the output.



What this means is

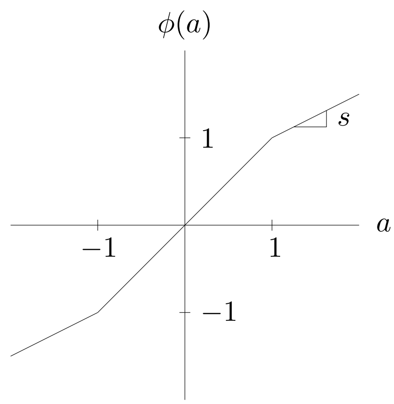
$$z(t) = \sum_{\tau=0}^{M-1} h(\tau)v(t-\tau), \qquad v(t) = \phi(u(t)), \qquad t \in \mathbf{Z}.$$

(Note that these signals are defined for all integer times, not just nonnegative times.)

Here $\phi : \mathbf{R} \to \mathbf{R}$, with the specific form

$$\phi(a) = \begin{cases} a & -1 \leq a \leq 1 \\ 1 - s + sa & a > 1 \\ -1 + s + sa & a < -1, \end{cases}$$

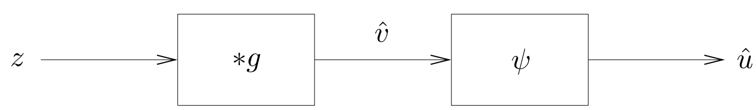
where $s > 0$ is a parameter. This function is shown below.



Here is an interpretation (that is not needed to solve the problem). The nonlinear function ϕ represents a power amplifier that is nonlinear for input signals larger than one in magnitude; s is called the *saturation gain* of the amplifier. The convolution system represents the transmission channel.

We are going to design an *equalizer* for the system, *i.e.*, another system that takes the signal z as input, and gives an output $\hat{u}$ which is an approximation of the input signal u.

Our equalizer will have the form shown below.



This means

$$\hat{v}(t) = \sum_{\tau=0}^{M-1} g(\tau) z(t-\tau), \qquad \hat{u}(t) = \psi(\hat{v}(t)), \qquad t \in \mathbf{Z}.$$

This equalizer will work well provided $g * h \approx \delta$ (in which case $\hat{v}(t) \approx v(t)$), and $\psi = \phi^{-1}$ (*i.e.*, $\psi(\phi(a)) = a$ for all a).

To make sure our (standard) notation here is clear, recall that

$$(g * h)(t) = \sum_{\tau=\max\{0, t-M+1\}}^{\min\{M-1, t\}} g(\tau) h(t-\tau), \quad t = 0, \ldots, 2M-1.$$

(Note: in matlab `conv(g,h)` gives the convolution of `g` and `h`, but these vectors are indexed from 1 to M, *i.e.*, `g(1)` corresponds to $g(0)$.) The term δ is the Kronecker delta, defined as $\delta(0) = 1$, $\delta(i) = 0$ for $i \neq 0$.

Now, finally, we come to the problem. You are given some input/output (I/O) data $u(1), \ldots, u(N)$, $z(1), \ldots, z(N)$, and M (the length of g, and also the length of h). You do *not* know the parameter s, or the channel impulse response $h(0), \ldots, h(M-1)$. You also don't know $u(t)$, $z(t)$ for $t \leq 0$.

(a) Explain how to find $\hat{s}$, an estimate of the saturation gain s, and $g(0), \ldots, g(M-1)$, that minimize

$$J = \frac{1}{N-M+1} \sum_{i=M}^{N} \Big(\hat{v}(i) - \phi(u(i))\Big)^2.$$

Here u refers to the given input data, and $\hat{v}$ comes from the given output data z. Note that if $g * h = \delta$ and $s = \hat{s}$, we have $J = 0$.

We exclude $i = 1, \ldots, M-1$ in the sum defining J because these terms depend (through $\hat{v}$) on $z(0), z(-1), \ldots$, which are unknown.

(b) Apply your method to the data given in the file `nleq_data.m`. Give the values of the parameters $\hat{s}$ and $g(0), \ldots, g(M-1)$ found, as well as J. Plot g using the matlab command `stem`.

(c) Using the values of $\hat{s}$ and $g(0), \ldots, g(M-1)$ found in part (b), find the equalized signal $\hat{u}(t)$, for $t = 1, \ldots, N$. For the purposes of finding $\hat{u}(t)$ you can assume that $z(t) = 0$ for $t \leq 0$. As a result, we can expect a large equalization error (*i.e.*, $\hat{u}(t) - u(t)$) for $t = 1, \ldots, M-1$.

Plot the input signal $u(t)$, the output signal $z(t)$, the equalized signal $\hat{u}(t)$, and the equalization error $\hat{u}(t) - u(t)$, for $t = 1, \ldots, N$.

Lecture 7 – Regularized least-squares and Gauss-Newton method

7.1 *Fitting a Gaussian function to data.* A Gaussian function has the form

$$f(t) = ae^{-(t-\mu)^2/\sigma^2}.$$

Here $t \in \mathbf{R}$ is the independent variable, and $a \in \mathbf{R}$, $\mu \in \mathbf{R}$, and $\sigma \in \mathbf{R}$ are parameters that affect its shape. The parameter a is called the *amplitude* of the Gaussian, μ is called its *center*, and σ is called the *spread* or *width*. We can always take $\sigma > 0$. For convenience we define $p \in \mathbf{R}^3$ as the vector of the parameters, *i.e.*, $p = [a\ \mu\ \sigma]^T$. We are given a set of data,

$$t_1, \ldots, t_N, \qquad y_1, \ldots, y_N,$$

and our goal is to fit a Gaussian function to the data. We will measure the quality of the fit by the root-mean-square (RMS) fitting error, given by

$$E = \left(\frac{1}{N} \sum_{i=1}^{N} (f(t_i) - y_i)^2 \right)^{1/2}.$$

Note that E is a function of the parameters a, μ, σ, *i.e.*, p. Your job is to choose these parameters to minimize E. You'll use the Gauss-Newton method.

(a) Work out the details of the Gauss-Newton method for this fitting problem. Explicitly describe the Gauss-Newton steps, including the matrices and vectors that come up. You can use the notation $\Delta p^{(k)} = [\Delta a^{(k)}\ \Delta \mu^{(k)}\ \Delta \sigma^{(k)}]^T$ to denote the update to the parameters, *i.e.*,

$$p^{(k+1)} = p^{(k)} + \Delta p^{(k)}.$$

(Here k denotes the kth iteration.)

(b) Get the data t, y (and N) from the file `gauss_fit_data.m`, available on the class website. Implement the Gauss-Newton (as outlined in part (a) above). You'll need an initial guess for the parameters. You can visually estimate them (giving a short justification), or estimate them by any other method (but you must explain your method). Plot the RMS error E as a function of the iteration number. (You should plot enough iterations to convince yourself that the algorithm has nearly converged.) Plot the final Gaussian function obtained along with the data on the same plot. Repeat for another reasonable, but different initial guess for the parameters. Repeat for another set of parameters that is *not* reasonable, *i.e.*, not a good guess for the parameters. (It's possible, of course, that the Gauss-Newton algorithm doesn't converge, or fails at some step; if this occurs, say so.) Briefly comment on the results you obtain in the three cases.

7.2 *E-911.* The federal government has mandated that cellular network operators must have the ability to locate a cell phone from which an emergency call is made. This problem concerns a simplified version of an E-911 system that uses time of arrival information at a number of base stations to estimate the cell phone location. A cell phone at location $x \in \mathbf{R}^2$ (we assume that the elevation is zero for simplicity) transmits an emergency signal at time τ. This signal is received at n base stations, located at locations $s_1, \ldots, s_n \in \mathbf{R}^2$. Each base station can measure the time of arrival of the emergency signal, within a few tens of nanoseconds. (This is possible because the base stations are synchronized using the Global Positioning System.) The measured times of arrival are

$$t_i = \frac{1}{c}\|s_i - x\| + \tau + v_i, \quad i = 1, \ldots, n,$$

where c is the speed of light, and v_i is the noise or error in the measured time of arrival. You can assume that v_i is on the order of a few tens of nanseconds. The problem is to estimate the cell phone position $x \in \mathbf{R}^2$, as well as the time of transmission τ, based on the time of arrival measurements $t_1, \ldots, t_n$. The mfile `e911_data.m`, available on the course web site, defines the data for this problem. Specifically, it defines a 2×9 matrix `S`, whose columns give the positions of the 9 basestations, a 1×9 vector `t` that contains the measured times of arrival, and the constant `c`, which is the speed of light. Distances are given in meters, times in nanoseconds, and the speed of light in meters/nanosecond. You can assume that the position x is somewhere in the box

$$|x_1| \leq 3000, \qquad |x_2| \leq 3000,$$

and that $|\tau| \leq 5000$ (although all that really matters are the time differences). Your solution must contain the following:

- An explanation of your approach to solving this problem, including how you will check that your estimate is reasonable.
- The matlab source code you use to solve the problem, and check the results.
- The numerical results obtained by your method, including the results of any verification you do.

7.3 *Curve-smoothing.* We are given a function $F : [0,1] \to \mathbf{R}$ (whose graph gives a curve in $\mathbf{R}^2$). Our goal is to find another function $G : [0,1] \to \mathbf{R}$, which is a *smoothed* version of F. We'll judge the smoothed version G of F in two ways:

- *Mean-square deviation from F,* defined as

$$D = \int_0^1 (F(t) - G(t))^2 \, dt.$$

- *Mean-square curvature,* defined as

$$C = \int_0^1 G''(t)^2 \, dt.$$

We want *both* D and C to be small, so we have a problem with two objectives. In general there will be a trade-off between the two objectives. At one extreme, we can choose $G = F$, which makes $D = 0$; at the other extreme, we can choose G to be an affine function (*i.e.*, to have $G''(t) = 0$ for all $t \in [0,1]$), in which case $C = 0$. The problem is to identify the optimal trade-off curve between C and D, and explain how to find smoothed functions G on the optimal trade-off curve. To reduce the problem to a finite-dimensional one, we will represent the functions F and G (approximately) by vectors $f,\ g \in \mathbf{R}^n$, where

$$f_i = F(i/n), \quad g_i = G(i/n).$$

You can assume that n is chosen large enough to represent the functions well. Using this representation we will use the following objectives, which approximate the ones defined for the functions above:

- *Mean-square deviation,* defined as

$$d = \frac{1}{n}\sum_{i=1}^{n}(f_i - g_i)^2.$$

- *Mean-square curvature,* defined as

$$c = \frac{1}{n-2}\sum_{i=2}^{n-1}\left(\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2}\right)^2.$$

In our definition of c, note that

$$\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2}$$

gives a simple approximation of $G''(i/n)$. You will only work with this approximate version of the problem, *i.e.*, the vectors f and g and the objectives c and d.

(a) Explain how to find g that minimizes $d + \mu c$, where $\mu \geq 0$ is a parameter that gives the relative weighting of sum-square curvature compared to sum-square deviation. Does your method always work? If there are some assumptions you need to make (say, on rank of some matrix, independence of some vectors, etc.), state them clearly. Explain how to obtain the two extreme cases: $\mu = 0$, which corresponds to minimizing d without regard for c, and also the solution obtained as $\mu \to \infty$ (*i.e.*, as we put more and more weight on minimizing curvature).

(b) Get the file `curve_smoothing.m` from the course web site. This file defines a specific vector f that you will use. Find and plot the optimal trade-off curve between d and c. Be sure to identify any critical points (such as, for example, any intersection of the curve with an axis). Plot the optimal g for the two extreme cases $\mu = 0$ and $\mu \to \infty$, and for three values of μ in between (chosen to show the trade-off nicely). On your plots of g, be sure to include also a plot of f, say with dotted line type, for reference. Submit your Matlab code.

7.4 *Optimal choice of initial temperature profile.* We consider a thermal system described by an n-element finite-element model. The elements are arranged in a line, with the temperature of element i at time t denoted $T_i(t)$. Temperature is measured in degrees Celsius above ambient; negative $T_i(t)$ corresponds to a temperature below ambient. The dynamics of the system are described by

$$c_1\dot{T}_1 = -a_1T_1 - b_1(T_1 - T_2),$$

$$c_i\dot{T}_i = -a_iT_i - b_i(T_i - T_{i+1}) - b_{i-1}(T_i - T_{i-1}), \quad i = 2, \ldots, n-1,$$

and

$$c_n\dot{T}_n = -a_nT_n - b_{n-1}(T_n - T_{n-1}).$$

where $c \in \mathbf{R}^n$, $a \in \mathbf{R}^n$, and $b \in \mathbf{R}^{n-1}$ are given and are all positive.

We can interpret this model as follows. The parameter c_i is the heat capacity of element i, so $c_i\dot{T}_i$ is the net heat flow into element i. The parameter a_i gives the thermal conductance between element i and the environment, so a_iT_i is the heat flow from element i to the environment (*i.e.*, the direct heat loss from element i.) The parameter b_i gives the thermal conductance between element i and element $i+1$, so $b_i(T_i - T_{i+1})$ is the heat flow from element i to element $i+1$. Finally, $b_{i-1}(T_i - T_{i-1})$ is the heat flow from element i to element $i-1$.

The goal of this problem is to choose the initial temperature profile, $T(0) \in \mathbf{R}^n$, so that $T(t^{\rm des}) \approx T^{\rm des}$. Here, $t^{\rm des} \in \mathbf{R}$ is a specific time when we want the temperature profile to closely match $T^{\rm des} \in \mathbf{R}^n$. We also wish to satisfy a constraint that $\|T(0)\|$ should be not be too large.

To formalize these requirements, we use the objective $(1/\sqrt{n})\|T(t^{\rm des}) - T^{\rm des}\|$ and the constraint $(1/\sqrt{n})\|T(0)\| \leq T^{\rm max}$. The first expression is the RMS temperature deviation, at $t = t^{\rm des}$, from the desired value, and the second is the RMS temperature deviation from ambient at $t = 0$. $T^{\rm max}$ is the (given) maximum inital RMS temperature value.

(a) Explain how to find $T(0)$ that minimizes the objective while satisfying the constraint.

(b) Solve the problem instance with the values of n, c, a, b, $t_{\rm des}$, $T^{\rm des}$ and $T^{\rm max}$ defined in the file `temp_prof_data.m`.

Plot, on one graph, your $T(0)$, $T(t^{\rm des})$ and $T^{\rm des}$. Give the RMS temperature error $(1/\sqrt{n})\|T(t^{\rm des}) - T^{\rm des}\|$, and the RMS value of initial temperature $(1/\sqrt{n})\|T(0)\|$.

Lecture 8 – Least-norm solutions of underdetermined equations

8.1 *Minimum fuel and minimum peak input solutions.* Suppose $A \in \mathbf{R}^{m\times n}$ is fat and full rank, so there are many x's that satisfy $Ax = y$. In lecture we encountered the *least-norm* solution given by $x_{\rm ln} = A^T(AA^T)^{-1}y$. This solution has the minimum (Euclidean) norm among all solutions of $Ax = y$. In many applications we want to minimize another norm of x (*i.e.*, measure of size of x) subject to $Ax = y$. Two common examples are the 1-norm and ∞-norm, which are defined as

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|, \qquad \|x\|_\infty = \max_{i=1,..,n} |x_i|.$$

The 1-norm, for example, is often a good measure of fuel use; the ∞-norm is the *peak* of the vector or signal x. There is no simple formula for the least 1-norm or ∞-norm solution of $Ax = y$, like there is for the least (Euclidean) norm solution. They can be computed very easily, however. (That's one of the topics of EE364.) The analysis is a bit trickier as well, since we can't just differentiate to verify that we have the minimizer. For example, how would you *know* that a solution of $Ax = y$ has minimum 1-norm? In this problem you will explore this idea. First verify the following inequality, which is like the Cauchy-Schwarz inequality (but even easier to prove): for any $v, w \in \mathbf{R}^p$, the following inequality holds: $w^Tv \le \|v\|_\infty \|w\|_1$. From this inequality it follows that whenever $v \neq 0$,

$$\|w\|_1 \ge \frac{w^Tv}{\|v\|_\infty}.$$

Now let z be any solution of $Az = y$, and let $\lambda \in \mathbf{R}^m$ be such that $A^T\lambda \neq 0$. Explain why we must have

$$\|z\|_1 \ge \frac{\lambda^T y}{\|A^T\lambda\|_\infty}.$$

Thus, any solution of $Az = y$ must have 1-norm at least as big as the righthand side expression. Therefore if you can find $x_{\rm mf} \in \mathbf{R}^n$ (mf stands for minimum fuel) and $\lambda \in \mathbf{R}^m$ such that $Ax_{\rm mf} = y$ and

$$\|x_{\rm mf}\|_1 = \frac{\lambda^T y}{\|A^T\lambda\|_\infty},$$

then $x_{\rm mf}$ is a minimum fuel solution. (Explain why.) Methods for computing $x_{\rm mf}$ and the mysterious vector λ are described in EE364. In the rest of this problem, you'll use these ideas to verify a statement made during lecture. Now consider the problem from the lecture notes of a unit mass acted on by forces $x_1, \ldots, x_{10}$ for one second each. The mass starts at position $p(0) = 0$ with zero velocity and is required to satisfy $p(10) = 1$, $\dot{p}(10) = 0$. There are, of course, many force vectors that satisfy these requirements. In the lecture notes, you can see a plot of the least (Euclidean) norm force profile. In class I stated that the minimum fuel solution is given by $x_{\rm mf} = (1/9, 0, \ldots, 0, -1/9)$, *i.e.*, an accelerating force at the beginning, 8 seconds of coasting, and a (braking) force at the end to decelerate the mass to zero velocity at $t = 10$. Prove this. *Hint:* try $\lambda = (1, -5)$. Verify that the 1-norm of $x_{\rm mf}$ is less than the 1-norm of $x_{\rm ln}$, the (Euclidean) least-norm solution. Feel free to use Matlab. There are several convenient ways to find the 1- and ∞-norm of a vector z, *e.g.*, `norm(z,1)` and `norm(z,inf)` or `sum(abs(z))` and `max(abs(z))`. One last question, for fun: what do you think is the minimum peak force vector $x_{\rm mp}$? How would you verify that a vector $x_{\rm mp}$ (mp for minimum peak) is a minimum ∞-norm solution of $Ax = y$? This input, by the way, is very widely used in practice. It is (basically) the input used in a disk drive to move the head from one track to another, while respecting a maximum possible current in the disk drive motor coil. *Hints:*

- The input is called *bang-bang*.
- Some people drive this way.

8.2 *Simultaneous left inverse of two matrices.* Consider a system where

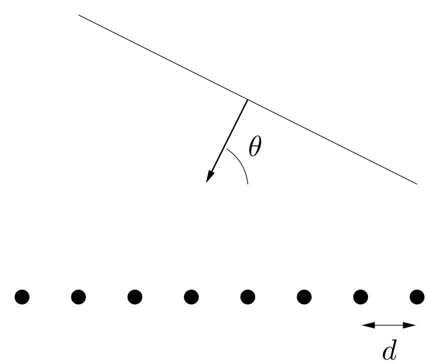
$$y = Gx, \quad \tilde{y} = \tilde{G}x$$

where $G \in \mathbf{R}^{m\times n}$, $\tilde{G} \in \mathbf{R}^{m\times n}$. Here x is some variable we wish to estimate or find, y gives the measurements with some set of (linear) sensors, and $\tilde{y}$ gives the measurements with some *alternate* set of (linear) sensors. We want to find a *reconstruction matrix* $H \in \mathbf{R}^{n\times m}$ such that $HG = H\tilde{G} = I$. Such a reconstruction matrix has the nice property that it recovers x perfectly from *either* set of measurements (y or $\tilde{y}$), *i.e.*, $x = Hy = H\tilde{y}$. Consider the specific case

$$G = \begin{bmatrix} 2 & 3 \\ 1 & 0 \\ 0 & 4 \\ 1 & 1 \\ -1 & 2 \end{bmatrix}, \quad \tilde{G} = \begin{bmatrix} -3 & -1 \\ -1 & 0 \\ 2 & -3 \\ -1 & -3 \\ 1 & 2 \end{bmatrix}.$$

Either find an explicit reconstruction matrix H, or explain why there is no such H.

8.3 *Phased-array antenna weight design.* We consider the phased-array antenna system shown below.



The array consists of n individual antennas (called *antenna elements*) spaced on a line, with spacing d between elements. A sinusoidal plane wave, with wavelength λ and angle of arrival θ, impinges on the array, which yields the output $e^{2\pi j(k-1)(d/\lambda)\cos\theta}$ (which is a complex number) from the kth element. (We've chosen the *phase center* as element 1, *i.e.*, the output of element 1 does not depend on the incidence angle θ.) A (complex) linear combination of these outputs is formed, and called the *combined array output*,

$$y(\theta) = \sum_{k=1}^{n} w_k e^{2\pi j(k-1)(d/\lambda)\cos\theta}.$$

The complex numbers $w_1, \ldots, w_n$, which are the coefficients of the linear combination, are called the *antenna weights*. We can choose, *i.e.*, design, the weights. The combined array output depends on the angle of arrival of the wave. The function $|y(\theta)|$, for $0^\circ \leq \theta \leq 180^\circ$, is called the antenna array *gain pattern*. By choosing the weights $w_1, \ldots, w_n$ intelligently, we can shape the gain pattern to satisfy some specifications. As a simple example, if we choose the weights as $w_1 = 1$, $w_2 = \cdots = w_n = 0$, then we get a uniform or omnidirectional gain pattern; $|y(\theta)| = 1$ for all θ. In this problem, we want a gain pattern that is one in a given ('target') direction θ_{target}, but small at other angles. Such a pattern would receive a signal coming from the direction θ_{target}, and attenuate signals (*e.g.*, 'jammers' or multipath reflections) coming from other directions. Here's the problem. You will design the weights for an array with $n = 20$ elements, and a spacing $d = 0.4\lambda$ (which is a typical value). We want $y(70^\circ) = 1$, and we want $|y(\theta)|$ small for $0^\circ \leq \theta \leq 60^\circ$ and $80^\circ \leq \theta \leq 180^\circ$. In other words, we want the antenna array to be relatively insensitive to plane waves arriving from angles more that 10° away from the target direction. (In the language of antenna arrays, we want a beamwidth of 20° around a target direction of 70°.) To solve this problem, you will first discretize the angles between 0° and 180° in 1° increments. Thus $y \in \mathbf{C}^{180}$ will be a (complex) vector, with y_k equal to $y(k^\circ)$, *i.e.*, $y(\pi k/180)$, for $k = 1, \ldots, 180$. You are to choose $w \in \mathbf{C}^{20}$ that minimizes

$$\sum_{k=1}^{60} |y_k|^2 + \sum_{k=80}^{180} |y_k|^2$$

subject to the constraint $y_{70} = 1$. As usual, you must explain how you solve the problem. Give the weights you find, and also a plot of the antenna array response, *i.e.*, $|y_k|$, versus k (which, hopefully, will achieve the desired goal of being relatively insensitive to plane waves arriving at angles more than 10° from $\theta = 70^\circ$). *Hints:*

- You'll probably want to rewrite the problem as one involving real variables (*i.e.*, the real and imaginary parts of the antenna weights), and real matrices. You can then rewrite your solution in a more compact formula that uses complex matrices and vectors (if you like).
- **Very important:** in Matlab, the prime is actually the Hermitian conjugate operator. In other words, if `A` is a complex matrix or vector, `A'` gives the conjugate transpose, or Hermitian conjugate, of `A`.
- Although we don't require you to, you might find it fun to also plot your antenna gain pattern on a polar plot, which allows you to easily visualize the pattern. In Matlab, this is done using the `polar` command.

8.4 *Modifying measurements to satisfy known conservation laws.* A vector $y \in \mathbf{R}^n$ contains n measurements of some physical quantities $x \in \mathbf{R}^n$. The measurements are good, but not perfect, so we have $y \approx x$. From physical principles it is known that the quantities x must satisfy some linear equations, *i.e.*,

$$a_i^T x = b_i, \qquad i = 1, \ldots, m,$$

where $m < n$. As a simple example, if x_1 is the current in a circuit flowing into a node, and x_2 and x_3 are the currents flowing out of the node, then we must have $x_1 = x_2 + x_3$. More generally, the linear equations might come from various conservation laws, or balance equations (mass, heat, energy, charge . . .). The vectors a_i and the constants b_i are known, and we assume that $a_1, \ldots, a_m$ are independent. Due to measurement errors, the measurement y won't satisfy the conservation laws (*i.e.*, linear equations above) exactly, although we would expect $a_i^T y \approx b_i$. An engineer proposes to adjust the measurements y by adding a correction term $c \in \mathbf{R}^n$, to get an adjusted estimate of x, given by

$$y_{\text{adj}} = y + c.$$

She proposes to find the smallest possible correction term (measured by $\|c\|$) such that the adjusted measurements y_{adj} satisfy the known conservation laws. Give an explicit formula for the correction term, in terms of y, a_i, b_i. If any matrix inverses appear in your formula, explain why the matrix to be inverted is nonsingular. Verify that the resulting adjusted measurement satisfies the conservation laws, *i.e.*, $a_i^T y_{\text{adj}} = b_i$.

8.5 *Estimator insensitive to certain measurement errors.* We consider the usual measurement setup: $y = Ax + v$, where

- $y \in \mathbf{R}^m$ is the vector of measurements
- $x \in \mathbf{R}^n$ is the vector of parameters we wish to estimate
- $v \in \mathbf{R}^m$ is the vector of measurement errors
- $A \in \mathbf{R}^{m \times n}$ is the coefficient matrix relating the parameters to the measurements

You can assume that $m > n$, and A is full rank. In this problem we assume that the measurement errors lie in the subspace

$$\mathcal{V} = \text{span}\{f_1, \ldots, f_k\},$$

where $f_1, \ldots, f_k \in \mathbf{R}^m$ are given, known vectors. Now consider a linear estimator of the form $\hat{x} = By$. Recall that the estimator is called *unbiased* if whenever $v = 0$, we have $\hat{x} = x$, for any $x \in \mathbf{R}^n$. In other words, an unbiased estimator predicts x perfectly when there is no measurement error. In this problem we consider the stronger condition that the estimator predicts x perfectly, for any measurement error in $\mathcal{V}$. In other words, we have $\hat{x} = x$, for any $x \in \mathbf{R}^n$, and any $v \in \mathcal{V}$. If this condition holds, we say that the estimator is insensitive to measurement errors in $\mathcal{V}$. (Note that this condition is a stronger condition than the estimator being unbiased.)

(a) Show that if $\mathcal{R}(A) \cap \mathcal{V} \neq \{0\}$, then there is no estimator insensitive to measurement errors in $\mathcal{V}$.

(b) Now we consider a specific example, with

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & -1 \\ 2 & 1 \\ -1 & 2 \end{bmatrix}, \qquad f_1 = \begin{bmatrix} 1 \\ 2 \\ -1 \\ 1 \\ 0 \end{bmatrix}, \qquad f_2 = \begin{bmatrix} 3 \\ 3 \\ 2 \\ 2 \\ 1 \end{bmatrix}.$$

Either construct a specific $B \in \mathbf{R}^{2 \times 5}$ for which the linear estimator $\hat{x} = By$ is insensitive to measurement errors in $\mathcal{V}$, or explain in detail why none exists. If you find such a B, you must explain how you found it, and verify (say, in Matlab) that it satisfies the required properties. (We'll be really annoyed if you just give a matrix and leave the verification to us!)

8.6 *Optimal flow on a data collection network.* We consider a communications network with m nodes, plus a special destination node, and n communication links. Each communication link connects two (distinct) nodes and is bidirectional, *i.e.*, information can flow in either direction. We will assume that the network is connected, *i.e.*, there is a path, or sequence of links, from every node (including the special destination node) to every other node. With each communication link we associate a directed arc, which defines the direction of information flow that we will call positive. Using these reference directions, the flow or traffic on link j is denoted f_j. (The units are bits per second, but that won't matter to us.) The traffic on the network (*i.e.*, the flow in each communication link) is given by a vector $f \in \mathbf{R}^n$. A small example is shown in part 2 of this problem. In this example, nodes 1 and 3 are connected by communication link 4, and the associated arc points from node 1 to node 3. Thus $f_4 = 12$ means the flow on that link is 12 (bits per second), from node 1 to node 3. Similarly, $f_4 = -3$ means the flow on link 4 is 3 (bits per second), from node 3 to node 1. External information enters each of the m regular nodes and flows across links to the special destination node. In other words, the network is used to collect information from the nodes and route it through the links to the special destination node. (That explains why we call it a data collection network.) At node i, an external information flow s_i (which is nonnegative) enters. The vector $s \in \mathbf{R}^m$ of external flows is sometimes called the *source vector*. Information flow is conserved. This means that at each node (except the special destination node) the sum of all flows entering the node from communication links connected to that node, plus the external flow, equals the sum of the flows leaving that node on communication links. As an example, consider node 3 in the network of part 2. Links 4 and 5 enter this node, and link 6 leaves the node. Therefore, flow conservation at node 3 is given by

$$f_4 + f_5 + s_3 = f_6.$$

The first two terms on the left give the flow entering node 3 on links 4 and 5; the last term on the left gives the external flow entering node 3. The term on the righthand side gives the flow leaving over link 6. Note that this equation correctly expresses flow conservation regardless of the signs of f_4, f_5, and f_6. Finally, here is the problem.

(a) The vector of external flows, $s \in \mathbf{R}^m$, and the network topology, are given, and you must find the flow f that satisfies the conservation equations, and minimizes the mean-square traffic on the network, *i.e.*,

$$\frac{1}{n}\sum_{j=1}^{n} f_j^2.$$

Your answer should be in terms of the external flow s, and the *node incidence matrix* $A \in \mathbf{R}^{m\times n}$ that describes the network topology. The node incidence matrix is defined as

$$A_{ij} = \begin{cases} 1 & \text{arc } j \text{ enters (or points into) node } i \\ -1 & \text{arc } j \text{ leaves (or points out of) node } i \\ 0 & \text{otherwise.} \end{cases}$$

Note that each row of A is associated with a node on the network (not including the destination node), and each column is associated with an arc or link.

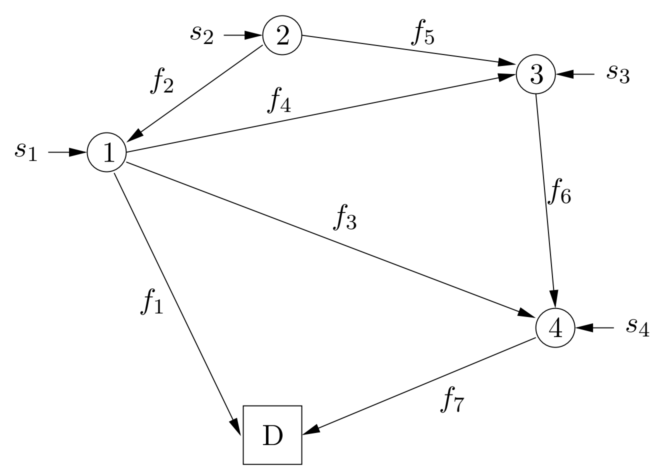
(b) Now consider the specific (and very small) network shown below. The nodes are shown as circles, and the special destination node is shown as a square. The external flows are

$$s = \begin{bmatrix} 1 \\ 4 \\ 10 \\ 10 \end{bmatrix}.$$

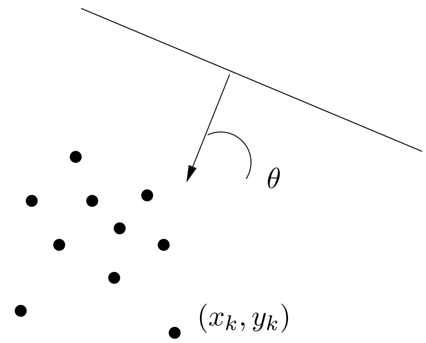
One simple feasible flow is obtained by routing all the external flow entering each node along a shortest path to the destination. For example, all the external flow entering node 2 goes to node 1, then to the destination node. For node 3, which has two shortest paths to the destination, we arbitrarily choose the path through node 4. This simple routing scheme results in the feasible flow

$$f_{\text{simple}} = \begin{bmatrix} 5 \\ 4 \\ 0 \\ 0 \\ 0 \\ 10 \\ 20 \end{bmatrix}.$$

Find the mean square optimal flow for this problem (as in part 1). Compare the mean square flow of the optimal flow with the mean square flow of f_{simple}.



8.7 *Random geometry antenna weight design.* We consider the phased-array antenna system shown below.



The array consists of n individual antennas (called *antenna elements*) randomly placed on $2d$ space, with the coordinates of the kth element being x_k and y_k. A sinusoidal plane wave, with wavelength λ and angle of arrival θ, impinges on the array, which yields the output $e^{j\frac{2\pi}{\lambda}(x_k \cos\theta + y_k \sin\theta)}$ (which is a complex number) from the kth element. A (complex) linear combination of these outputs is formed, and called the *combined array output*,

$$r(\theta) = \sum_{k=1}^{n} w_k e^{j\frac{2\pi}{\lambda}(x_k \cos\theta + y_k \sin\theta)}.$$

The complex numbers $w_1, \ldots, w_n$, which are the coefficients of the linear combination, are called the *antenna weights.* We can choose, *i.e.*, design, the weights. The combined array output depends on the angle of arrival of the wave. The function $|r(\theta)|$, for $0^\circ \le \theta < 360^\circ$, is called the antenna array *gain pattern.* By choosing the weights $w_1, \ldots, w_n$ intelligently, we can shape the gain pattern to satisfy some specifications. In this problem, we want a gain pattern that is one in a given ('target') direction θ_{target}, but small at other angles. Such a pattern would receive a signal coming from the direction θ_{target}, and attenuate signals (*e.g.*, 'jammers' or multipath reflections) coming from other directions. Here's the problem. Design the weights for the antenna array, whose elements have coordinates given in the file `antenna_geom.mat`. We want $r(70^\circ) = 1$, and we want $|r(\theta)|$ small for $0^\circ \le \theta \le 60^\circ$ and $80^\circ \le \theta < 360^\circ$. In other words, we want the antenna array to be relatively insensitive to plane waves arriving from angles more that 10° away from the target direction. (In the language of antenna arrays, we want a beamwidth of 20° around a target direction of 70°.) You are told that $\lambda = 1$. To solve this problem, you will first discretize the angles between 1° and 360° in 1° increments. Thus $r \in \mathbf{C}^{360}$ will be a (complex) vector, with r_k equal to $r(k^\circ)$, *i.e.*, $r(\pi k/180)$, for $k = 1, \ldots, 360$. You are to choose $w \in \mathbf{C}^n$ that minimizes

$$\sum_{k=1}^{60} |r_k|^2 + \sum_{k=80}^{360} |r_k|^2$$

subject to the constraint $r_{70} = 1$. As usual, you must explain how you solve the problem. Give the weights you find, and also a plot of the antenna array response, *i.e.*, $|r_k|$, versus k (which, hopefully, will achieve the desired goal of being relatively insensitive to plane waves arriving at angles more than 10° from $\theta = 70^\circ$). *Hints:*

- You'll probably want to rewrite the problem as one involving real variables (*i.e.*, the real and imaginary parts of the antenna weights), and real matrices. You can then rewrite your solution in a more compact formula that uses complex matrices and vectors (if you like).

- **Very important:** in Matlab, the prime is actually the Hermitian conjugate operator. In other words, if `A` is a complex matrix or vector, `A'` gives the conjugate transpose, or Hermitian conjugate, of `A`.
- Although we don't require you to, you might find it fun to also plot your antenna gain pattern on a polar plot, which allows you to easily visualize the pattern. In Matlab, this is done using the `polar` command.

8.8 *Estimation with known input norm.* We consider a standard estimation setup: $y = Ax + v$, where $A \in \mathbf{R}^{m\times n}$ is a full rank, skinny matrix, $x \in \mathbf{R}^n$ is the vector we wish to estimate, $v \in \mathbf{R}^m$ is an unknown noise vector, and $y \in \mathbf{R}^m$ is the measurement vector. As usual, we assume that smaller values of $\|v\|$ are more plausible than larger values. In this problem, we add one more piece of prior information: we know that $\|x\| = 1$. (In other words, the vector we are estimating is known ahead of time to have norm one.) This might occur in a communications system, where the transmitted signal power is known to be equal to one. (You may assume that the norm of the least-squares approximate solution exceeds one, *i.e.*, $\|(A^TA)^{-1}A^Ty\| > 1$.)

(a) Explain clearly how would you find the best estimate of x, taking into account the prior information $\|x\| = 1$. Explain how you would compute your estimate $\hat{x}$, given A and y. Is your estimate $\hat{x}$ a linear function of y?

(b) On the EE263 webpage, you will find the file `mtprob4.m`, which gives the matrix A and the observed vector y. Carry out the estimation procedure you developed in part (a). Give your estimate $\hat{x}$, and verify that it satisfies $\|\hat{x}\| = 1$. Give the Matlab source you use to compute $\hat{x}$.

8.9 *Minimum energy rendezvous.* The dynamics of two vehicles, at sampling times $t = 0, 1, 2, \ldots$, are given by

$$x(t+1) = Ax(t) + bu(t), \qquad z(t+1) = Fz(t) + gv(t)$$

where

- $x(t) \in \mathbf{R}^n$ is the state of vehicle 1
- $z(t) \in \mathbf{R}^n$ is the state of vehicle 2
- $u(t) \in \mathbf{R}$ is the (scalar) input to vehicle 1
- $v(t) \in \mathbf{R}$ is the (scalar) input to vehicle 2

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

$$x(0) = x_0, \qquad z(0) = z_0.$$

We are interested in finding inputs for the two vehicles over the time interval $t = 0, 1, \ldots, N-1$ so that they *rendezvous* at state $w \in \mathbf{R}^n$ at time $t = N$, *i.e.*, $x(N) = w$, $z(N) = w$. (The point $w \in \mathbf{R}^n$ is called the *rendezvous point*.) You can select the inputs to the two vehicles,

$$u(0), u(1), \ldots, u(N-1), \qquad v(0), v(1), \ldots, v(N-1),$$

as well as the rendezvous point $w \in \mathbf{R}^n$. Among choices of u, v, and w that satisfy the rendezvous condition, we want the one that minimizes the total input energy defined as

$$E = \sum_{t=0}^{N-1} u(t)^2 + \sum_{t=0}^{N-1} v(t)^2.$$

Give explicit formulas for the optimal u, v, and w in terms of the problem data, *i.e.*, A, b, F, g, x_0, and z_0. If you need to assume that one or more matrices that arise in your solution are invertible, full rank, etc., that's fine, but be sure to make very clear what you're assuming.

8.10 *Least-norm solution of nonlinear equations.* Suppose $f : \mathbf{R}^n \to \mathbf{R}^m$ is a function, and $y \in \mathbf{R}^m$ is a vector, where $m < n$ (*i.e.*, x has larger dimension than y). We say that $x \in \mathbf{R}^n$ is a least-norm solution of $f(x) = y$ if for any $z \in \mathbf{R}^n$ that satisfies $f(z) = y$, we have $\|z\| \geq \|x\|$. When the function f is linear or affine (*i.e.*, linear plus a constant), the equations $f(x) = y$ are linear, and we know how to find the least-norm solution for such problems. In general, however, it is an extremely difficult problem to compute a least-norm solution to a set of nonlinear equations. There are, however, some good heuristic iterative methods that work well when the function f is not too far from affine, *i.e.*, its nonlinear terms are small compared to its linear and constant part. You may assume that you have a starting guess, which we'll call $x^{(0)}$. This guess doesn't necessarily satisfy the equations $f(x) = y$.

(a) Suggest an iterative method for (approximately) solving the nonlinear least-norm problem, starting from the initial guess $x^{(0)}$. Use the notation $x^{(k)}$ to denote the kth iteration of your method. Explain clearly how you obtain $x^{(k+1)}$ from $x^{(k)}$. If you need to make any assumptions about rank of some matrix, do so. (You don't have to worry about what happens if the matrix is not full rank.) Your method should have the property that $f(x^{(k)})$ converges to y as k increases. (In particular, we don't need to have the iterates satsfiy the nonlinear equations exactly.) Suggest a name for the method you invent. Your method should not be complicated or require a long explanation. You do not have to prove that the method converges, or that when it converges, it converges to a least-norm solution. All you have to do is suggest a sensible, simple method that ought to work well when f is not too nonlinear, and the starting guess $x^{(0)}$ is good.

(b) Now we consider a specific example, with the function $f : \mathbf{R}^5 \to \mathbf{R}^2$ given by

$$\begin{aligned} f_1(x) &= 2x_1 - 3x_3 + x_5 + 0.1x_1x_2 - 0.5x_2x_5, \\ f_2(x) &= -x_2 + x_3 - x_4 + x_5 - 0.6x_1x_4 + 0.3x_3x_4. \end{aligned}$$

Note that each component of f consists of a linear part, and also a quadratic part. Use the method you invented in part a to find the least-norm solution of

$$f(x) = y = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

(We repeat that you do not have to prove that the solution you found is really the least-norm one.) As initial guess, you can use the least-norm solution of the linear equations resulting if you ignore the quadratic terms in f. Make sure to turn in your Matlab code as well as to identify the least-norm x you find, its norm, and the equation residual, *i.e.*, $f(x) - y$ (which should be very small).

8.11 *The smoothest input that takes the state to zero.* We consider the discrete-time linear dynamical system $x(t+1) = Ax(t) + Bu(t)$, with

$$A = \begin{bmatrix} 1.0 & 0.5 & 0.25 \\ 0.25 & 0 & 1.0 \\ 1.0 & -0.5 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 1.0 \\ 0.1 \\ 0.5 \end{bmatrix}, \qquad x(0) = \begin{bmatrix} 25 \\ 0 \\ -25 \end{bmatrix}.$$

The goal is to choose an input sequence $u(0), u(1), \ldots, u(19)$ that yields $x(20) = 0$. Among the input sequences that yield $x(20) = 0$, we want the one that is *smoothest*, *i.e.*, that minimizes

$$J_{\text{smooth}} = \left(\frac{1}{20} \sum_{t=0}^{19} (u(t) - u(t-1))^2 \right)^{1/2},$$

where we take $u(-1) = 0$ in this formula. Explain how to solve this problem. Plot the smoothest input u_{smooth}, and give the associated value of J_{smooth}.

8.12 *Minimum energy input with way-point constraints.* We consider a vehicle that moves in $\mathbf{R}^2$ due to an applied force input. We will use a discrete-time model, with time index $k = 1, 2, \ldots$; time index k corresponds to time $t = kh$, where $h > 0$ is the sample interval. The position at time index k is denoted by $p(k) \in \mathbf{R}^2$, and the velocity by $v(k) \in \mathbf{R}^2$, for $k = 1, \ldots, K+1$. These are related by the equations

$$p(k+1) = p(k) + hv(k), \quad v(k+1) = (1-\alpha)v(k) + (h/m)f(k), \quad k = 1, \ldots, K,$$

where $f(k) \in \mathbf{R}^2$ is the force applied to the vehicle at time index k, $m > 0$ is the vehicle mass, and $\alpha \in (0,1)$ models drag on the vehicle: In the absence of any other force, the vehicle velocity decreases by the factor $1-\alpha$ in each time index. (These formulas are approximations of more accurate formulas that we will see soon, but for the purposes of this problem, we consider them exact.) The vehicle starts at the origin, at rest, *i.e.*, we have $p(1) = 0$, $v(1) = 0$. (We take $k = 1$ as the initial time, to simplify indexing.)

The problem is to find forces $f(1), \ldots, f(K) \in \mathbf{R}^2$ that minimize the cost function

$$J = \sum_{k=1}^{K} \|f(k)\|^2,$$

subject to *way-point constraints*

$$p(k_i) = w_i, \quad i = 1, \ldots, M,$$

where k_i are integers between 1 and K. (These state that at the time $t_i = hk_i$, the vehicle must pass through the location $w_i \in \mathbf{R}^2$.) Note that there is no requirement on the vehicle velocity at the way-points.

(a) Explain how to solve this problem, given all the problem data (*i.e.*, h, α, m, K, the way-points $w_1, \ldots, w_M$, and the way-point indices $k_1, \ldots, k_M$).

(b) Carry out your method on the specific problem instance with data $h = 0.1$, $m = 1$, $\alpha = 0.1$, $K = 100$, and the $M = 4$ way-points

$$w_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \quad w_2 = \begin{bmatrix} -2 \\ 3 \end{bmatrix}, \quad w_3 = \begin{bmatrix} 4 \\ -3 \end{bmatrix}, \quad w_4 = \begin{bmatrix} -4 \\ -2 \end{bmatrix},$$

with way-point indices $k_1 = 10$, $k_2 = 30$, $k_3 = 40$, and $k_4 = 80$.

Give the optimal value of J.

Plot $f_1(k)$ and $f_2(k)$ versus k, using

```
subplot(211); plot(f(1,:));
subplot(212); plot(f(2,:));
```

We assume here that `f` is a $2 \times K$ matrix, with columns $f(1), \dots, f(K)$.

Plot the vehicle trajectory, using `plot(p(1,:),p(2,:))`. Here `p` is a $2 \times (K+1)$ matrix with columns $p(1), \dots, p(K+1)$.

8.13 In this problem you will show that, for *any* matrix A, and any positive number μ, the matrices $A^TA + \mu I$ and $AA^T + \mu I$ are both invertible, and

$$(A^TA + \mu I)^{-1}A^T = A^T(AA^T + \mu I)^{-1}.$$

(a) Let's first show that $A^TA + \mu I$ is invertible, assuming $\mu > 0$. (The same argument, with A^T substituted for A, will show that $AA^T + \mu I$ is invertible.) Suppose that $(A^TA + \mu I)z = 0$. Multiply on the left by z^T, and argue that $z = 0$. This is what we needed to show. (Your job is to fill all details of the argument.)

(b) Now let's establish the identity above. First, explain why

$$A^T(AA^T + \mu I) = (A^TA + \mu I)A^T$$

holds. Then, multiply on the left by $(A^TA + \mu I)^{-1}$, and on the right by $(AA^T + \mu I)^{-1}$. (These inverses exist, by part (a).)

(c) Now assume that A is fat and full rank. Show that as μ tends to zero from above (*i.e.*, μ is positive) we have

$$(A^TA + \mu I)^{-1}A^T \to A^T(AA^T)^{-1}.$$

(This is asserted, but not shown, in the lecture notes on page 8-12.)

8.14 *Singularity of the KKT matrix.* This problem concerns the general norm minimization with equality constraints problem (described in the lectures notes on pages 8-13),

$$\begin{array}{ll} \text{minimize} & \|Ax - b\| \\ \text{subject to} & Cx = d, \end{array}$$

where the variable is $x \in \mathbf{R}^n$, $A \in \mathbf{R}^{m \times n}$, and $C \in \mathbf{R}^{k \times n}$. We asume that C is fat ($k \leq n$), *i.e.*, the number of equality constraints is no more than the number of variables.

Using Lagrange multipliers, we found that the solution can be obtained by solving the linear equations

$$\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} A^Tb \\ d \end{bmatrix}$$

for x and λ. (The vector x gives the solution of the norm minimization problem above.) The matrix above, which we will call $K \in \mathbf{R}^{(n+k)\times(n+k)}$, is called the *KKT matrix* for the problem. (KKT are the initials of some of the people who came up with the optimality conditions for a more general type of problem.)

One question that arises is, when is the KKT matrix K nonsingular? The answer is: K is nonsingular if and only if C is full rank and $\mathcal{N}(A) \cap \mathcal{N}(C) = \{0\}$.

You will fill in all details of the argument below.

(a) Suppose C is not full rank. Show that K is singular.

(b) Suppose that there is a nonzero $u \in \mathcal{N}(A) \cap \mathcal{N}(C)$. Use this u to show that K is singular.

(c) Suppose that K is singular, so there exists a nonzero vector $[u^T \ v^T]^T$ for which

$$\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 0.$$

Write this out as two block equations, $A^TAu + C^Tv = 0$ and $Cu = 0$. Conclude that $u \in \mathcal{N}(C)$. Multiply $A^TAu + C^Tv = 0$ on the left by u^T, and use $Cu = 0$ to conclude that $\|Au\| = 0$, which implies $u \in \mathcal{N}(A)$. Finish the argument that leads to the conclusion that either C is not full rank, or $\mathcal{N}(A) \cap \mathcal{N}(C) \neq \{0\}$.

Lecture 9 – Autonomous linear dynamical systems

9.1 *A simple population model.* We consider a certain population of fish (say) each (yearly) season. $x(t) \in \mathbf{R}^3$ will describe the population of fish at year $t \in \mathbf{Z}$, as follows:

- $x_1(t)$ denotes the number of fish less than one year old
- $x_2(t)$ denotes the number of fish between one and two years old
- $x_3(t)$ denotes the number of fish between two and three years

(We will ignore the fact that these numbers are integers.) The population evolves from year t to year $t+1$ as follows.

- The number of fish less than one year old in the next year $(t+1)$ is equal to the total number of offspring born during the current year. Fish that are less than one year old in the current year (t) bear no offspring. Fish that are between one and two years old in the current year (t) bear an average of 2 offspring each. Fish that are between two and three years old in the current year (t) bear an average of 1 offspring each.
- 40% of the fish less than one year old in the current year (t) die; the remaining 60% live on to be between one and two years old in the next year $(t+1)$. fish, and 50% of the old fish.
- 30% of the one-to-two year old fish in the current year die, and 70% live on to be two-to-three year old fish in the next year.
- All of the two-to-three year old fish in the current year die.

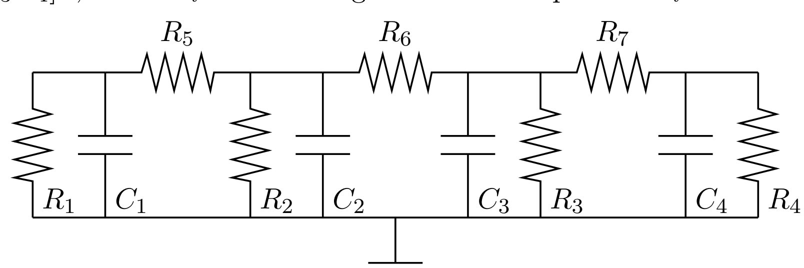
Express the population dynamics as an autonomous linear system with state $x(t)$, *i.e.*, in the form $x(t+1) = Ax(t)$. **Remark:** this example is silly, but more sophisticated population dynamics models are very useful and widely used.

9.2 *Tridiagonal systems.* A square matrix A is called tridiagonal if $A_{ij} = 0$ whenever $|i-j| > 1$. Tridiagonal matrices arise in many applications.

(a) Draw a pretty block diagram of $\dot{x} = Ax$, where $A \in \mathbf{R}^{4\times 4}$ is tridiagonal.

(b) Consider a Markov chain with four states labeled 1,2,3,4. Let $z(k)$ denote the state at time k. The state transition probabilities are described as follows: when z is not 4, it increases by one with probability 0.3; when z is not 1, it decreases by one with probability 0.2. (If z neither increases nor decreases, it stays the same, *i.e.*, $z(k+1) = z(k)$). Draw a graph of this Markov chain as in the lecture notes. Give the discrete time linear system equations that govern the evolution of the state distribution.

(c) Find the linear dynamical system description for the circuit shown below. Use state $x = [v_1\ v_2\ v_3\ v_4]^T$, where v_i is the voltage across the capacitor C_i.



9.3 *A distributed congestion control scheme.* A data network is modeled as a set of l directed links that connect n nodes. There are p *routes* in the network, which is a path from a *source node*, along one or more links in the network, to the *destination node*. The routes are determined and known. Each route has a *source rate* (in, say, bits per second). We denote the source rate for route j at time t as $x_j(t)$, $t = 0, 1, 2, \ldots$. (We assume the system operates in discrete time.) The total *traffic* on a link is the sum of the source rates for the routes that pass through it. We use $T_i(t)$ to denote the total traffic on link i at time t, for $i = 1, \ldots, l$. Each link has a *target traffic level*, which we denote T_i^{target}, $i = 1, \ldots, l$. We define the *congestion* on link i as $T_i(t) - T_i^{\text{target}}$, $i = 1, \ldots, l$. The congestion is positive if the traffic exceeds the target rate, and negative if it is below the target rate. The goal in *congestion control* is to adjust the source rates in such a way that the traffic levels converge to the target levels if possible, or close to the target levels otherwise. In this problem we consider a very simple congestion control protocol. Each route monitors the congestion for the links along its route. It then adjusts its source rate proportional to the sum of the congestion along its route. This can be expressed as:

$$x_j(t+1) = x_j(t) - \alpha \, (\text{sum of congestion along route } j)\,, \quad j = 1, \ldots, p,$$

where α is a positive scalar that determines how aggressively the source rates react to congestion. Note that this congestion control method is *distributed*; each source only needs to know the congestion along its own route, and does not directly coordinate its adjustments with the other routes. In real congestion control, the rates and traffic are nonnegative, and the traffic on each link must be below a maximum allowed level called the link capacity. In this problem, however, we ignore these effects; we do not take into account the link capacities, and allow the source rates and total traffic levels to become negative. Before we get to the questions, we define a matrix that may be useful. The *route-link matrix* $R \in \mathbf{R}^{l \times p}$, is defined as

$$R_{ij} = \begin{cases} 1 & \text{route } j \text{ utilizes link } i \\ 0 & \text{otherwise.} \end{cases}$$

(a) Show that $x(t)$, the vector of source rates, can be expressed as a linear dynamical system with constant input, *i.e.*, we have $x(t+1) = Ax(t) + b$. Be as explicit as you can about what A and b are. Try to use the simplest notation you can. *Hint:* use the matrix R.

(b) Simulate the congestion control algorithm for the network shown in figure 3, from two different initial source rates, using algorithm parameter $\alpha = 0.1$, and all target traffic levels equal to one. Plot the traffic level $T_i(t)$ for each link (on the same plot) versus t, for each of the initial source rates. (You are welcome to simulate the system from more than two initial source rates; we only ask you to hand in the plots for two, however.) Make a brief comment on the results.

(c) Now we come back to the general case (and *not* just the specific example from part (b)). Assume the congestion control update (*i.e.*, the linear dynamical system found in part (a)) has a unique equilibrium point $\bar{x}$, and that the rate $x(t)$ converges to it as $t \to \infty$. What can you say about $\bar{x}$? Limit yourself to a few sentences. Does the rate $\bar{x}$ always correspond to zero congestion on every link? Is it optimal in any way?

9.4 *Sequence transmission with constraints.* A communication system is based on 3 symbols: 1,

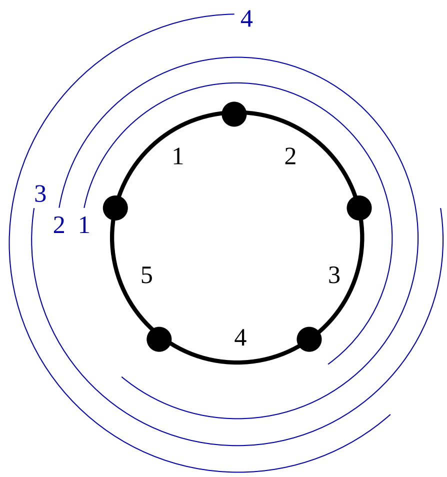


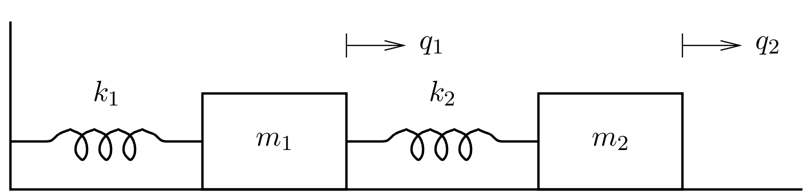
Figure 3: Data network for part (b), with links shown darker. Route 1 is $(1,2,3)$, route 2 is $(1,2,3,4)$, route 3 is $(3,4,5)$, and route 4 is $(4,5,1)$, where routes are defined as sequences of links. All traffic and routes flow counterclockwise (although this doesn't matter).

0, and -1. For this communication system, a valid sequence of symbols $x_1, x_2, \ldots, x_k$, must satisfy several constraints:

- *Transition constraint:* Consecutive symbols cannot be more than one apart from each other: we must have $|x_{i+1} - x_i| \leq 1$, for $i = 1, \ldots, k-1$. Thus, for example, a 0 or a 1 can follow a 1, but a -1 cannot follow a 1.
- *Power constraint:* The sum of the squares of any three consecutive symbols cannot exceed 2:
$$x_i^2 + x_{i+1}^2 + x_{i+2}^2 \leq 2,$$
for $i = 1, \ldots, k-2$.
- *Average constraint:* The sum of any three consecutive symbols must not exceed one in absolute value:
$$|x_i + x_{i+1} + x_{i+2}| \leq 1,$$
for $i = 1, \ldots, k-2$. So, for example, a sequence that contains 1100 would not be valid, because the sum of the first three consecutive symbols is 2.

How many different (valid) sequences of length 20 are there?

9.5 Consider the mechanical system shown below:



Here q_i give the displacements of the masses, m_i are the values of the masses, and k_i are the spring stiffnesses, respectively. The dynamics of this system are

$$\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1+k_2}{m_1} & \frac{k_2}{m_1} & 0 & 0 \\ \frac{k_2}{m_2} & -\frac{k_2}{m_2} & 0 & 0 \end{bmatrix} x$$

where the state is given by

$$x = \begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}.$$

Immediately before $t = 0$, you are able to apply a strong impulsive force α_i to mass i, which results in initial condition

$$x(0) = \begin{bmatrix} 0 \\ 0 \\ \alpha_1/m_1 \\ \alpha_2/m_2 \end{bmatrix}.$$

(i.e., each mass starts with zero position and a velocity determined by the impulsive forces.) This problem concerns selection of the impulsive forces α_1 and α_2. For parts a–c below, the parameter values are

$$m_1 = m_2 = 1, \quad k_1 = k_2 = 1.$$

Consider the following specifications:

(a) $q_2(10) = 2$

(b) $q_1(10) = 1$, $q_2(10) = 2$

(c) $q_1(10) = 1$, $q_2(10) = 2$, $\dot{q}_1(10) = 0$, $\dot{q}_2(10) = 0$

(d) $q_2(10) = 2$ when the parameters have the values used above (*i.e.*, $m_1 = m_2 = 1$, $k_1 = k_2 = 1$), and *also*, $q_2(10) = 2$ when the parameters have the values $m_1 = 1$, $m_2 = 1.3$, $k_1 = k_2 = 1$.

Determine whether each of these specifications is feasible or not (*i.e.*, whether there exist $\alpha_1, \alpha_2 \in \mathbf{R}$ that make the specification hold). If the specification is feasible, find the particular α_1, α_2 that satisfy the specification and minimize $\alpha_1^2+\alpha_2^2$. If the specification is infeasible, find the particular α_1, α_2 that come closest, in a least-squares sense, to satisfying the specification. (For example, if you cannot find α_1, α_2 that satisfy $q_1(10) = 1$, $q_2(10) = 2$, then find α_i that minimize $(q_1(10)-1)^2+(q_2(10)-2)^2$.) Be sure to be very clear about which alternative holds for each specification.

9.6 *Invariance of the unit square.* Consider the linear dynamical system $\dot{x} = Ax$ with $A \in \mathbf{R}^{2\times 2}$. The unit square in $\mathbf{R}^2$ is defined by

$$S = \{\, x \mid -1 \le x_1 \le 1, \ -1 \le x_2 \le 1 \,\}.$$

(a) Find the exact conditions on A for which the unit square S is invariant under $\dot{x} = Ax$. Give the conditions as explicitly as possible.

(b) Consider the following statement: if the eigenvalues of A are real and negative, then S is invariant under $\dot{x} = Ax$. Either show that this is true, or give an explicit counterexample.

9.7 *Iterative solution of linear equations.* In many applications we need to solve a set of linear equations $Ax = b$, where A is nonsingular (square) and x is very large (*e.g.*, $x \in \mathbf{R}^{100000}$). We assume that Az can be computed at reasonable cost, for any z, but the standard methods for computing $x = A^{-1}b$ (*e.g.*, LU decomposition) are not feasible. A common approach is to use an *iterative* method, which computes a sequence $x(1), x(2)$,... that *converges* to the solution $x = A^{-1}b$. These methods rely on another matrix $\hat{A}$, which is supposed to be 'close' to A. More importantly, $\hat{A}$ has the property that $\hat{A}^{-1}z$ is easily or cheaply computed for any given z. As a simple example, the matrix $\hat{A}$ might be the diagonal part of the matrix A (which, presumably, has relatively small off-diagonal elements). Obviously computing $\hat{A}^{-1}z$ is fast; it's just scaling the entries of z. There are many, many other examples. A simple iterative method, sometimes called *relaxation*, is to set $\hat{x}(0)$ equal to some approximation of x (*e.g.*, $\hat{x}(0) = \hat{A}^{-1}b$) and repeat, for $t = 0, 1, \ldots$

$$r(t) = A\hat{x}(t) - b; \qquad \hat{x}(t+1) = \hat{x}(t) - \hat{A}^{-1}r(t);$$

(The hat reminds us that $\hat{x}(t)$ is an approximation, after t iterations, of the true solution $x = A^{-1}b$.) This iteration uses only 'cheap' calculations: multiplication by A and $\hat{A}^{-1}$. Note that $r(t)$ is the residual after the tth iteration.

(a) Let $\beta = \|\hat{A}^{-1}(A - \hat{A})\|$ (which is a measure of how close $\hat{A}$ and A are). Show that if we choose $\hat{x}(0) = \hat{A}^{-1}b$, then $\|\hat{x}(t) - x\| \leq \beta^{t+1}\|x\|$. Thus if $\beta < 1$, the iterative method works, *i.e.*, for any b we have $\hat{x}(t) \to x$ as $t \to \infty$. (And if $\beta < 0.8$, say, then convergence is pretty fast.)

(b) Find the exact conditions on A and $\hat{A}$ such that the method works for any starting approximation $\hat{x}(0)$ and any b. Your condition can involve norms, singular values, condition number, and eigenvalues of A and $\hat{A}$, or some combination, etc. Your condition should be as explicit as possible; for example, it should not include any limits. Try to avoid the following two errors:

- Your condition guarantees convergence but is too restrictive. (For example: $\beta = \|\hat{A}^{-1}(A - \hat{A})\| < 0.8$)
- Your condition doesn't guarantee convergence.

9.8 *Periodic solution of periodic linear dynamical system.* Consider the linear dynamical system $\dot{x} = A(t)x$ where

$$A(t) = \begin{cases} A_1 & 2k \leq t < 2k+1, \quad k = 0, 1, 2, \ldots \\ A_2 & 2k+1 \leq t < 2k+2, \quad k = 0, 1, 2, \ldots \end{cases}$$

In other words, $A(t)$ switches between the two values $A_1 \in \mathbf{R}^{n\times n}$ and $A_2 \in \mathbf{R}^{n\times n}$ every second. The matrix $A(t)$ is periodic with period 2, *i.e.*, $A(t+2) = A(t)$ for all $t \geq 0$.

(a) *Existence of a periodic trajectory.* What are the conditions on A_1 and A_2 under which the system has a nonzero periodic trajectory, with period 2? By this we mean: there exists $x : \mathbf{R}_+ \to \mathbf{R}^n$, x not identically zero, with $x(t+2) = x(t)$ and $\dot{x} = A(t)x$.

(b) *All trajectories are asymptotically periodic.* What are the conditions on A_1 and A_2 under which all trajectories of the system are asymptotically 2-periodic? By this we mean: for every $x : \mathbf{R}_+ \to \mathbf{R}^n$ with $\dot{x} = A(t)x$, we have

$$\lim_{t\to\infty} \|x(t+2) - x(t)\| = 0.$$

(Note that this holds when x converges to zero ...)

Please note:

- Your conditions should be as explicit as possible. You can refer to the matrices A_1 and A_2, or any matrices derived from them using standard matrix operations, their eigenvalues and eigenvectors or Jordan forms, singular values and singular vectors, etc.
- We do not want you to give us *a* condition under which the property described holds. We want you to give us the most general conditions under which the property holds.

9.9 *Analysis of a power control algorithm.* In this problem we consider again the power control method described in homework problem 1. Please refer to this problem for the setup and background. In that problem, you expressed the power control method as a discrete-time linear dynamical system, and simulated it for a specific set of parameters, with several values of initial power levels, and two target SINRs. You found that for the target SINR value $\gamma = 3$, the powers converged to values for which each SINR exceeded γ, no matter what the initial power was, whereas for the larger target SINR value $\gamma = 5$, the powers appeared to diverge, and the SINRs did not appear to converge. You are going to analyze this, now that you know alot more about linear systems.

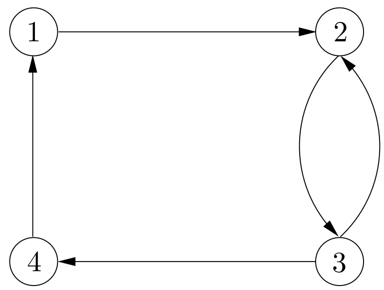
(a) *Explain the simulations.* Explain your simulation results from the problem 1(b) for the given values of G, α, σ, and the two SINR threshold levels $\gamma = 3$ and $\gamma = 5$.

(b) *Critical SINR threshold level.* Let us consider fixed values of G, α, and σ. It turns out that the power control algorithm works provided the SINR threshold γ is less than some critical value γ_{crit} (which might depend on G, α, σ), and doesn't work for $\gamma > \gamma_{\text{crit}}$. ('Works' means that no matter what the initial powers are, they converge to values for which each SINR exceeds γ.) Find an expression for γ_{crit} in terms of $G \in \mathbf{R}^{n\times n}$, α, and σ. Give the simplest expression you can. Of course you must explain how you came up with your expression.

9.10 *Paths and cycles in a directed graph.* We consider a directed graph with n nodes. The graph is specified by its *node adjacency matrix* $A \in \mathbf{R}^{n\times n}$, defined as

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge from node } j \text{ to node } i \\ 0 & \text{otherwise.} \end{cases}$$

Note that the edges are *oriented*, *i.e.*, $A_{34} = 1$ means there is an edge from node 4 to node 3. For simplicity we do not allow self-loops, *i.e.*, $A_{ii} = 0$ for all i, $1 \le i \le n$. A simple example illustrating this notation is shown below.



The node adjacency matrix for this example is

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

In this example, nodes 2 and 3 are connected in both directions, *i.e.*, there is an edge from 2 to 3 and also an edge from 3 to 2. A *path* of length $l > 0$ from node j to node i is a sequence $s_0 = j, s_1, \ldots, s_l = i$ of nodes, with $A_{s_{k+1},s_k} = 1$ for $k = 0, 1, \ldots, l-1$. For example, in the graph shown above, $1, 2, 3, 2$ is a path of length 3. A *cycle* of length l is a path of length l, with the same starting and ending node, with no repeated nodes other than the endpoints. In other words, a cycle is a sequence of nodes of the form $s_0, s_1, \ldots, s_{l-1}, s_0$, with

$$A_{s_1,s_0} = 1, \quad A_{s_2,s_1} = 1, \quad \ldots \quad A_{s_0,s_{l-1}} = 1,$$

and

$$s_i \neq s_j \text{ for } i \neq j, \quad i, j = 0, \ldots, l-1.$$

For example, in the graph shown above, $1, 2, 3, 4, 1$ is a cycle of length 4. The rest of this problem concerns a specific graph, given in the file `directed_graph.m` on the course web site. For each of the following questions, you must give the answer explicitly (for example, enclosed in a box). You must also explain clearly how you arrived at your answer.

(a) What is the length of a shortest cycle? (Shortest means minimum length.)

(b) What is the length of a shortest path from node 13 to node 17? (If there are no paths from node 13 to node 17, you can give the answer as 'infinity'.)

(c) What is the length of a shortest path from node 13 to node 17, that *does not* pass through node 3?

(d) What is the length of a shortest path from node 13 to node 17, that *does* pass through node 9?

(e) Among all paths of length 10 that start at node 5, find the most common ending node.

(f) Among all paths of length 10 that end at node 8, find the most common starting node.

(g) Among all paths of length 10, find the most common pair of starting and ending nodes. In other words, find i, j which maximize the number of paths of length 10 from i to j.

9.11 *Stability of a time-varying system.* We consider a discrete-time linear dynamical system

$$x(t+1) = A(t)x(t),$$

where $A(t) \in \{A_1, A_2, A_3, A_4\}$. These 4 matrices, which are 4×4, are given in `tv_data.m`.

Show that this system is stable, *i.e.*, for any trajectory x, we have $x(t) \to 0$ as $t \to \infty$. (This means that for any $x(0)$, and for any sequence $A(0), A(1), A(2), \ldots$, we have $x(t) \to 0$ as $t \to \infty$.)

You may use any methods or concepts used in the class, *e.g.*, least-squares, eigenvalues, singular values, controllability, and so on. Your proof will consist of two parts:

- An explanation of how you are going to show that any trajectory converges to zero. Your argument of course will require certain conditions (that you will find) to hold for the given data $A_1, \ldots, A_4$.
- The numerical calculations that verify the conditions hold for the given data. You must provide the source code for these calculations, and show the results as well.

Lecture 10 – Solution via Laplace transform and matrix exponential

10.1 Suppose $\dot{x} = Ax$ and $\dot{z} = \sigma z + Az = (A + \sigma I)z$ where $\sigma \in \mathbf{R}$, and $x(0) = z(0)$. How are $z(t)$ and $x(t)$ related? Find the simplest possible expression for $z(t)$ in terms of $x(t)$. Justify your answer. When $\sigma < 0$, some people refer to the system $\dot{z} = \sigma z + Az$ as a *damped version* of $\dot{x} = Ax$. Another way to think of the damped system is in terms of *leaky integrators.* A leaky integrator satisfies $\dot{y} - \sigma y = u$; to get the damped system, you replace every integrator in the original system with a leaky integrator.

10.2 *Harmonic oscillator.* The system $\dot{x} = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix} x$ is called a *harmonic oscillator.*

(a) Find the eigenvalues, resolvent, and state transition matrix for the harmonic oscillator. Express $x(t)$ in terms of $x(0)$.

(b) Sketch the vector field of the harmonic oscillator.

(c) The state trajectories describe circular orbits, *i.e.*, $\|x(t)\|$ is constant. Verify this fact using the solution from part (a).

(d) You may remember that circular motion (in a plane) is characterized by the velocity vector being orthogonal to the position vector. Verify that this holds for any trajectory of the harmonic oscillator. Use only the differential equation; do not use the explicit solution you found in part (a).

10.3 *Properties of the matrix exponential.*

(a) Show that $e^{A+B} = e^A e^B$ if A and B commute, *i.e.*, $AB = BA$.

(b) Carefully show that $\frac{d}{dt} e^{At} = Ae^{At} = e^{At}A$.

10.4 *Two-point boundary value problem.* Consider the system described by $\dot{x} = Ax$, where $A = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$.

(a) Find e^A.

(b) Suppose $x_1(0) = 1$ and $x_2(1) = 2$. Find $x(2)$. (This is called a *two-point boundary value problem*, since we are given conditions on the state at two time points instead of the usual single initial point.)

10.5 *Determinant of matrix exponential.*

(a) Suppose the eigenvalues of $A \in \mathbf{R}^{n \times n}$ are $\lambda_1, \ldots, \lambda_n$. Show that the eigenvalues of e^A are $e^{\lambda_1}, \ldots, e^{\lambda_n}$. You can assume that A is diagonalizable, although it is true in the general case.

(b) Show that $\det e^A = e^{\mathbf{Tr}\, A}$. *Hint:* $\det X$ is the product of the eigenvalues of X, and $\mathbf{Tr}\, Y$ is the sum of the eigenvalues of Y.

10.6 *Linear system with a quadrant detector.* In this problem we consider the specific system

$$\dot{x} = Ax = \begin{bmatrix} 0.5 & 1.4 \\ -0.7 & 0.5 \end{bmatrix} x.$$

We have a detector or sensor that gives us the sign of each component of the state $x = [x_1\ x_2]^T$ each second:

$$y_1(t) = \mathrm{sgn}(x_1(t)), \quad y_2(t) = \mathrm{sgn}(x_2(t)), \quad t = 0, 1, 2, \ldots$$

where the function sgn : $\mathbf{R} \to \mathbf{R}$ is defined by

$$\mathrm{sgn}(a) = \begin{cases} 1 & a > 0 \\ 0 & a = 0 \\ -1 & a < 0 \end{cases}$$

There are several ways to think of these sensor measurements. You can think of $y(t) = [y_1(t)\ y_2(t)]^T$ as determining which quadrant the state is in at time t (thus the name *quadrant detector*). Or, you can think of $y(t)$ as a one-bit quantized measurement of the state at time t. Finally, the problem. You observe the sensor measurements

$$y(0) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad y(1) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Based on these measurements, what values could $y(2)$ possibly take on? In terms of the quadrants, the problem can be stated as follows. $x(0)$ is in quadrant IV, and $x(1)$ is also in quadrant IV. The question is: which quadrant(s) can $x(2)$ possibly be in? You do not know the initial state $x(0)$. Of course, you must completely justify and explain your answer.

10.7 *Linear system with one-bit quantized output.* We consider the system

$$\dot{x} = Ax, \quad y(t) = \mathrm{sign}\,(cx(t))$$

where

$$A = \begin{bmatrix} -0.1 & 1 \\ -1 & 0.1 \end{bmatrix}, \quad c = \begin{bmatrix} 1 & -1 \end{bmatrix},$$

and the sign function is defined as

$$\mathrm{sign}(a) = \begin{cases} +1 & \text{if } a > 0 \\ -1 & \text{if } a < 0 \\ 0 & \text{if } a = 0 \end{cases}$$

Rougly speaking, the output of this autonomous linear system is quantized to one-bit precision. The following outputs are observed:

$$y(0.4) = +1, \quad y(1.2) = -1, \quad y(2.3) = -1, \quad y(3.8) = +1$$

What can you say (if anything) about the following:

$$y(0.7), \quad y(1.8), \quad \text{and } y(3.7)?$$

Your response might be, for example: "$y(0.7)$ is definitely $+1$, and $y(1.8)$ is definitely -1, but $y(3.7)$ can be anything (*i.e.*, -1, 0, or 1)". Of course you *must* fully explain how you arrive at your conclusions. (What we mean by "$y(0.7)$ is definitely $+1$" is: for any trajectory of the system for which $y(0.4) = +1$, $y(1.2) = -1$, $y(2.3) = -1$, and $y(3.8) = +1$, we also have $y(0.7) = +1$.)

10.8 *Some basic properties of eigenvalues.* Show that

(a) the eigenvalues of A and A^T are the same

(b) A is invertible if and only if A does not have a zero eigenvalue

(c) if the eigenvalues of A are $\lambda_1, \ldots, \lambda_n$ and A is invertible, then the eigenvalues of A^{-1} are $1/\lambda_1, \ldots, 1/\lambda_n$,

(d) the eigenvalues of A and $T^{-1}AT$ are the same.

Hint: you'll need to use the facts $\det AB = \det A \det B$ and $\det A^{-1} = 1/\det A$ (provided A is invertible).

10.9 *Characteristic polynomial.* Consider the characteristic polynomial $\mathcal{X}(s) = \det(sI - A)$ of the matrix $A \in \mathbf{R}^{n\times n}$.

(a) Show that $\mathcal{X}$ is *monic*, which means that its leading coefficient is one: $\mathcal{X}(s) = s^n + \cdots$.

(b) Show that the s^{n-1} coefficient of $\mathcal{X}$ is given by $-\mathbf{Tr}\, A$. ($\mathbf{Tr}\, X$ is the *trace* of a matrix: $\mathbf{Tr}\, X = \sum_{i=1}^n X_{ii}$.)

(c) Show that the constant coefficient of $\mathcal{X}$ is given by $\det(-A)$.

(d) Let $\lambda_1, \ldots, \lambda_n$ denote the eigenvalues of A, so that

$$\mathcal{X}(s) = s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0 = (s-\lambda_1)(s-\lambda_2)\cdots(s-\lambda_n).$$

By equating coefficients show that $a_{n-1} = -\sum_{i=1}^n \lambda_i$ and $a_0 = \prod_{i=1}^n (-\lambda_i)$.

10.10 *The adjoint system.* The adjoint system associated with the linear dynamical system $\dot{x} = Ax$ is $\dot{z} = A^T z$. Evidently the adjoint system and the system have the same eigenvalues.

(a) How are the state-transition matrices of the system and the adjoint system related?

(b) Show that $z(0)^T x(t) = z(t)^T x(0)$.

10.11 *Spectral resolution of the identity.* Suppose $A \in \mathbf{R}^{n\times n}$ has n linearly independent eigenvectors $p_1, \ldots, p_n$, $p_i^T p_i = 1$, $i = 1, \ldots, n$, with associated eigenvalues λ_i. Let $P = [p_1 \ \cdots \ p_n]$ and $Q = P^{-1}$. Let q_i^T be the ith row of Q.

(a) Let $R_k = p_k q_k^T$. What is the range of R_k? What is the rank of R_k? Can you describe the null space of R_k?

(b) Show that $R_i R_j = 0$ for $i \neq j$. What is R_i^2?

(c) Show that

$$(sI - A)^{-1} = \sum_{k=1}^n \frac{R_k}{s - \lambda_k}.$$

Note that this is a partial fraction expansion of $(sI - A)^{-1}$. For this reason the R_i's are called the *residue* matrices of A.

(d) Show that $R_1 + \cdots + R_n = I$. For this reason the residue matrices are said to constitute a *resolution of the identity.*

(e) Find the residue matrices for

$$A = \begin{bmatrix} 1 & 0 \\ 1 & -2 \end{bmatrix}$$

both ways described above (*i.e.*, find P and Q and then calculate the R's, and then do a partial fraction expansion of $(sI - A)^{-1}$ to find the R's).

10.12 *Using Matlab to find an invariant plane.* Consider the continuous-time system $\dot{x} = Ax$ with A given by

$$A = \begin{bmatrix} -0.1005 & 1.0939 & 2.0428 & 4.4599 \\ -1.0880 & -0.1444 & 5.9859 & -3.0481 \\ -2.0510 & -5.9709 & -0.1387 & 1.9229 \\ -4.4575 & 3.0753 & -1.8847 & -0.1164 \end{bmatrix}$$

You can verify that the eigenvalues of A are

$$\lambda_{1,2} = -0.10 \pm j5, \quad \lambda_{3,4} = -0.15 \pm j7.$$

(a) Find an orthonormal basis (q_1, q_2) for the invariant plane associated with λ_1 and λ_2.

(b) Find $q_3, q_4 \in \mathbf{R}^4$ such that $Q = [q_1 \; q_2 \; q_3 \; q_4]$ is orthogonal. You might find the Matlab command `null` useful; it computes an orthonormal basis of the null space of a matrix.

(c) Plot the individual states constituting the trajectory $x(t)$ of the system starting from an initial point in the invariant plane, say $x(0) = q_1$, for $0 \le t \le 40$.

(d) If $x(t)$ is in the invariant plane what can you say about the components of the vector $Q^T x(t)$?

(e) Using the result of part (12d) verify that the trajectory you found in part (12c) is in the invariant plane.

Note: The A matrix is available on the class web site in the file `inv_plane_matrix.m`.

10.13 *Positive quadrant invariance.* We consider a system $\dot{x} = Ax$ with $x(t) \in \mathbf{R}^2$ (although the results of this problem can be generalized to systems of higher dimension). We say the system is *positive quadrant invariant* (PQI) if whenever $x_1(T) \ge 0$ and $x_2(T) \ge 0$, we have $x_1(t) \ge 0$ and $x_2(t) \ge 0$ for all $t \ge T$. In other words, if the state starts inside (or enters) the positive (*i.e.*, first) quadrant, then the state remains indefinitely in the positive quadrant.

(a) Find the precise conditions on A under which the system $\dot{x} = Ax$ is PQI. Try to express the conditions in the simplest form.

(b) *True or False:* if $\dot{x} = Ax$ is PQI, then the eigenvalues of A are real.

10.14 *Some Matlab exercises.* Consider the continuous-time system $\dot{x} = Ax$ with

$$A = \begin{bmatrix} -0.1005 & 1.0939 & 2.0428 & 4.4599 \\ -1.0880 & -0.1444 & 5.9859 & -3.0481 \\ -2.0510 & -5.9709 & -0.1387 & 1.9229 \\ -4.4575 & 3.0753 & -1.8847 & -0.1164 \end{bmatrix}.$$

(a) What are the eigenvalues of A? Is the system stable? You can use the command `eig` in Matlab.

(b) Plot a few trajectories of $x(t)$, *i.e.*, $x_1(t)$, $x_2(t)$, $x_3(t)$ and $x_4(t)$, for a few initial conditions. To do this you can use the matrix exponential command in Matlab `expm` (*not* `exp` which gives the element-by-element exponential of a matrix), or more directly, the Matlab command `initial` (use `help initial` for details.) Verify that the qualitative behavior of the system is consistent with the eigenvalues you found in part (14a).

(c) Find the matrix Z such that $Zx(t)$ gives $x(t+15)$. Thus, Z is the '15 seconds forward predictor matrix'.

(d) Find the matrix Y such that $Yx(t)$ gives $x(t-20)$. Thus Y reconstructs what the state was 20 seconds ago.

(e) Briefly comment on the size of the elements of the matrices Y and Z.

(f) Find $x(0)$ such that $x(10) = [1\ 1\ 1\ 1]^T$.

Note: The A matrix is available on the class web page.

10.15 *Volume preserving flows.* Suppose we have a set $S \subseteq \mathbf{R}^n$ and a linear dynamical system $\dot{x} = Ax$. We can propagate S along the 'flow' induced by the linear dynamical system by considering

$$S(t) = e^{At}S = \{\ e^{At}s \mid s \in S\ \}.$$

Thus, $S(t)$ is the image of the set S under the linear transformation e^{tA}. What are the conditions on A so that the flow preserves volume, *i.e.*, $\mathbf{vol}\, S(t) = \mathbf{vol}\, S$ for all t? Can the flow $\dot{x} = Ax$ be stable? *Hint:* if $F \in \mathbf{R}^{n\times n}$ then $\mathbf{vol}(FS) = |\det F|\,\mathbf{vol}\, S$, where $FS = \{\ Fs \mid s \in S\ \}$.

10.16 *Stability of a periodic system.* Consider the linear dynamical system $\dot{x} = A(t)x$ where

$$A(t) = \begin{cases} A_1 & 2n \le t < 2n+1, \quad n = 0, 1, 2, \ldots \\ A_2 & 2n+1 \le t < 2n+2, \quad n = 0, 1, 2, \ldots \end{cases}$$

In other words, $A(t)$ switches between the two values A_1 and A_2 every second. We say that this (time-varying) linear dynamical system is stable if every trajectory converges to zero, *i.e.*, we have $x(t) \to 0$ as $t \to \infty$ for any $x(0)$. Find the conditions on A_1 and A_2 under which the periodic system is stable. Your conditions should be as explicit as possible.

10.17 *Computing trajectories of a continuous-time LDS.* We have seen in class that if $x(t)$ is the solution to the continuous-time, time-invariant, linear dynamical system

$$\dot{x} = Ax, \quad x(0) = x_0,$$

then the Laplace transform of $x(t)$ is given by

$$X(s) = (sI - A)^{-1}\, x_0.$$

Hence, we can obtain $x(t)$ from the inverse Laplace transform of the resolvent of A:

$$x(t) = \mathcal{L}^{-1}\left((sI - A)^{-1}\right) x_0.$$

(a) Assuming that $A \in \mathbf{R}^{n\times n}$ has n independent eigenvectors, write $x(t)$ in terms of the residue matrices R_i and associated eigenvalues λ_i, $i = 1, \ldots, n$. (The residue matrices are defined in the previous problem.)

(b) Consider once again the matrix

$$A = \begin{bmatrix} 1 & 3 \\ 0 & -1 \end{bmatrix}.$$

Write the solution $x(t)$ for this dynamics matrix, with the initial condition $x_0 = [\,2 \quad -1\,]^T$. Compute $x_1(2)$, *i.e.*, the value of the first entry of $x(t)$ at $t = 2$.

(c) *Forward Euler approximation.* With this same A and x_0, compute an approximation to the trajectory $x(t)$ by Euler approximation, with different step-sizes h. Run your simulation from $t = 0$ to $t = 2$, with N steps. For the number of steps N, use the values 10, 100, 1000, and 10000 (with the corresponding step-size $h = 2/N$). For each run, you'll obtain the sequence resulting from the discrete-time LDS

$$y(k+1) = (I + hA)y(k), \quad k = 0, \ldots, N-1$$

with $y(0) = x_0$. On the same graph, plot the first entry, $y_1(k)$, of each of the four sequences you obtain (with hk on the horizontal axis).

(d) *Error in Euler approximation.* For each of the four runs, compute the final error in x_1, given by $\epsilon = y_1(N) - x_1(2)$. Plot ϵ as a function of N on a logarithmic scale (*hint:* use the Matlab function `loglog`). How many steps do you estimate you would you need to achieve a precision of 10^{-6}?

(e) *Matrix exponential.* The matrix exponential is defined by the series

$$e^A = I + \sum_{k=1}^{+\infty} \frac{1}{k!} A^k.$$

With A as above and $h = 0.5$, compute an approximation of the matrix exponential of hA by adding the first ten term of the series:

$$B = I + \sum_{k=1}^{10} \frac{1}{k!} (hA)^k.$$

Compute 4 iterates of the discrete-time LDS

$$z(k+1) = Bz(k), \quad k = 0, \ldots, 3,$$

with $z(0) = x_0$. Add $z_1(k)$ to the plot of the $y_1(k)$. What is the final error $\epsilon = z_1(4) - x_1(2)$? *Note:* The Matlab function `expm` uses a much more efficient algorithm to compute the matrix exponential. For this example, `expm` requires about the same computational effort as is needed to add the first ten terms of the series, but the result is much more accurate. (If you're curious, go ahead and compute the corresponding final error ϵ.)

10.18 Suppose $\dot{x} = Ax$ with $A \in \mathbf{R}^{n\times n}$. Two one-second experiments are performed. In the first, $x(0) = [1\ 1]^T$ and $x(1) = [4\ \ -2]^T$. In the second, $x(0) = [1\ 2]^T$ and $x(1) = [5\ \ -2]^T$.

(a) Find $x(1)$ and $x(2)$, given $x(0) = [3\ \ -1]^T$.

(b) Find A, by first computing the matrix exponential.

(c) Either find $x(1.5)$ or explain why you cannot ($x(0) = [3\ \ -1]^T$).

(d) More generally, for $\dot{x} = Ax$ with $A \in \mathbf{R}^{n\times n}$, describe a procedure for finding A using experiments with different initial values. What conditions must be satisfied for your procedure to work?

10.19 *Output response envelope for linear system with uncertain initial condition.* We consider the autonomous linear dynamical system $\dot{x} = Ax$, $y(t) = Cx(t)$, where $x(t) \in \mathbf{R}^n$ and $y(t) \in \mathbf{R}$. We do not know the initial condition exactly; we only know that it lies in a ball of radius r centered at the point x_0:

$$\|x(0) - x_0\| \leq r.$$

We call x_0 the *nominal* initial condition, and the resulting output, $y_{\mathrm{nom}}(t) = Ce^{tA}x_0$, the *nominal output.* We define the *maximum output* or *upper output envelope* as

$$\overline{y}(t) = \max\{y(t) \mid \|x(0) - x_0\| \leq r\},$$

i.e., the maximum possible value of the output at time t, over all possible initial conditions. (Here you can choose a different initial condition for each t; you are not required to find a single initial condition.) In a similar way, we define the *minimum output* or *lower output envelope* as

$$\underline{y}(t) = \min\{y(t) \mid \|x(0) - x_0\| \leq r\},$$

i.e., the minimum possible value of the output at time t, over all possible initial conditions.

(a) Explain how to find $\overline{y}(t)$ and $\underline{y}(t)$, given the problem data A, C, x_0, and r.

(b) Carry out your method on the problem data in `uie_data.m`. On the same axes, plot y_{nom}, $\overline{y}$, and $\underline{y}$, versus t, over the range $0 \leq t \leq 10$.

10.20 *Alignment of a fleet of vehicles.* We consider a fleet of vehicles, labeled $1, \ldots, n$, which move along a line with (scalar) positions $y_1, \ldots, y_n$. We let $v_1, \ldots, v_n$ denote the velocities of the vehicles, and $u_1, \ldots, u_n$ the net forces applied to the vehicles. The vehicle motions are governed by the equations

$$\dot{y}_i = v_i, \qquad \dot{v}_i = u_i - v_i.$$

(Here we take each vehicle mass to be one, and include a damping term in the equations.) We assume that $y_1(0) < \cdots < y_n(0)$, *i.e.*, the vehicles start out with vehicle 1 in the leftmost position, followed by vehicle 2 to its right, and so on, with vehicle n in the rightmost position. The goal is for the vehicles to converge to the configuration

$$y_i = i, \quad v_i = 0, \quad i = 1, \ldots, n,$$

i.e., first vehicle at position 1, with unit spacing between adjacent vehicles, and all stationary. We call this configuration *aligned*, and the goal is to drive the vehicles to this configuration, *i.e.*, to align the vehicles. We define the spacing between vehicle i and $i+1$ as $s_i(t) = y_{i+1}(t) - y_i(t)$, for $i = 1, \ldots, n-1$. (When the vehicles are aligned, these spacings are all one.) We will investigate three control schemes for aligning the fleet of vehicles.

- *Right looking control* is based on the spacing to the vehicle to the right. We use the control law

$$u_i(t) = s_i(t) - 1, \quad i = 1, \dots, n-1,$$

for vehicles $i = 1, \dots, n-1$. In other words, we apply a force on vehicle i proportional to its spacing error with respect to the vehicle to the right (*i.e.*, vehicle $i+1$). The rightmost vehicle uses the control law

$$u_n(t) = -(y_n(t) - n),$$

which applies a force proportional to its position error, in the opposite direction. This control law has the advantage that only the rightmost vehicle needs an absolute measurement sensor; the others only need a measurement of the distance to their righthand neighbor.

- *Left and right looking control* adjusts the input force based on the spacing errors to the vehicle to the left and the vehicle to the right:

$$u_i(t) = \frac{s_i(t) - 1}{2} - \frac{s_{i-1}(t) - 1}{2}, \quad i = 2, \dots, n-1,$$

The rightmost vehicle uses the same absolute error method as in right looking control, *i.e.*,

$$u_n(t) = -(y_n(t) - n),$$

and the first vehicle, which has no vehicle to its left, uses a right looking control scheme,

$$u_1(t) = s_1(t) - 1.$$

This scheme requires vehicle n to have an absolute position sensor, but the other vehicles only need to measure the distance to their neighbors.

- *Independent alignment* is based on each vehicle independently adjusting its position with respect to its required position:

$$u_i(t) = -(y_i(t) - i), \quad i = 1, \dots, n.$$

This scheme requires all vehicles to have absolute position sensors.

In the questions below, we consider the specific case with $n = 5$ vehicles.

(a) Which of the three schemes work? By 'work' we mean that the vehicles converge to the alignment configuration, no matter what the initial positions and velocities are. Among the schemes that do work, which one gives the fastest asymptotic convergence to alignment? (If there is a tie between two or three schemes, say so.) In this part of the problem you can ignore the issue of vehicle collisions, *i.e.*, spacings that pass through zero.

(b) *Collisions.* In this problem we analyze vehicle collisions, which occur when any spacing between vehicles is equal to zero. (For example, $s_3(5.7) = 0$ means that vehicles 3 and 4 collide at $t = 5.7$.) We take the particular starting configuration

$$y = (0, 2, 3, 5, 7), \qquad v = (0, 0, 0, 0, 0),$$

which corresponds to the vehicles with zero initial velocity, but not in the aligned positions. For each of the three schemes above (whether or not they work), determine if a collision occurs. If a collision does occur, find the earliest collision, giving the time and the vehicles involved. (For example, 'Vehicles 3 and 4 collide at $t = 7.7$.') If there is a tie, *i.e.*, two pairs of vehicles collide at the same time, say so. If the vehicles do not collide, find the point of closest approach, *i.e.*, the minimum spacing that occurs, between any pair of vehicles, for $t \geq 0$. (Give the time, the vehicles involved, and the minimum spacing.) If there is a tie, *i.e.*, two or more pairs of vehicles have the same distance of closest approach, say so. Be sure to give us times of collisions or closest approach with an absolute precision of at least 0.1.

Lecture 11 – Eigenvectors and diagonalization

11.1 *Left eigenvector properties.* Suppose w is a left eigenvector of $A \in \mathbf{R}^{n\times n}$ with real negative eigenvalue λ.

(a) Find a simple expression for $w^T e^{At}$.

(b) Let $\alpha < \beta$. The set $\{ z \mid \alpha \leq w^T z \leq \beta \}$ is referred to as a *slab*. Briefly explain this terminology. Draw a picture in $\mathbf{R}^2$.

(c) Show that the slab $\{ z \mid 0 \leq w^T z \leq \beta \}$ is invariant for $\dot{x} = Ax$.

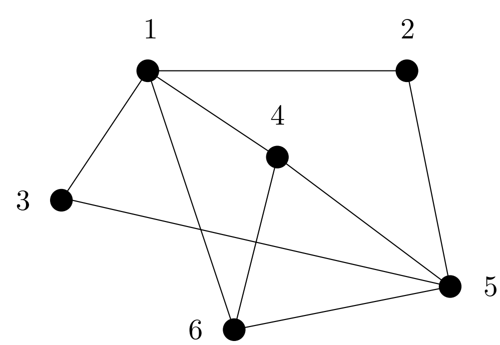
11.2 Consider the linear dynamical system $\dot{x} = Ax$ where $A \in \mathbf{R}^{n\times n}$ is diagonalizable with eigenvalues λ_i, eigenvectors v_i, and left eigenvectors w_i for $i = 1, \ldots, n$. Assume that $\lambda_1 > 0$ and $\Re\lambda_i < 0$ for $i = 2, \ldots, n$. Describe the trajectories qualitatively. Specifically, what happens to $x(t)$ as $t \to \infty$? Give the answer geometrically, in terms of $x(0)$.

11.3 *Another formula for the matrix exponential.* You might remember that for any complex number $a \in \mathbf{C}$, $e^a = \lim_{k\to\infty}(1 + a/k)^k$. You will establish the matrix analog: for any $A \in \mathbf{R}^{n\times n}$,

$$e^A = \lim_{k\to\infty} (I + A/k)^k.$$

To simplify things, you can assume A is diagonalizable. *Hint:* diagonalize.

11.4 *Synchronizing a communication network.* The graph below shows a communication network, with communication links shown as lines between the nodes, which are labeled $1, \ldots, 6$. We refer to one node as a *neighbor* of another if they are connected by a link.



Each node has a clock. The clocks run at the same speed, but are not (initially) synchronized. The shift or offset of clock i, with respect to some absolute clock (*e.g.*, NIST's atomic clocks or the clock for the GPS system) will be denoted x_i. Thus $x_i > 0$ means the clock at node i is running in advance of the standard clock, while $x_i < 0$ means the ith clock is running behind the standard clock. The nodes *do not* know their own clock offsets (or the offsets of any of the other clocks); we introduce the numbers x_i only so we can analyze the system. At discrete intervals, which we denote $t = 0, 1, 2 \ldots$, the nodes exchange communications messages. Through this exchange each node is able to find out the *relative time offset* of its own clock compared to the clocks of its neighboring nodes. For example, node 2 is able to find out the differences $x_1 - x_2$ and $x_5 - x_2$. (But remember, node 2 does not know any of the absolute clock offsets x_1, x_2, or x_5.) While node i does not know its absolute offset x_i, it is able to *adjust it* by adding a delay or advance to it. The new offset takes effect at the next interval. Thus we have $x_i(t+1) = x_i(t) + a_i(t)$, where $a_i(t)$ is the adjustment made by the ith node to its clock in the tth interval. An engineer suggests the following scheme of adjusting the clock offsets. At each interval, each node determines its relative offset with each of its neighboring nodes. Then it computes the average of these relative offsets. The node then adjusts its offset by this average. For example, for node 2 we would have the adjustment

$$a_2(t) = \frac{(x_1(t) - x_2(t)) + (x_5(t) - x_2(t))}{2}.$$

Finally, the question.

(a) What happens?

(b) Why?

We are interested in questions such as: do all the clocks become synchronized with the standard clock (*i.e.*, $x(t) \to 0$ as $t \to \infty$)? Do the clocks become synchronized with each other (*i.e.*, do all $x_i(t) - x_j(t)$ converge to zero as $t \to \infty$)? Does the system become synchronized no matter what the initial offsets are, or only for some initial offsets? You are welcome to use Matlab to do some relevant numerical computations, but you *must* explain what you are doing and why. We will *not* accept simulations of the network as an explanation. Another engineer suggests a modification of the scheme described above. She notes that if the scheme above were applied to a simple network consisting of two connected nodes, then the two nodes would just trade their offsets each time interval, so synchronization does not occur. To avoid this, she proposes to adjust each node's clock by only half the average offset with its neighbors. Thus, for node 2, this means:

$$a_2(t) = \frac{1}{2}\,\frac{(x_1(t) - x_2(t)) + (x_5(t) - x_2(t))}{2}.$$

(c) Would you say this scheme is better or worse than the original one described above? If one is better than the other, how is it better? (For example, does it achieve synchronization from a bigger set of initial offsets, does it achieve synchronization faster, etc.)

11.5 *Population dynamics.* In this problem we will study how some population distribution (say, of people) evolves over time, using a discrete-time linear dynamical system model. Let $t = 0, 1, \ldots$ denote time in years (since the beginning of the study). The vector $x(t) \in \mathbf{R}^n$ will give the population distribution at year t (on some fixed census date, *e.g.*, January 1). Specifically, $x_i(t)$ is the number of people at year t, of age $i-1$. Thus $x_5(3)$ denotes the number of people of age 4, at year 3, and $x_1(t)$ (the number of 0 year-olds) denotes the number of people born since the last census. We assume n is large enough that no one lives to age n. We'll also ignore the fact that x_i are integers, and treat them as real numbers. (If $x_3(4) = 1.2$ bothers you, you can imagine the units as millions, say.) The total population at year t is given by $\mathbf{1}^T x(t)$, where $\mathbf{1} \in \mathbf{R}^n$ is the vector with all components 1.

- *Death rate.* The death rate depends only on age, and not on time t. The coefficient d_i is the fraction of people of age $i-1$ who will die during the year. Thus we have, for $t = 0, 1, \ldots,$

$$x_{k+1}(t+1) = (1 - d_k)x_k(t), \quad k = 1, \ldots, n-1.$$

(As mentioned above, we assume that $d_n = 1$, *i.e.*, all people who make it to age $n-1$ die during the year.) The death rate coefficients satisfy $0 < d_i < 1$, $i = 1, \ldots, n-1$. We define the *survival rate* coefficients as $s_k = 1 - d_k$, so $0 < s_k < 1$, $k = 1, \ldots, n-1$.

- *Birth rate.* The birth rate depends only on age, and not on time t. The coefficient b_i is the fraction of people of age $i-1$ who will have a child during the year (taking into account multiple births). Thus the total births during a year is given by

$$x_1(t+1) = b_1 x_1(t) + \cdots + b_n x_n(t).$$

  The birth rate coefficients satisfy $b_i \geq 0$, $i = 1, \ldots, n$. We'll assume that at least one of the b_k's is positive. (Of course you'd expect that b_i would be zero for non-fertile ages, *e.g.*, age below 11 and over 60, but we won't make that explicit assumption.)

The assumptions imply the following important property of our model: if $x_i(0) > 0$ for $i = 1, \ldots, n$, then $x_i(t) > 0$ for $i = 1, \ldots, n$. Therefore we don't have to worry about negative $x_i(t)$, so long as our initial population distribution $x(0)$ has all positive components. (To use fancy language we'd say the system is *positive orthant invariant*.)

(a) Express the population dynamics model described above as a discrete-time linear dynamical system. That is, find a matrix A such that $x(t+1) = Ax(t)$.

(b) Draw a block diagram of the system found in part (a).

(c) Find the characteristic polynomial of the system explicitly in terms of the birth and death rate coefficients (or, if you prefer, the birth and survival rate coefficients).

(d) *Survival normalized variables.* For each person born, s_1 make it to age 1, $s_1 s_2$ make it to age 2, and in general, $s_1 \cdots s_k$ make it to age k. We define

$$y_k(t) = \frac{x_k(t)}{s_1 \cdots s_{k-1}}$$

(with $y_1(t) = x_1(t)$) as new population variables that are normalized to the survival rate. Express the population dynamics as a linear dynamical system using the variable $y(t) \in \mathbf{R}^n$. That is, find a matrix $\tilde{A}$ such $y(t+1) = \tilde{A} y(t)$.

Determine whether each of the next four statements is true or false. (Of course by 'true' we mean true for any values of the coefficients consistent with our assumptions, and by 'false' we mean false for some choice of coefficients consistent with our assumptions.)

(e) Let x and z both satisfy our population dynamics model, *i.e.*, $x(t+1) = Ax(t)$ and $z(t+1) = Az(t)$, and assume that all components of $x(0)$ and $z(0)$ are positive. If $\mathbf{1}^T x(0) > \mathbf{1}^T z(0)$, then $\mathbf{1}^T x(t) > \mathbf{1}^T z(t)$ for $t = 1, 2, \ldots$. (In words: we consider two populations that satisfy the same dynamics. Then the population that is initially larger will always be larger.)

(f) All the eigenvalues of A are real.

(g) If $d_k \geq b_k$ for $k = 1, \ldots, n$, then $\mathbf{1}^T x(t) \to 0$ as $t \to \infty$, *i.e.*, the population goes extinct.

(h) Suppose that $(b_1 + \cdots + b_n)/n \leq (d_1 + \cdots d_n)/n$, *i.e.*, the 'average' birth rate is less than the 'average' death rate. Then $\mathbf{1}^T x(t) \to 0$ as $t \to \infty$.

11.6 *Rate of a Markov code.* Consider the Markov language described in exercise 13, with five symbols $1, 2, 3, 4, 5$, and the following symbol transition rules:

- 1 must be followed by 2 or 3
- 2 must be followed by 2 or 5
- 3 must be followed by 1
- 4 must be followed by 4 or 2 or 5
- 5 must be followed by 1 or 3

(a) *The rate of the code.* Let K_N denote the number of allowed sequences of length N. The number

$$R = \lim_{N\to\infty} \frac{\log_2 K_N}{N}$$

(if it exists) is called the *rate* of the code, in bits per symbol. Find the rate of this code. Compare it to the rate of the code which consists of all sequences from an alphabet of 5 symbols (*i.e.*, with no restrictions on which symbols can follow which symbols).

(b) *Asymptotic fraction of sequences with a given starting or ending symbol.* Let $F_{N,i}$ denote the number of allowed sequences of length N that start with symbol i, and let $G_{N,i}$ denote the number of allowed sequences of length N that end with symbol i. Thus, we have

$$F_{N,1} + \cdots + F_{N,5} = G_{N,1} + \cdots + G_{N,5} = K_N.$$

Find the asymptotic fractions

$$f_i = \lim_{N\to\infty} F_{N,i}/K_N, \quad g_i = \lim_{N\to\infty} G_{N,i}/K_N.$$

Please don't find your answers by simple simulation or relatively mindless computation; we want to see (and understand) your method.

11.7 *Companion matrices.* A matrix A of the form

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

is said to be a (top) *companion matrix.* There can be four forms of companion matrices depending on whether the a_i's occur in the first or last row, or first or last column. These are referred to as top-, bottom-, left-, or right-companion matrices. Let $\dot{x} = Ax$ where A is top-campanion.

(a) Draw a block diagram for the system $\dot{x} = Ax$.

(b) Find the characteristic polynomial of the system using the block diagram and show that A is nonsingular if and only if $a_n \neq 0$.

(c) Show that if A is nonsingular, then A^{-1} is a bottom-companion matrix with last row $-[1\ a_1\ \cdots\ a_{n-1}]/a_n$.

(d) Find the eigenvector of A associated with the eigenvalue λ.

(e) Suppose that A has distinct eigenvalues $\lambda_1, \ldots, \lambda_n$. Find T such that $T^{-1}AT$ is diagonal.

11.8 *Squareroot and logarithm of a (diagonalizable) matrix.* Suppose that $A \in \mathbf{R}^{n\times n}$ is diagonalizable. Therefore, an invertible matrix $T \in \mathbf{C}^{n\times n}$ and diagonal matrix $\Lambda \in \mathbf{C}^{n\times n}$ exist such that $A = T\Lambda T^{-1}$. Let $\Lambda = \mathbf{diag}(\lambda_1, \ldots, \lambda_n)$.

(a) We say $B \in \mathbf{R}^{n\times n}$ is a squareroot of A if $B^2 = A$. Let μ_i satisfy $\mu_i^2 = \lambda_i$. Show that $B = T\,\mathbf{diag}(\mu_1, \ldots, \mu_n)T^{-1}$ is a squareroot of A. A squareroot is sometimes denoted $A^{1/2}$ (but note that there are in general many squareroots of a matrix). When λ_i are real and nonnegative, it is conventional to take $\mu_i = \sqrt{\lambda_i}$ (*i.e.*, the nonnegative squareroot), so in this case $A^{1/2}$ is unambiguous.

(b) We say B is a logarithm of A if $e^B = A$, and we write $B = \log A$. Following the idea of part a, find an expression for a logarithm of A (which you can assume is invertible). Is the logarithm unique? What if we insist on B being real?

11.9 *Separating hyperplane for a linear dynamical system.* A *hyperplane* (passing through 0) in $\mathbf{R}^n$ is described by the equation $c^Tx = 0$, where $c \in \mathbf{R}^n$ is nonzero. (Note that if $\beta \neq 0$, the vector $\tilde{c} = \beta c$ defines the same hyperplane.) Now consider the autonomous linear dynamic system $\dot{x} = Ax$, where $A \in \mathbf{R}^{n\times n}$ and $x(t) \in \mathbf{R}^n$. We say that the hyperplane defined by c is a *separating hyperplane* for this system if no trajectory of the system ever crosses the hyperplane. This means it is impossible to have $c^Tx(t) > 0$ for some t, and $c^Tx(\tilde{t}) < 0$ for some other $\tilde{t}$, for any trajectory x of the system. Explain how to find *all* separating hyperplanes for the system $\dot{x} = Ax$. In particular, give the conditions on A under which there is no separating hyperplane. (If you think there is always a separating hyperplane for a linear system, say so.) You can assume that A has distinct eigenvalues (and therefore is diagonalizable).

11.10 *Equi-angle sets.* Let $x_1, \ldots, x_n \in \mathbf{R}^n$. We say that they form a (normalized) *equi-angle set*, with angle θ, if $\|x_i\| = 1$, $i = 1, \ldots, n$, and

$$\angle(x_i, x_j) = \theta, \quad i, j = 1, \ldots, n, \quad i \neq j.$$

In other words, each of the vectors has unit norm, and the angle between any pair of the vectors is θ. We'll take θ to be between 0 and π. An orthonormal set is a familiar example of an equi-angle set, with $\theta = \pi/2$. In $\mathbf{R}^2$, there are equi-angle sets for every value of θ. It's easy to find such sets: just take

$$x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad x_2 = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}.$$

In $\mathbf{R}^n$, with $n > 2$, however, you can't have an equi-angle set with angle $\theta = \pi$. To see this, suppose that $x_1, \ldots, x_n$ is an equi-angle set in $\mathbf{R}^n$, with $n > 2$. Then we have $x_2 = -x_1$ (since $\angle(x_1, x_2) = \pi$), but also $x_3 = -x_1$ (since $\angle(x_1, x_3) = \pi$), so $\angle(x_2, x_3) = 0$. The question then arises, for what values of θ (between 0 and π) can you have an equi-angle set on $\mathbf{R}^n$? The angle $\theta = 0$ always has an equi-angle set (just choose any unit vector u and set $x_1 = \cdots = x_n = u$), and so does $\theta = \pi/2$ (just choose any orthonormal basis, *e.g.*, $e_1, \ldots, e_n$. But what other angles are possible? For $n = 2$, we know the answer: any value of θ between 0 and π is possible, *i.e.*, for every value of θ there is an equi-angle set with angle θ.

(a) For general n, describe the values of θ for which there is an equi-angle set with angle θ. In particular, what is the maximum possible value θ can have?

(b) Construct a specific equi-angle set in $\mathbf{R}^4$ for angle $\theta = 100^\circ = 5\pi/9$. Attach Matlab output to verify that your four vectors are unit vectors, and that the angle between any two of them is 100°. (Since $\angle(u, v) = \angle(v, u)$, you only have to check 6 angles. Also, you might find a clever way to find all the angles at once.)

11.11 *Optimal control for maximum asymptotic growth.* We consider the controllable linear system

$$x(t+1) = Ax(t) + Bu(t), \qquad x(0) = 0,$$

where $x(t) \in \mathbf{R}^n$, $u(t) \in \mathbf{R}^m$. You can assume that A is diagonalizable, and that it has a single dominant eigenvalue (which here, means that there is one eigenvalue with largest magnitude). An input $u(0), \ldots, u(T-1)$ is applied over time period $0, 1, \ldots, T-1$; for $t \geq T$, we have $u(t) = 0$. The input is subject to a total energy constraint:

$$\|u(0)\|^2 + \cdots + \|u(T-1)\|^2 \leq 1.$$

The goal is to choose the inputs $u(0), \ldots, u(T-1)$ that maximize the norm of the state for large t. To be more precise, we're searching for $u(0), \ldots, u(T-1)$, that satisfies the total energy constraint, and, for any other input sequence $\tilde{u}(0), \ldots, \tilde{u}(T-1)$ that satisfies the total energy constraint, satisfies $\|x(t)\| \geq \|\tilde{x}(t)\|$ for t large enough. Explain how to do this. You can use any of the ideas from the class, *e.g.*, eigenvector decomposition, SVD, pseudo-inverse, etc. Be sure to summarize your final description of how to solve the problem. Unless you have to, you should *not* use limits in your solution. For example you cannot explain how to make $\|x(t)\|$ as large as possible (for a specific value of t), and then say, "Take the limit as $t \to \infty$" or "Now take t to be really large".

11.12 *Estimating a matrix with known eigenvectors.* This problem is about estimating a matrix $A \in \mathbf{R}^{n \times n}$. The matrix A is not known, but we do have a noisy measurement of it, $A^{\text{meas}} = A + E$. Here the matrix E is measurement error, which is assumed to be small. While A is not known, we do know real, independent eigenvectors $v_1, \ldots, v_n$ of A. (Its eigenvalues $\lambda_1, \ldots, \lambda_n$, however, are not known.) We will combine our measurement of A with our prior knowledge to find an estimate $\hat{A}$ of A. To do this, we choose $\hat{A}$ as the matrix that minimizes

$$J = \frac{1}{n^2} \sum_{i,j=1}^{n} (A_{ij}^{\text{meas}} - \hat{A}_{ij})^2$$

among all matrices which have eigenvectors $v_1, \ldots, v_n$. (Thus, $\hat{A}$ is the matrix closest to our measurement, in the mean-square sense, that is consistent with the known eigenvectors.)

(a) Explain how you would find $\hat{A}$. If your method is iterative, say whether you can guarantee convergence. Be sure to say whether your method finds the exact minimizer of J (except, of course, for numerical error due to roundoff), or an approximate solution. You can use any of the methods (least-squares, least-norm, Gauss-Newton, low rank approximation, *etc.*) or decompositions (QR, SVD, eigenvalue decomposition, *etc.*) from the course.

(b) Carry out your method with the data

$$A^{\text{meas}} = \begin{bmatrix} 2.0 & 1.2 & -1.0 \\ 0.4 & 2.0 & -0.5 \\ -0.5 & 0.9 & 1.0 \end{bmatrix}, \quad v_1 = \begin{bmatrix} 0.7 \\ 0 \\ 0.7 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 0.3 \\ 0.6 \\ 0.7 \end{bmatrix}, \quad v_3 = \begin{bmatrix} 0.6 \\ 0.6 \\ 0.3 \end{bmatrix}.$$

Be sure to check that $\hat{A}$ does indeed have v_1, v_2, v_3 as eigenvectors, by (numerically) finding its eigenvectors and eigenvalues. Also, give the value of J for $\hat{A}$. **Hint.** You might find the following useful (but then again, you might not.) In Matlab, if `A` is a matrix, then `A(:)` is a (column) vector consisting of all the entries of `A`, written out column by column. Therefore `norm(A(:))` gives the squareroot of the sum of the squares of entries of the matrix `A`, *i.e.*, its Frobenius norm. The inverse operation, *i.e.*, writing a vector out as a matrix with some given dimensions, is done using the function `reshape`. For example, if `a` is an mn vector, then `reshape(a,m,n)` is an $m \times n$ matrix, with elements taken from `a` (column by column).

11.13 *Real modal form.* Generate a matrix A in $\mathbf{R}^{10\times 10}$ using `A=randn(10)`. (The entries of A will be drawn from a unit normal distribution.) Find the eigenvalues of A. If by chance they are all real, please generate a new instance of A. Find the real modal form of A, *i.e.*, a matrix S such that $S^{-1}AS$ has the real modal form given in lecture 11. Your solution should include a clear explanation of how you will find S, the source code that you use to find S, and some code that checks the results (*i.e.*, computes $S^{-1}AS$ to verify it has the required form).

Lecture 12 – Jordan canonical form

12.1 *Some true/false questions.* Determine if the following statements are true or false. No justification or discussion is needed for your answers. What we mean by "true" is that the statement is true for all values of the matrices and vectors that appear in the statement. You can't assume anything about the dimensions of the matrices (unless it's explicitly stated), but you can assume that the dimensions are such that all expressions make sense. For example, the statement "$A+B=B+A$" is true, because no matter what the dimensions of A and B are (they must, however, be the same), and no matter what values A and B have, the statement holds. As another example, the statement $A^2=A$ is false, because there are (square) matrices for which this doesn't hold. (There are also matrices for which it does hold, *e.g.*, an identity matrix. But that doesn't make the statement true.) "False" means the statement isn't true, in other words, it can fail to hold for some values of the matrices and vectors that appear in it.

(a) If $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{n\times p}$ are both full rank, and $AB=0$, then $n \geq m+p$.

(b) If $A \in \mathbf{R}^{3\times 3}$ satisfies $A+A^T=0$, then A is singular.

(c) If $A^k=0$ for some integer $k \geq 1$, then $I-A$ is nonsingular.

(d) If $A, B \in \mathbf{R}^{n\times n}$ are both diagonalizable, then AB is diagonalizable.

(e) If $A, B \in \mathbf{R}^{n\times n}$, then every eigenvalue of AB is an eigenvalue of BA.

(f) If $A, B \in \mathbf{R}^{n\times n}$, then every eigenvector of AB is an eigenvector of BA.

(g) If A is nonsingular and A^2 is diagonalizable, then A is diagonalizable.

12.2 Consider the discrete-time system $x(t+1)=Ax(t)$, where $x(t) \in \mathbf{R}^n$.

(a) Find $x(t)$ in terms of $x(0)$.

(b) Suppose that $\det(zI-A)=z^n$. What are the eigenvalues of A? What (if anything) can you say about $x(k)$ for $k<n$ and $k \geq n$, without knowing $x(0)$?

12.3 *Asymptotically periodic trajectories.* We say that $x:\mathbf{R}_+ \to \mathbf{R}^n$ is *asymptotically T-periodic* if $\|x(t+T)-x(t)\|$ converges to 0 as $t \to \infty$. (We assume $T>0$ is fixed.) Now consider the (time-invariant) linear dynamical system $\dot{x}=Ax$, where $A \in \mathbf{R}^{n\times n}$. Describe the precise conditions on A under which *all* trajectories of $\dot{x}=Ax$ are asymptotically T-periodic. Give your answer in terms of the Jordan form of A. (The period T can appear in your answer.) Make sure your answer works for 'silly' cases like $A=0$ (for which all trajectories are constant, hence asymptotically T-periodic), or stable systems (for which all trajectories converge to 0, hence are asymptotically T-periodic). Mark your answer clearly, to isolate it from any (brief) discussion or explanation. You do not need to formally prove your answer; a brief explanation will suffice.

12.4 *Jordan form of a block matrix.* We consider the block 2×2 matrix

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

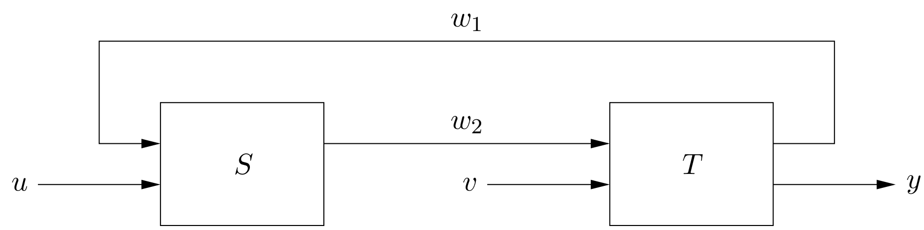
Here $A \in \mathbf{R}^{n\times n}$, and is diagonalizable, with real, distinct eigenvalues $\lambda_1,\dots,\lambda_n$. We'll let $v_1,\dots,v_n$ denote (independent) eigenvectors of A associated with $\lambda_1,\dots,\lambda_n$.

(a) Find the Jordan form J of C. Be sure to explicitly describe its block sizes.

(b) Find a matrix T such that $J = T^{-1}CT$.

Lecture 13 – Linear dynamical systems with inputs and source

13.1 *Interconnection of linear systems.* Often a linear system is described in terms of a block diagram showing the interconnections between components or subsystems, which are themselves linear systems. In this problem you consider the specific interconnection shown below:



Here, there are two subsystems S and T. Subsystem S is characterized by

$$\dot{x} = Ax + B_1 u + B_2 w_1, \qquad w_2 = Cx + D_1 u + D_2 w_1,$$

and subsystem T is characterized by

$$\dot{z} = Fz + G_1 v + G_2 w_2, \qquad w_1 = H_1 z, \qquad y = H_2 z + J w_2.$$

We don't specify the dimensions of the signals (which can be vectors) or matrices here. You can assume all the matrices are the correct (*i.e.*, compatible) dimensions. Note that the subscripts in the matrices above, as in B_1 and B_2, refer to different matrices. Now the problem. Express the overall system as a single linear dynamical system with input, state, and output given by

$$\begin{bmatrix} u \\ v \end{bmatrix}, \qquad \begin{bmatrix} x \\ z \end{bmatrix}, \qquad y,$$

respectively. Be sure to explicitly give the input, dynamics, output, and feedthrough matrices of the overall system. If you need to make any assumptions about the rank or invertibility of any matrix you encounter in your derivations, go ahead. But be sure to let us know what assumptions you are making.

13.2 *Minimum energy control.* Consider the discrete-time linear dynamical system

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

where $x(t) \in \mathbf{R}^n$, and the input $u(t)$ is a scalar (hence, $A \in \mathbf{R}^{n\times n}$ and $B \in \mathbf{R}^{n\times 1}$). The initial state is $x(0) = 0$.

(a) Find the matrix $\mathcal{C}_T$ such that

$$x(T) = \mathcal{C}_T \begin{bmatrix} u(T-1) \\ \vdots \\ u(1) \\ u(0) \end{bmatrix}.$$

(b) For the remainder of this problem, we consider a specific system with $n = 4$. The dynamics and input matrices are

$$A = \begin{bmatrix} 0.5 & 0.7 & -0.9 & -0.5 \\ 0.4 & -0.7 & 0.1 & 0.3 \\ 0.7 & 0.0 & -0.6 & 0.1 \\ 0.4 & -0.1 & 0.8 & -0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

Suppose we want the state to be x_{des} at time T. Consider the desired state

$$x_{\text{des}} = \begin{bmatrix} 0.8 \\ 2.3 \\ -0.7 \\ -0.3 \end{bmatrix}.$$

What is the smallest T for which we can find inputs $u(0), \ldots, u(T-1)$, such that $x(T) = x_{\text{des}}$? What are the corresponding inputs that achieve x_{des} at this minimum time? What is the smallest T for which we can find inputs $u(0), \ldots, u(T-1)$, such that $x(T) = x_{\text{des}}$ for any $x_{\text{des}} \in \mathbf{R}^4$? We'll denote this T by T_{min}.

(c) Suppose the energy expended in applying inputs $u(0), \ldots, u(T-1)$ is

$$E(T) = \sum_{t=0}^{T-1} (u(t))^2.$$

For a given T (greater than T_{min}) and x_{des}, how can you compute the inputs which achieve $x(T) = x_{\text{des}}$ with the minimum expense of energy? Consider now the desired state

$$x_{\text{des}} = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

For each T ranging from T_{min} to 30, find the minimum energy inputs that achieve $x(T) = x_{des}$. For each T, evaluate the corresponding input energy, which we denote by $E_{\text{min}}(T)$. Plot $E_{min}(T)$ as a function of T. (You should include in your solution a description of how you computed the minimum-energy inputs, and the plot of the minimum energy as a function of T. But you don't need to list the actual inputs you computed!)

(d) You should observe that $E_{min}(T)$ is non-increasing in T. Show that this is the case in general (*i.e.*, for any A, B, and x_{des}).

Note: There is a direct way of computing the assymptotic limit of the minimum energy as $T \to \infty$. We'll cover these ideas in more detail in *ee363*.

13.3 *Output feedback for maximum damping.* Consider the discrete-time linear dynamical system

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned}$$

with $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^{n\times m}$, $C \in \mathbf{R}^{p\times n}$. In *output feedback control* we use an input which is a linear function of the output, that is,

$$u(t) = Ky(t),$$

where $K \in \mathbf{R}^{m\times p}$ is the *feedback gain matrix.* The resulting state trajectory is identical to that of an autonomous system,

$$x(t+1) = \bar{A}x(t).$$

(a) Write $\bar{A}$ explicitly in terms of A, B, C, and K.

(b) Consider the single-input, single-output system with

$$A = \begin{bmatrix} 0.5 & 1.0 & 0.1 \\ -0.1 & 0.5 & -0.1 \\ 0.2 & 0.0 & 0.9 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}.$$

In this case, the feedback gain matrix K is a scalar (which we call simply the *feedback gain.*) The question is: find the feedback gain K_{opt} such that the feedback system is maximally damped. By maximally damped, we mean that the state goes to zero with the fastest asymptotic decay rate (measured for an initial state $x(0)$ with non-zero coefficient in the slowest mode.) *Hint:* You are only required to give your answer K_{opt} up to a precision of ± 0.01, and you can assume that $K_{\mathrm{opt}} \in [-2, 2]$.

13.4 *Affine dynamical systems.* A function $f : \mathbf{R}^n \to \mathbf{R}^m$ is called *affine* if it is a linear function plus a constant, *i.e.*, of the form $f(x) = Ax + b$. Affine functions are more general than linear functions, which result when $b = 0$. We can generalize linear dynamical systems to *affine dynamical systems*, which have the form

$$\dot{x} = Ax + Bu + f, \quad y = Cx + Du + g.$$

Fortunately we don't need a whole new theory for (or course on) affine systems; a simple shift of coordinates converts it to a linear dynamical system. Assuming A is invertible, define $\tilde{x} = x + A^{-1}f$ and $\tilde{y} = y - g + CA^{-1}f$. Show that $\tilde{x}$, u, and $\tilde{y}$ are the state, input, and output of a linear dynamical system.

13.5 Two separate experiments are performed for $t \geq 0$ on the single-input single-output (SISO) linear system

$$\dot{x} = Ax + Bu, \quad y = Cx + Du, \quad x(0) = [1\ 2\ -1\ ]^T$$

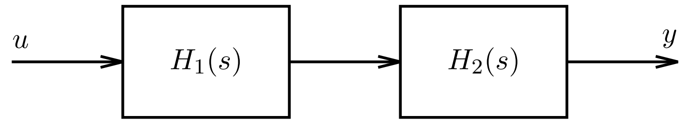
(the initial condition is the same in each experiment). In the first experiment, $u(t) = e^{-t}$ and the resulting output is $y(t) = e^{-3t} + e^{-2t}$. In the second, $u(t) = e^{-3t}$ and the resulting output is $y(t) = 3e^{-3t} - e^{-2t}$.

(a) Can you determine the transfer function $C(sI - A)^{-1}B + D$ from this information? If it is possible, do so. If not, find two linear systems consistent with all the data given which have different transfer functions.

(b) Can you determine A, B, C, or D?

13.6 *Cascade connection of systems.*

(a) Two linear systems (A_1, B_1, C_1, D_1) and (A_2, B_2, C_2, D_2) with states x_1 and x_2 (these are two *column vectors*, not two scalar components of one vector), have transfer functions $H_1(s)$ and $H_2(s)$, respectively. Find state equations for the cascade system:



Use the state $x = [x_1^T \; x_2^T]^T$.

(b) Use the state equations above to verify that the cascade system has transfer function $H_2(s)H_1(s)$. (To simplify, you can assume $D_1 = 0$, $D_2 = 0$.)

(c) Find the dual of the LDS found in (a). Draw a block diagram of the dual system as a cascade connection of two systems. (To simplify, you can assume $D_1 = 0$, $D_2 = 0$.) *Remark:* quite generally, the block diagram corresponding to the dual system is the original block diagram, "turned around," with all arrows reversed.

13.7 *Inverse of a linear system.* Suppose $H(s) = C(sI - A)^{-1}B + D$, where D is square and invertible. You will find a linear system with transfer function $H(s)^{-1}$.

(a) Start with $\dot{x} = Ax + Bu$, $y = Cx + Du$, and solve for $\dot{x}$ and u in terms of x and y. Your answer will have the form: $\dot{x} = Ex + Fy$, $u = Gx + Hy$. Interpret the result as a linear system with state x, input y, and output u.

(b) Verify that

$$(G(sI - E)^{-1}F + H)(C(sI - A)^{-1}B + D) = I.$$

Hint: use the following "resolvent identity:"

$$(sI - X)^{-1} - (sI - Y)^{-1} = (sI - X)^{-1}(X - Y)(sI - Y)^{-1}$$

which can be verified by multiplying by $sI - X$ on the left and $sI - Y$ on the right.

13.8 *Offset or skewed discretization.* In the lecture notes we considered sampling a continuous-time system in which the input update and output sampling occur at the same time, *i.e.*, are synchronized. In this problem we consider what happens when there is a constant time offset or skew between them (which often happens in practice). Consider the continuous-time LDS $\dot{x} = Ax + Bu$, $y = Cx + Du$. We define the sequences x_d and y_d as

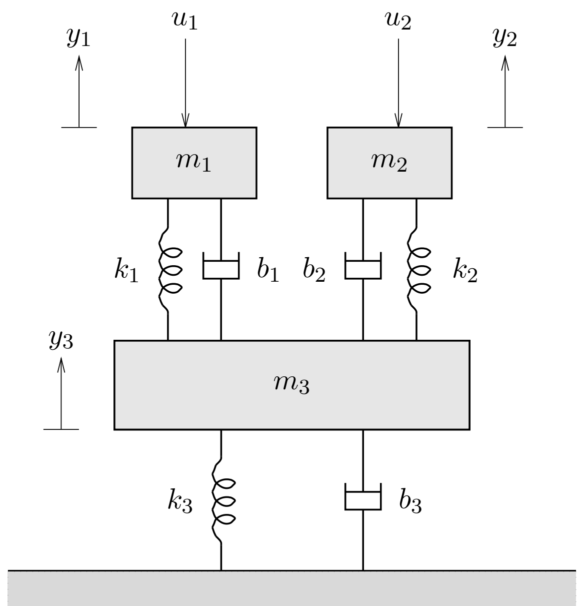
$$x_d(k) = x(kh), \quad y_d(k) = y(kh), \quad k = 0, 1, \ldots$$

where $h > 0$ (*i.e.*, the state and output are sampled every h seconds). The input u is given by

$$u(t) = u_d(k) \quad \text{for} \quad kh + \delta \le t < (k+1)h + \delta, \quad k = 0, 1, \ldots$$

where δ is a delay or offset in the input update, with $0 \le \delta < h$. Find a discrete-time LDS with u_d as input and y_d as output. Give the matrices that describe this LDS.

13.9 *Static decoupling.* Consider the mechanical system shown below.



Two masses with values $m_1 = 1$ and $m_2 = 2$ are attached via spring/damper suspensions with stiffnesses $k_1 = 1$, $k_2 = 2$ and damping $b_1 = 1$, $b_2 = 2$ to a platform, which is another mass of value $m_3 = 3$. The platform is attached to the ground by a spring/damper suspension with stiffness $k_3 = 3$ and damping $b_3 = 3$. The displacements of the masses (with respect to ground) are denoted y_1, y_2, and y_3. Forces u_1 and u_2 are applied to the first two masses.

(a) Find matrices $A \in \mathbf{R}^{6\times 6}$ and $B \in \mathbf{R}^{6\times 2}$ such that the dynamics of the mechanical system is given by $\dot{x} = Ax + Bu$ where

$$x = [y_1\ y_2\ y_3\ \dot{y}_1\ \dot{y}_2\ \dot{y}_3]^T, \quad u = [u_1\ u_2]^T.$$

Ignore the effect of gravity (or you can assume the effect of gravity has already been taken into account in the definition of y_1, y_2 and y_3).

(b) Plot the step responses matrix, *i.e.*, the step responses from inputs u_1 and u_2 to outputs y_1, y_2 and y_3. Briefly interpret and explain your plots.

(c) Find the DC gain matrix $H(0)$ from inputs u_1 and u_2 to outputs y_1 and y_2.

(d) *Design of an asymptotic decoupler.* In order to make the steady-state deflections of masses 1 and 2 independent of each other, we let $u = H(0)^{-1}y_{\rm cmd}$, where $y_{\rm cmd} : \mathbf{R}_+ \to \mathbf{R}^2$. Plot the step responses from $y_{\rm cmd}$ to y_1 and y_2, and compare with the original ones found in part b.

13.10 *A method for rapidly driving the state to zero.* We consider the discrete-time linear dynamical system

$$x(t+1) = Ax(t) + Bu(t),$$

where $A \in \mathbf{R}^{n\times n}$ and $B \in \mathbf{R}^{n\times k}$, $k < n$, is full rank. The goal is to choose an input u that causes $x(t)$ to converge to zero as $t \to \infty$. An engineer proposes the following simple method: at time t, choose $u(t)$ that minimizes $\|x(t+1)\|$. The engineer argues that this scheme will work well, since the norm of the state is made as small as possible at every step. In this problem you will analyze this scheme.

(a) Find an explicit expression for the proposed input $u(t)$ in terms of $x(t)$, A, and B.

(b) Now consider the linear dynamical system $x(t+1) = Ax(t) + Bu(t)$ with $u(t)$ given by the proposed scheme (*i.e.*, as found in (10a)). Show that x satisfies an autonomous linear dynamical system equation $x(t+1) = Fx(t)$. Express the matrix F explicitly in terms of A and B.

(c) Now consider a specific case:

$$A = \begin{bmatrix} 0 & 3 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Compare the behavior of $x(t+1) = Ax(t)$ (*i.e.*, the orginal system with $u(t) = 0$) and $x(t+1) = Fx(t)$ (*i.e.*, the original system with $u(t)$ chosen by the scheme described above) for a few initial conditions. Determine whether each of these systems is stable.

13.11 *Analysis of investment allocation strategies.* Each year or period (denoted $t = 0, 1, \ldots$) an investor buys certain amounts of one-, two-, and three-year certificates of deposit (CDs) with interest rates 5%, 6%, and 7%, respectively. (We ignore minimum purchase requirements, and assume they can be bought in any amount.)

- $B_1(t)$ denotes the amount of one-year CDs bought at period t.
- $B_2(t)$ denotes the amount of two-year CDs bought at period t.
- $B_3(t)$ denotes the amount of three-year CDs bought at period t.

We assume that $B_1(0) + B_2(0) + B_3(0) = 1$, *i.e.*, a total of 1 is to be invested at $t = 0$. (You can take $B_j(t)$ to be zero for $t < 0$.) The total payout to the investor, $p(t)$, at period t is a sum of six terms:

- $1.05B_1(t-1)$, *i.e.*, principle plus 5% interest on the amount of one-year CDs bought one year ago.
- $1.06B_2(t-2)$, *i.e.*, principle plus 6% interest on the amount of two-year CDs bought two years ago.
- $1.07B_3(t-3)$, *i.e.*, principle plus 7% interest on the amount of three-year CDs bought three years ago.
- $0.06B_2(t-1)$, *i.e.*, 6% interest on the amount of two-year CDs bought one year ago.
- $0.07B_3(t-1)$, *i.e.*, 7% interest on the amount of three-year CDs bought one year ago.
- $0.07B_3(t-2)$, *i.e.*, 7% interest on the amount of three-year CDs bought two years ago.

The total wealth held by the investor at period t is given by

$$w(t) = B_1(t) + B_2(t) + B_2(t-1) + B_3(t) + B_3(t-1) + B_3(t-2).$$

Two re-investment allocation strategies are suggested.

- *The 35-35-30 strategy.* The total payout is re-invested 35% in one-year CDs, 35% in two-year CDs, and 30% in three-year CDs. The initial investment allocation is the same: $B_1(0) = 0.35$, $B_2(0) = 0.35$, and $B_3(0) = 0.30$.

- *The 60-20-20 strategy.* The total payout is re-invested 60% in one-year CDs, 20% in two-year CDs, and 20% in three-year CDs. The initial investment allocation is $B_1(0) = 0.60$, $B_2(0) = 0.20$, and $B_3(0) = 0.20$.

(a) Describe the investments over time as a linear dynamical system $x(t+1) = Ax(t)$, $y(t) = Cx(t)$ with $y(t)$ equal to the total wealth at time t. Be *very clear* about what the state $x(t)$ is, and what the matrices A and C are. You will have two such linear systems: one for the 35-35-30 strategy and one for the 60-20-20 strategy.

(b) *Asymptotic wealth growth rate.* For each of the two strategies described above, determine the asymptotic growth rate, defined as $\lim_{t\to\infty} w(t+1)/w(t)$. (If this limit doesn't exist, say so.) *Note:* simple numerical simulation of the strategies (*e.g.*, plotting $w(t+1)/w(t)$ versus t to guess its limit) is *not* acceptable. (You can, of course, simulate the strategies to *check* your answer.)

(c) *Asymptotic liquidity.* The total wealth at time t can be divided into three parts:
   - $B_1(t)+B_2(t-1)+B_3(t-2)$ is the amount that matures in one year (*i.e.*, the amount of principle we could get back next year)
   - $B_2(t) + B_3(t-1)$ is the amount that matures in two years
   - $B_3(t)$ is the amount that matures in three years (*i.e.*, is least liquid)

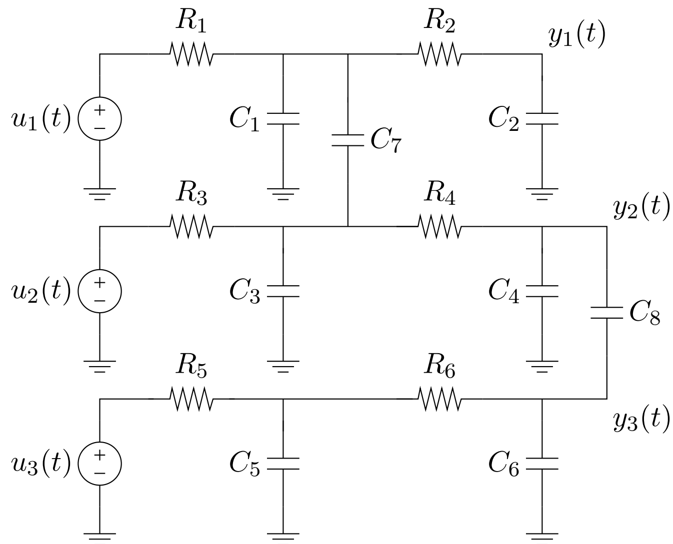
   We define liquidity ratios as the ratio of these amounts to the total wealth:

$$\begin{aligned} L_1(t) &= (B_1(t) + B_2(t-1) + B_3(t-2))/w(t), \\ L_2(t) &= (B_2(t) + B_3(t-1))/w(t), \\ L_3(t) &= B_3(t)/w(t). \end{aligned}$$

   For the two strategies above, do the liquidity ratios converge as $t \to \infty$? If so, to what values? *Note:* as above, simple numerical simulation alone is *not* acceptable.

(d) Suppose you could change the *initial* investment allocation for the 35-35-30 strategy, *i.e.*, choose some other nonnegative values for $B_1(0)$, $B_2(0)$, and $B_3(0)$ that satisfy $B_1(0)+B_2(0)+B_3(0) = 1$. What allocation would you pick, and how would it be better than the $(0.35,\ 0.35,\ 0.30)$ initial allocation? (For example, would the asymptotic growth rate be larger?) How much better is your choice of initial investment allocations? *Hint for part d:* think *very carefully* about this one. *Hint for whole problem:* watch out for nondiagonalizable, or nearly nondiagonalizable, matrices. Don't just blindly type in Matlab commands; check to make sure you're computing what you think you're computing.

13.12 *Analysis of cross-coupling in interconnect wiring.* In integrated circuits, wires which connect the output of one gate to the inputs of one (or more) other gates are called *nets*. As feature sizes shrink to well below a micron (*i.e.*, 'deep submicron') the capacitance of a wire to the substrate (which in a simple analysis can be approximated as ground), as well as to neighboring wires, must be taken into account. A simple lumped model of three nets is shown below. The inputs are the voltage sources u_1, u_2, u_3, and the outputs are the three voltages labeled y_1, y_2, y_3. The resistances $R_1, \ldots, R_6$ represent the resistance of the wire segments. The capacitances $C_1, \ldots, C_6$ are capacitances from the interconnect wires to the substrate; the capacitances C_7 and C_8 are capacitances between wires 1 and 2, and wires 2 and 3, respectively. (The different locations of the these cross-coupling capacitances models the wire 1 crossing over wire 2 near the driving gate, and wire 2 crossing over wire 3 near the end of the wire, but you don't need to know this to do the problem ...) In static conditions, the circuit reduces to three wires (with resistance R_1+R_2, R_3+R_4, and R_5+R_6, respectively) connecting the inputs to the outputs.



To simplify the problem we'll assume that all resistors have value 1 and all capacitors have value 1. We recognize that some of you don't know how to write the equations that govern this circuit, so we've done it for you. (If you're an EE student in this category, then shame on you.) The equations are

$$C\dot{v} + Gv = Fu, \quad y = Kv,$$

where

$$C = \begin{bmatrix} 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 2 \end{bmatrix}, \quad G = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix},$$

$$F = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad K = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and $v \in \mathbf{R}^6$ is the vector of voltages at capacitors $C_1, \ldots, C_6$, respectively. To save you the trouble of typing these in, we've put an mfile `interconn.m` on the course web page, which defines these matrices. The inputs (which represent the gates that drive the three nets) are

Boolean valued, *i.e.*, $u_i(t) \in \{0,1\}$ for all t. In this problem we will only consider inputs that switch (change value from 0 to 1 or 1 to 0) at most once.

(a) 50%-*threshold delay.* For $t < 0$, the system is in static condition, and the inputs have values $u(t) = f$ for $t < 0$, where $f_i \in \{0,1\}$. At $t = 0$, the input switches to the Boolean vector g, *i.e.*, for $t \geq 0$, $u(t) = g$, where $g_i \in \{0,1\}$. Since the DC gain matrix of this system is I, and the system is stable, the output converges to the input value: $y(t) \to g$ as $t \to \infty$. We define the 50%-threshold delay of the transition as smallest T such that $|y_i(t) - g_i| \leq 0.5$ for $t \geq T$, and for $i = 1,2,3$. (If the following gate thresholds were set at 0.5, then this would be first time after which the outputs would be guaranteed correct.) Among the 64 possible transitions, find the largest (*i.e.*, worst) 50%-threshold delay. Give the largest delay, and also describe which transition gives the largest delay (*e.g.*, the transition with $f = (0,0,1)$ to $g = (1,0,0)$).

(b) *Maximum bounce due to cross-coupling.* Now suppose that input 2 remains zero, but inputs 1 and 3 undergo transitions at times $t = T_1$ and $t = T_3$, respectively. (In part 1, in contrast, all transitions occured at $t = 0$.) To be more precise (and also so nobody can say we weren't clear),

$$u_1(t) = \begin{cases} f_1 & \text{for } t < T_1 \\ g_1 & \text{for } t \geq T_1 \end{cases}, \quad u_3(t) = \begin{cases} f_3 & \text{for } t < T_3 \\ g_3 & \text{for } t \geq T_3 \end{cases}, \quad u_2(t) = 0 \text{ for all } t,$$

where $f_1, f_3, g_1, g_3 \in \{0,1\}$. The transitions in inputs 1 and 3 induce a nonzero response in output 2. (But y_2 does converge back to zero, since $u_2 = 0$.) This phenomenon of y_2 deviating from zero (which is what it would be if there were no cross-coupling capacitance) is called *bounce* (induced by the cross-coupling between the nets). If for any t, $y_2(t)$ is large enough to trigger the following gate, things can get very, very ugly. What is the maximum possible bounce? In other words, what is the maximum possible value of $y_2(t)$, over all possible t, T_1, T_3, f_1, f_3, g_1, g_3? Be sure to give not only the maximum value, but also the times t, T_1, and T_3, and the transitions f_1, f_3, g_1, g_3, which maximize $y(t)$.

Note: in this problem we don't consider multiple transitions, but it's not hard to do so.

13.13 *Periodic solution with intermittent input.* We consider the *stable* linear dynamical system $\dot{x} = Ax + Bu$, where $x(t) \in \mathbf{R}^n$, and $u(t) \in \mathbf{R}$. The input has the specific form

$$u(t) = \begin{cases} 1 & kT \leq t < (k+\theta)T, \quad k = 0,1,2,\ldots \\ 0 & (k+\theta)T \leq t < (k+1)T, \quad k = 0,1,2,\ldots \end{cases}$$

Here $T > 0$ is the *period*, and $\theta \in [0,1]$ is called the *duty cycle* of the input. You can think of u as a constant input value one, that is applied over a fraction θ of each cycle, which lasts T seconds. Note that when $\theta = 0$, the input is $u(t) = 0$ for all t, and when $\theta = 1$, the input is $u(t) = 1$ for all t.

(a) Explain how to find an initial state $x(0)$ for which the resulting state trajectory is T-periodic, *i.e.*, $x(t+T) = x(t)$ for all $t \geq 0$. Give a formula for $x(0)$ in terms of the problem data, *i.e.*, A, B, T, and θ. Try to give the simplest possible formula.

(b) Explain why there is always *exactly one* value of $x(0)$ that results in $x(t)$ being T-periodic. In addition, explain why the formula you found in part (a) always makes sense and is valid. (For example, if your formula involves a matrix inverse, explain why the matrix to be inverted is nonsingular.)

(c) We now consider the specific system with

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -2 & -1 \end{bmatrix}, \qquad B = \begin{bmatrix} 8 \\ 2 \\ -14 \end{bmatrix}, \qquad T = 5.$$

Plot J, the mean-square norm of the state,

$$J = \frac{1}{T} \int_0^T \|x(t)\|^2 \, dt,$$

versus θ, for $0 \leq \theta \leq 1$, where $x(0)$ is the periodic initial condition that you found in part (a). You may approximate J as

$$J \approx \frac{1}{N} \sum_{i=0}^{N-1} \|x(iT/N)\|^2,$$

for N large enough (say 1000). Estimate the value of θ that maximizes J.

13.14 *System identification of a linear dynamical system.* In *system identification*, we are given some time series values for a discrete-time input vector signal,

$$u(1),\ u(2), \ldots, u(N) \in \mathbf{R}^m,$$

and also a discrete-time state vector signal,

$$x(1),\ x(2), \ldots, x(N) \in \mathbf{R}^n,$$

and we are asked to find matrices $A \in \mathbf{R}^{n\times n}$ and $B \in \mathbf{R}^{n\times m}$ such that we have

$$x(t+1) \approx Ax(t) + Bu(t), \quad t = 1, \ldots, N-1. \tag{2}$$

We use the symbol $\approx$ since there may be small measurement errors in the given signal data, so we don't expect to find matrices A and B for which the linear dynamical system equations hold exactly. Let's give a quantitative measure of how well the linear dynamical system model (2) holds, for a particular choice of matrices A and B. We define the RMS (root-mean-square) value of the residuals associated with our signal data and a candidate pair of matrices A, B as

$$R = \left(\frac{1}{N-1} \sum_{t=1}^{N-1} \|x(t+1) - Ax(t) - Bu(t)\|^2 \right)^{1/2}.$$

We define the RMS value of x, over the same period, as

$$S = \left(\frac{1}{N-1} \sum_{t=1}^{N-1} \|x(t+1)\|^2 \right)^{1/2}.$$

We define the *normalized residual*, denoted ρ, as $\rho = R/S$. If we have $\rho = 0.05$, for example, it means that the state equation (2) holds, roughly speaking, to within 5%. Given the signal data, we will choose the matrices A and B to minimize the RMS residual R (or, equivalently, the normalized residual ρ).

(a) Explain how to do this. Does the method always work? If some conditions have to hold, specify them.

(b) Carry out this procedure on the data in `lds_sysid.m` on the course web site. Give the matrices A and B, and give the associated value of the normalized residual. Of course you must show your Matlab source code and the output it produces.

13.15 *System identification with selection of inputs & states.* This problem continues, or rather extends, the previous one on system identification, problem 14. Here too we need to fit a linear dynamical system model to some given signal data. To complicate things, though, we are not told which of the scalar signals are input components and which are state components. That's part of what we have to decide. We are given the time series data, *i.e.*, a vector signal,

$$z(1),\ z(2), \ldots, z(N) \in \mathbf{R}^p.$$

We will assign each component of z as either an input component, or a state component. For example, if z has four components we might assign its first and third to be the state, and its second and fourth to be the input, *i.e.*,

$$x(t) = \begin{bmatrix} z_1(t) \\ z_3(t) \end{bmatrix}, \qquad u(t) = \begin{bmatrix} z_2(t) \\ z_4(t) \end{bmatrix}.$$

You can assume that we always assign at least one component to the state, so the dimension of the state is always at least one. Once we assign components of z to either x or u, we then proceed as in problem (14): we find matrices A and B that minimize the RMS residuals as defined in problem (14). One measure of the complexity of the model is the number of components assigned to the input u; the larger the dimension of u, the more complex the model. If the dimension of u is small, then we have a compact model, in the sense that the data are explained by a linear dynamical system driven by only a few inputs. As an extreme case, if all components of z are assigned to x, then we have an autonomous linear dynamical system model for the data, *i.e.*, one with no inputs at all. Finally, here is the problem. Get the data given in `lds_sysid2.m` on the class web server, which contains a vector $z(t) \in \mathbf{R}^8$ for $t = 1, \ldots, 100$. Assign the components of z to either state or input, and develop a linear dynamical system model (*i.e.*, find matrices A and B) for your choice of x and u. We seek the simplest model, *i.e.*, the one with the smallest dimension of u, for which the normalized RMS residuals is smaller than around 5%. Your solution should consist of the following:

- *Your approach.* Explain how you solved the problem.
- *Your assignments to state and input.* Give a clear description of what x and u are. Please order the components in x and u in the same order as in z.
- Your matrices A and B.
- The relative RMS residuals obtained by your matrices.

- The Matlab code used to solve the problem, and its output.

13.16 *A greedy control scheme.* Our goal is to choose an input $u : \mathbf{R}_+ \to \mathbf{R}^m$, that is not too big, and drives the state $x : \mathbf{R}_+ \to \mathbf{R}^n$ of the system $\dot{x} = Ax + Bu$ to zero quickly. To do this, we will choose $u(t)$, for each t, to minimize the quantity

$$\frac{d}{dt}\|x(t)\|^2 + \rho\|u(t)\|^2,$$

where $\rho > 0$ is a given parameter. The first term gives the rate of decrease (if it is negative) of the norm-squared of the state vector; the second term is a penalty for using a large input.

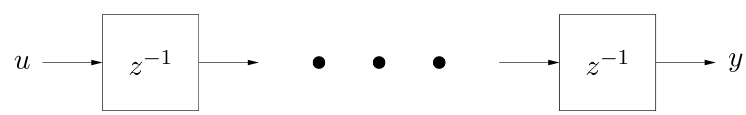
This scheme is greedy because at each instant t, $u(t)$ is chosen to minimize the compositive objective above, without regard for the effects such an input might have in the future.

(a) Show that $u(t)$ can be expressed as $u(t) = Kx(t)$, where $K \in \mathbf{R}^{m \times n}$. Give an explicit formula for K. (In other words, the control scheme has the form of a constant linear state feedback.)

(b) What are the conditions on A, B, and ρ under which we have $(d/dt)\|x(t)\|^2 < 0$ whenever $x(t) \neq 0$, using the scheme described above? (In other words, when does this control scheme result in the norm squared of the state always decreasing?)

(c) Find an example of a system (*i.e.*, A and B), for which the open-loop system $\dot{x} = Ax$ is stable, but the closed-loop system $\dot{x} = Ax + Bu$ (with u as above) is unstable, when $\rho = 1$. Try to find the simplest example you can, and be sure to show us verification that the open-loop system is stable and that the closed-loop system is not. (We will *not* check this for you. You must explain how to check this, and attach code and associated output.)

13.17 *FIR filter with small feedback.* Consider a cascade of 100 one-sample delays:

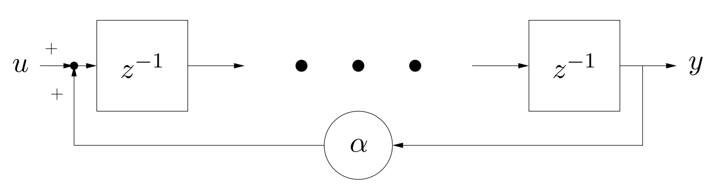


(a) Express this as a linear dynamical system

$$x(t+1) = Ax(t) + Bu(t), \qquad y(t) = Cx(t) + Du(t)$$

(b) What are the eigenvalues of A?

(c) Now we add simple feedback, with gain $\alpha = 10^{-5}$, to the system:



Express this as a linear dynamical system

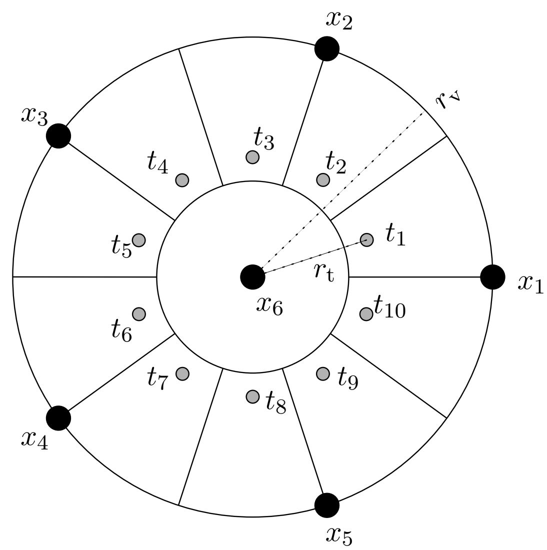
$$x(t+1) = A_f x(t) + B_f u(t), \qquad y(t) = C_f x(t) + D_f u(t)$$

(d) What are the eigenvalues of A_f?

(e) How different is the impulse response of the system with feedback ($\alpha = 10^{-5}$) and without feedback ($\alpha = 0$)?

Lecture 14 – Symmetric matrices, quadratic forms, matrix norm, and SVD

14.1 *Simplified temperature control.* A circular room consists of 10 identical cubicles around a circular shaft. There are 6 temperature-control systems in the room. Of those, 5 have vents evenly distributed around the perimeter of the room, and one is located in the centre. Each vent j blows a stream of air at temperature x_j, measured relative to the surrounding air (ambient air temperature.) The temperatures may be hotter ($x_j > 0$) or colder ($x_j < 0$) than the ambient air temperature. The temperature in each cubicle (measured at its center



as shown in the figure) is t_i, and the effect of vent j on temperature t_i is given by

$$A_{ij} = \frac{1}{r_{ij}^2}$$

where r_{ij} is the distance between vent j and measuring point i. So the system can be described by $t = Ax$ (where A is tall.) The temperature preferences differ among the inhabitants of the 10 cubicles. More precisely, the inhabitant of cubicle i wants the temperature to be y_i hotter than the surrounding air (which is colder if $y_i < 0$!) The objective is then to choose the x_j to best match these preferences (*i.e.*, obtain *exactly* the least possible sum of squares error in temperature), with minimal cost. Here, "cost" means the total power spent on the temperature-control systems, which is the sum of the power consumed by each heater/cooler, which in turn is proportional to x_j^2.

(a) How would you choose the x_j to best match the given preferences y_i, with minimal power consumption?

(b) Temperature measurement points are at distance r_t from the center, and vents are at distance r_v. Vent 1 lies exactly on the horizontal. The file `temp_control.m` on the course webpage defines `r_t`, `r_v`, and a preferences vector `y`. It also provides code for computing the distances from each vent to each desired temperature location. Using these data, find the optimal vent temperatures x, and the corresponding RMS error in temperature, as well as the power usage.

Comment: In this problem we ignore the fact that, in general, cooling requires more power (per unit of temperature difference) than heating ... But this was not meant to be an entirely realistic problem to start with!

14.2 *Norm expressions for quadratic forms.* Let $f(x) = x^T Ax$ (with $A = A^T \in \mathbf{R}^{n\times n}$) be a quadratic form.

(a) Show that f is positive semidefinite (*i.e.*, $A \geq 0$) if and only if it can be expressed as $f(x) = \|Fx\|^2$ for some matrix $F \in \mathbf{R}^{k\times n}$. Explain how to find such an F (when $A \geq 0$). What is the size of the smallest such F (*i.e.*, how small can k be)?

(b) Show that f can be expressed as a difference of squared norms, in the form $f(x) = \|Fx\|^2 - \|Gx\|^2$, for some appropriate matrices F and G. How small can the sizes of F and G be?

14.3 *Congruences and quadratic forms.* Suppose $A = A^T \in \mathbf{R}^{n\times n}$.

(a) Let $Z \in \mathbf{R}^{n\times p}$ be any matrix. Show that $Z^T AZ \geq 0$ if $A \geq 0$.

(b) Suppose that $T \in \mathbf{R}^{n\times n}$ is invertible. Show that $T^T AT \geq 0$ *if and only if* $A \geq 0$. When T is invertible, TAT^T is called a *congruence* of A and TAT^T and A are said to be *congruent*. This problem shows that congruences preserve positive semidefiniteness.

14.4 *Positive semidefinite (PSD) matrices.*

(a) Show that if A and B are PSD and $\alpha \in \mathbf{R}$, $\alpha \geq 0$, then so are αA and $A + B$.

(b) Show that any (symmetric) submatrix of a PSD matrix is PSD. (To form a symmetric submatrix, choose any subset of $\{1, \dots, n\}$ and then throw away all other columns and rows.)

(c) Show that if $A \geq 0$, $A_{ii} \geq 0$.

(d) Show that if $A \geq 0$, $|A_{ij}| \leq \sqrt{A_{ii}A_{jj}}$. In particular, if $A_{ii} = 0$, then the entire ith row and column of A are zero.

14.5 *A Pythagorean inequality for the matrix norm.* Suppose that $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{p\times n}$. Show that

$$\left\| \begin{bmatrix} A \\ B \end{bmatrix} \right\| \leq \sqrt{\|A\|^2 + \|B\|^2}.$$

Under what conditions do we have equality?

14.6 *Gram matrices.* Given functions $f_i : [a, b] \to \mathbf{R}$, $i = 1, \dots, n$, the *Gram matrix* $G \in \mathbf{R}^{n\times n}$ associated with them is defined by

$$G_{ij} = \int_a^b f_i(t)f_j(t)\, dt.$$

(a) Show that $G = G^T \geq 0$.

(b) Show that G is singular if and only if the functions $f_1, \dots, f_n$ are linearly dependent.

14.7 *Properties of symmetric matrices.* In this problem P and Q are symmetric matrices. For each statement below, either give a proof or a specific counterexample.

(a) If $P \geq 0$ then $P + Q \geq Q$.

(b) If $P \geq Q$ then $-P \leq -Q$.

(c) If $P > 0$ then $P^{-1} > 0$.

(d) If $P \geq Q > 0$ then $P^{-1} \leq Q^{-1}$.

(e) If $P \geq Q$ then $P^2 \geq Q^2$.

Hint: you might find it useful for part (d) to prove $Z \geq I$ implies $Z^{-1} \leq I$.

14.8 Express $\sum_{i=1}^{n-1}(x_{i+1} - x_i)^2$ in the form $x^T P x$ with $P = P^T$. Is $P \geq 0$? $P > 0$?

14.9 Suppose A and B are symmetric matrices that yield the same quadratic form, *i.e.*, $x^T A x = x^T B x$ for all x. Show that $A = B$. *Hint:* first try $x = e_i$ (the ith unit vector) to conclude that the entries of A and B on the diagonal are the same; then try $x = e_i + e_j$.

14.10 *A power method for computing* $\|A\|$*.* The following method can be used to compute the largest singular value (σ_1), and also the corresponding left and right singular vectors (u_1 and v_1) of $A \in \mathbf{R}^{m \times n}$. You can assume (to simplify) that the largest singular value of A is isolated, *i.e.*, $\sigma_1 > \sigma_2$. Let $z(0) = a \in \mathbf{R}^n$ be nonzero, and then repeat the iteration

$$w(t) = Az(t); \quad z(t+1) = A^T w(t);$$

for $t = 1, 2, \ldots$. For large t, $w(t)/\|w(t)\| \approx u_1$ and $z(t)/\|z(t)\| \approx v_1$. Analyze this algorithm. Show that it 'usually' works. Be very explicit about when it fails. (In practice it always works.)

14.11 Suppose that $A \in \mathbf{R}^{n \times n}$. Show that $\|A\| < 1$ implies $I + A$ is invertible.

14.12 *Some problems involving matrix inequalities.* In the following problems you can assume that $A = A^T \in \mathbf{R}^{n \times n}$ and $B = B^T \in \mathbf{R}^{n \times n}$. We *do not,* however, assume that A or B is positive semidefinite. For $X = X^T \in \mathbf{R}^{n \times n}$, $\lambda_i(X)$ will denote its ith eigenvalue, sorted so $\lambda_1(X) \geq \lambda_2(X) \geq \cdots \geq \lambda_n(X)$. As usual, the symbol $\leq$ between symmetric matrices denotes matrix inequality (*e.g.*, $A \leq B$ means $B - A$ is positive semidefinite). Decide whether each of the following statements is true or false. ('True' means the statement holds for all A and B; 'false' means there is at least one pair A, B for which the statement does not hold.)

(a) $A \geq B$ if $\lambda_i(A) \geq \lambda_i(B)$ for $i = 1, \ldots, n$.

(b) If $\{x | x^T A x \leq 1\} \subseteq \{x | x^T B x \leq 1\}$, then $A \geq B$.

(c) If $A \leq B$, then $\{x | x^T A x \leq 1\} \subseteq \{x | x^T B x \leq 1\}$.

(d) If the eigenvalues of A and B are the same, *i.e.*, $\lambda_i(A) = \lambda_i(B)$ for $i = 1, \ldots, n$, then there is an orthogonal matrix Q such that $A = Q^T B Q$.

(e) If there is an orthogonal matrix Q such that $A = Q^T B Q$, then the eigenvalues of A and B are the same, *i.e.*, $\lambda_i(A) = \lambda_i(B)$ for $i = 1, \ldots, n$.

(f) If $A \geq B$ then for all $t \geq 0$, $e^{At} \geq e^{Bt}$.

(g) If $A \geq B$ then $A_{ij} \geq B_{ij}$ for $i, j = 1, \ldots, n$.

(h) If $A_{ij} \geq B_{ij}$ for $i, j = 1, \ldots, n$, then $A \geq B$.

14.13 *Eigenvalues and singular values of a symmetric matrix.* Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues, and let $\sigma_1, \ldots, \sigma_n$ be the singular values of a matrix $A \in \mathbf{R}^{n\times n}$, which satisfies $A = A^T$. (The singular values are based on the full SVD: If $\mathbf{Rank}(A) < n$, then some of the singular values are zero.) You can assume the eigenvalues (and of course singular values) are sorted, *i.e.*, $\lambda_1 \geq \cdots \geq \lambda_n$ and $\sigma_1 \geq \cdots \geq \sigma_n$. How are the eigenvalues and singular values related?

14.14 *More facts about singular values of matrices.* For each of the following statements, prove it if it is true; otherwise give a *specific* counterexample. Here $X, Y, Z \in \mathbf{R}^{n\times n}$.

(a) $\sigma_{\max}(X) \geq \max_{1\leq i\leq n} \sqrt{\sum_{1\leq j\leq n} |X_{ij}|^2}$.

(b) $\sigma_{\min}(X) \geq \min_{1\leq i\leq n} \sqrt{\sum_{1\leq j\leq n} |X_{ij}|^2}$.

(c) $\sigma_{\max}(XY) \leq \sigma_{\max}(X)\sigma_{\max}(Y)$.

(d) $\sigma_{\min}(XY) \geq \sigma_{\min}(X)\sigma_{\min}(Y)$.

(e) $\sigma_{\min}(X+Y) \geq \sigma_{\min}(X) - \sigma_{\max}(Y)$.

14.15 A matrix can have all entries large and yet have small gain in some directions, that is, it can have a small $\sigma_{\min}$. For example,

$$A = \begin{bmatrix} 10^6 & 10^6 \\ 10^6 & 10^6 \end{bmatrix}$$

has "large" entries while $\|A[1 \ \ -1]^T\| = 0$. Can a matrix have all entries small and yet have a large gain in some direction, that is, a large $\sigma_{\max}$? Suppose, for example, that $|A_{ij}| \leq \epsilon$ for $1 \leq i, j \leq n$. What can you say about $\sigma_{\max}(A)$?

14.16 *Frobenius norm of a matrix.* The Frobenius norm of a matrix $A \in \mathbf{R}^{n\times n}$ is defined as $\|A\|_{\mathrm{F}} = \sqrt{\mathbf{Tr}\, A^T A}$. (Recall $\mathbf{Tr}$ is the trace of a matrix, *i.e.*, the sum of the diagonal entries.)

(a) Show that

$$\|A\|_{\mathrm{F}} = \left(\sum_{i,j} |A_{ij}|^2\right)^{1/2}.$$

Thus the Frobenius norm is simply the Euclidean norm of the matrix when it is considered as an element of $\mathbf{R}^{n^2}$. Note also that it is much easier to compute the Frobenius norm of a matrix than the (spectral) norm (*i.e.*, maximum singular value).

(b) Show that if U and V are orthogonal, then $\|UA\|_{\mathrm{F}} = \|AV\|_{\mathrm{F}} = \|A\|_{\mathrm{F}}$. Thus the Frobenius norm is not changed by a pre- or post- orthogonal transformation.

(c) Show that $\|A\|_{\mathrm{F}} = \sqrt{\sigma_1^2 + \cdots + \sigma_r^2}$, where $\sigma_1, \ldots, \sigma_r$ are the singular values of A. Then show that $\sigma_{\max}(A) \leq \|A\|_{\mathrm{F}} \leq \sqrt{r}\sigma_{\max}(A)$. In particular, $\|Ax\| \leq \|A\|_{\mathrm{F}}\|x\|$ for all x.

14.17 *Drawing a graph.* We consider the problem of drawing (in two dimensions) a graph with n vertices (or nodes) and m undirected edges (or links). This just means assigning an x- and a y- coordinate to each node. We let $x \in \mathbf{R}^n$ be the vector of x- coordinates of the nodes, and $y \in \mathbf{R}^n$ be the vector of y- coordinates of the nodes. When we draw the graph, we draw node i at the location $(x_i, y_i) \in \mathbf{R}^2$. The problem, of course, is to make the drawn graph look good. One goal is that neighboring nodes on the graph (*i.e.*, ones connected by an edge) should not be too far apart as drawn. To take this into account, we will choose the x- and y-coordinates so as to minimize the objective

$$J = \sum_{i<j,\ i\sim j} \left((x_i - x_j)^2 + (y_i - y_j)^2 \right),$$

where $i \sim j$ means that nodes i and j are connected by an edge. The objective J is precisely the sum of the squares of the lengths (in $\mathbf{R}^2$) of the drawn edges of the graph. We have to introduce some other constraints into our problem to get a sensible solution. First of all, the objective J is not affected if we shift all the coordinates by some fixed amount (since J only depends on differences of the x- and y-coordinates). So we can assume that

$$\sum_{i=1}^{n} x_i = 0, \qquad \sum_{i=1}^{n} y_i = 0,$$

i.e., the sum (or mean value) of the x- and y-coordinates is zero. These two equations 'center' our drawn graph. Another problem is that we can minimize J by putting all the nodes at $x_i = 0$, $y_i = 0$, which results in $J = 0$. To force the nodes to spread out, we impose the constraints

$$\sum_{i=1}^{n} x_i^2 = 1, \qquad \sum_{i=1}^{n} y_i^2 = 1, \qquad \sum_{i=1}^{n} x_i y_i = 0.$$

The first two say that the variance of the x- and y- coordinates is one; the last says that the x- and y- coordinates are uncorrelated. (You don't have to know what variance or uncorrelated mean; these are just names for the equations given above.) The three equations above enforce 'spreading' of the drawn graph. Even with these constraints, the coordinates that minimize J are not unique. For example, if x and y are any set of coordinates, and $Q \in \mathbf{R}^{2\times 2}$ is any orthogonal matrix, then the coordinates given by

$$\begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \end{bmatrix} = Q \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \qquad i = 1, \ldots, n$$

satisfy the centering and spreading constraints, and have the same value of J. This means that if you have a proposed set of coordinates for the nodes, then by rotating or reflecting them, you get another set of coordinates that is just as good, according to our objective. We'll just live with this ambiguity. Here's the question:

(a) Explain how to solve this problem, *i.e.*, how to find x and y that minimize J subject to the centering and spreading constraints, given the graph topology. You can use any method or ideas we've encountered in the course. Be clear as to whether your approach solves the problem exactly (*i.e.*, finds a set of coordinates with J as small as it can possibly be), or whether it's just a good heuristic (*i.e.*, a choice of coordinates that achieves a reasonably small value of J, but perhaps not the absolute best). In describing your method, you may not refer to any Matlab commands or operators; your description must be entirely in mathematical terms.

(b) Implement your method, and carry it out for the graph given in `dg_data.m`. This mfile contains the *node adjacency matrix* of the graph, denoted A, and defined as $A_{ij} = 1$ if nodes i and j are connected by an edge, and $A_{ij} = 0$ otherwise. (The graph is undirected, so A is symmetric. Also, we do not have self-loops, so $A_{ii} = 0$, for $i = 1, \ldots, n$.) Draw your final result using the commands `gplot(A, [x y],'o-')`, which plots the graph and `axis('square')`, which sets the x- y- scales to be equal. Give the value of J achieved by your choice of x and y. Verify that your x and y satisfy the centering and spreading conditions, at least approximately. If your method is iterative, plot the value of J versus iteration. The mfile `dg_data.m` also contains the vectors `x_circ` and `y_circ`. These coordinates are obtained using a standard technique for drawing a graph, by placing the nodes in order on a circle. The radius of the circle is chosen so that `x_circ` and `y_circ` satisfy the centering and spread constraints. You can draw the given graph this way using `gplot(A, [x_circ y_circ],'o-'); axis('square');`.

Hint. You are welcome to use the results described below, without proving them. Let $A \in \mathbf{R}^{n \times n}$ be symmetric, with eigenvalue decomposition $A = \sum_{i=1}^{n} \lambda_i q_i q_i^T$, with $\lambda_1 \geq \cdots \geq \lambda_n$, and $\{q_1, \ldots, q_n\}$ orthonormal. You know that a solution of the problem

$$\begin{array}{ll} \text{minimize} & x^T A x \\ \text{subject to} & x^T x = 1, \end{array}$$

where the variable is $x \in \mathbf{R}^n$, is $x = q_n$. The related maximization problem is

$$\begin{array}{ll} \text{maximize} & x^T A x \\ \text{subject to} & x^T x = 1, \end{array}$$

with variable $x \in \mathbf{R}^n$. A solution to this problem is $x = q_1$. Now consider the following generalization of the first problem:

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(X^T A X) \\ \text{subject to} & X^T X = I_k, \end{array}$$

where the variable is $X \in \mathbf{R}^{n \times k}$, and I_k denotes the $k \times k$ identity matrix, and we assume $k \leq n$. The constraint means that the columns of X, say, $x_1, \ldots, x_k$, are orthonormal; the objective can be written in terms of the columns of X as $\mathbf{Tr}(X^T A X) = \sum_{i=1}^{k} x_i^T A x_i$. A solution of this problem is $X = [q_{n-k+1} \cdots q_n]$. Note that when $k = 1$, this reduces to the first problem above. The related maximization problem is

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(X^T A X) \\ \text{subject to} & X^T X = I_k, \end{array}$$

with variable $X \in \mathbf{R}^{n \times k}$. A solution of this problem is $X = [q_1 \cdots q_k]$.

14.18 *Approximate left inverse with norm constraints.* Suppose $A \in \mathbf{R}^{m \times n}$ is full rank with $m \geq n$. We seek a matrix $F \in \mathbf{R}^{n \times m}$ that minimizes $\|I - FA\|$ subject to the constraint $\|F\| \leq \alpha$, where $\alpha > 0$ is given. Note that $\|I - FA\|$ gives a measure of how much F fails to be a left inverse of A. Give an explicit description of an optimal F. Your description can involve standard matrix operations and decompositions (eigenvector/eigenvalue, QR, SVD, ...).

14.19 *Finding worst-case inputs.* The single-input, single output system $x(t+1) = Ax(t) + Bu(t)$, $y(t) = Cx(t)$, $x(0) = 0$, where

$$A = \begin{bmatrix} 0.9 & 0.5 \\ -0.5 & 0.7 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 2 \end{bmatrix},$$

is a very simple (discretized and lumped) dynamical model of a building. The input u is ground displacement (during an earthquake), and y gives the displacement of the top of the building. The input u is known to satisfy $\sum_{t=0}^{49} u(t)^2 \leq 1$ and $u(t) = 0$ for $t \geq 50$, *i.e.*, the earthquake has energy less than one, and only lasts 50 samples.

(a) How large can $\sum_{t=0}^{99} y(t)^2$ be? Plot an input u that maximizes $\sum_{t=0}^{99} y(t)^2$, along with the resulting output y.

(b) How large can $|y(100)|$ be? Plot an input u that maximizes $|y(100)|$, along with the resulting output y.

As usual, you must explain how you solve the problem, as well as give explicit numerical answers and plots.

14.20 *Worst and best direction of excitation for a suspension system.* A suspension system is connected at one end to a base (that can move or vibrate) and at the other to the load (that it is supposed to isolate from vibration of the base). Suitably discretized, the system is described by

$$x(t+1) = Ax(t) + Bu(t), \quad y(t) = Cx(t), \quad x(0) = 0,$$

where $u(t) \in \mathbf{R}^3$ represents the (x-, y-, and z- coordinates of the) displacement of base, and $y(t) \in \mathbf{R}^3$ represents the (x-, y-, and z- coordinates of the) displacement of the load. The input u has the form $u(t) = qv(t)$, where $q \in \mathbf{R}^3$ is a (constant) vector with $\|q\| = 1$, and $v(t) \in \mathbf{R}$ gives the displacement amplitude versus time. In other words, the driving displacement u is always in the direction q, with amplitude given by the (scalar) signal v. The response of the system is judged by the RMS deviation of the load over a 100 sample interval, *i.e.*,

$$D = \left(\frac{1}{100} \sum_{t=1}^{100} \|y(t)\|^2 \right)^{1/2}.$$

The data A, B, C, $v(0), \ldots, v(99)$ are known (and available in the mfile `worst_susp_data.m` on the course web site). The problem is to find the direction $q_{\text{max}} \in \mathbf{R}^3$ that maximizes D, and the direction $q_{\text{min}} \in \mathbf{R}^3$ that minimizes D. Give the directions and the associated values of D (D_{max} and D_{min}, respectively).

14.21 *Two representations of an ellipsoid.* In the lectures, we saw two different ways of representing an ellipsoid, centered at 0, with non-zero volume. The first uses a quadratic form:

$$\mathcal{E}_1 = \left\{ x \,\middle|\, x^T S x \leq 1 \right\},$$

with $S^T = S > 0$. The second is as the image of a unit ball under a linear mapping:

$$\mathcal{E}_2 = \{ y \,|\, y = Ax, \|x\| \leq 1 \},$$

with $\det(A) \neq 0$.

(a) Given S, explain how to find an A so that $\mathcal{E}_1 = \mathcal{E}_2$.

(b) Given A, explain how to find an S so that $\mathcal{E}_1 = \mathcal{E}_2$.

(c) What about uniqueness? Given S, explain how to find *all* A that yield $\mathcal{E}_1 = \mathcal{E}_2$. Given A, explain how to find *all* S that yield $\mathcal{E}_1 = \mathcal{E}_2$.

14.22 *Determining initial bacteria populations.* We consider a population that consists of three strains of a bacterium, called strain 1, strain 2, and strain 3. The vector $x(t) \in \mathbf{R}^3$ will denote the amount, or *biomass* (in grams) of the strains present in the population at time t, measured in hours. For example, $x_2(3.4)$ denotes the amount of strain 2 (in grams) in the sample at time $t = 3.4$ hours. Over time, the biomass of each strain changes through several mechanisms including cell division, cell death, and mutation. (But you don't need to know any biology to answer this question!) The population dynamics is given by $\dot{x} = Ax$, where

$$A = \begin{bmatrix} -0.1 & 0.3 & 0 \\ 0 & -0.2 & 0.1 \\ 0.1 & 0 & -0.1 \end{bmatrix}.$$

You can assume that we always have $x_i(t) > 0$, *i.e.*, the biomass of each strain is always positive. The total biomass at time t is given by $\mathbf{1}^T x(t) = x_1(t) + x_2(t) + x_3(t)$, where $\mathbf{1} \in \mathbf{R}^3$ denotes the vector with all components one.

(a) Give a very brief interpretation of the entries of the matrix A. For example, what is the significance of $a_{13} = 0$? What is the significance of the sign of a_{11}? Limit yourself to 100 words. You may use phrases such as 'the presence of strain i enhances (or inhibits) growth of strain j'.

(b) As $t \to \infty$, does the total biomass converge to ∞ (*i.e.*, grow without bound), converge to zero, or not converge at all (for example, oscillate)? Explain how you arrive at your conclusion and show any calculations (by hand or Matlab) that you need to do. You can assume that $x_i(0) > 0$ for $i = 1,\ 2,\ 3$. *Posterior intuitive explanation.* In 100 words or less, give a plausible story that explains, intuitively, the result you found.

(c) *Selection of optimal assay time.* A biologist wishes to estimate the original biomass of each of the three strains, *i.e.*, the vector $x(0) \in \mathbf{R}^3$, based on measurements of the total biomass taken at $t = 0$, $t = 10$, and $t = T$, where T satisfies $0 < T < 10$. The three measurements of total biomass (which are called *assays*) will include a small additive error, denoted v_1 (for the assay at $t = 0$), v_2 (for the assay at $t = T$ and v_3 (for the assay at $t = 10$). You can assume that $v_1^2 + v_2^2 + v_3^2 \leq 0.01^2$, *i.e.*, the sum of the squares of the measurement errors is not more than 0.01^2. You can also assume that a good method for computing the estimate of $x(0)$, given the measurements, will be used. (The estimation method won't make any use of the information that $x_i(0) > 0$.) The problem here is to choose the time T of the intermediate assay in such a way that the estimate of $x(0)$, denoted $\hat{x}(0)$, is as accurate as possible. We'll judge accuracy by the maximum value that $\|\hat{x}(0) - x(0)\|$ can have, over all measurement errors that satisfy $v_1^2 + v_2^2 + v_3^2 \leq 0.01^2$. Find the optimal value for T (of course, between 0 and 10), *i.e.*, the value of T that minimizes the maximum value $\|\hat{x}(0) - x(0)\|$ can have. We are looking for an answer that is accurate to within ± 0.1. Of course you must explain exactly what you are doing, and submit your Matlab code as well the output it produces. Be sure to say what the optimal T is, and what the optimal accuracy is (*i.e.*, what the maximum value $\|\hat{x}(0) - x(0)\|$ is, for the T you choose).

14.23 *A measure of connectedness in a graph.* We consider an undirected graph with n nodes, described by its adjacency matrix $A \in \mathbf{R}^{n\times n}$, defined by

$$A_{ij} = \begin{cases} 1 & \text{if there is a link connecting nodes } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases}$$

We assume the graph has no self-loops, *i.e.*, $A_{ii} = 0$. Note that $A = A^T$. We assume that the graph has at least one link, so $A \neq 0$. A *path* from node i to node j, of length $m > 0$, is an $m+1$-long sequence of nodes, connected by links, that start at i and end at j. More precisely it is a sequence $i = k_1,\ k_2, \ldots, k_{m+1} = j$, with the property that $A_{k_1,k_2} = \cdots = A_{k_m,k_{m+1}} = 1$. Note that a path can include loops; there is no requirement that each node be visited only once. For example, if node 3 and node 5 are connected by a link (*i.e.*, $A_{35} = 1$), then the sequence $3, 5, 3, 5$ is a path between node 3 and node 5 of length 3. We say that each node is connected to itself by a path of length zero. Let $P_m(i, j)$ denote the total number of paths of length m from node i to node j. We define

$$C_{ij} = \lim_{m\to\infty} \frac{P_m(i,j)}{\sum_{i,j=1}^n P_m(i,j)},$$

when the limits exist. When the limits don't, we say that C_{ij} isn't defined. In the fraction in this equation, the numerator is the number of paths of length m between nodes i and j, and the denominator is the total number of paths of length m, so the ratio gives the fraction of all paths of length m that go between nodes i and j. When C_{ij} exists, it gives the asymptotic fraction of all (long) paths that go from node i to node j. The number C_{ij} gives a good measure of how "connected" nodes i and j are in the graph. You can make *one* of the following assumptions:

(a) A is full rank.

(b) A has distinct eigenvalues.

(c) A has distinct singular values.

(d) A is diagonalizable.

(e) A has a dominant eigenvalue, *i.e.*, $|\lambda_1| > |\lambda_i|$ for $i = 2, \ldots, n$, where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A.

(Be very clear about which one you choose.) Using your assumption, explain why C_{ij} exists, and derive an expression for C_{ij}. You can use any of the concepts from the class, such as singular values and singular vectors, eigenvalues and eigenvectors, pseudo-inverse, etc., but you cannot leave a limit in your expression. You must explain why your expression is, in fact, equal to C_{ij}.

14.24 *Recovering an ellipsoid from boundary points.* You are given a set of vectors $x^{(1)}, \ldots, x^{(N)} \in \mathbf{R}^n$ that are thought to lie on or near the surface of an ellipsoid centered at the origin, which we represent as

$$\mathcal{E} = \{x \in \mathbf{R}^n \mid x^T A x = 1\},$$

where $A = A^T \in \mathbf{R}^{n\times n} \geq 0$. Your job is to recover, at least approximately, the matrix A, given the observed data $x^{(1)}, \ldots, x^{(N)}$. Explain your approach to this problem, and then carry it out on the data given in the mfile `ellip_bdry_data.m`. Be sure to explain how you check that the ellipsoid you find is reasonably consistent with the given data, and also that the matrix A you find does, in fact, correspond to an ellipsoid. To simplify the explanation, you can give it for the case $n = 4$ (which is the dimension of the given data). But it should be clear from your discussion how it works in general.

14.25 *Predicting zero crossings.* We consider a linear system of the form

$$\dot{x} = Ax, \qquad y = Cx,$$

where $x(t) \in \mathbf{R}^n$ and $y(t) \in \mathbf{R}$. We know A and C, but we do not know $x(0)$. We cannot directly observe the output, but we do know the times at which the output is zero, *i.e.*, we are given the *zero-crossing times* $t_1, \ldots, t_p$ at which $y(t_i) = 0$. You can assume these times are given in increasing order, *i.e.*, $0 \leq t_1 < \cdots < t_p$, and that $y(t) \neq 0$ for $0 \leq t < t_p$ and $t \neq t_1, \ldots, t \neq t_p$. (Note that this definition of zero-crossing times doesn't require the output signal to *cross* the value zero; it is enough to just have the value zero.) We are interested in the following question: given A, C, and the zero-crossing times $t_1, \ldots, t_p$, can we *predict* the next zero-crossing time t_{p+1}? (This means, of course, that $y(t) \neq 0$ for $t_p < t < t_{p+1}$, and $y(t_{p+1}) = 0$.) You will answer this question for the specific system

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -18 & -11 & -12 & -2 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix},$$

and zero-crossing times

$$t_1 = 0.000, \quad t_2 = 1.000, \quad t_3 = 2.000, \quad t_4 = 3.143.$$

(So here we have $p = 4$.) Note that the zero-crossing times are given to three significant digits. Specifically, you must do one of the following:

- If you think that you can determine the next zero-crossing time t_5, explain in detail how to do it, and find the next time t_5 (to at least two significant figures).
- If you think that you cannot determine the next zero-crossing time t_5, explain in detail why, and find two trajectories of the system which have $t_1, \ldots, t_4$ as the first 4 zero-crossings, but have different 5th zero-crossings. (The zero-crossings should differ substantially, and not just in the last significant digit.)

Be sure to make it clear which one of these options you choose. *Hint:* Be careful estimating rank or determining singularity, if that's part of your procedure; remember that the zero-crossing times are only given to three significant figures.

14.26 *Optimal time compression equalizer.* We are given the (finite) impulse response of a communications channel, *i.e.*, the real numbers

$$c_1, c_2, \ldots, c_n.$$

Our goal is to design the (finite) impulse response of an equalizer, *i.e.*, the real numbers

$$w_1, w_2, \ldots, w_n.$$

(To make things simple, the equalizer has the same length as the channel.) The equalized channel response h is given by the convolution of w and c, *i.e.*,

$$h_i = \sum_{j=1}^{i-1} w_j c_{i-j}, \quad i = 2, \ldots, 2n.$$

This is shown below.

The goal is to choose w so that most of the energy of the equalized impulse response h is *concentrated* within k samples of $t = n + 1$, where $k < n - 1$ is given. To define this formally, we first define the total energy of the equalized response as

$$E_{\text{tot}} = \sum_{i=2}^{2n} h_i^2,$$

and the energy in the desired time interval as

$$E_{\text{des}} = \sum_{i=n+1-k}^{n+1+k} h_i^2.$$

For any w for which $E_{\text{tot}} > 0$, we define the *desired to total energy ratio*, or DTE, as $\text{DTE} = E_{\text{des}}/E_{\text{tot}}$. Thus number is clearly between 0 and 1; it tells us what fraction of the energy in h is contained in the time interval $t = n + 1 - k, \ldots, t = n + 1 + k$. You can assume that h is such that for any $w \neq 0$, we have $E_{\text{tot}} > 0$.

(a) How do you find a $w \neq 0$ that maximizes DTE? You must give a very clear description of your method, and explain why it works. Your description and justification must be *very clear*. You can appeal to any concepts used in the class, *e.g.*, least-squares, least-norm, eigenvalues and eigenvectors, singular values and singular vectors, matrix exponential, and so on.

(b) Carry out your method for time compression length $k = 1$ on the data found in `time_comp_data.m`. Plot your solution w, the equalized response h, and give the DTE for your w.

Please note: You do not need to know anything about equalizers, communications channels, or even convolution; everything you need to solve this problem is clearly defined in the problem statement.

14.27 *Minimum energy required to leave safe operating region.* We consider the stable controllable system $\dot{x} = Ax + Bu$, $x(0) = 0$, where $x(t) \in \mathbf{R}^n$ and $u(t) \in \mathbf{R}^m$. The input u is beyond our control, but we have some idea of how large its total energy

$$\int_0^\infty \|u(\tau)\|^2 \, d\tau$$

is likely to be. The *safe operating region* for the state is the unit ball

$$\mathcal{B} = \{ \, x \mid \|x\| \leq 1 \, \}.$$

The hope is that input u will not drive the state outside the safe operating region. One measure of system security that is used is the minimum energy $E_{\min}$ that is required to drive the state outside the safe operating region:

$$E_{\min} = \min \left\{ \int_0^t \|u(\tau)\|^2 \, d\tau \;\middle|\; x(t) \notin \mathcal{B} \right\}.$$

(Note that we do not specify t, the time at which the state is outside the safe operating region.) If $E_{\min}$ is much larger than the energy of the u's we can expect, we can be fairly confident that the state will not leave the safe operating region. ($E_{\min}$ can also be justified as a system security measure on statistical grounds, but we won't go into that here.)

(a) Find $E_{\min}$ explicitly. Your solution should be in terms of the matrices A, B, or other matrices derived from them such as the controllability matrix $\mathcal{C}$, the controllability Gramian W_c, and its inverse $P = W_c^{-1}$. Make sure you give the simplest possible expression for $E_{\min}$.

(b) Suppose the safe operating region is the unit cube $C = \{ \, x \mid |x_i| \leq 1, \; i = 1, \ldots, n \, \}$ instead of the unit ball $\mathcal{B}$. Let $E_{\min}^{\text{cube}}$ denote the minimum energy required to drive the state outside the unit cube C. Repeat part (a) for $E_{\min}^{\text{cube}}$.

14.28 *Energy storage efficiency in a linear dynamical system.* We consider the discrete-time linear dynamic system

$$x(t+1) = Ax(t) + Bu(t), \quad y(t) = Cx(t),$$

where $x(t) \in \mathbf{R}^n$, and $u(t), y(t) \in \mathbf{R}$. The initial state is zero, *i.e.*, $x(0) = 0$. We apply an input sequence $u(0), \ldots, u(N-1)$, and are interested in the output over the next N samples, *i.e.*, $y(N), \ldots, y(2N-1)$. (We take $u(t) = 0$ for $t \geq N$.) We define the *input energy* as

$$\mathcal{E}_{\text{in}} = \sum_{t=0}^{N-1} u(t)^2,$$

and similarly, the output energy is defined as

$$\mathcal{E}_{\text{out}} = \sum_{t=N}^{2N-1} y(t)^2.$$

How would you choose the (nonzero) input sequence $u(0), \ldots, u(N-1)$ to maximize the ratio of output energy to input energy, *i.e.*, to maximize $\mathcal{E}_{\text{out}}/\mathcal{E}_{\text{in}}$? What is the maximum value the ratio $\mathcal{E}_{\text{out}}/\mathcal{E}_{\text{in}}$ can have?

14.29 *Energy-optimal evasion.* A vehicle is governed by the following discrete-time linear dynamical system equations:

$$x(t+1) = Ax(t) + Bu(t), \quad y(t) = Cx(t), \quad x(0) = 0.$$

Here $x(t) \in \mathbf{R}^n$ is the vehicle state, $y(t) \in \mathbf{R}^3$ is the vehicle position, and $u(t) \in \mathbf{R}^m$ is the input signal. (The vehicle dynamics are really continuous; the equation above is the result of a suitable sampling.) The system is controllable.

(a) *Minimum energy to reach a position.* Find the input $u(0), \ldots, u(T-1)$ that reaches position $f \in \mathbf{R}^3$ at time T (where $T \geq n$), *i.e.*, $y(T) = f$, and minimizes the input 'energy'

$$\|u(0)\|^2 + \cdots + \|u(T-1)\|^2.$$

The input u is the (energy) optimal input for the vehicle to arrive at the position f at time T. Give an expression for E, the energy of the minimum energy input. (Of course E will depend on the data A, B, C, and f.)

(b) *Energy-optimal evasion.* Now consider a second vehicle governed by

$$z(t+1) = Fz(t) + Gv(t), \quad w(t) = Hz(t), \quad z(0) = 0$$

where $z(t) \in \mathbf{R}^n$ is the state of the vehicle, $w(t) \in \mathbf{R}^3$ is the vehicle position, and $v(t) \in \mathbf{R}^m$ is the input signal. This vehicle is to be overtaken (intercepted) by the first vehicle at time T, where $T \geq n$. This means that $w(T) = y(T)$. How would you find $v(0), \ldots, v(T-1)$ that maximizes the minimum energy the first vehicle must expend to intercept the second vehicle at time T, subject to a limit on input energy,

$$\|v(0)\|^2 + \cdots + \|v(T-1)\|^2 \leq 1?$$

The input v is maximally evasive, in the sense that is requires the first vehicle to expend the largest amount of input energy to overtake it, given the limit on input energy the second vehicle is allowed to use. Express your answer in terms of the data A, B, C, F, G, H, and standard matrix functions (inverse, transpose, SVD, ...). *Remark:* This problem is obviously not a very realistic model of a real pursuit-evasion situation, for several reasons: both vehicles start from the zero initial state, the time of intercept (T) is known to the second vehicle, and the place of intercept ($w(T)$) is known ahead of time to the first vehicle. Still, it's possible to extend the results of this problem to handle a realistic model of a pursuit/evasion.

14.30 *Worst-case analysis of impact.* We consider a (time-invariant) linear dynamical system

$$\dot{x} = Ax + Bu, \qquad x(0) = x_{\text{init}},$$

with state $x(t) \in \mathbf{R}^n$, and input $u(t) \in \mathbf{R}^m$. We are interested in the state trajectory over the time interval $[0, T]$. In this problem the input u represents an *impact* on the system, so it has the form

$$u(t) = g\delta(t - T_{\text{imp}}),$$

where $g \in \mathbf{R}^m$ is a vector that gives the direction and magnitude of the impact, and T_{imp} is the time of the impact. We assume that $0 \leq T_{\text{imp}} \leq T_{-}$. ($T_{\text{imp}} = T_{-}$ means that the impact occurs right at the end of the period of interest, and does affect $x(T)$.) We let $x_{\rm nom}(T)$ denote the state, at time $t = T$, of the linear system $\dot{x}_{\rm nom} = Ax_{\rm nom}$, $x_{\rm nom}(0) = x_{\rm init}$. The vector $x_{\rm nom}(T)$ is what the final state $x(T)$ of the system above would have been at time $t = T$, had the impact not occured (*i.e.*, with $u = 0$). We are interested in the deviation D between $x(T)$ and $x_{\rm nom}(T)$, as measured by the norm:

$$D = \|x(T) - x_{\rm nom}(T)\|.$$

D measures how far the impact has shifted the state at time T. We would like to know how large D can be, over all possible impact directions and magnitudes no more than one (*i.e.*, $\|g\| \leq 1$), and over all possible impact times between 0 and T_{-}. In other words, we would like to know the maximum possible state deviation, at time T, due to an impact of magnitude no more than one. We'll call the choices of $T_{\rm imp}$ and g that maximize D the *worst-case impact time* and *worst-case impact vector*, respectively.

(a) Explain how to find the worst-case impact time, and the worst-case impact vector, given the problem data A, B, $x_{\rm init}$, and T. Your explanation should be as short and clear as possible. You can use any of the concepts we have encountered in the class. Your approach can include a simple numerical search (such as plotting a function of one variable to find its maximum), if needed. If either the worst-case impact time or the worst-case impact vector do not depend on some of the problem data (*i.e.*, A, B, $x_{\rm init}$, and T) say so.

(b) Get the data from `wc_impact_data.m`, which defines A, B, $x_{\rm init}$, and T, and carry out the procedure described in part (a). Be sure to give us the worst-case impact time (with absolute precision of 0.01), the worst-case impact vector, and the corresponding value of D.

14.31 *Worst time for control system failure.* In this problem we consider a system that under normal circumstances is governed by the equations

$$\dot{x}(t) = Ax(t) + Bu(t), \qquad u(t) = Kx(t). \tag{3}$$

(This is called *state feedback*, and is very commonly used in automatic control systems.) Here the application is a regulator, which means that input u is meant to drive the state to zero as $t \to \infty$, no matter what $x(0)$ is. At time $t = T_f$, however, a fault occurs, and the input signal becomes zero. The fault is cleared (*i.e.*, corrected) T_c seconds after it occurs. Thus, for $T_f \leq t \leq T_f + T_c$, we have $\dot{x}(t) = Ax(t)$; for $t < T_f$ and $t > T_f + T_c$, the system is governed by the equations (3). You'll find the specific matrices A, B, and K, in the mfile `fault_ctrl_sys.m` on the class web site. Here's the problem: suppose the system fails for one second, some time in the time interval $[0, 9]$. In other words, we have $0 \leq T_f \leq 9$, and $T_c = 1$. We don't know what $x(0)$ is, but you can assume that $\|x(0)\| \leq 1$. We also don't know the time of failure T_f. The problem is to find the time of failure T_f (in $[0, 9]$) and the initial condition $x(0)$ (with $\|x(0)\| \leq 1$) that maximizes $\|x(10)\|$. In essence, you are carrying out a worst-case analysis of the effects of a one second control system failure. As usual, you must explain your approach clearly and completely. You must also give your source code, and the results, *i.e.*, the worse possible $x(0)$, the worst failure time T_f, and the resulting value of $\|x(10)\|$. An accuracy of 0.01 for T_f is fine.

14.32 *Some proof or counterexample questions.* Determine if the following statements are true or false. If the statement is true, prove it; if you think it is false, provide a *specific* (numerical) counterexample. You get five points for the correct solution (*i.e.*, the right answer and a valid proof or counterexample), two points for the right answer (*i.e.*, true or false), and zero points otherwise. What we mean by "true" is that the statement is true for all values of the matrices and vectors given. (You can assume the entries of the matrices and vectors are all real.) You can't assume anything about the dimensions of the matrices (unless it's explicitly stated), but you can assume that the dimensions are such that all expressions make sense. For example, the statement "$A+B=B+A$" is true, because no matter what the dimensions of A and B (which must, however, be the same), and no matter what values A and B have, the statement holds. As another example, the statement $A^2=A$ is false, because there are (square) matrices for which this doesn't hold. In such a case, provide a specific counterexample, for example, $A=2$ (which is a matrix in $\mathbf{R}^{1\times 1}$).

(a) Suppose $A=A^T\in\mathbf{R}^{n\times n}$ satisfies $A\geq 0$, and $A_{kk}=0$ for some k (between 1 and n). Then A is singular.

(b) Suppose $A,\ B\in\mathbf{R}^{n\times n}$, with $\|A\|>\|B\|$. Then, for all $k\geq 1$, $\|A^k\|\geq\|B^k\|$.

(c) Suppose $\tilde{A}$ is a submatrix of a matrix $A\in\mathbf{R}^{m\times n}$. (This means $\tilde{A}$ is obtained from A by removing some rows and columns from A; as an extreme case, any element of A is a (1×1) submatrix of A.) Then $\|\tilde{A}\|\leq\|A\|$.

(d) For any A, B, C, D with compatible dimensions (see below),

$$\left\|\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right\| \leq \left\|\begin{bmatrix} \|A\| & \|B\| \\ \|C\| & \|D\| \end{bmatrix}\right\|.$$

Compatible dimensions means: A and B have the same number of rows, C and D have the same number of rows, A and C have the same number of columns, and B and D have the same number of columns.

(e) For any A and B with the same number of columns, we have

$$\max\{\|A\|,\|B\|\} \leq \left\|\begin{bmatrix} A \\ B \end{bmatrix}\right\| \leq \sqrt{\|A\|^2+\|B\|^2}.$$

(f) Suppose the fat (including, possibly, square) and full rank matrices A and B have the same number of rows. Then we have $\kappa(A)\leq\kappa([\ A\ \ B\ ])$, where $\kappa(Z)$ denotes, as usual, the condition number of the matrix Z, *i.e.*, the ratio of the largest to the smallest singular value.

14.33 *Uncovering a hidden linear explanation.* Consider a set of vectors $y_1,\ldots,y_N\in\mathbf{R}^n$, which might represent a collection of measurements or other data. Suppose we have

$$y_i \approx Ax_i+b, \quad i=1,\ldots,N,$$

where $A\in\mathbf{R}^{n\times m}$, $x_i\in\mathbf{R}^m$, and $b\in\mathbf{R}^n$, with $m<n$. (Our main interest is in the case when N is much larger than n, and m is smaller than n.) Then we say that $y=Ax+b$ is a *linear explanation* of the data y. We refer to x as the vector of *factors* or *underlying causes* of the

data y. For example, suppose $N = 500$, $n = 30$, and $m = 5$. In this case we have 500 vectors; each vector y_i consists of 30 scalar measurements or data points. But these 30-dimensional data points can be 'explained' by a much smaller set of 5 'factors' (the components of x_i). This problem is about uncovering, or discovering, a linear explanation of a set of data, given only the data. In other words, we are given $y_1, \ldots, y_N$, and the goal is to find m, A, b, and $x_1, \ldots, x_N$ so that $y_i \approx Ax_i + b$. To judge the accuracy of a proposed explanation, we'll use the RMS explanation error, *i.e.*,

$$J = \left(\frac{1}{N} \sum_{i=1}^{N} \|y_i - Ax_i - b\|^2 \right)^{1/2}.$$

One rather simple linear explanation of the data is obtained with $x_i = y_i$, $A = I$, and $b = 0$. In other words, the data explains itself! In this case, of course, we have $y_i = Ax_i + b$, so the RMS explanation error is zero. But this is not what we're after. To be a useful explanation, we want to have m substantially smaller than n, *i.e.*, substantially fewer factors than the dimension of the original data (and for this smaller dimension, we'll accept a nonzero, but hopefully small, value of J.) Generally, we want m, the number of factors in the explanation, to be as small as possible, subject to the constraint that J is not too large. Even if we fix the number of factors as m, a linear explanation of a set of data is not unique. Suppose A, b, and $x_1, \ldots, x_N$ is a linear explanation of our data, with $x_i \in \mathbf{R}^m$. Then we can multiply the matrix A by two (say), and divide each vector x_i by two, and we have another linear explanation of the original data. More generally, let $F \in \mathbf{R}^{m \times m}$ be invertible, and $g \in \mathbf{R}^m$. Then we have

$$y_i \approx Ax_i + b = (AF^{-1})(Fx_i + g) + (b - AF^{-1}g).$$

Thus,

$$\tilde{A} = AF^{-1}, \qquad \tilde{b} = b - AF^{-1}g, \qquad \tilde{x}_1 = Fx_1 + g, \quad \ldots, \quad \tilde{x}_N = Fx_N + g$$

is another equally good linear explanation of the data. In other words, we can apply any affine (*i.e.*, linear plus constant) mapping to the underlying factors x_i, and generate another equally good explanation of the original data by appropriately adjusting A and b. To standardize or normalize the linear explanation, it is usually assumed that

$$\frac{1}{N} \sum_{i=1}^{N} x_i = 0, \qquad \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T = I.$$

In other words, the underlying factors have an average value zero, and unit sample covariance. (You don't need to know what covariance is — it's just a definition here.) Finally, the problem.

(a) Explain clearly how you would find a hidden linear explanation for a set of data $y_1, \ldots, y_N$. Be sure to say how you find m, the dimension of the underlying causes, the matrix A, the vector b, and the vectors $x_1, \ldots, x_N$. Explain clearly why the vectors $x_1, \ldots, x_N$ have the required properties.

(b) Carry out your method on the data in the file `linexp_data.m` available on the course web site. The file gives the matrix $Y = [y_1 \ \cdots \ y_N]$. Give your final A, b, and $x_1, \ldots, x_N$, and verify that $y_i \approx Ax_i - b$ by calculating the norm of the error vector, $\|y_i - Ax_i - b\|$, for $i = 1, \ldots, N$. Sort these norms in descending order and plot them. (This gives a good picture of the distribution of explanation errors.) By explicit computation verify that the vectors $x_1, \ldots, x_N$ obtained, have the required properties.

14.34 *Some bounds on singular values.* Suppose $A \in \mathbf{R}^{6\times 3}$, with singular values 7, 5, 3, and $B \in \mathbf{R}^{6\times 3}$, with singular values 2, 2, 1. Let $C = [A\ B] \in \mathbf{R}^{6\times 6}$, with full SVD $C = U\Sigma V^T$, with $\Sigma = \mathbf{diag}(\sigma_1, \ldots, \sigma_6)$. (We allow the possibility that some of these singular values are zero.)

(a) How large can σ_1 be?

(b) How small can σ_1 be?

(c) How large can σ_6 be?

(d) How small can σ_6 be?

What we mean is, how large (or small) can the specified quantity be, for any A and B with the given sizes and given singular values.

Give your answers with 3 digits after the decimal place, as in

$$(a)\ 12.420, \qquad (b)\ 10.000, \qquad (c)\ 0.552, \qquad (d)\ 0.000.$$

(This is just an example.) Briefly justify your answers. find A and B that achieve the values you give.

Lecture 15 – SVD applications

15.1 *Smallest matrix with given row and column sums.* Explain how to find the matrix $A \in \mathbf{R}^{m\times n}$ that minimizes

$$J = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2,$$

subject to the constraints

$$\sum_{j=1}^{n} A_{ij} = r_i, \quad i = 1,\ldots,m, \qquad \sum_{i=1}^{m} A_{ij} = c_j, \quad j = 1,\ldots,n.$$

Here, r_i (which give the rows sums) are given, as are c_j (which give the column sums). You can assume that $\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} c_j$; if this doesn't hold, there is no A that can satisfy the constraints. Using matrix notation, the objective can be written as $J = \mathbf{Tr}(A^TA)$, and the constraints are

$$A\mathbf{1} = r, \qquad A^T\mathbf{1} = c,$$

where $\mathbf{1}$ denotes a vector (of appropriate size in each case) with all components one. The data $r \in \mathbf{R}^m$ and $c \in \mathbf{R}^n$ must satisfy $\mathbf{1}^Tr = \mathbf{1}^Tc$. Explain your method in the general case. If you can give a nice formula for the optimal A, do so. In addition, carry out your method for the specific data

$$r = \begin{bmatrix} 30\\ 18\\ 26\\ 22\\ 14\\ 34 \end{bmatrix}, \qquad c = \begin{bmatrix} 24\\ 20\\ 16\\ 8\\ 28\\ 32\\ 4\\ 12 \end{bmatrix},$$

with $m = 6$ and $n = 8$. (Entries in A do not have to be integers.)

15.2 *Condition number.* Show that $\kappa(A) = 1$ if and only if A is a multiple of an orthogonal matrix. Thus the best conditioned matrices are precisely (scaled) orthogonal matrices.

15.3 *Tightness of the condition number sensitivity bound.* Suppose A is invertible, $Ax = y$, and $A(x + \delta x) = y + \delta y$. In the lecture notes we showed that $\|\delta x\|/\|x\| \le \kappa(A)\|\delta y\|/\|y\|$. Show that this bound is not conservative, *i.e.*, there are x, y, δx, and δy such that equality holds. *Conclusion:* the bound on relative error can be taken on, if the data x is in a particularly unlucky direction and the data error δx is in (another) unlucky direction.

15.4 *Sensitivity to errors in A.* This problem concerns the relative error incurred in solving a set of linear equations when there are errors in the *matrix* A (as opposed to error in the data vector b). Suppose A is invertible, $Ax = b$, and $(A + \delta A)(x + \delta x) = b$. Show that $\|\delta x\|/\|x + \delta x\| \le \kappa(A)\|\delta A\|/\|A\|$.

15.5 *Minimum and maximum RMS gain of an FIR filter.* Consider an FIR filter with impulse response

$$h_1 = 1, \quad h_2 = 0.6, \quad h_3 = 0.2, \quad h_4 = -0.2, \quad h_5 = -0.1.$$

We'll consider inputs of length 50 (*i.e.*, input signal that are zero for $t > 50$), so the output will have length (up to) 54, since the FIR filter has length 5. Let $u \in \mathbf{R}^{50}$ denote the input signal, and $y \in \mathbf{R}^{54}$ denote the resulting output signal. The RMS gain of the filter for a signal u is defined as

$$g = \frac{\frac{1}{\sqrt{54}}\|y\|}{\frac{1}{\sqrt{50}}\|u\|},$$

which is the ratio of the RMS value of the output to the RMS value of the input. Obviously, the gain g depends on the input signal.

(a) Find the maximum RMS gain $g_{\max}$ of the FIR filter, *i.e.*, the largest value g can have. Plot the corresponding input signal whose RMS gain is $g_{\max}$.

(b) Find the minimum RMS gain $g_{\min}$ of the FIR filter, *i.e.*, the smallest value g can have. Plot the corresponding input signal whose RMS gain is $g_{\min}$.

(c) Plot the magnitude of the transfer function of the FIR filter, *i.e.*, $|H(e^{j\Omega})|$, where

$$H(e^{j\Omega}) = \sum_{k=1}^{5} h_k e^{-jk\Omega}.$$

Find the maximum and minimum absolute values of the transfer function, and the frequencies at which they are attained. Compare to the results from parts a and b. *Hint:* To plot the magnitude of the transfer function, you may want to use the `freqz` Matlab command. Make sure you understand what `freqz` does (using `help freqz`, for example).

(d) (This part is for fun.) Make a conjecture about the maximum and minimum singular values of a Toeplitz matrix, and the associated left and right singular vectors.

15.6 *Detecting linear relations.* Suppose we have N measurements $y_1, \ldots, y_N$ of a vector signal $x_1, \ldots, x_N \in \mathbf{R}^n$:

$$y_i = x_i + d_i, \; i = 1, \ldots, N.$$

Here d_i is some small measurement or sensor noise. We hypothesize that there is a linear relation among the components of the vector signal x, *i.e.*, there is a nonzero vector q such that $q^T x_i = 0$, $i = 1, \ldots, N$. The geometric interpretation is that all of the vectors x_i lie in the hyperplane $q^T x = 0$. We will assume that $\|q\| = 1$, which does not affect the linear relation. Even if the x_i's do lie in a hyperplane $q^T x = 0$, our measurements y_i will not; we will have $q^T y_i = q^T d_i$. These numbers are small, assuming the measurement noise is small. So the problem of determing whether or not there is a linear relation among the components of the vectors x_i comes down to finding out whether or not there is a unit-norm vector q such that $q^T y_i$, $i = 1, \ldots, N$, are all small. We can view this problem geometrically as well. Assuming that the x_i's all lie in the hyperplane $q^T x = 0$, and the d_i's are small, the y_i's will all lie close to the hyperplane. Thus a scatter plot of the y_i's will reveal a sort of flat cloud, concentrated near the hyperplane $q^T x = 0$. Indeed, for any z and $\|q\| = 1$, $|q^T z|$ is the distance from the vector z to the hyperplane $q^T x = 0$. So we seek a vector q, $\|q\| = 1$, such that all the measurements $y_1, \ldots, y_N$ lie close to the hyperplane $q^T x = 0$ (that is, $q^T y_i$ are all small). How can we determine if there is such a vector, and what is its value? We define the following normalized measure:

$$\rho = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(q^T y_i)^2} \Bigg/ \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|y_i\|^2}.$$

This measure is simply the ratio between the *root mean square distance* of the vectors to the hyperplane $q^T x = 0$ and the *root mean square length* of the vectors. If ρ is small, it means that the measurements lie close to the hyperplane $q^T x = 0$. Obviously, ρ depends on q. Here is the problem: explain how to find the minimum value of ρ over all unit-norm vectors q, and the unit-norm vector q that achieves this minimum, given the data set $y_1, \ldots, y_N$.

15.7 *Stability conditions for the distributed congestion control scheme.* We consider the congestion control scheme in problem 3, and will use the notation from that problem. In this problem, we study the dynamics and convergence properties of the rate adjustment scheme. To simplify things, we'll assume that the route matrix R is skinny and full rank. You can also assume that $\alpha > 0$. Let $\bar{x}_{\text{ls}} = (R^T R)^{-1} R^T T^{\text{target}}$ denote the least-squares approximate solution of the (over-determined) equations $Rx \approx T^{\text{target}}$. (The source rates given by $\bar{x}_{\text{ls}}$ minimize the sum of the squares of the congestion measures on all paths.)

(a) Find the conditions on the update parameter α under which the rate adjustment scheme converges to $\bar{x}_{\text{ls}}$, no matter what the initial source rate is.

(b) Find the value of α that results in the fastest possible asymptotic convergence of the rate adjustment algorithm. Find the associated asymptotic convergence rate. We define the convergence rate as the smallest number c for which $\|x(t) - \bar{x}_{\text{ls}}\| \leq ac^t$ holds for all trajectories and all t (the constant a can depend on the trajectory).

You can give your solutions in terms of any of the concepts we have studied, *e.g.*, matrix exponential, eigenvalues, singular values, condition number, and so on. Your answers can, of course, depend on R, T^{target}, and $\bar{x}_{\text{ls}}$. If your answer doesn't depend on some of these (or even all of them) be sure to point this out. We'll take points off if you give a solution that is correct, but not as simple as it can be.

15.8 Consider the system $\dot{x} = Ax$ with

$$A = \begin{bmatrix} 0.3132 & 0.3566 & 0.2545 & 0.2579 & 0.2063 \\ -0.0897 & 0.2913 & 0.1888 & 0.4392 & 0.1470 \\ 0.0845 & 0.2433 & -0.5888 & -0.0407 & 0.1744 \\ 0.2478 & -0.1875 & 0.2233 & 0.3126 & -0.6711 \\ 0.1744 & 0.2315 & -0.1004 & -0.2111 & 0.0428 \end{bmatrix}.$$

(a) Find the initial state $x(0) \in \mathbf{R}^5$ satisfying $\|x(0)\| = 1$ such that $\|x(3)\|$ is maximum. In other words, find an initial condition of unit norm that produces the *largest* state at $t = 3$.

(b) Find the initial state $x(0) \in \mathbf{R}^5$ satisfying $\|x(0)\| = 1$ such that $\|x(3)\|$ is minimum.

To save you the trouble of typing in the matrix A, you can find it on the web page in the file `max_min_init_state.m`.

15.9 *Regularization and SVD.* Let $A \in \mathbf{R}^{n\times n}$ be full rank, with SVD

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T.$$

(We consider the square, full rank case just for simplicity; it's not too hard to consider the general nonsquare, non-full rank case.) Recall that the regularized approximate solution of $Ax = y$ is defined as the vector $x_{\rm reg} \in \mathbf{R}^n$ that minimizes the function

$$\|Ax - y\|^2 + \mu\|x\|^2,$$

where $\mu > 0$ is the regularization parameter. The regularized solution is a linear function of y, so it can be expressed as $x_{\rm reg} = By$ where $B \in \mathbf{R}^{n\times n}$.

(a) Express the SVD of B in terms of the SVD of A. To be more specific, let

$$B = \sum_{i=1}^{n} \tilde{\sigma}_i \tilde{u}_i \tilde{v}_i^T$$

denote the SVD of B. Express $\tilde{\sigma}_i$, $\tilde{u}_i$, $\tilde{v}_i$, for $i = 1, \ldots, n$, in terms of σ_i, u_i, v_i, $i = 1, \ldots, n$ (and, possibly, μ). Recall the convention that $\tilde{\sigma}_1 \geq \cdots \geq \tilde{\sigma}_n$.

(b) Find the norm of B. Give your answer in terms of the SVD of A (and μ).

(c) Find the worst-case relative inversion error, defined as

$$\max_{y\neq 0} \frac{\|ABy - y\|}{\|y\|}.$$

Give your answer in terms of the SVD of A (and μ).

15.10 *Optimal binary signalling.* We consider a communication system given by

$$y(t) = Au(t) + v(t), \quad t = 0, 1, \ldots.$$

Here

- $u(t) \in \mathbf{R}^n$ is the transmitted (vector) signal at time t
- $y(t) \in \mathbf{R}^m$ is the received (vector) signal at time t
- $v(t) \in \mathbf{R}^m$ is noise at time t
- $t = 0,\ 1,\ \ldots$ is the (discrete) time

Note that the system has no memory: $y(t)$ depends only on $u(t)$. For the noise, we assume that $\|v(t)\| < V_{\rm max}$. Other than this maximum value for the norm, we know nothing about the noise (for example, we do not assume it is random). We consider binary signalling, which means that at each time t, the transmitter sends one of two signals, *i.e.*, we have either $u(t) = s_1 \in \mathbf{R}^n$ or $u(t) = s_2 \in \mathbf{R}^n$. The receiver then guesses which of the two signals was sent, based on $y(t)$. The process of guessing which signal was sent, based on the received signal $y(t)$, is called *decoding*. In this problem we are only interested in the case when the communication is completely reliable, which means that the receiver's estimate of which signal was sent is always correct, no matter what $v(t)$ is (provided $\|v(t)\| < V_{\rm max}$, of course). Intuition suggests that this is possible when $V_{\rm max}$ is small enough.

(a) Your job is to design the signal constellation, *i.e.*, the vectors $s_1 \in \mathbf{R}^n$ and $s_2 \in \mathbf{R}^n$, and the associated (reliable) decoding algorithm used by the receiver. Your signal constellation should minimize the maximum transmitter power, *i.e.*,

$$P_{\max} = \max\{\|s_1\|, \|s_2\|\}.$$

You must describe:

- your analysis of this problem,
- how you come up with s_1 and s_2,
- the exact decoding algorithm used,
- how you know that the decoding algorithm is reliable, *i.e.*, the receiver's guess of which signal was sent is always correct.

(b) The `file opt_bin_data.m` contains the matrix A and the scalar $V_{\max}$. Using your findings from part 1, determine the optimal signal constellation.

15.11 *Some input optimization problems.* In this problem we consider the system $x(t+1) = Ax(t) + Bu(t)$, with

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad x(0) = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 1 \end{bmatrix}.$$

(a) *Least-norm input to steer state to zero in minimum time.* Find the minimum T, $T_{\min}$, such that $x(T) = 0$ is possible. Among all $(u(0), u(1), \ldots u(T_{\min}-1))$ that steer $x(0)$ to $x(T_{\min}) = 0$, find the one of minimum norm, *i.e.*, the one that minimizes

$$J_{T_{\min}} = \|u(0)\|^2 + \cdots + \|u(T_{\min}-1)\|^2.$$

Give the minimum value of $J_{T_{\min}}$ achieved.

(b) *Least-norm input to achieve $\|x(10)\| \leq 0.1$.* In lecture we worked out the least-norm input that drives the state exactly to zero at $t = 10$. Suppose instead we only require the state to be *small* at $t = 10$, for example, $\|x(10)\| \leq 0.1$. Find $u(0), u(1), \ldots, u(9)$ that minimize

$$J_9 = \|u(0)\|^2 + \cdots + \|u(9)\|^2$$

subject to the condition $\|x(10)\| \leq 0.1$. Give the value of J_9 achieved by your input.

15.12 *Determining the number of signal sources.* The signal transmitted by n sources is measured at m receivers. The signal transmitted by each of the sources at sampling period k, for $k = 1, \ldots, p$, is denoted by an n-vector $x(k) \in \mathbf{R}^n$. The gain from the j-th source to the i-th receiver is denoted by $a_{ij} \in \mathbf{R}$. The signal measured at the receivers is then

$$y(k) = A\,x(k) + v(k), \quad k = 1, \ldots, p,$$

where $v(k) \in \mathbf{R}^m$ is a vector of sensor noises, and $A \in \mathbf{R}^{m \times n}$ is the matrix of source to receiver gains. However, we do not know the gains a_{ij}, nor the transmitted signal $x(k)$, nor even the number of sources present n. We only have the following additional *a priori* information:

- We expect the number of sources to be less than the number of receivers (*i.e.*, $n < m$, so that A is skinny);
- A is full-rank and well-conditioned;
- All sources have roughly the same average power, the signal $x(k)$ is unpredictable, and the source signals are unrelated to each other; Hence, given enough samples (*i.e.*, p large) the vectors $x(k)$ will 'point in all directions';
- The sensor noise $v(k)$ is small relative to the received signal $A\,x(k)$.

Here's the question:

(a) You are given a large number of vectors of sensor measurements $y(k) \in \mathbf{R}^m$, $k = 1, \ldots, p$. How would you estimate the number of sources, n? Be sure to clearly describe your proposed method for determining n, and to explain when and why it works.

(b) Try your method on the signals given in the file `nsources.m`. Running this script will define the variables:

- `m`, the number of receivers;
- `p`, the number of signal samples;
- `Y`, the receiver sensor measurements, an array of size `m` by `p` (the k-th column of `Y` is $y(k)$.)

What can you say about the number of signal sources present?

Note: Our problem description and assumptions are not precise. An important part of this problem is to explain your method, and clarify the assumptions.

15.13 *The EE263 search engine.* In this problem we examine how linear algebra and low-rank approximations can be used to find matches to a search query in a set of documents. Let's assume we have four documents: **A**, **B**, **C**, and **D**. We want to search these documents for three terms: *piano*, *violin*, and *drum*. We know that:
in **A**, the word *piano* appears 4 times, *violin* 3 times, and *drum* 1 time;
in **B**, the word *piano* appears 6 times, *violin* 1 time, and *drum* 0 times;
in **C**, the word *piano* appears 7 time, *violin* 4 times, and *drum* 39 times; and
in **D**, the word *piano* appears 0 times, *violin* 0 times, and *drum* 5 times.
We can tabulate this as follows:

	A	**B**	**C**	**D**
piano	4	6	7	0
violin	3	1	4	0
drum	1	0	39	5

This information is used to form a *term-by-document* matrix A, where A_{ij} specifies the frequency of the ith term in the jth document, *i.e.*,

$$A = \begin{bmatrix} 4 & 6 & 7 & 0 \\ 3 & 1 & 4 & 0 \\ 1 & 0 & 39 & 5 \end{bmatrix}.$$

Now let q be a *query vector*, with a non-zero entry for each term. The query vector expresses a criterion by which to select a document. Typically, q will have 1 in the entries corresponding to the words we want to search for, and 0 in all other entries (but other weighting schemes are possible.) A simple measure of how relevant document j is to the query is given by the inner product of the jth column of A with q:

$$a_j^T q.$$

However, this criterion is biased towards large documents. For instance, a query for *piano* ($q = [1\ 0\ 0]^T$) by this criterion would return document **C** as most relevant, even though document **B** (and even **A**) is probably much more relevant. For this reason, we use the inner product normalized by the norm of the vectors,

$$\frac{a_j^T q}{\|a_j\| \, \|q\|}.$$

Note that our criterion for measuring how well a document matches the query is now the cosine of the angle between the document and query vectors. Since all entries are non-negative, the cosine is in $[0, 1]$ (and the angle is in $[-\pi/2, \pi/2]$.) Define $\tilde{A}$ and $\tilde{q}$ as normalized versions of A and q (A is normalized column-wise, *i.e.*, each column is divided by its norm.) Then,

$$c = \tilde{A}^T \tilde{q}$$

is a column vector that gives a measure of the relevance of each document to the query. And now, the question. In the file `term_by_doc.m` you are given m search terms, n documents, and the corresponding term-by-document matrix $A \in \mathbf{R}^{m \times n}$. (They were obtained randomly from Stanford's *Portfolio* collection of internal documents.) The variables `term` and `document` are lists of strings. The string `term{i}` contains the ith word. Each document is specified by its URL, *i.e.*, if you point your web browser to the URL specified by the string `document{j}` you can inspect the contents of the jth document. The matrix entry `A(i,j)` specifies how many times term i appears in document j.

(a) Compute $\tilde{A}$, the normalized term-by-document matrix. Compute and plot the singular values of $\tilde{A}$.

(b) Perform a query for the word *students* ($i = 53$) on $\tilde{A}$. What are the 5 top results?

(c) We will now consider low-rank approximations of $\tilde{A}$, that is

$$\hat{A}_r = \min_{\hat{A},\, \mathbf{rank}(\hat{A}) \leq r} \|\tilde{A} - \hat{A}\|.$$

Compute $\hat{A}_{32}$, $\hat{A}_{16}$, $\hat{A}_8$, and $\hat{A}_4$. Perform a query for the word *students* on these matrices. Comment on the results.

(d) Are there advantages of using low-rank approximations over using the full-rank matrix? (You can assume that a very large number of searches will be performed before the term-by-document matrix is updated.)

Note: Variations and extensions of this idea are actually used in commercial search engines (although the details are closely guarded secrets ...) Issues in real search engines include the fact that m and n are enormous and change with time. These methods are very interesting because they can recover documents that don't include the term searched for. For example, a search for *automobile* could retrieve a document with no mention of *automobile*, but many references to cars (can you give a justification for this?) For this reason, this approach is sometimes called *latent semantic indexing. Matlab hints:* You may find the command `sort` useful. It sorts a vector in *ascending* order, and can also return a vector with the original indexes of the sorted elements. Here's a sample code that sorts the vector c in *descending* order, and prints the URL of the top document and its corresponding c_j.

```
[c,j]=sort(-c);
c=-c;
disp(document{j(1)})
disp(c(1))
```

15.14 *Condition number and angles between columns.* Suppose $A \in \mathbf{R}^{n\times n}$ has columns $a_1, \dots, a_n \in \mathbf{R}^n$, each of which has unit norm:

$$A = [a_1\ a_2\ \cdots\ a_n], \qquad \|a_i\| = 1, \quad i = 1, \dots, n.$$

Suppose that two of the columns have an angle less than 10° between them, *i.e.*, $a_k^T a_l \geq \cos 10^\circ$. Show that $\kappa(A) \geq 10$, where κ denotes the condition number. (If A is singular, we take $\kappa(A) = \infty$, and so $\kappa(A) \geq 10$ holds.) *Interpretation:* If the columns were orthogonal, *i.e.*, $\angle(a_i, a_j) = 90^\circ$ for $i \neq j$, $i, j = 1, \dots, n$, then A would be an orthogonal matrix, and therefore its condition number would be one (which is the smallest a condition number can be). At the other extreme, if two columns are the same (*i.e.*, have zero angle between them), the matrix is singular, and the condition number is infinite. Intuition suggests that if some pair of columns has a *small* angle, such as 10°, then the condition number must be large. (Although in many applications, a condition number of 10 is not considered large.)

15.15 *Analysis and optimization of a communication network.* A communication network is modeled as a set of m directed links connecting nodes. There are n *routes* in the network. A route is a path, along one or more links in the network, from a *source node* to a *destination node.* In this problem, the routes are fixed, and are described by an $m \times n$ *route-link matrix* A, defined as

$$A_{ij} = \begin{cases} 1 & \text{route } j \text{ passes through link } i \\ 0 & \text{otherwise.} \end{cases}$$

Over each route we have a nonnegative *flow*, measured in (say) bits per second. We denote the flow along route j as f_j, and we call $f \in \mathbf{R}^n$ the *flow vector.* The *traffic* on a link i, denoted t_i, is the sum of the flows on all routes passing through link i. The vector $t \in \mathbf{R}^m$ is called the *traffic vector.* handle. We're

Each link has an associated nonnegative *delay*, measured in (say) seconds. We denote the delay for link i as d_i, and refer to $d \in \mathbf{R}^m$ as the *link delay vector.* The *latency* on a route j, denoted l_j, is the sum of the delays along each link constituting the route, *i.e.*, the time it takes for bits entering the source to emerge at the destination. The vector $l \in \mathbf{R}^n$ is the *route latency vector.*

The total number of bits in the network at an instant in time is given by $B = f^T l = t^T d$.

(a) *Worst-case flows and delays.* Suppose the flows and link delays satisfy

$$(1/n)\sum_{j=1}^{n} f_j^2 \le F^2, \qquad (1/m)\sum_{i=1}^{m} d_i^2 \le D^2,$$

where F and D are given. What is the maximum possible number of bits in the network? What values of f and d achieve this maximum value? (For this problem you can ignore the constraint that the flows and delays must be nonnegative. It turns out, however, that the worst-case flows and delays can always be chosen to be nonnegative.)

(b) *Utility maximization.* per unit time, For a flow f_j, the network operator derives income at a rate $p_j f_j$, where p_j is the price per unit flow on route j. The network operator's total rate of income is thus $\sum_{j=1}^{n} p_j f_j$. (The route prices are known and positive.)

The network operator is charged at a rate $c_i t_i$ for having traffic t_i on link i, where c_i is the cost per unit of traffic on link i. The total charge rate for link traffic is $\sum_{i=1}^{m} t_i c_i$. (The link costs are known and positive.) The net income rate (or utility) to the network operator is therefore

$$U^{\text{net}} = \sum_{j=1}^{n} p_j f_j - \sum_{i=1}^{m} c_i t_i.$$

Find the flow vector f that maximizes the operator's net income rate, subject to the constraint that each f_j is between 0 and $F^{\max}$, where $F^{\max}$ is a given positive maximum flow value.

15.16 *A heuristic for MAXCUT.* Consider a graph with n nodes and m edges, with the nodes labeled $1, \ldots, n$ and the edges labeled $1, \ldots, m$. We partition the nodes into two groups, B and C, *i.e.*, $B \cap C = \emptyset$, $B \cup C = \{1, \ldots, n\}$. We define the number of *cuts* associated with this partition as the number of edges between pairs of nodes when one of the nodes is in B and the other is in C. A famous problem, called the MAXCUT problem, involves choosing a partition (*i.e.*, B and C) that maximizes the number of cuts for a given graph. For any partition, the number of cuts can be no more than m. If the number of cuts is m, nodes in group B connect only to nodes in group C and the graph is bipartite.

The MAXCUT problem has many applications. We describe one here, although you do not need it to solve this problem. Suppose we have a communication system that operates with a two-phase clock. During periods $t = 0, 2, 4, \ldots$, each node in group B transmits data to nodes in group C that it is connected to; during periods $t = 1, 3, 5, \ldots$, each node in group C transmits to the nodes in group B that it is connected to. The number of cuts, then, is exactly the number of successful transmissions that can occur in a two-period cycle. The MAXCUT problem is to assign nodes to the two groups so as to maximize the overall efficiency of communication.

It turns out that the MAXCUT problem is hard to solve exactly, at least if we don't want to resort to an exhaustive search over all, or most of, the 2^{n-1} possible partitions. In this problem we explore a sophisticated heuristic method for finding a good (if not the best) partition in a way that scales to large graphs.

We will encode the partition as a vector $x \in \mathbf{R}^n$, with $x_i \in \{-1, 1\}$. The associated partition has $x_i = 1$ for $i \in B$ and $x_i = -1$ for $i \in C$. We describe the graph by its node adjacency matrix $A \in \mathbf{R}^{n \times n}$ C), with

$$A_{ij} = \begin{cases} 1 & \text{there is an edge between node } i \text{ and node } j \\ 0 & \text{otherwise} \end{cases}$$

Note that A is symmetric and $A_{ii} = 0$ (since we do not have self-loops in our graph).

(a) Find a symmetric matrix P, with $P_{ii} = 0$ for $i = 1, \ldots, n$, and a constant d, for which $x^T P x + d$ is the number of cuts encoded by any partitioning vector x. Explain how to calculate P and d from A. Of course, P and d cannot depend on x.

The MAXCUT problem can now be stated as the optimization problem

$$\begin{array}{ll} \text{maximize} & x^T P x + d \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n, \end{array}$$

with variable $x \in \mathbf{R}^n$.

(b) A famous heuristic for approximately solving MAXCUT is to replace the n constraints $x_i^2 = 1$, $i = 1, \ldots, n$, with a single constraint $\sum_{i=1}^n x_i^2 = n$, creating the so-called *relaxed* problem

$$\begin{array}{ll} \text{maximize} & x^T P x + d \\ \text{subject to} & \sum_{i=1}^n x_i^2 = n. \end{array}$$

Explain how to solve this relaxed problem (even if you could not solve part (a)).

Let $x^\star$ be a solution to the relaxed problem. We generate our candidate partition with $x_i = \mathbf{sign}(x_i^\star)$. (This means that $x_i = 1$ if $x_i^\star \geq 0$, and $x_i = -1$ if $x_i^\star < 0$.) really

Remark: We can give a geometric interpretation of the relaxed problem, which will also explain why it's called relaxed. The constraints in the problem in part (a), that $x_i^2 = 1$, require x to lie on the vertices of the unit hypercube. In the relaxed problem, the constraint set is the unit ball of unit radius. Because this constraint set is larger than the original constraint set (*i.e.*, it includes it), we say the constraints have been relaxed.

(c) Run the MAXCUT heuristic described in part (b) on the data given in `mc_data.m`. How many cuts does your partition yield?

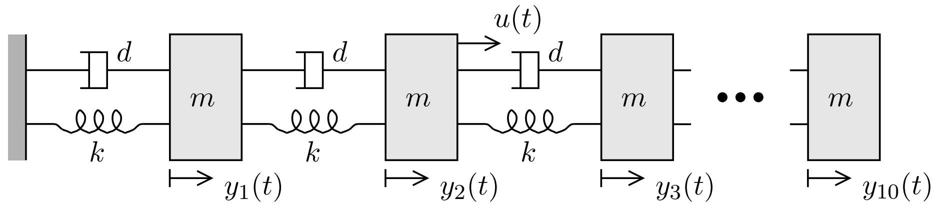
A simple alternative to MAXCUT is to generate a large number of random partitions, using the random partition that maximizes the number of cuts as an approximate solution. Carry out this method with 1000 random partitions generated by `x = sign(rand(n,1)-0.5)`. What is the largest number of cuts obtained by these random partitions?

Note: There are many other heuristics for approximately solving the MAXCUT problem. However, we are not interested in them. In particular, please do not submit any other method for approximately solving MAXCUT.

Lecture 16 – Controllability and state transfer

16.1 This problem has two parts that are mostly independent.

(a) *Actuator placement (10 masses).*



Ten masses are connected in series by springs and light dampers, as shown in the figure above. The mass positions (deviation from rest) are denoted by $y_1, \ldots, y_{10}$. The masses, spring constants, and damping constants are all identical and given by

$$m = 1, \quad k = 1, \quad d = 0.01.$$

An actuator is used to apply a force $u(t)$ to one of the masses. In the figure, the actuator is shown located on the second mass from the left, but it could also have been placed in any of the other nine masses. Use state $x = [y^T \ \dot{y}^T]^T$.

i. For which of the ten possible actuator placements is the system controllable?

ii. You are given the following design specification: any state should be reachable without the need for very large actuation forces. Where would you place the actuator? (Since the design specification is somewhat vague, you should clearly explain and justify your decision.)

Note: To avoid error propagation in solutions, use the Matlab script `spring_series.m`, available at the course web site, which constructs the dynamics and input matrices A and B.

(b) *Optimal control (4 masses).* Consider now a system with the same characteristics, but with only four masses. Four unit masses are connected in series by springs and light dampers (with $k = 1$, and $d = 0.01$.) A force actuator is placed on the *third* mass from the left. As before, use state $x = [y^T \ \dot{y}^T]^T$.

i. Is the system controllable?

ii. You are given the initial state of the system, $x(0) = e_8 = [0 \ \cdots \ 0 \ 1]^T$, and asked to drive the state to as close to zero as possible at time $t_\mathrm{f} = 20$ (*i.e.*, a velocity disturbance in the fourth mass is to be attenuated as much as possible in 20 seconds.) In other words, you are to choose an input $u(t)$, $t \in [0, t_\mathrm{f}]$, that minimizes $\|x(t_\mathrm{f})\|^2$. Furthermore, from among all inputs that achieve the minimum $\|x(t_f)\|^2$, we want the smallest one, *i.e.*, the one for which the energy

$$\mathcal{E}_u = \int_0^{t_\mathrm{f}} u(t)^2 \, dt$$

is minimized. Your answer should include (i) a plot of the minimizing input $u_{\mathrm{opt}}(t)$; (ii) the corresponding energy $\mathcal{E}_{u,\mathrm{min}}$; and (iii) the resulting $\|x(t_f)\|^2$. You must explain and justify how you obtained your solution. *Notes:*

- We will be happy with an approximate solution (by using, say, an input that is piece-wise constant in small intervals.) You may want to discretize the system, in which case we suggest you use 100 discretization intervals (or more.)
- You may (or may not) want to use the result

$$A \int_0^h e^{At} B \, dt = \left(e^{Ah} - I \right) B.$$

16.2 *Horizon selection.* Consider the (scalar input) system

$$x(t+1) = \begin{bmatrix} 0 & 0 & 0.8 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(t), \qquad x(0) = 0.$$

For $N \geq 3$ let $E_N(z)$ denote the minimum input energy, *i.e.*, the minimum value of

$$u(0)^2 + \cdots + u(N-1)^2,$$

required to reach $x(N) = z$. Let $E_\infty(z)$ denote the minimum energy required to reach the state $x(N) = z$, without fixing the final time N, *i.e.*, $E_\infty(z) = \lim_{N \to \infty} E_N(z)$. Find the minimum value of N such that $E_N(z) \leq 1.1 E_\infty(z)$ for all z. (This is the shortest horizon that requires no more than 10% more control energy than infinite horizon control, for any final state). *Hint:* the Matlab command `P=dlyap(A,W)` computes the solution of the Lyapunov equation $APA^T + W = P$.

16.3 *Minimum energy required to steer the state to zero.* Consider a controllable discrete-time system $x(t+1) = Ax(t) + Bu(t)$, $x(0) = x_0$. Let $E(x_0)$ denote the minimum energy required to drive the state to zero, *i.e.*

$$E(x_0) = \min \left\{ \sum_{\tau=0}^{t-1} \|u(\tau)\|^2 \mid x(t) = 0 \right\}.$$

An engineer argues as follows:

> This problem is like the minimum energy reachability problem, but 'turned backwards in time' since here we steer the state from a given state to zero, and in the reachability problem we steer the state from zero to a given state. The system $z(t+1) = A^{-1}z(t) - A^{-1}Bv(t)$ is the same as the given one, except time is running backwards. Therefore $E(x_0)$ is the same as the minimum energy required for z to reach x_0 (a formula for which can be found in the lecture notes).

Either justify or refute the engineer's statement. You can assume that A is invertible.

16.4 *Minimum energy inputs with coasting.* We consider the controllable system $\dot{x} = Ax + Bu$, $x(0) = 0$, where $A \in \mathbf{R}^{n \times n}$ and $B \in \mathbf{R}^{n \times m}$. You are to determine an input u that results in $x(t_\mathrm{f}) = x_\mathrm{des}$, where t_f and x_des are given. You are also given t_a, where $0 < t_\mathrm{a} \leq t_\mathrm{f}$, and have the constraint that $u(t) = 0$ for $t > t_\mathrm{a}$. Roughly speaking, you are allowed to apply a (nonzero) input u during the 'controlled portion' of the trajectory, *i.e.*, from $t = 0$ until

$t = t_{\rm a}$; from $t = t_{\rm a}$ until the final time $t_{\rm f}$, the system 'coasts' or 'drifts' with $u(t) = 0$. Among all u that satisfy these specifications, $u_{\rm ln}$ will denote the one that minimizes the 'energy'

$$\int_0^{t_{\rm a}} \|u(t)\|^2 \, dt.$$

(a) Give an explicit formula for $u_{\rm ln}(t)$.

(b) Now suppose that $t_{\rm a}$ is increased (but still less than $t_{\rm f}$). An engineer asserts that the minimum energy required will decrease. Another engineer disagrees, pointing out that the final time has not changed. Who is right? Justify your answer. (It is possible that neither is right.)

(c) *Matlab exercise.* Consider the mechanical system on page 11-9 of the notes. Let $x_{\rm des} = [1\ 0\ -1\ 0\ 0\ 0]^T$ and $t_{\rm f} = 6$. Plot the minimum energy required as a function of $t_{\rm a}$, for $0 < t_{\rm a} < t_{\rm f}$. You can use a simple method to numerically approximate any integrals you encounter. You must explain what you are doing; just submitting some code and a plot is not enough.

16.5 *Some True/False questions.* By 'True', of course, we mean that the statement holds for all values of the matrices, vectors, dimensions, etc., mentioned in the statement. 'False' means that the statement fails to hold in at least one case.

(a) Suppose $A \in \mathbf{R}^{n\times n}$ and $p(s) = s^n + a_1 s^{n-1} + \cdots + a_n$ is polynomial of degree n, with leading coefficient one, that satisfies $p(A) = 0$. Then p is the characteristic polynomial of A.

(b) Suppose $x : \mathbf{R}_+ \to \mathbf{R}^n$ is a trajectory of the linear dynamical system $\dot{x} = Ax$, which is stable. Then for any $t \geq 0$, we have $\|x(t)\| \leq \|x(0)\|$.

(c) Let $A \in \mathbf{R}^{p\times q}$ and let $a_i \in \mathbf{R}^p$ denote the ith column of A. Then we have

$$\|A\| \geq \max_{i=1,\ldots,q} \|a_i\|.$$

(d) Suppose the two linear dynamical systems $\dot{x} = Fx$ and $\dot{z} = Gz$, where $F,\ G \in \mathbf{R}^{n\times n}$, are both stable. Then the linear dynamical system $\dot{w} = (F+G)w$ is stable.

(e) Suppose P and Q are symmetric $n \times n$ matrices, and let $\{v_1, v_2, \ldots, v_n\}$ be a basis for $\mathbf{R}^n$. Then if we have $v_i^T P v_i \geq v_i^T Q v_i$ for $i = 1, \ldots, n$, we must have $P \geq Q$.

(f) Let $A \in \mathbf{R}^{n\times n}$, and suppose $v \in \mathbf{R}^n$, $v \neq 0$, satisfies $v^T A = \lambda v^T$, where $\lambda \in \mathbf{R}$. Let $x : \mathbf{R}_+ \to \mathbf{R}^n$ be any trajectory of the linear dynamical system $\dot{x} = Ax$. Then at least one of the following statements hold:

- $v^T x(t) \geq v^T x(0)$ for all $t \geq 0$
- $v^T x(t) \leq v^T x(0)$ for all $t \geq 0$

(g) Suppose $A \in \mathbf{R}^{p\times q}$ is fat (*i.e.*, $p \leq q$) and full rank, and $B \in \mathbf{R}^{q\times r}$ is skinny (*i.e.*, $q \geq r$) and full rank. Then AB is full rank.

(h) Suppose $A \in \mathbf{R}^{n\times n}$ has all eigenvalues equal to zero, and the nullspace of A is the same as the nullspace of A^2. Then $A = 0$.

(i) Consider the discrete-time linear dynamical system $x(t+1) = Ax(t) + Bu(t)$, where $A \in \mathbf{R}^{n \times n}$. Suppose there is an input that steers the state from a particular initial state x_{init} at time $t = 0$ to a particular final state x_{final} at time $t = T$, where $T > n$. Then there is an input that steers the state from x_{init} at time $t = 0$ to x_{final} at time $t = n$.

Lecture 17 – Observability and state estimation

17.1 *Sensor selection and observer design.* Consider the system $\dot{x} = Ax$, $y = Cx$, with

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

(This problem concerns observer design so we've simplified things by not even including an input.) (The matrix A is the same as in problem 11, just to save you typing; there is no other connection between the problems.) We consider observers that (exactly and instantaneously) reconstruct the state from the output and its derivatives. Such observers have the form

$$x(t) = F_0 y(t) + F_1 \frac{dy}{dt}(t) + \cdots + F_k \frac{d^k y}{dt^k}(t),$$

where $F_0, \ldots, F_k$ are matrices that specify the observer. (Of course we require this formula to hold for any trajectory of the system and any t, *i.e.*, the observer has to work!) Consider an observer defined by $F_0, \ldots, F_k$. We say the *degree* of the observer is the largest j such that $F_j \neq 0$. The degree gives the highest derivative of y used to reconstruct the state. If the ith columns of $F_0, \ldots, F_k$ are all zero, then the observer doesn't use the ith sensor signal $y_i(t)$ to reconstruct the state. We say the observer *uses* or *requires* the sensor i if at least one of the ith columns of $F_0, \ldots, F_k$ is nonzero.

(a) What is the minimum number of sensors required for such an observer? List all combinations (*i.e.*, sets) of sensors, of this minimum number, for which there is an observer using only these sensors.

(b) What is the minimum degree observer? List all combinations of sensors for which an observer of this minimum degree can be found.