

EE263 final exam

If you received this document via email, print out this page, sign it, and submit it with your completed final exam. You may scan and email your final if you like, but we prefer that you turn it in to the dropbox in Packard if possible.

Instructions

- You will need access to a computer with Matlab.
- You may use the lecture slides, any notes you took yourself, and any material posted on the course website (including homework solutions).
- You may also use supplementary material such as textbooks, wikipedia (at your own risk!), etc. but there shouldn't be any need to consult such outside sources.
- The exam is graded out of 70 points. Point values for problems are indicated at the start of each problem. Sub-parts are labeled (a), (b), etc. and are worth 3 points each for the first problem, and 5 points each for every other problem.
- Turn in all your work, and be sure to attempt every problem. Submit all Matlab code that you write.
- Be neat, and be concise. We will take points off if we can't read your solutions or if they are unnecessarily messy and/or verbose.
- There are no hints available, so don't ask for any. If you need clarification on one of the problems, please send your questions to: ee263-sum1011-staff@lists.stanford.edu
- You must turn in your completed final exam to the grader no later than 24 hours after you picked it up or received it via email.
- Some of the problems require you to download supplementary Matlab files. You can find all files related to the final exam here: www.stanford.edu/class/ee263s/final_data/
- You are not permitted to work in groups, or discuss exam problems with anybody. You must work alone and turn in your own work. By signing below, you are asserting that you have read and understood the Stanford Honor Code, and promise to uphold it.

Signature

Date/Time received

Name (printed)

Date/Time submitted

1. *Short Problems.* [15 points] Suppose $A \in \mathbf{R}^{m \times n}$, $B \in \mathbf{R}^{n \times m}$, and $Q \in \mathbf{R}^{n \times n}$. Provide a short proof for each of the following problems.

(a) Show that $(I - AB)^{-1} = I + A(I - BA)^{-1}B$. You may assume that both inverses exist.

Solution. Expand the product:

$$\begin{aligned} (I - AB) [I + A(I - BA)^{-1}B] &= I - AB + (I - AB)A(I - BA)^{-1}B \\ &= I - AB + (A - ABA)(I - BA)^{-1}B \\ &= I - AB + A(I - BA)(I - BA)^{-1}B \\ &= I - AB + AB \\ &= 0 \end{aligned}$$

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

(b) Suppose $\mathbf{Rank}(A) = 1$. Show that $A = uv^T$ for some $u \in \mathbf{R}^m$ and $v \in \mathbf{R}^n$.

Solution. The rank is defined as the dimension of the range. So if $\text{range}(A)$ has dimension 1, that means that every column of A is a multiple of a_1 , the first column of A . In particular: $a_2 = v_2 a_1, a_3 = v_3 a_1, \dots, a_n = v_n a_1$. Therefore:

$$A = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix} = \begin{bmatrix} a_1 & v_2 a_1 & \dots & v_n a_1 \end{bmatrix} = a_1 \begin{bmatrix} 1 & v_2 & \dots & v_n \end{bmatrix}$$

So let $u = a_1$ and $v = \begin{bmatrix} 1 & v_2 & \dots & v_n \end{bmatrix}^T$. and the result follows. Note that we can replace u by αu and v by $\alpha^{-1}v$ for any $\alpha \neq 0$ and we obtain another valid factorization.

Alternate Solution. Let $A = U\Sigma V^T$ be the SVD of A . Since $r = 1$, we have $U \in \mathbf{R}^{m \times 1}$, $V \in \mathbf{R}^{n \times 1}$, and $\Sigma = \sigma_1 \in \mathbf{R}$. A factorization of the desired form can be obtained for example by setting $u = U$ and $v = \sigma_1 V$.

(c) Suppose $Q = Q^T$ and $Q > 0$. Show that Q is invertible, and $Q^{-1} > 0$.

Solution. Since Q is positive definite, all of its eigenvalues λ_i are strictly positive. Because none of the eigenvalues are zero, Q must be invertible. Since Q is symmetric, it has an orthogonal diagonalization: $Q = U\Lambda U^T$. The inverse is simply $Q^{-1} = U\Lambda^{-1}U^T$, and so the eigenvalues of Q^{-1} are λ_i^{-1} , which are again positive, and so $Q^{-1} > 0$.

(d) How would you find vectors y and x , that maximize $y^T Ax$, subject to $\|y\| = 1, \|x\| = 1$? What is the resulting value of $y^T Ax$?

Solution. Note that $y^T Ax$ is a scalar, and we can find an upper-bound:

$$y^T Ax \leq \|y^T Ax\| \leq \|y\| \|A\| \|x\| = \|A\| = \sigma_1$$

Let $A = U\Sigma V^T$. The upper bound σ_1 is attained by setting $y = u_1$ and $x = v_1$.

(e) The *Frobenius norm* of a matrix is defined as $\|A\|_F^2 = \mathbf{Trace}(A^T A)$. Show that

$$\|A\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n A_{ij}^2 = \sum_{i=1}^r \sigma_i^2$$

where $\{\sigma_1, \sigma_2, \dots, \sigma_r\}$ are the singular values of A .

Solution. Write A in terms of its columns: $A = [a_1 \ \dots \ a_n]$. Then compute:

$$\|A\|_F^2 = \mathbf{Trace} \begin{bmatrix} a_1^T a_1 & \dots & a_1^T a_n \\ \vdots & \ddots & \vdots \\ a_n^T a_1 & \dots & a_n^T a_n \end{bmatrix} = \sum_{j=1}^n a_j^T a_j = \sum_{j=1}^n \sum_{i=1}^m A_{ij}^2$$

Write the compact SVD of A : $A = U\Sigma V^T$. Now compute:

$$\|A\|_F^2 = \mathbf{Trace} (A^T A) = \mathbf{Trace} (V\Sigma^2 V^T) = \mathbf{Trace} (V^T V\Sigma^2) = \mathbf{Trace} (\Sigma^2) = \sum_{i=1}^r \sigma_i^2$$

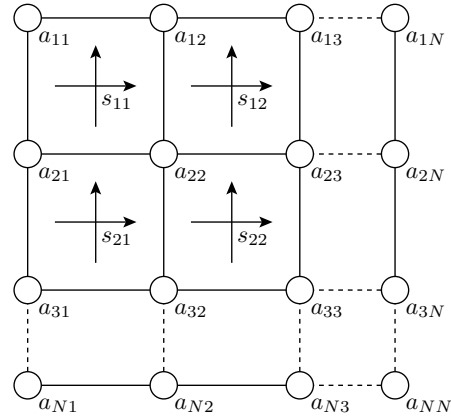
where we used the property of the trace: $\mathbf{Trace}(AB) = \mathbf{Trace}(BA)$ in the third step. Alternatively, we can recall the fact that the trace of a matrix is the sum of its eigenvalues. Since the nonzero eigenvalues of $A^T A$ are $\{\sigma_1^2, \dots, \sigma_r^2\}$, the result follows directly.

2. *Adaptive Optics*. [15 points] Ground-based telescopes use a technology known as *adaptive optics* to cancel out the distortions of incoming light caused by the turbulent atmosphere. One way of doing this is by using a guide star. Since we know a priori what the non-distorted star should look like, we can use it as a reference. Any differences measured must be caused by atmospheric distortion, and can be subtracted out. This enables us to cancel distortion in close proximity to bright stars, for use in applications such a planet-finding.

As light enters the telescope, it has a distribution of phase offsets ϕ across the face of the aperture, and we want to estimate this function. Our measurement device is an array of sensors arranged in a grid (see figure on the right). The grid is $N \times N$, and the sensors s_{ij} are located at the centers of the faces. Each sensor records a horizontal gradient and a vertical gradient, each corrupted by noise. For example, the sensor s_{12} will measure

$$y_{12} = \frac{1}{2} \begin{bmatrix} \phi(a_{13}) - \phi(a_{12}) + \phi(a_{23}) - \phi(a_{22}) \\ \phi(a_{13}) + \phi(a_{12}) - \phi(a_{23}) - \phi(a_{22}) \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

where $x_{ij} = \phi(a_{ij})$ is the phase offset at location (i, j) . Our goal is to estimate the x_{ij} given the measurements y_{ij} .



If the phase offsets are stored in a matrix $X \in \mathbf{R}^{N \times N}$, we can represent X as a vector $x \in \mathbf{R}^{N^2 \times 1}$ by stacking the columns of X . Then, our measurement equation becomes:

$$y = Gx + v$$

where $G \in \mathbf{R}^{2(N-1)^2 \times N^2}$ is called the *influence matrix*. Note that you can go back and forth between these representations by using the `reshape` command in Matlab:

```
x = reshape(X,N^2,1); % convert matrix to vector
X = reshape(x,N,N); % convert vector to matrix
```

Now, the problem:

- (a) Consider the case $N = 17$. The astronomer in charge tells you: “there are two undetectable modes: the *piston mode* and the *checkerboard mode*”. Explain what this means. Specifically:

- i. What is an *undetectable mode* as it relates to G and concepts we covered in class?
- ii. Show that there are exactly two such modes, and compute them.
- iii. Explain why you think they are called “piston” and “checkerboard”.

To save you some time, code that computes G has been provided in the file `ao_data.m`.

- (b) The astronomer wants to reconstruct x given the measurement y , and wants the RMS error to be as small as possible. Furthermore, he wants a solution that contains no piston or checkerboard mode, because these modes are not realistic (do not occur in nature).
 - i. Explain how you would do this and justify your answer.
 - ii. Apply your method to the measurement vector `y_quiet` defined in `ao_data.m`, and plot your estimate x_{est} using the code:

```
figure; imagesc( reshape(xest,N,N), [-10 10] );
axis equal; axis off;
```

- (c) Now apply your method to the measurement `y_noisy`, also defined in `ao_data.m`. This is a noisier measurement of the same x from part (b). The astronomer is not pleased – “the estimate isn’t smooth enough!”. To remedy this problem, you decide to remove some of the higher-frequency modes. Consider the modes $W \in \mathbf{R}^{N \times N}$ defined by

$$W_{ij} = f\left(\frac{2\pi p}{N}(i-1)\right) g\left(\frac{2\pi q}{N}(j-1)\right) \quad i, j \in \{1, 2, \dots, N\}$$

where f, g are any combination of sin and cos, and $p, q \in \{6, 7, 8\}$. This defines a family of $2 \times 2 \times 3 \times 3 = 36$ different modes. Explain how you would reconstruct x from the noisy measurements while penalizing these high-frequency modes in addition to removing the piston and checkerboard modes. Plot your estimate x_{est} with and without smoothing.

Solution.

- (a) An undetectable mode is a value of x that produces no measurement. In other words, $Gx = 0$, so x is in the nullspace of G . Using the code provided to compute G , we can then find the nullspace in any number of ways. For example:

```
r = rank(G);
[U,S,V] = svd(G);
H = V(:,end-r+1:end);
```

And the columns of H will be a basis for the nullspace of G . If we examine these modes in the original $N \times N$ space using this code

```
X1 = reshape(H(:,1),N,N)
X2 = reshape(H(:,2),N,N)
```

we see that:

$$X_1 = \begin{bmatrix} 0.0656 & -0.0869 & 0.0656 & -0.0869 & \dots \\ -0.0869 & 0.0656 & -0.0869 & 0.0656 & \dots \\ 0.0656 & -0.0869 & 0.0656 & -0.0869 & \dots \\ -0.0869 & 0.0656 & -0.0869 & 0.0656 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$X_2 = \begin{bmatrix} -0.0864 & -0.0660 & -0.0864 & -0.0660 & \dots \\ -0.0660 & -0.0864 & -0.0660 & -0.0864 & \dots \\ -0.0864 & -0.0660 & -0.0864 & -0.0660 & \dots \\ -0.0660 & -0.0864 & -0.0660 & -0.0864 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

both modes have repeated numbers occurring at every other entry, in a checkerboard pattern. Indeed, any linear combination of X_1 and X_2 is also an undetectable mode. In particular, instead of using X_1 and X_2 , we can use

$$X_{\text{piston}} = \begin{bmatrix} 1 & 1 & 1 & \dots \\ 1 & 1 & 1 & \dots \\ 1 & 1 & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad \text{and} \quad X_{\text{checker}} = \begin{bmatrix} 1 & -1 & 1 & \dots \\ -1 & 1 & -1 & \dots \\ 1 & -1 & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

which explains the names ‘‘piston’’ and ‘‘checkerboard’’. If we return to the definition of G from the problem statement, it is clear that the gradients of either a constant signal or a checkerboard signal will be zero, and thus undetectable.

- (b) A solution that minimizes the RMS error is simply $x_{\text{est}} = G^\dagger y$. Since G is not full-rank, there may be other values of x that also achieve this same minimum. We will now show that choosing $x_{\text{est}} = G^\dagger y$ yields a solution which contains no piston or checkerboard mode. First, take the full SVD of G :

$$G = [U_1 \quad U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}$$

Note that the columns of V_2 are precisely the piston and checkerboard modes we wish to remove. Now notice that:

$$V_2^T x_{\text{est}} = V_2^T G^\dagger y = V_2^T V_1 \Sigma_1^{-1} U_1^T y = 0$$

where we used the fact that the columns of V_2 are orthogonal to the columns of V_1 . Since $V_2^T x_{\text{est}} = 0$, we conclude that x_{est} contains no piston or checkerboard modes. Using the code:

```
xest = pinv(G)*y_quiet;
```

and the provided plotting routine, we obtain the following estimate:

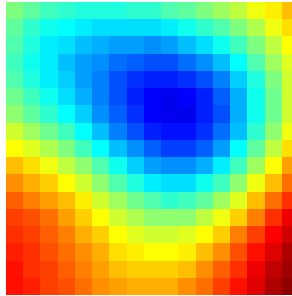


Figure 1: x_{est} , generated from y_{quiet}

- (c) One way to solve this problem is to use a multi-objective least-squares approach. If the modes we want to penalize are written as columns of the matrix: $W = [w_1 \quad \dots \quad w_{36}]$, then we seek a solution that makes both the residual $\|y - Gx\|^2$ and the penalty cost $\|W^T x\|^2$ small. We must also make the undetectable modes small. So our cost function is:

$$J = \|y - Gx\|^2 + \mu_1 \|W^T x\|^2 + \mu_2 \|V_2^T x\|^2$$

As μ_1 and μ_2 tend to infinity, we will approach the x for which $W^T x = 0$, $V_2^T x = 0$, and $\|y - Gx\|^2$ is as small as possible. The optimal x is then:

$$x_{\text{opt}} = \lim_{\mu_1, \mu_2 \rightarrow \infty} (G^T G + \mu_1 W W^T + \mu_2 V_2 V_2^T)^{-1} G^T y$$

Picking a sufficiently large μ_1 and μ_2 yields the smoothed estimate. The following code generates the required modes, and plots the solutions

```

ao_data;
N = 17;
P = diag(ones(N-1,1),1) - diag(ones(N,1)); P = P(1:N-1,:);
Q = diag(ones(N-1,1),1) + diag(ones(N,1)); Q = Q(1:N-1,:);
G = 1/2*[kron(P,Q);kron(-Q,P)];

% generate undetectable modes
V = null(G);

% generate high-frequency modes
W = zeros(N^2,36);
range = [6 7 8];
vals = 2*pi/N*(0:N-1)';
k = 0;
for i = range
    for j = range
        k = k+1; W(:,k) = reshape( cos(i*vals)*cos(j*vals)', N^2, 1 );
        k = k+1; W(:,k) = reshape( cos(i*vals)*sin(j*vals)', N^2, 1 );
        k = k+1; W(:,k) = reshape( sin(i*vals)*cos(j*vals)', N^2, 1 );
        k = k+1; W(:,k) = reshape( sin(i*vals)*sin(j*vals)', N^2, 1 );
    end
end
x_est = (G'*G + V*V')\G'*y_noisy;
mu = 1e6;
x_est_smooth = (G'*G + mu*V*V' + mu*W*W')\G'*y_noisy;

% plot solutions
figure; LIM = [-10 10];
subplot(121); imagesc( reshape(x_est,N,N), LIM); axis equal; axis off;
subplot(122); imagesc( reshape(x_est_smooth,N,N), LIM); axis equal; axis off;

```

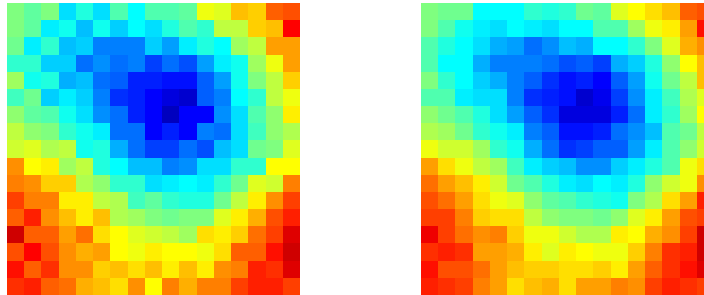


Figure 2: x_{est} (normal and smoothed), generated from y_{noisy}

Alternate Solution. We may also compute the permissible modes for x directly. We know the forbidden modes are the columns of $[W \ V_2]$, as defined above. Thus, we must have:

$$x \in \text{Range} [W \ V_2]^\perp = \text{Range } \bar{U}_2$$

where \bar{U}_2 is found by taking the full SVD:

$$[W \ V_2] = [\bar{U}_1 \ \bar{U}_2] \begin{bmatrix} \bar{\Sigma}_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{V}_1^T \\ \bar{V}_2^T \end{bmatrix}$$

Therefore, $x = \bar{U}_2 z$ for some z . Substituting this into the cost, we can transform our problem:

$$\begin{aligned} \text{minimize} \quad & \|y - Gx\|^2 & \iff & \text{minimize} \quad \|y - G\bar{U}_2 z\|^2 \\ \text{subject to:} \quad & [W \quad V_2]^T x = 0 \end{aligned}$$

The problem on the right is a standard least-squares problem, and since $G\bar{U}_2$ is skinny and full-rank, we have:

$$x_{\text{opt}} = \bar{U}_2 z_{\text{opt}} = \bar{U}_2 (G\bar{U}_2)^\dagger y = \bar{U}_2 (\bar{U}_2^T G^T G \bar{U}_2)^{-1} \bar{U}_2^T G^T y$$

3. *Left-inverses of a matrix.* [20 points] We saw in class that a matrix may have more than one left-inverse. In this problem, we will show two ways of parameterizing *all* left-inverses of a matrix. We will start with a simpler case, and work our way up to the general case.

(a) Suppose $U_1 \in \mathbf{R}^{m \times n}$ is an orthogonal matrix. Show that every left-inverse of U_1 is of the form

$$L = U_1^T + YU_2^T \tag{1}$$

for some choice of Y . Here, U_2 is chosen such that $[U_1 \quad U_2]$ is square and orthogonal.

Hint: begin by arguing that we may write $L = XU_1^T + YU_2^T$ without losing any generality.

(b) We would like to show that any left-inverse of U_1 can also be expressed as

$$L = (U_1^T Q U_1)^{-1} U_1^T Q \tag{2}$$

where Q is some positive-definite matrix. First, show that for any $Q > 0$, you can find a Y such that (1) and (2) describe the *same* left-inverse. Verify that your Y works!

(c) The result of part (b) only partially proves the equivalence of (1) and (2). We must also show that for any Y , we can find a $Q > 0$ such that (1) and (2) describe the same left-inverse. Show that $Q = I + (U_1 + 2U_2 Y^T)(U_1 + 2U_2 Y^T)^T$ is such a Q .

(d) Now we tackle the general case. Suppose $A \in \mathbf{R}^{m \times n}$ is left-invertible. Show that every left-inverse of A can be expressed in two equivalent ways:

$$\begin{aligned} \tilde{L} &= A^\dagger + Z \quad \text{where } Z \text{ satisfies } ZA = 0, \text{ or} \\ \tilde{L} &= (A^T Q A)^{-1} A^T Q \quad \text{where } Q \text{ is positive-definite} \end{aligned}$$

Solution.

(a) Since $U = [U_1 \quad U_2]$ is an orthogonal matrix, we have $UU^T = I$. Thus, $L = LUU^T$. So any L can be written in the form $L = WU^T$, for some W . Now partition W into a block 1×2 matrix so that we can write:

$$L = WU^T = [X \quad Y] \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} = XU_1^T + YU_2^T$$

Now, we use the fact that L is a left-inverse of U_1 . Thus,

$$I = LU_1 = (XU_1^T + YU_2^T)U_1 = X$$

where we made use of the fact that $U_1^T U_1 = I$ and $U_2^T U_1 = 0$ (property of orthogonal vectors). It follows that $L = U_1^T + YU_2^T$ for some Y .

- (b) First, let's make sure that the matrix being inverted in (2) is actually invertible. Note that for any nonzero z , $U_1 z$ is nonzero (because the columns of U_1 are independent). Therefore, $(U_1 z)^T Q (U_1 z) > 0$, because $Q > 0$. So for any nonzero z , $z^T (U_1^T Q U_1) z > 0$, and so $U_1^T Q U_1$ is positive definite, so all its eigenvalues are positive, and it must therefore be invertible. Setting (1) and (2) equal to one another, we have:

$$U_1^T + Y U_2^T = (U_1^T Q U_1)^{-1} U_1^T Q$$

To solve for Y , multiply on the right by U_2 , and we obtain:

$$Y = (U_1^T Q U_1)^{-1} U_1^T Q U_2$$

Now let's check that this choice of Y makes both left-inverses the same:

$$\begin{aligned} U_1^T + Y U_2^T &= U_1^T + (U_1^T Q U_1)^{-1} U_1^T Q U_2 U_2^T \\ &= (U_1^T Q U_1)^{-1} (U_1^T Q U_1 U_1^T + U_1^T Q U_2 U_2^T) \\ &= (U_1^T Q U_1)^{-1} U_1^T Q (U_1 U_1^T + U_2 U_2^T) \\ &= (U_1^T Q U_1)^{-1} U_1^T Q (U U^T) \\ &= (U_1^T Q U_1)^{-1} U_1^T Q \end{aligned}$$

So for any Q , using $Y = (U_1^T Q U_1)^{-1} U_1^T Q U_2$ makes both left-inverses the same.

- (c) First, we will show that the provided Q is positive-definite. It is clearly symmetric, and for any nonzero vector x , we have:

$$\begin{aligned} x^T Q x &= x^T \left[I + (U_1 + 2U_2 Y^T) (U_1 + 2U_2 Y^T)^T \right] x \\ &= \|x\|^2 + \left\| (U_1 + 2U_2 Y^T)^T x \right\|^2 > 0 \end{aligned}$$

which holds because the first norm is positive and the second norm is nonnegative. Now, let's substitute Q into (2) and see what we get. First note that:

$$\begin{aligned} U_1^T Q U_1 &= I + U_1^T (U_1 + 2U_2 Y^T) (U_1 + 2U_2 Y^T)^T U_1^T \\ &= I + (I + 0)(I + 0)^T \\ &= 2I \end{aligned}$$

Therefore,

$$\begin{aligned} (U_1^T Q U_1)^{-1} U_1^T Q &= (2I)^{-1} U_1^T \left[I + (U_1 + 2U_2 Y^T) (U_1 + 2U_2 Y^T)^T \right] \\ &= \frac{1}{2} (U_1^T + U_1^T + 2Y U_2^T) \\ &= U_1^T + Y U_2^T \end{aligned}$$

This matches (1), so this choice of Q is valid!

- (d) Since A is left-invertible, it must be skinny and full-rank. So we may write $A = U_1 R$ where U_1 is a skinny matrix with orthonormal columns and R is invertible (QR factorization).

Finding a left-inverse of A amounts to finding a left-inverse of U_1 . To see why, suppose \tilde{L} is a left-inverse of A . Then: $\tilde{L} A = I \implies \tilde{L} U_1 R = I \implies \tilde{L} U_1 = R^{-1} \implies R \tilde{L} U_1 = I$. Therefore, $R \tilde{L}$ is a left-inverse of U_1 . Similarly, if L is a left-inverse of U_1 , then $R^{-1} L$ is a left-inverse of A . So there is a bijection between inverses of A and inverses of U_1 .

For the first part of the question, use the result from (a). Every left-inverse of U_1 is of the form $U_1^T + YU_2^T$ for some Y . So any left-inverse of A can be written as $\tilde{L} = R^{-1}U_1^T + R^{-1}YU_2^T$. Recall that $A^\dagger = R^{-1}U_1^T$ when A is left-invertible. Furthermore, if we let $Z = R^{-1}YU_2^T$, then $ZA = 0$. Conversely, any matrix of the form $A^\dagger + Z$ is a left-inverse of A because $(A^\dagger + Z)A = I + ZA = I$. For the second part of the question, use (b) and (c). We can write any left-inverse of U_1 as $(U_1^T Q U_1)^{-1} U_1^T Q$ for some $Q > 0$. So any left-inverse of A can be written as

$$\tilde{L} = R^{-1}(U_1^T Q U_1)^{-1} U_1^T Q = R^{-1}(U_1^T Q U_1)^{-1} R^{-T} R^T U_1^T Q = (R^T U_1^T Q U_1 R)^{-1} R^T U_1^T Q$$

Since $A = U_1 R$, the last expression is equal to $(A^T Q A)^{-1} A^T Q$. Conversely, any matrix of the form $(A^T Q A)^{-1} A^T Q$ is a left-inverse of A because $(A^T Q A)^{-1} A^T Q A = I$.

4. *Waypoint Tracking.* [20 points] You are in charge of designing a controller for a small hovercraft. The hovercraft moves around in the plane with the aid of three individually actuated thrusters. You are given a set of waypoints, and the exact times at which the vehicle must reach each waypoint. The objective is to find the smallest sequence of thruster inputs that achieves this goal.

We have a discrete state-space model for the hovercraft dynamics:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned}$$

where:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \frac{1}{100} \begin{bmatrix} 2 & -1 & 2 \\ 4 & -2 & 4 \\ 0 & 2 & 2 \\ 0 & 4 & 4 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$y(t) \in \mathbf{R}^2$ is the hovercraft position at time t , and $u(t) \in \mathbf{R}^3$ are the thrust inputs at time t . The hovercraft starts at rest at the origin ($x(0) = 0$), and our waypoint constraints are:

$$y(20) = \begin{bmatrix} 5 \\ 4 \end{bmatrix}, \quad y(40) = \begin{bmatrix} 10 \\ -1 \end{bmatrix}, \quad y(70) = \begin{bmatrix} 4 \\ -1 \end{bmatrix}, \quad y(100) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

so at time $t = 100$, the hovercraft has returned to its starting point.

- (a) Rewrite the waypoint constraints so that they are in the more useful form: $y_{\text{wp}} = Fu$, where

$$y_{\text{wp}} = \begin{bmatrix} y(20) \\ y(40) \\ y(70) \\ y(100) \end{bmatrix} \quad \text{and} \quad u = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(100) \end{bmatrix}$$

Find an expression for the matrix F in terms of A , B , and C .

- (b) Explain how you would solve this problem in general. Specifically: how would you find u such that $y_{\text{wp}} = Fu$ and the cost $J_1 = \|u\|^2$ is minimized?
- (c) Carry out your solution in Matlab, and plot the optimal trajectory. Label the waypoints so that you can verify that the trajectory does indeed cross them. To save you some time, the file `hovercraft.m` contains the definitions of A , B , and C .

- (d) Suppose you would like to save even more fuel, and are willing to sacrifice some accuracy in exchange. Rather than imposing the constraint $y_{\text{wp}} = Fu$ let's instead try to minimize both

$$J_1 = \|u\|^2 \quad \text{and} \quad J_2 = \|y_{\text{wp}} - Fu\|^2$$

- i. Plot the optimal trade-off curve (J_1 on the x -axis and J_2 on the y -axis).
- ii. Plot a trajectory where both costs have equal weight, again labeling the waypoints.

Solution.

- (a) Express a general output $y(t)$ in terms of the initial state and the previous inputs by expanding:

$$y(t) = CA^t x(0) + \sum_{i=0}^{t-1} CA^{t-i-1} Bu(i) + Du(t)$$

Putting this together for each $t = 0, 1, \dots, 100$, we obtain:

$$y = \mathcal{O}x(0) + \mathcal{T}u \tag{3}$$

where:

$$y = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(100) \end{bmatrix}, \quad u = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(100) \end{bmatrix}, \quad \mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{100} \end{bmatrix}, \quad \mathcal{T} = \begin{bmatrix} D & & & \\ CB & D & & \\ \vdots & \ddots & \ddots & \\ CA^{99}B & \dots & CB & D \end{bmatrix}$$

For this problem, $x(0) = 0$, and $D = 0$. We can extract the required constraint by selecting the appropriate block-rows from the \mathcal{T}_t matrix:

$$y_{\text{wp}} = Fu$$

where:

$$y_{\text{wp}} = \begin{bmatrix} y(20) \\ y(40) \\ y(70) \\ y(100) \end{bmatrix}, \quad F = \begin{bmatrix} CA^{19}B & CA^{18}B & \dots & CB & & & \\ CA^{39}B & CA^{38}B & \dots & \dots & CB & & \\ CA^{69}B & CA^{68}B & \dots & \dots & \dots & CB & \\ CA^{99}B & CA^{98}B & \dots & \dots & \dots & \dots & CB & 0 \end{bmatrix}$$

- (b) We would like to minimize $\|u\|$ subject to the constraint $y_{\text{wp}} = Fu$. This is a standard norm-minimization problem. First, let us verify that F is full-rank. There are many ways to do this. For example, you could compute the rank in Matlab by constructing the full F and using the `rank` command. A simpler way is to notice that due to the staggered nature of the F matrix, it is sufficient to show that CB is full-rank. So we compute:

$$CB = \frac{1}{100} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & -1 & 2 \\ 4 & -2 & 4 \\ 0 & 2 & 2 \\ 0 & 4 & 4 \end{bmatrix} = \frac{1}{100} \begin{bmatrix} 2 & -1 & 2 \\ 0 & 2 & 2 \end{bmatrix}$$

which is full-rank, as required. Thus the solution to our problem is the pseudo-inverse of F :

$$u_{\text{opt}} = F^T (FF^T)^{-1} y_{\text{wp}}$$

(c) Once we have the optimal inputs, we can compute the optimal trajectory by using (3):

$$y_{\text{opt}} = \mathcal{T}u_{\text{opt}} = \mathcal{T}F^T(FF^T)^{-1}y_{\text{wp}}$$

See below for the Matlab code and plots.

(d) This is a multi-objective least-squares problem. The cost function is:

$$J = \underbrace{\|y_{\text{wp}} - Fu\|^2}_{J_2} + \mu \underbrace{\|u\|^2}_{J_1}$$

The solution to this problem is:

$$u_{\text{opt}}(\mu) = (F^T F + \mu I)^{-1} F^T y_{\text{wp}}$$

And once again, the optimal trajectory is found using (3):

$$y_{\text{opt}} = \mathcal{T}u_{\text{opt}} = \mathcal{T}(F^T F + \mu I)^{-1} F^T y_{\text{wp}}$$

Note that if we write $J = \mu J_2 + J_1$ instead, this simply replaces μ with μ^{-1} in the formula above, so it does not affect what the trade-off curve looks like. We are also asked to plot a trajectory where J_1 and J_2 have equal weight. This amounts to setting $\mu = 1$. Here is Matlab code that computes all the optimal input sequences and produces the desired plots.

```
clear all; clc;
vehicle;
n=size(A,1); m=size(B,2); k=size(C,1);
N=101; % time horizon is 101, since indexing starts at 1 in Matlab

% build T one block-row at a time
T = [D zeros(k,(N-1)*m)];
for t = 2:N
    T = [T; [C*A^(t-2)*B T(end-k+1:end,1:end-m)]];
end

% declare waypoints (times and positions)
twp = [ 21  41  71  101 ];
Ywp = [ 5   10   4   0
        4  -1  -1   0 ];

% construct F matrix by extracting appropriate block-rows from T
F=[]; for t=twp; F=[F; T(k*(t-1)+1:k*t,:)]; end

% compute minimum-fuel trajectory
ywp = reshape(Ywp, [], 1);
uopt = F'*((F*F')\ywp);
Yopt = reshape(T*uopt, 2, N);

% compute "equal weight" regularized trajectory
ureg = (F'*F + eye(size(F,2)))\ (F'*ywp);
Yreg = reshape(T*ureg, 2, N);

% plot the trajectories
```

```

set(0,'DefaultTextInterpreter','latex');
figure(1); clf;
subplot(121); hold on; plot( Yopt(1,:), Yopt(2,:), 'b.-' );
title('Optimal minimum-fuel trajectory')
subplot(122); hold on; plot( Yreg(1,:), Yreg(2,:), 'b.-' );
title('Regularized trajectory,  $\mu=1$ ')
for i = [1 2]
    subplot(1,2,i); plot( Ywp(1,:), Ywp(2,:), 'ro', 'MarkerSize', 7 );
    axis equal; axis([-2 12 -4 6]);
    xlabel('$y_1(t)$, $x$-position'); ylabel('$y_2(t)$, $y$-position');
    lh = legend('trajectory','waypoints'); set(lh,'Interpreter','latex');
end;

% compute and plot the trade-off curve
mu_vals = logspace(-5,9,100);
J1 = []; J2 = [];
for mu = mu_vals
    u = (F'*F + mu*eye(size(F,2)))\ (F'*ywp);
    J1 = [J1 norm(u)^2]; J2 = [J2 norm(F*u-ywp)^2];
end;

```

