Lectures on Dynamic Systems and Control

Mohammed Dahleh Munther A. Dahleh George Verghese
Department of Electrical Engineering and Computer Science
Massachuasetts Institute of Technology[1]

[1]©

Chapter 2

Least Squares Estimation

2.1 Introduction

If the criterion used to measure the error $e = y - Ax$ in the case of inconsistent system of equations is the sum of squared magnitudes of the error components, i.e. $e'e$, or equivalently the square root of this, which is the usual Euclidean norm or 2-norm $\|e\|_2$, then the problem is called a *least squares* problem. Formally it can be written as

$$\min_x \|y - Ax\|_2. \tag{2.1}$$

The x that minimizes this criterion is called the least square error estimate, or more simply, the *least squares estimate.* The choice of this criterion and the solution of the problem go back to Legendre (1805) and Gauss (around the same time).

Example 2.1 Suppose we make some measurements y_i of an unknown function $f(t)$ at discrete points t_i, $i = 1, \ldots, N$:

$$y_i = f(t_i) \ , \quad i = 1, \ldots, N.$$

We want to find the function $g(t)$ in the space χ of polynomials of order $m - 1 < N - 1$ that best approximates $f(t)$ at the measured points t_i, where

$$\chi = \left\{ g(t) = \sum_{i=0}^{m-1} \alpha_i t^i, \ \alpha_i \ \text{real} \right\}$$

For any $g(t) \in \chi$, we will have $y_i = g(t_i) + e_i$ for $i = 1, \ldots, N$. Writing this in matrix form for the available data, we have

$$\underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{m-1} \\ \vdots & & & \vdots & \\ 1 & t_N & t_N^2 & \cdots & t_N^{m-1} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{m-1} \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix}}_{e}$$

The problem is to find $\alpha_0, \ldots, \alpha_{m-1}$ such that $e'e = \sum_{i-1}^{N} e_i^2$ is minimized.

2.2 Computing the Estimate

The solution, $\hat{x}$, of Equation 2.1 is characterized by:

$$(y - A\hat{x}) \perp \mathcal{R}(A).$$

All elements in a basis of $\mathcal{R}(A)$ must be orthogonal to $(y - A\hat{x})$. Equivalently this is true for the set of columns of A, $[a_1, \ldots, a_n]$. Thus

$$\begin{aligned} (y - A\hat{x}) \perp \mathcal{R}(A) \quad & \Leftrightarrow a_i'\,(y - A\hat{x}) = 0 \qquad \text{for } i = 1, \ldots, n \\ & \Leftrightarrow A'(y - A\hat{x}) = 0 \\ & \Leftrightarrow A'A\hat{x} = A'y \end{aligned}$$

This system of m equations in the m unknowns of interest is referred to as the **normal equations**. We can solve for the unique $\hat{x}$ iff $A'A$ is invertible. Conditions for this will be derived shortly. In the sequel, we will present the generalization of the above ideas for infinite dimensional vector spaces.

2.3 Preliminary: The Gram Product

Given the array of n_A vectors $A = [a_1 \mid \cdots \mid a_{n_A}]$ and the array of n_B vectors $B = [b_1 \mid \cdots \mid b_{n_B}]$ from a given inner product space, let $\prec A, B \succ$ denote the $n_A \times n_B$ *matrix* whose (i, j)-th element is $< a_i, b_j >$. We shall refer to this object as the *Gram product* (but note that this terminology is not standard!).

If the vector space under consideration is $\mathbf{R}^m$ or $\mathbf{C}^m$, then both A and B are matrices with m rows, but our definition of $\prec A, B \succ$ can actually handle more general A, B. In fact, the vector space can be infinite dimensional, as long as we are only examining finite collections of vectors from this space. For instance, we could use the same notation to treat finite collections of vectors chosen from the infinite-dimensional vector space $\mathcal{L}^2$ of *square* *integrable* functions, i.e. functions $a(t)$ for which $\int_{-\infty}^{\infty} a^2(t)\,dt < \infty$. The inner product in $\mathcal{L}^2$ is $< a(t), b(t) >= \int_{-\infty}^{\infty} a^*(t)b(t)\,dt$. (The space $\mathcal{L}^2$ is an example of an infinite dimensional *Hilbert space*, and most of what we know for finite dimensional spaces — which are also Hilbert spaces! — has natural generalizations to infinite dimensional Hilbert spaces. Many of these generalizations involve introducing notions of topology and measure, so we shall *not* venture too far there. It is worth also mentioning here another important infinite dimensional Hilbert space that is central to the *probabilistic* treatment of least squares estimation: the space of zero-mean *random variables*, with the expected value $E(ab)$ serving as the inner product $< a, b >$.)

For the usual Euclidean inner product in an m-dimensional space, where $< a_i, b_j >= a_i' b_j$, we simply have $\prec A, B \succ = A'B$. For the inner product defined by $< a_i, b_j >= a_i' S\, b_j$ for a positive definite, Hermitian matrix S, we have $\prec A, B \succ = A'SB$.

- Verify that the symmetry and linearity of the inner product imply the same for the Gram product, so $\prec AF, BG + CH \succ = F' \prec A, B \succ G + F' \prec A, C \succ H$, for any constant matrices F, G, H (a constant matrix is a matrix of scalars), with A, B, C denoting arrays whose columns are vectors.

2.4 The Least Squares Estimation Problem

The problem of interest is to find the least square error (LSE) estimate of the parameter vector x that arises in the linear model $y \approx Ax$, where A is an array of n vectors, $A = [a_1, \cdots, a_n]$. Defining the *error* e by

$$e = y - Ax$$

what we want to determine is

$$\widehat{x} = \arg\min_x \|e\| = \arg\min_x \|y - Ax\| \ , \qquad y,\ A \ \text{given}$$

(where "arg min$_x$" should be read as "the value of the argument x that minimizes"). To state this yet another way, note that as x is varied, Ax ranges over the subspace $\mathcal{R}(A)$, so we are looking for the point

$$\widehat{y} = A\widehat{x}$$

in $\mathcal{R}(A)$ that comes closest to y, as measured by whatever norm we are using.

Rather than restricting the norm in the above expression to be the Euclidean 2-norm used in Lecture 1, we shall now actually permit it to be any norm induced by an inner product, so $\|e\| = \sqrt{< e, e >}$. This will allow us to solve the so-called *weighted* least squares problem in a finite dimensional space with no additional work, because error criteria of the form $e'Se$ for positive definite Hermitian S are thereby included. Also, our problem formulation then applies to infinite dimensional spaces that have an inner product defined on them, with the restriction that our model Ax be confined to a finite dimensional subspace. This actually covers the cases of most interest to us; treatment of the more general case involves introducing further topological notions (*closed* subspaces, etc.), and we avoid doing this.

We shall also assume that the vectors a_i, $i = 1, \ldots, n$ in A are *independent.* This assumption is satisfied by any reasonably parametrized model, for otherwise there would be an infinite number of choices of x that attained any achievable value of the error $y - Ax$. If the vectors in A are discovered to be dependent, then a re-parametrization of the model is needed to yield a well-parametrized model with independent vectors in the new A. (A subtler problem — and one that we shall say something more about in the context of ill-conditioning and the singular value decomposition — is that the vectors in A can be *nearly* dependent, causing practical difficulties in numerical estimation of the parameters.)

Gram Matrix Lemma

An important route to verifying the independence of the vectors that make up the columns of A is a lemma that we shall refer to as the *Gram Matrix Lemma.* This states that the vectors in A are independent iff the associated Gram matrix (or *Gramian*) $\prec A, A \succ = [< a_i, a_j >]$ is invertible; all norms are equivalent, as far as this result is concerned — one can pick any norm. As noted above, for the case of the usual Euclidean inner product, $\prec A, A \succ = A'A$. For an inner product of the form $< a_i, a_j > = a_i' S a_j$, where S is Hermitian and positive definite, we have $\prec A, A \succ = A'SA$. The lemma applies to the infinite dimensional setting as well (e.g. $\mathcal{L}^2$), provided we are only considering the independence of a finite subset of vectors.

Proof: If the vectors in A are dependent, there is some nonzero vector η such that $A\eta = \sum_j a_j \eta_j = 0$. But then $\sum_j < a_i, a_j > \eta_j = 0$, by the linearity of the inner product; in matrix form, we can write $\prec A, A \succ \eta = 0$ — so $\prec A, A \succ$ is not invertible.

Conversely, if $\prec A, A \succ$ is not invertible, then $\prec A, A \succ \eta = 0$ for some nonzero η. But then $\eta' \prec A, A \succ \eta = 0$, so by the linearity of inner products $< \sum \eta_i a_i, \sum a_j \eta_j > = 0$, i.e. the norm of the vector $\sum a_j \eta_j = A\eta$ is zero, so the vectors in A are dependent.

2.5 The Projection Theorem and the Least Squares Estimate

The solution to our least squares problem is now given by the *Projection Theorem*, also referred to as the **Orthogonality Principle**, which states that

$$\widehat{e} = (y - A\widehat{x}) \quad \perp \quad \mathcal{R}(A)$$

from which - — as we shall see — $\widehat{x}$ can be determined. In words, the theorem/"principle" states that the point $\widehat{y} = A\widehat{x}$ in the subspace $\mathcal{R}(A)$ that comes closest to y is characterized by the fact that the associated error $\widehat{e} = y - \widehat{y}$ is orthogonal to $\mathcal{R}(A)$, i.e., orthogonal to the space spanned by the vectors in A. This principle was presented and proved in the previous chapter. We repeat the proof here in the context of the above problem.

Proof: We first show that y has a unique decomposition of the form $y = y_1 + y_2$, where $y_1 \in \mathcal{R}(A)$ and $y_2 \in \mathcal{R}^{\perp}(A)$. We can write any $y_1 \in \mathcal{R}(A)$ in the form $y_1 = A\alpha$ for some vector α.

If we want $(y - y_1) \in \mathcal{R}^{\perp}(A)$, we must see if there is an α that satisfies

$$< a_i, (y - A\alpha) >= 0 \ , \quad i = 1, \ldots, n$$

or, using our Gram product notation,

$$\prec A, (y - A\alpha) \succ = 0$$

Rearranging this equation and using the linearity of the Gram product, we get

$$\prec A, A \succ \alpha = \prec A, y \succ$$

which is in the form of the normal equations that we encountered in Lecture 1. Under our assumption that the vectors making up the columns of A are independent, the Gram matrix lemma shows that $\prec A, A \succ$ is invertible, so the unique solution of the preceding equation is

$$\alpha = \prec A, A \succ^{-1} \prec A, y \succ$$

We now have the decomposition that we sought.

To show that the preceding decomposition is unique, let $y = y_{1a} + y_{2a}$ be another such decomposition, with $y_{1a} \in \mathcal{R}(A)$ and $y_{2a} \in \mathcal{R}^{\perp}(A)$. Then

$$y_1 - y_{1a} = y_2 - y_{2a}$$

and the left side is in $\mathcal{R}(A)$ while the right side is in its orthogonal complement. It is easy to show that the only vector common to a subspace and its orthogonal complement is the zero vector, so $y_1 - y_{1a} = 0$ and $y_2 - y_{2a} = 0$, i.e., the decomposition of y is unique.

To proceed, decompose the error $e = y - Ax$ similarly (and uniquely) into the sum of $e_1 \in \mathcal{R}(A)$ and $e_2 \in \mathcal{R}^{\perp}(A)$. Note that

$$\|e\|^2 = \|e_1\|^2 + \|e_2\|^2$$

Now we can rewrite $e = y - Ax$ as

$$e_1 + e_2 = y_1 + y_2 - Ax$$

or

$$e_2 - y_2 = y_1 - e_1 - Ax$$

Since the right side of the above equation lies in $\mathcal{R}(A)$ and the left side lies in $\mathcal{R}^{\perp}(A)$, each side separately must equal 0 — again because this is the only vector common to a subspace and its orthogonal complement. We thus have $e_2 = y_2$, and the choice of x can do nothing to affect e_2. On the other hand, $e_1 = y_1 - Ax = A(\alpha - x)$, and the best we can do as far as minimizing $\|e\|^2$ is to make $e_1 = 0$ by choosing $x = \alpha$, so $\widehat{x} = \alpha$, i.e.,

$$\boxed{\widehat{x} = \prec A, A \succ^{-1} \prec A, y \succ}$$

This solves the least squares estimation problem that we have posed.

The above result, though rather abstractly developed, is immediately applicable to many concrete cases of interest.

- Specializing to the case of $\mathbf{R}^m$ or $\mathbf{C}^m$, and choosing x to minimize the usual Euclidean norm,
$$\|e\|^2 = e'e = \sum_{i=1}^{m} |e_i|^2$$
we have
$$\widehat{x} = (A'A)^{-1}A'y$$
Note that if the columns of A form a mutually orthogonal set (i.e. an orthogonal basis for $\mathcal{R}(A)$), then $A'A$ is diagonal, and its inversion is trivial.

- If instead we choose to minimize $e'Se$ for some positive definite Hermitian S ($\neq I$), we have a **weighted least squares** problem, with solution given by
$$\widehat{x} = (A'SA)^{-1}A'Sy$$
For instance, with a *diagonal* S, the criterion that we are trying to minimize becomes
$$\sum_{i=1}^{m} s_{ii}|e_i|^2$$
where the s_{ii} are all positive. We can thereby preferentially weight those equations in our linear system for which we want a smaller error in the final solution; a larger value of s_{ii} will encourage a smaller e_i.

  Such weighting is important in any practical situation, where different measurements y_i may have been subjected to different levels of noise or uncertainty. One might expect that s_{ii} should be inversely proportional to the noise intensity on the ith equation. In fact, a probabilistic derivation, assuming zero-mean noise on each equation in the system but noise that is uncorrelated across equations, shows that s_{ii} should vary inversely with the *variance* of e_i.

  A full matrix S rather than a diagonal one would make sense if the errors were correlated across measurements. A probabilistic treatment shows that the proper weighting matrix is $S = (E[ee'])^{-1}$, the inverse of the *covariance matrix* of e. In the deterministic setting, one has far less guidance on picking a good S.

- The boxed result also allows us to immediately write down the choice of coefficients x_i that minimizes the integral

$$\int [\, y(t) - a_1(t)x_1 - a_2(t)x_2 - \cdots - a_n(t)x_n \,]^2 \, dt$$

for specified functions $y(t)$ and $a_i(t)$. If, for instance, $y(t)$ is of finite extent (or finite "support") T, and the $a_i(t)$ are sinusoids whose frequencies are integral multiples of $2\pi/T$, then the formulas that we obtain for the x_i are just the familiar Fourier series expressions. A simplification in this example is that the vectors in A are orthogonal, so $\prec A, A \succ$ is diagonal.

2.6 Recursive Least Squares (optional)

What if the data is coming in sequentially? Do we have to recompute everything each time a new data point comes in, or can we write our new, updated estimate in terms of our old estimate?

Consider the model

$$y_i = A_i x + e_i \ , \quad i = 0, 1, \ldots , \tag{2.2}$$

where $y_i \in \mathbf{C}^{m\times 1}$, $A_i \in \mathbf{C}^{m\times n}$, $x \in \mathbf{C}^{n\times 1}$, and $e_i \in \mathbf{C}^{m\times 1}$. The vector e_k represents the mismatch between the measurement y_k and the model for it, $A_k x$, where A_k is known and x is the vector of parameters to be estimated. At each time k, we wish to find

$$\widehat{x}_k = \arg\min_x \left(\sum_{i=1}^{k} (y_i - A_i x)_i' S_i (y_i - A_i x) \right) = \arg\min_x \left(\sum_{i=1}^{k} e_i' S_i e_i \right) \ , \tag{2.3}$$

where $S_i \in \mathbf{C}^{m\times m}$ is a positive definite Hermitian matrix of weights, so that we can vary the importance of the e_i's and components of the e_i's in determining $\widehat{x}_k$.

To compute $\widehat{x}_{k+1}$, let:

$$\overline{y}_{k+1} = \begin{bmatrix} y_0 \\ y_1 \\ \cdot \\ \cdot \\ y_{k+1} \end{bmatrix} ; \qquad \overline{A}_{k+1} = \begin{bmatrix} A_0 \\ A_1 \\ \cdot \\ \cdot \\ A_{k+1} \end{bmatrix} ; \qquad \overline{e}_{k+1} = \begin{bmatrix} e_0 \\ e_1 \\ \cdot \\ \cdot \\ e_{k+1} \end{bmatrix} ;$$

and

$$\overline{S}_{k+1} = \text{diag}\ (S_0\ ,\ S_1\ ,\ \ldots\ ,\ S_{k+1})$$

where S_i is the weighting matrix for e_i.

Our problem is then equivalent to

$$\min(\overline{e}'_{k+1}\overline{S}_{k+1}\overline{e}_{k+1})$$

$$\text{subject to:} \quad \overline{y}_{k+1} = \overline{A}_{k+1}x_{k+1} + \overline{e}_{k+1}$$

The solution can thus be written as

$$(\overline{A}'_{k+1}\overline{S}_{k+1}\overline{A}_{k+1})\widehat{x}_{k+1} = \overline{A}'_{k+1}\overline{S}_{k+1}\overline{y}_{k+1}$$

or in summation form as

$$\left(\sum_{i=0}^{k+1} A'_i S_i A_i\right)\widehat{x}_{k+1} = \sum_{i=0}^{k+1} A'_i S_i y_i$$

Defining

$$Q_{k+1} = \sum_{i=0}^{k+1} A'_i S_i A_i.$$

we can write a recursion for Q_{k+1} as follows:

$$Q_{k+1} = Q_k + A'_{k+1}S_{k+1}A_{k+1}.$$

Rearranging the summation form equation for $\widehat{x}_{k+1}$, we get

$$\begin{aligned}\widehat{x}_{k+1} &= Q_{k+1}^{-1}\left[\left(\textstyle\sum_{i=0}^{k} A'_i S_i A_i\right)\widehat{x}_k + A'_{k+1}S_{k+1}y_{k+1}\right]\\ &= Q_{k+1}^{-1}\left[Q_k\widehat{x}_k + A'_{k+1}S_{k+1}y_{k+1}\right]\end{aligned}$$

This clearly displays the new estimate as a weighted combination of the old estimate and the new data, so we have the desired recursion. Another useful form of this result is obtained by substituting from the recursion for Q_{k+1} above to get

$$\widehat{x}_{k+1} = \widehat{x}_k - Q_{k+1}^{-1}\left(A'_{k+1}S_{k+1}A_{k+1}\widehat{x}_k - A'_{k+1}S_{k+1}y_{k+1}\right) \quad ,$$

which finally reduces to

$$\widehat{x}_{k+1} = \widehat{x}_k + \underbrace{Q_{k+1}^{-1}A'_{k+1}S_{k+1}}_{\text{Kalman Filter Gain}}\underbrace{(y_{k+1} - A_{k+1}\widehat{x}_k)}_{\text{innovations}}$$

The quantity $Q_{k+1}^{-1}A'_{k+1}S_{k+1}$ is called the *Kalman gain*, and $y_{k+1} - A_{k+1}\widehat{x}_k$ is called the *innovations*, since it compares the difference between a data update and the prediction given the last estimate.

Unfortunately, as one acquires more and more data, i.e. as k grows large, the Kalman gain goes to zero. One data point cannot make much headway against the mass of previous data which has 'hardened' the estimate. If we leave this estimator as is—without modification—the estimator 'goes to sleep' after a while, and thus doesn't adapt well to parameter changes. The homework investigates the concept of a 'fading memory' so that the estimator doesn't go to sleep.

An Implementation Issue

Another concept which is important in the implementation of the RLS algorithm is the computation of Q_{k+1}^{-1}. If the dimension of Q_k is very large, computation of its inverse can be computationally expensive, so one would like to have a recursion for Q_{k+1}^{-1}.

This recursion is easy to obtain. Applying the handy matrix identity

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B\Big(DA^{-1}B + C^{-1}\Big)^{-1}DA^{-1}$$

to the recursion for Q_{k+1} yields

$$Q_{k+1}^{-1} = Q_k^{-1} - Q_k^{-1}A'_{k+1}\Big(A_{k+1}Q_k^{-1}A'_{k+1} + S_{k+1}^{-1}\Big)^{-1}A_{k+1}Q_k^{-1} \quad .$$

Upon defining

$$P_{k+1} = Q_{k+1}^{-1} \quad ,$$

this becomes

$$P_{k+1} = P_k - P_kA'_{k+1}\Big(S_{k+1}^{-1} + A_{k+1}P_kA'_{k+1}\Big)^{-1}A_{k+1}P_k \quad .$$

which is called the (discrete-time) *Riccati equation.*

Interpretation

We have $\widehat{x}_k$ and y_{k+1} available for computing our updated estimate. Interpreting $\widehat{x}_k$ as a measurement, we see our model becomes

$$\begin{bmatrix} \widehat{x}_k \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} I \\ A_{k+1} \end{bmatrix} x + \begin{bmatrix} e_k \\ e_{k+1} \end{bmatrix} .$$

The criterion, then, by which we choose $\widehat{x}_{k+1}$ is thus

$$\widehat{x}_{k+1} = \text{argmin}\left(e'_kQ_ke_k + e'_{k+1}S_{k+1}e_{k+1}\right) \quad .$$

In this context, one interprets Q_k as the weighting factor for the previous estimate.

Exercises

Exercise 2.1 Least Squares Fit of an Ellipse

Suppose a particular object is modeled as moving in an elliptical orbit centered at the origin. Its nominal trajectory is described in rectangular coordinates (r, s) by the constraint equation $x_1 r^2 + x_2 s^2 + x_3 rs = 1$, where x_1, x_2, and x_3 are unknown parameters that specify the orbit. We have available the following noisy measurements of the object's coordinates (r, s) at ten different points on its orbit:

(0.6728, 0.0589) (0.3380, 0.4093) (0.2510, 0.3559) (-0.0684, 0.5449)
(-0.4329, 0.3657) (-0.6921, 0.0252) (-0.3681, -0.2020) (0.0019, -0.3769)
(0.0825, -0.3508) (0.5294, -0.2918)

The ten measurements are believed to be equally reliable. For your convenience, these ten pairs of measured (r, s) values have been stored in column vectors named r and s that you can access through the 6.241 locker on Athena.* After `add 6.241`, and once in the directory in which you are running Matlab, you can copy the data using `cp /mit/6.241/Public/fall95/hw1rs.mat hw1rs.mat`. Then, in Matlab, type `load hw1rs` to load the desired data; type `who` to confirm that the vectors r and s are indeed available.

Using the assumed constraint equation, we can arrange the given information in the form of the linear system of (approximate) equations $Ax \approx b$, where A is a known 10×3 matrix, b is a known 10×1 vector, and $x = (\, x_1, x_2, x_3 \,)^T$. This system of 10 equations in 3 unknowns is inconsistent. We wish to find the solution x that minimizes the Euclidean norm (or length) of the error $Ax - b$. Compare the solutions obtained by using the following four Matlab invocations, each of which in principle gives the desired least-square-error solution:

(a) $x = A \backslash b$

(b) $x = \text{pinv}(A) * b$

(c) $x = \text{inv}(A' * A) * A' * b$

(d) $[q, r] = qr(A)$, followed by implementation of the approach described in Exercise 3.1.

For more information on these commands, try `help slash`, `help qr`, `help pinv`, `help inv`, etc. [Incidentally, the prime, $'$, in Matlab takes the transpose of the *complex conjugate* of a matrix; if you want the ordinary transpose of a complex matrix C, you have to write $C.'$ or `transp`(C).]

You should include in your solutions a plot the ellipse that corresponds to your estimate of x. If you create the following function file in your Matlab directory, with the name `ellipse.m`, you can obtain the polar coordinates `theta, rho` of n points on the ellipse specified by the parameter vector x. To do this, enter `[theta,rho]=ellipse(x,n);` at the Matlab prompt. You can then plot the ellipse by using the `polar(theta,rho)` command.

```
function [theta,rho]=ellipse(x,n)
% [theta,rho]=ellipse(x,n)
%
% The vector x = [x(1),x(2),x(3)]', defines an ellipse centered at the origin
% via the equation x(1)*r^ 2 + x(2)*s^ 2 +x(3)*r*s = 1.
% This routine generates the polar coordinates of points on the ellipse,
% to send to a plot command. It does this by solving for the radial
% distance in n equally spaced angular directions.
% Use polar(theta,rho) to actually plot the ellipse.
```

* Athena is MIT's UNIX-based computing environment. OCW does not provide access to it.

```
theta = 0:(2*pi/n):(2*pi);
a = x(1)*cos(theta).^ 2 + x(2)*sin(theta).^ 2 + x(3)*(cos(theta).*sin(theta));
rho = ones(size(a))./sqrt(a);
```

Exercise 2.2 Approximation by a Polynomial

Let $f(t) = 0.5e^{0.8t}$, $t \in [0, 2]$.

(a) Suppose 16 exact measurements of $f(t)$ are available to you, taken at the times t_i listed in the array T below:

$$T = \begin{array}{llllllll} [2 \cdot 10^{-3}, & 0.136, & 0.268, & 0.402, & 0.536, & 0.668, & 0.802, & 0.936, \\ 1.068, & 1.202, & 1.336, & 1.468, & 1.602, & 1.736, & 1.868, & 2.000] \end{array}$$

Use Matlab to generate these measurements:

$$y_i = f(t_i) \qquad i = 1, \ldots, 16 \qquad t_i \in T$$

Now determine the coefficients of the least square error polynomial approximation of the measurements, for

1. a polynomial of degree 15, $p_{15}(t)$;
2. a polynomial of degree 2, $p_2(t)$.

Compare the quality of the two approximations by plotting $y(t_i)$, $p_{15}(t_i)$ and $p_2(t_i)$ for all t_i in T. To see how well we are approximating the function on the whole interval, also plot $f(t)$, $p_{15}(t)$ and $p_2(t)$ on the interval $[0, 2]$. (Pick a very fine grid for the interval, e.g. t=[0:1000]'/500.) Report your observations and comments.

(b) Now suppose that your measurements are affected by some noise. Generate the measurements using

$$y_i = f(t_i) + e(t_i) \qquad i = 1, \ldots, 16 \qquad t_i \in T$$

where the vector of noise values can be generated in the following way:

$$randn('seed', 0);$$
$$e = randn(size(T));$$

Again determine the coefficients of the least square error polynomial approximation of the measurements for

1. a polynomial of degree 15, $p_{15}(t)$;
2. a polynomial of degree 2, $p_2(t)$.

Compare the two approximations as in part (a). Report your observations and comments. Explain any surprising results.

(c) So far we have obtained polynomial approximations of $f(t)$, $t \in [0,2]$, by approximating the measurements at $t_i \in T$. We are now interested in minimizing the square error of the polynomial approximation over the whole interval $[0,2]$:

$$\min \|f(t) - p_n(t)\|_2^2 = \min \int_0^2 |f(t) - p_n(t)|^2 \, dt$$

where $p_n(t)$ is some polynomial of degree n. Find the polynomial $p_2(t)$ of degree 2 that solves the above problem. Are the optimal $p_2(t)$ in this case and the optimal $p_2(t)$ of parts (a) and (b) very different from each other? Elaborate.

Exercise 2.3 Combining Estimates

Suppose $y_1 = C_1 x + e_1$ and $y_2 = C_2 x + e_2$, where x is an n-vector, and C_1, C_2 have full column rank. Let $\hat{x}_1$ denote the value of x that minimizes $e_1^T S_1 e_1$, and $\hat{x}_2$ denote the value that minimizes $e_2^T S_2 e_2$, where S_1 and S_2 are positive definite matrices. Show that the value $\hat{x}$ of x that minimizes $e_1^T S_1 e_1 + e_2^T S_2 e_2$ can be written entirely in terms of $\hat{x}_1$, $\hat{x}_2$, and the $n \times n$ matrices $Q_1 = C_1^T S_1 C_1$ and $Q_2 = C_2^T S_2 C_2$. What is the significance of this result?

Exercise 2.4 Exponentially Windowed Estimates

Suppose we observe the *scalar* measurements

$$y_i = c_i x + e_i \ , \qquad i = 1, 2, \ldots$$

where c_i and x are possibly vectors (row- and column-vectors respectively).

(a) Show (by reducing this to a problem that we already know how to solve — don't start from scratch!) that the value $\hat{x}_k$ of x that minimizes the criterion

$$\sum_{i=1}^{k} f^{k-i} e_i^2, \qquad \text{some fixed } f, \quad 0 < f \leq 1$$

is given by

$$\hat{x}_k = \Big(\sum_{i=1}^{k} f^{k-i} c_i^T c_i \Big)^{-1} \Big(\sum_{i=1}^{k} f^{k-i} c_i^T y_i \Big)$$

The so-called *fade* or *forgetting* factor f allows us to preferentially weight the more recent measurements by picking $0 < f < 1$, so that old data is discounted at an exponential rate. We then say that the data has been subjected to exponential fading or forgetting or weighting or windowing or tapering or ... . This is usually desirable, in order to keep the filter adaptive to changes that may occur in x. Otherwise the filter becomes progressively less attentive to new data and falls asleep, with its gain approaching 0.

(b) Now show that

$$\hat{x}_k = \hat{x}_{k-1} + Q_k^{-1} c_k^T (y_k - c_k \hat{x}_{k-1})$$

where

$$Q_k = f Q_{k-1} + c_k^T c_k , \qquad Q_0 = 0$$

The vector $g_k = Q_k^{-1} c_k^T$ is termed the *gain* of the estimator.

(c) If x and c_i are scalars, and c_i is a constant c, determine g_k as a function of k. What is the *steady-state gain* g_∞? Does g_∞ increase or decrease as f increases — and why do you expect this?

Exercise 2.5 Suppose our model for some waveform $y(t)$ is $y(t) = \alpha \sin(\omega t)$, where α is a scalar, and suppose we have measurements $y(t_1), \ldots, y(t_p)$. Because of modeling errors and the presence of measurement noise, we will generally not find any choice of model parameters that allows us to precisely account for all p measurements.

(a) If ω is known, find the value of α that minimizes

$$\sum_{i=1}^{p} [y(t_i) - \alpha \sin(\omega t_i)]^2$$

(b) Determine this value of α if $\omega = 2$ and if the measured values of $y(t)$ are:

$$y(1) = +2.31 \quad y(2) = -2.01 \quad y(3) = -1.33 \quad y(4) = +3.23$$

$$y(5) = -1.28 \quad y(6) = -1.66 \quad y(7) = +3.28 \quad y(8) = -0.88$$

(I generated this data using the equation $y(t) = 3\sin(2t) + e(t)$ evaluated at the integer values $t = 1, \ldots, 8$, and with $e(t)$ for each t being a random number uniformly distributed in the interval - 0.5 to +0.5.)

(c) Suppose that α *and* ω are unknown, and that we wish to determine the values of these two variables that minimize the above criterion. Assume you are given initial estimates α_0 and ω_0 for the minimizing values of these variables. Using the Gauss-Newton algorithm for this nonlinear least squares problem, i.e. applying LLSE to the problem obtained by linearizing about the initial estimates, determine explicitly the estimates α_1 and ω_1 obtained after one iteration of this algorithm. Use the following notation to help you write out the solution in a condensed form:

$$a = \sum \sin^2(\omega_0 t_i) \ , \quad b = \sum t_i^2 \cos^2(\omega_0 t_i) \ , \quad c = \sum t_i [\sin(w_0 t_i)][\cos(w_0 t_i)]$$

(d) What values do you get for α_1 and ω_1 with the data given in (b) above if the initial guesses are $\alpha_0 = 3.2$ and $\omega_0 = 1.8$? Continue the iterative estimation a few more steps. Repeat the procedure when the initial guesses are $\alpha_0 = 3.5$ and $\omega_0 = 2.5$, verifying that the algorithm does not converge.

(e) Since only ω enters the model nonlinearly, we might think of a decomposed algorithm, in which α is estimated using *linear* least squares and ω is estimated via nonlinear least squares. Suppose, for example, that our initial estimate of ω is $\omega_0 = 1.8$. Now obtain an estimate α_1 of α using the linear least squares method that you used in (b). Then obtain an (improved?) estimate ω_1 of ω, using one iteration of a Gauss-Newton algorithm (similar to what is needed in (c), except that now you are only trying to estimate ω). Next obtain the estimate α_2 via linear least squares, and so on. Compare your results with what you obtain via this decomposed procedure when your initial estimate is $\omega_0 = 2.5$ instead of 1.8.

Exercise 2.6 Comparing Different Estimators

This problem asks you to compare the behavior of different parameter estimation algorithms by fitting a model of the type $y(t) = a\sin(2\pi t) + b\cos(4\pi t)$ to noisy data taken at values of t that are .02 apart in the interval (0,2].

First synthesize the data on which you will test the algorithms. Even though your estimation algorithms will assume that a and b are constant, we are interested in seeing how they track parameter changes as well. Accordingly, let $a = 2$, $b = 2$ for the first 50 points, and $a = 1$, $b = 3$ for the next 50 points. To get (approximately) normally distributed random variables, we use the function *randn* to produce variables with mean 0 and variance 1.

An elegant way to generate the data in Matlab, exploiting Matlab's facility with vectors, is to define the vectors $t1 = 0.02 : 0.02 : 1.0$ and $t2 = 1.02 : 0.02 : 2.0$, then set

$$y1 = 2 * \sin(2 * \text{pi} * t1) + 2 * \cos(4 * \text{pi} * t1) + s * randn(size(t1))$$

and

$$y2 = \sin(2 * \text{pi} * t2) + 3 * \cos(4 * \text{pi} * t2) + s * randn(size(t2))$$

where s determines the standard deviation of the noise. Pick $s = 1$ for this problem. Finally, set $y = [y1, y2]$. No loops, no counters, no fuss!!

Now estimate a and b from y using the following algorithms. Assume prior estimates $\hat{a}_0 = 3$ and $\hat{b}_0 = 1$, weighted equally with the measurements (so all weights can be taken as 1 without loss of generality). Plot your results to aid comparison.

(i) Recursive least squares.

(ii) Recursive least squares with exponentially fading memory, as in Problem 3. Use $f = .96$.

(iii) The algorithm in (ii), but with Q_k of Problem 3 replaced by $q_k = (1/n)\times\text{trace}(Q_k)$, where n is the number of parameters, so $n = 2$ in this case. (Recall that the trace of a matrix is the sum of its diagonal elements. Note that q_k itself satisfies a recursion, which you should write down.)

(iv) An algorithm of the form

$$\hat{x}_k = \hat{x}_{k-1} + \frac{.04}{c_k c_k^T} c_k^T (y_k - c_k \hat{x}_{k-1})$$

where $c_k = [\sin(2\pi t), \cos(4\pi t)]$ evaluated at the kth sampling instant, so $t = .02k$.

Exercise 2.7 Recursive Estimation of a State Vector

This course will soon begin to consider *state-space models* of the form

$$x_\ell = A x_{\ell-1} \tag{2.4}$$

where x_ℓ is an n-vector denoting the state at time ℓ of our model of some system, and A is a known $n \times n$ matrix. For example, suppose the system of interest is a rotating machine, with angular position d_ℓ and angular velocity ω_ℓ at time $t = \ell T$, where T is some fixed sampling interval. If we believed the machine to be rotating at constant speed, we would be led to the model

$$\begin{pmatrix} d_\ell \\ \omega_\ell \end{pmatrix} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} d_{\ell-1} \\ \omega_{\ell-1} \end{pmatrix}$$

Assume A to be nonsingular throughout this problem.

For the rotating machine example above, it is often of interest to obtain least-square-error estimates of the position and (constant) velocity, using noisy measurements of the angular position d_j at the sampling instants. More generally, it is of interest to obtain a least-square-error estimate of the state vector x_i in the model (2.4) from noisy p-component measurements y_j that are related to x_j by a linear equation of the form

$$y_j = Cx_j + e_j \ , \qquad j = 1, \ldots, i$$

where C is a $p \times n$ matrix. We shall also assume that a prior estimate $\hat{x}_0$ of x_0 is available:

$$\hat{x}_0 = x_0 + e_0$$

Let $\hat{x}_{i|i}$ denote the value of x_i that minimizes

$$\sum_{j=0}^{i} \|e_j\|^2$$

This is the estimate of x_i given the prior estimate and measurements up to time i, or the "filtered estimate" of x_i. Similarly, let $\hat{x}_{i|i-1}$ denote the value of x_i that minimizes

$$\sum_{j=0}^{i-1} \|e_j\|^2$$

This is the least-square-error estimate of x_i given the prior estimate and measurements up to time $i-1$, and is termed the "one-step prediction" of x_i.

a) Set up the linear system of equations whose least square error solution would be $\hat{x}_{i|i}$. Similarly, set up the linear system of equations whose least square error solution would be $\hat{x}_{i|i-1}$.

b) Show that $\hat{x}_{i|i-1} = A\hat{x}_{i-1|i-1}$.

c) Determine a recursion that expresses $\hat{x}_{i|i}$ in terms of $\hat{x}_{i-1|i-1}$ and y_i. This is the prototype of what is known as the *Kalman filter*. A more elaborate version of the Kalman filter would include additive noise driving the state-space model, and other embellishments, all in a stochastic context (rather than the deterministic one given here).

Exercise 2.8 Let $\hat{x}$ denote the value of x that minimizes $\|y - Ax\|^2$, where A has full column rank. Let $\overline{x}$ denote the value of x that minimizes this same criterion, but now subject to the constraint that $z = Dx$, where D has full *row* rank. Show that

$$\overline{x} = \hat{x} + (A^TA)^{-1}D^T\left(D(A^TA)^{-1}D^T\right)^{-1}(z - D\hat{x})$$

(Hint: One approach to solving this is to use our recursive least squares formulation, but modified for the limiting case where one of the measurement sets — namely $z = Dx$ in this case — is known to have no error. You may have to use some of the matrix identities from the previous chapter).