

# Least Squares

Math 45 — Linear Algebra

David Arnold

David-Arnold@Eureka.redwoods.cc.ca.us

## Abstract

Scientists and mathematicians often have need to “fit” a line to a set of data points. In this activity, you will learn how linear algebra greatly simplifies this task. *Prerequisites: Matrix operations, transpose and inverse, knowledge of the column space and null space of a matrix.*

## 1 Fundamentals

We begin by establishing some fundamentals involving the projection of vectors onto subspaces of  $R^n$ .

### 1.1 The Scalar Product

Suppose that  $\mathbf{x}$  and  $\mathbf{y}$  are two vectors from  $R^n$ . We define their *dot product* as follows.

#### Definition 1

If  $\mathbf{x}$  and  $\mathbf{y} \in R^n$ , then

$$\begin{aligned}\mathbf{x} \cdot \mathbf{y} &= \mathbf{x}^T \mathbf{y} \\ &= \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \\ &= x_1 y_1 + x_2 y_2 + \cdots + x_n y_n\end{aligned}$$

Note that the dot product of two vectors is a scalar, which is why the dot product is often called the *scalar product*. For example, if  $\mathbf{x} = (1, 2, 3)^T$  and  $\mathbf{y} = (4, 5, 6)^T$ , then

$$\begin{aligned}\mathbf{x} \cdot \mathbf{y} &= (1)(4) + (2)(5) + (3)(6) \\ &= 4 + 10 + 18 \\ &= 32.\end{aligned}$$

Of course, this is a simple operation in Matlab. Just use Matlab's transpose operator.

## Least Squares

```
>> x=[1;2;3],y=[4;5;6]
x =
    1
    2
    3
y =
    4
    5
    6
>> x.'*y
ans =
    32
```

It is a simple matter to prove a number of useful properties of the dot product. These properties are very useful when performing algebraic manipulations that involve expressions that contain the dot product. We leave the proof of these properties to the exercises.

### Proposition 1

If  $\mathbf{x}$ ,  $\mathbf{y}$ , and  $\mathbf{z}$  are vectors in  $R^n$  and  $\alpha$  is any scalar number from  $R$ , then

1.  $\mathbf{x} \cdot \mathbf{x} \geq 0$
2.  $\mathbf{x} \cdot \mathbf{x} = 0$  if and only if  $\mathbf{x} = \mathbf{0}$
3.  $(\alpha\mathbf{x}) \cdot \mathbf{y} = \mathbf{x} \cdot (\alpha\mathbf{y}) = \alpha(\mathbf{x} \cdot \mathbf{y})$
4.  $\mathbf{x} \cdot (\mathbf{y} + \mathbf{z}) = \mathbf{x} \cdot \mathbf{y} + \mathbf{x} \cdot \mathbf{z}$

## 1.2 The Norm of a Vector

Suppose that  $\mathbf{x} = (x_1, x_2)^T$  is a vector in the plane, as shown in [Figure 1](#). Then, if we let  $\|\mathbf{x}\|$  represent the length of a vector, then the Pythagorean theorem tells us that

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2}.$$

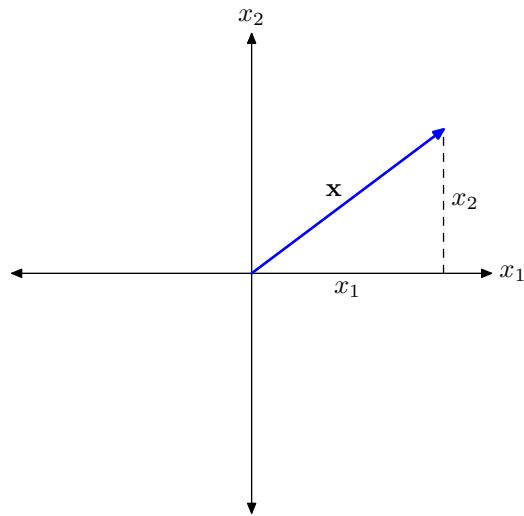
This result easily extends to vectors in  $R^n$ .

### Definition 2

If  $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in R^n$ , then the length of  $\mathbf{x}$  is defined as

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}.$$

## Least Squares



**Figure 1** The norm of vector  $\mathbf{x}$  is  $\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2}$ .

Sometimes we use equivalent words for the length of a vector, words like *norm* or *magnitude*.

If  $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ , then  $\|\mathbf{x}\|^2 = x_1^2 + x_2^2 + \dots + x_n^2$  and  $\mathbf{x} \cdot \mathbf{x} = x_1^2 + x_2^2 + \dots + x_n^2$ , so we have the following proposition.

**Proposition 2**

If  $\mathbf{x} = (x_1, x_2, \dots, x_n) \in R^n$  then

$$\|\mathbf{x}\|^2 = \mathbf{x} \cdot \mathbf{x}.$$

It is easy to compute the length of a vector in Matlab. For example, this computation computes the square of the length of  $\mathbf{x} = (1, 2, 3)^T$ .

```
>> x=[1;2;3]
x =
     1
     2
     3
>> x.'*x
ans =
    14
```

And this computation computes the length.

```
>> sqrt(x.'*x)
ans =
    3.7417
```

An important inequality relating the magnitude or norm and the dot product is the Cauchy-Schwartz inequality.

**Proposition 3**

If  $\mathbf{x}$  and  $\mathbf{y}$  are vectors in  $R^n$ , then

$$|\mathbf{x} \cdot \mathbf{y}| \leq \|\mathbf{x}\| \|\mathbf{y}\|.$$

Again, it is a simple matter to establish a number of important properties involving the length or magnitude of vectors. We leave the proof of these properties as exercises.

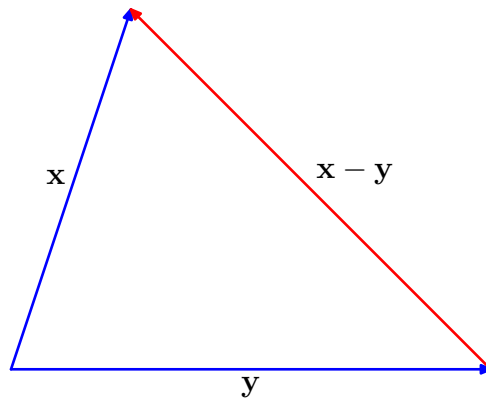
**Proposition 4**

If  $\mathbf{x}$  and  $\mathbf{y}$  are vectors in  $R^n$  and  $\alpha$  is any scalar number from  $R$ , then

1.  $\|\mathbf{x}\| \geq 0$
2.  $\|\mathbf{x}\| = 0$  if and only if  $\mathbf{x} = \mathbf{0}$
3.  $\|\alpha\mathbf{x}\| = |\alpha| \|\mathbf{x}\|$
4.  $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$

## 1.3 Orthogonality

It is important that we know how to test when one vector is perpendicular to another, which leads immediately to the question of finding the angle between two vectors. If that angle is  $90^\circ$ , then the vectors will be perpendicular. In [Figure 2](#), pictured are two vectors  $\mathbf{x}$  and  $\mathbf{y}$ , and their difference,  $\mathbf{x} - \mathbf{y}$ .



**Figure 2** Two vectors and their difference.

The law of cosines provides a meaningful relationship between the lengths of the sides of the triangle and the angle between the vectors  $\mathbf{x}$  and  $\mathbf{y}$ .

$$\|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\|\mathbf{x}\| \|\mathbf{y}\| \cos \theta$$

We can use [Proposition 1](#) and [Proposition 2](#) to simplify the left-hand side.

## Least Squares

$$\begin{aligned}\|\mathbf{x} - \mathbf{y}\|^2 &= (\mathbf{x} - \mathbf{y}) \cdot (\mathbf{x} - \mathbf{y}) \\ &= \mathbf{x} \cdot \mathbf{x} - 2\mathbf{x} \cdot \mathbf{y} + \mathbf{y} \cdot \mathbf{y} \\ &= \|\mathbf{x}\|^2 - 2\mathbf{x} \cdot \mathbf{y} + \|\mathbf{y}\|^2\end{aligned}$$

Thus,

$$\|\mathbf{x}\|^2 - 2\mathbf{x} \cdot \mathbf{y} + \|\mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\|\mathbf{x}\|\|\mathbf{y}\|\cos\theta,$$

and cancelling equal quantities from each side of this last relation leads to the following result.

### Proposition 5

If  $\mathbf{x}$  and  $\mathbf{y}$  are vectors in  $R^2$ , then

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\|\|\mathbf{y}\|\cos\theta.$$

If two nonzero vectors  $\mathbf{x}$  and  $\mathbf{y}$  are perpendicular in the plane, then the angle between them is  $90^\circ$  and

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\|\|\mathbf{y}\|\cos 90^\circ = 0.$$

Conversely, if the dot product of two nonzero vectors  $\mathbf{x}$  and  $\mathbf{y}$  in the plane is zero, then

$$\|\mathbf{x}\|\|\mathbf{y}\|\cos\theta = \mathbf{x} \cdot \mathbf{y} = 0.$$

Because the vectors are nonzero, this last result implies that the cosine of the angle between the vectors is 0, making the angle between the vectors  $90^\circ$ . Consequently, the vectors are perpendicular. This argument applies equally well to vectors in  $R^3$ .

### Proposition 6

In  $R^2$  and  $R^3$ , two nonzero vectors  $\mathbf{x}$  and  $\mathbf{y}$  are perpendicular if and only if their dot product is zero.

It is difficult to imagine what perpendicular vectors would look like in higher dimensions. Still, we are inclined to apply earlier results to vectors in higher dimensions.

### Definition 3

Two vectors  $\mathbf{x}$  and  $\mathbf{y}$  in  $R^n$  are said to be *orthogonal* (think perpendicular) if and only if their dot product is zero.

Matlab makes it easy to check orthogonality in higher dimensions.

```
>> x=[1;1;2;-1;0;3];y=[2;1;0;0;0;-1];
>> x.'*y
ans =
    0
```

Thus, the vectors  $x = (1, 1, 2, -1, 0, 3)^T$  and  $y = (2, 1, 0, 0, 0, -1)^T$  are orthogonal.

## 1.4 Projecting a Vector Onto a Line

In this section, we want to project a vector  $\mathbf{b}$  onto another vector  $\mathbf{a}$ . Thinking geometrically, this means that we want to find the vector on  $\mathbf{a}$  that is “closest” to the vector  $\mathbf{b}$ . Orthogonality naturally comes into play.

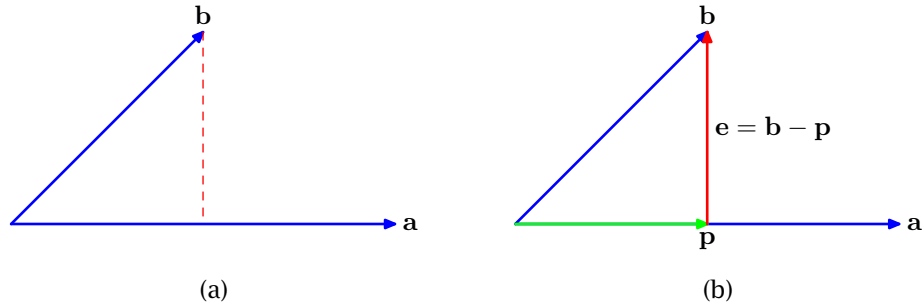


Figure 3 Projecting vector  $\mathbf{b}$  onto vector  $\mathbf{a}$ .

In **Figure 3a**, we “project” the vector  $\mathbf{b}$  onto the vector  $\mathbf{a}$ . Of course, this is an orthogonal (perpendicular) projection. The result is the projection vector  $\mathbf{p}$  shown in **Figure 3b**. Note that this selection of  $\mathbf{p}$  on the vector  $\mathbf{a}$  makes the “error vector”  $\mathbf{e} = \mathbf{b} - \mathbf{p}$  as small as possible.

Note that the projection vector  $\mathbf{p}$  is parallel to  $\mathbf{a}$ . Thus,  $\mathbf{p}$  must be a scalar multiple of  $\mathbf{a}$ . That is,  $\mathbf{p} = \hat{x}\mathbf{a}$ , where  $\hat{x}$  is some scalar real number. At this point, we proceed in three orderly steps.

1. Find the scalar  $\hat{x}$ .
2. Find the projection vector  $\mathbf{p}$ .
3. Find the projection matrix  $P$  that will map  $\mathbf{b}$  onto  $\mathbf{p}$ .

Note that the “error vector” in **Figure 3b** is orthogonal to  $\mathbf{a}$ . Thus, the dot product of  $\mathbf{a}$  and  $\mathbf{e}$  must equal zero.

$$\begin{aligned} \mathbf{a} \cdot \mathbf{e} &= 0 \\ \mathbf{a} \cdot (\mathbf{b} - \mathbf{p}) &= 0 \\ \mathbf{a} \cdot (\mathbf{b} - \hat{x}\mathbf{a}) &= 0 \\ \mathbf{a} \cdot \mathbf{b} - \hat{x}(\mathbf{a} \cdot \mathbf{a}) &= 0 \\ \hat{x} &= \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a} \cdot \mathbf{a}} \end{aligned}$$

That completes our first task. Next,  $\mathbf{p} = \hat{x}\mathbf{a}$ , so the projection vector is easily obtained.

### Proposition 7

If  $\mathbf{b}$  and  $\mathbf{a}$  are vectors in  $R^n$ , then the projection of vector  $\mathbf{b}$  onto vector  $\mathbf{a}$  is given by

$$\mathbf{p} = \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a} \cdot \mathbf{a}} \mathbf{a}.$$

## Least Squares

For example, suppose that we want to project the vector  $\mathbf{b} = (2, -1, 0)$  onto the vector  $\mathbf{a} = (1, 1, 1)$ . This is easily accomplished by using **Proposition 7** and Matlab.

```
>> a=[1;1;1];b=[2;-1;0];
>> p=(a.'*b)/(a.'*a)*a
p =
    0.3333
    0.3333
    0.3333
```

Thus,  $\mathbf{p} = (1/3, 1/3, 1/3)^T$ .

For our third task, the projection matrix  $P$  is easily found by reordering the factors of the projection formula in **Proposition 7**. First, in makes no difference whether the scalar quantity  $(\mathbf{a} \cdot \mathbf{b})/(\mathbf{a} \cdot \mathbf{a})$  comes before or after that vector  $\mathbf{a}$ , so we can write

$$\mathbf{p} = \mathbf{a} \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a} \cdot \mathbf{a}}.$$

Recalling that  $\mathbf{x} \cdot \mathbf{y}$  equals  $\mathbf{x}^T \mathbf{y}$ , we write

$$\mathbf{p} = \mathbf{a} \frac{\mathbf{a}^T \mathbf{b}}{\mathbf{a}^T \mathbf{a}}.$$

But matrix multiplication is associative, and the scalar  $(\mathbf{a}^T \mathbf{a})$  can be placed anywhere we wish. Thus,

$$\mathbf{p} = \frac{\mathbf{a} \mathbf{a}^T}{\mathbf{a}^T \mathbf{a}} \mathbf{b}.$$

Note that this last result has the form  $\mathbf{p} = P\mathbf{b}$ , revealing the projection matrix  $P$ .

### Proposition 8

If  $\mathbf{a}$  is a vector in  $R^n$ , then

$$P = \frac{\mathbf{a} \mathbf{a}^T}{\mathbf{a}^T \mathbf{a}}$$

will project all vectors  $\mathbf{b}$  in  $R^n$  onto a line through the vector  $\mathbf{a}$ .

Theory is great, but it's a lot more reassuring when we see the results of our computation come alive on the computer screen. Let's perform a little experiment where we use a projection matrix to project a number of random points in the plane onto the line through the vector  $\mathbf{a} = (1, 1)^T$ .

The Matlab command `rand` produces a random number between 0 and 1. That is,

$$0 \leq \text{rand} < 1.$$

Multiplying by 2,

$$0 \leq 2\text{rand} < 2.$$

Subtracting 1,

$$-1 \leq 2\text{rand} - 1 < 1.$$

## Least Squares

Thus, the command `2*rand-1` produces a random number between  $-1$  and  $1$ . The command `2*rand(2,100)-1` will fill a  $2 \times 100$  matrix with random numbers between  $-1$  and  $1$ . Each column of the resulting matrix gives a random point in the plane.

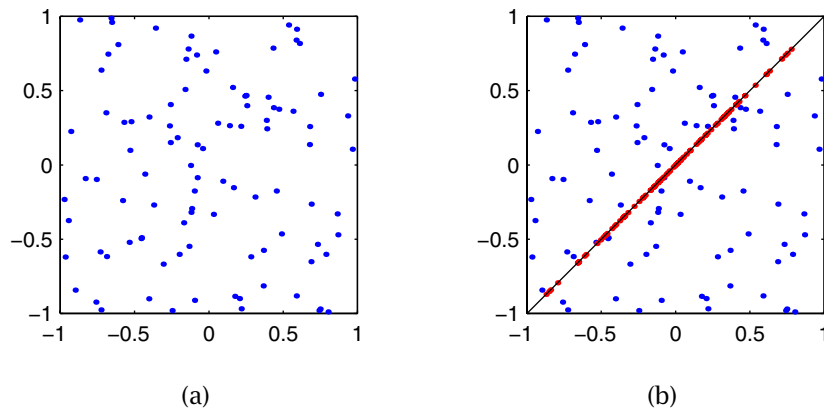
```
>> X=2*rand(2,100)-1;
```

Removing the semicolon that suppresses the output of the last command is revealing. We can plot these points in the plane. The first row of matrix  $X$  contains the  $x$ -values of the randomly generated points; the second row contains the  $y$ -values.

```
>> x=X(1, :);  
>> y=X(2, :);
```

We can now plot these points in the plane. The following command was used to generate the image in [Figure 4\(a\)](#).

```
plot(x,y,'b.')
```



**Figure 4** Projecting points in the plane onto the line through  $(1, 1)^T$ .

Next, use [Proposition 8](#) to compute the matrix  $P$  that will project vectors onto the vector  $\mathbf{a} = (1, 1)$ .

```
>> a=[1;1];  
>> P=(a*a. ')/(a. '*a)
```

Finally, we want to hit each point in [Figure 4a](#) with the projection matrix  $P$ , then plot the resulting points. We have a number of different ways to multiply matrices. One such form of matrix multiplication is particularly suited for the task at hand. Recall that

$$PX = P[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{100}] = [P\mathbf{x}_1, P\mathbf{x}_2, \dots, P\mathbf{x}_{100}].$$

That is, the first column of matrix  $PX$  contains the product of  $P$  and the first column of  $X$ , which is one of our 100 randomly plotted points in the plane. In like manner, the matrix multiplication  $PX$  continues along, multiplying each point in matrix  $X$  with the projection matrix  $P$ . This is exactly what we want.

```
>> U=P*X;
```

Again, it is instructive to remove the suppressing semicolon to examine the output of this command. The columns of matrix  $U$  now contains the projection of each point in matrix  $X$  onto the vector  $\mathbf{a}$ . The  $x$  values of these projections lie in the first row of  $U$ , the  $y$ -values in the second row.

```
>> ux=U(1,:);
>> uy=U(2,:);
```

We can now plot the original points and their projections with this command.

```
>> plot(x,y,'b.',ux,uy,'r.')
```

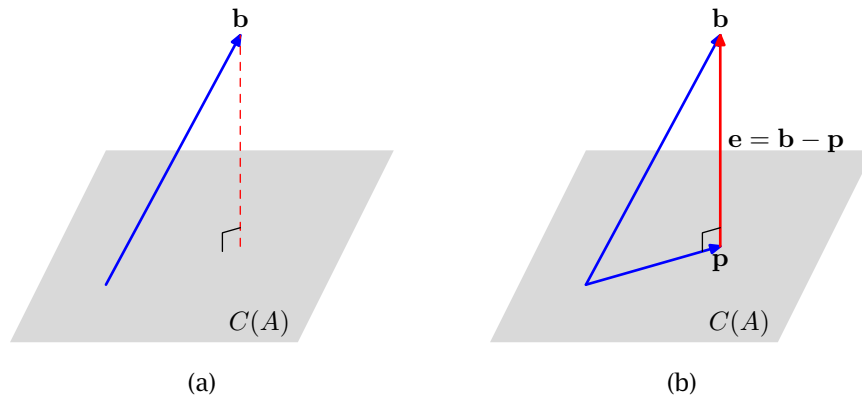
Drawing a line connecting the point  $(-1, -1)$  to the point  $(1, 1)$  solidifies the point that the projection matrix is projecting points in the plane onto the line through the vector  $\mathbf{a} = (1, 1)^T$ . The following command was used to generate the image in [Figure 4b](#).

```
>> plot(x,y,'b.',ux,uy,'r.',[-1,1],[-1,1],'k')
```

## 1.5 Projecting a Vector Onto a Subspace

In this section we learn how to project a vector  $\mathbf{b}$  onto the subspace spanned by the vectors  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$ . Let us assume that the vectors  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$  are independent. Thus, the vectors  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$  form a basis for the column space of matrix  $A = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k]$ . If  $\mathbf{b}$  lies in the column space of  $A$ , there is nothing to do. The projection of  $\mathbf{b}$  onto the column space of  $A$  is  $\mathbf{b}$ . So, we assume that  $\mathbf{b}$  does not lie in the column space of  $A$  and we seek the projection of  $\mathbf{b}$  onto the column space of  $A$ .

It is helpful to think of the column space as a plane. The vector  $\mathbf{b}$  does not lie in that plane. This is pictured in [Figure 5a](#). In [Figure 5b](#), we project the vector  $\mathbf{b}$  onto the vector  $\mathbf{p}$ , which lies in the “plane” spanned by the vectors  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$ . Note again that the “error vector,”  $\mathbf{e} = \mathbf{b} - \mathbf{p}$ , is orthogonal to the column space of  $A$ .



**Figure 5** Projecting vector  $\mathbf{b}$  onto the column space of  $A$ .

Thinking geometrically, what is required is that our “error vector” be orthogonal to every vector in the column space of  $A$ . This requirement is satisfied if  $\mathbf{e}$  is orthogonal to each of the basis elements  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$ , for if  $\mathbf{x}$  is in the column space of  $A$ , then  $\mathbf{x}$  is a linear combination of the columns of  $A$  and

$$\begin{aligned} \mathbf{e} \cdot \mathbf{x} &= \mathbf{e} \cdot (c_1 \mathbf{a}_1 + c_2 \mathbf{a}_2 + \dots + c_k \mathbf{a}_k), \\ &= \mathbf{e} \cdot \mathbf{a}_1 + \mathbf{e} \cdot \mathbf{a}_2 + \dots + \mathbf{e} \cdot \mathbf{a}_k, \\ &= 0 + 0 + \dots + 0, \\ &= 0, \end{aligned}$$

making  $\mathbf{e}$  orthogonal to  $\mathbf{x}$ . Thus, we need

## Least Squares

$$\begin{aligned}\mathbf{a}_1 \cdot \mathbf{e} &= 0, \\ \mathbf{a}_2 \cdot \mathbf{e} &= 0, \\ &\vdots \\ \mathbf{a}_k \cdot \mathbf{e} &= 0,\end{aligned}$$

or, in matrix notation,

$$\begin{aligned}\mathbf{a}_1^T \mathbf{e} &= 0, \\ \mathbf{a}_2^T \mathbf{e} &= 0, \\ &\vdots \\ \mathbf{a}_k^T \mathbf{e} &= 0.\end{aligned}$$

But  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$  are the columns of matrix  $A$ , so  $\mathbf{a}_1^T, \mathbf{a}_2^T, \dots, \mathbf{a}_k^T$  are the rows of matrix  $A^T$ . Thinking in terms of *block multiplication*,

$$\begin{pmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_k^T \end{pmatrix} \mathbf{e} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Of course, this is the same as

$$A^T \mathbf{e} = \mathbf{0}.$$

In [Figure 5b](#), notice that the projection vector  $\mathbf{p}$  lies in the column space of  $A$ . Thus,  $\mathbf{p}$  can be written as a linear combination of the columns of  $A$ .

$$\begin{aligned}\mathbf{p} &= x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \cdots + x_k \mathbf{a}_k, \\ &= (\mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_k) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix}, \\ &= A\hat{\mathbf{x}}.\end{aligned}$$

Thus,  $\mathbf{e} = \mathbf{b} - A\hat{\mathbf{x}}$  and we can write

$$A^T(\mathbf{b} - A\hat{\mathbf{x}}) = \mathbf{0}.$$

Following the lead we took in finding the projection of a vector onto a line, we will again proceed in three orderly steps.

1. Find  $\hat{\mathbf{x}}$ .
2. Find the projection vector  $\mathbf{p}$ .
3. Find the projection matrix  $P$  that maps vectors onto the column space of  $A$ .

First,

$$\begin{aligned}A^T(\mathbf{b} - A\hat{\mathbf{x}}) &= \mathbf{0}, \\ A^T \mathbf{b} - A^T A\hat{\mathbf{x}} &= \mathbf{0}, \\ A^T A\hat{\mathbf{x}} &= A^T \mathbf{b}.\end{aligned}$$

## Least Squares

If the columns of  $A$  are independent, then you can show that  $A^T A$  is nonsingular.

**Proposition 9**

If  $A$  is an  $m \times n$  matrix with linearly independent columns, then  $A^T A$  is nonsingular.

If  $A^T A$  is nonsingular, then we can multiply the last equation by  $(A^T A)^{-1}$  to obtain

$$\hat{\mathbf{x}} = (A^T A)^{-1} A^T \mathbf{b},$$

which allows us to calculate the projection vector  $\mathbf{p}$ .

$$\begin{aligned} \mathbf{p} &= A \hat{\mathbf{x}} \\ &= A(A^T A)^{-1} A^T \mathbf{b} \end{aligned}$$

Note that this last result has the form  $\mathbf{p} = P\mathbf{b}$ , revealing the projection matrix  $P$ .

**Proposition 10**

If  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$  is a collection of linearly independent vectors in  $R^n$ , then

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

will project any vector in  $R^n$  onto the column space of the matrix  $A = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k]$ .

It will help our confidence if we can see that the projection matrix does exactly what it is designed to do, project vectors onto the column space of  $A$ . First, generate a set of points in three space.

```
>> X=3*rand(3,100)-1;
```

Strip off the  $x$ ,  $y$ , and  $z$ -values.

```
>> x=X(1,:);  
>> y=X(2,:);  
>> z=X(3,:);
```

Plot the points in three space. The following command was used to generate the image in [Figure 6a](#).

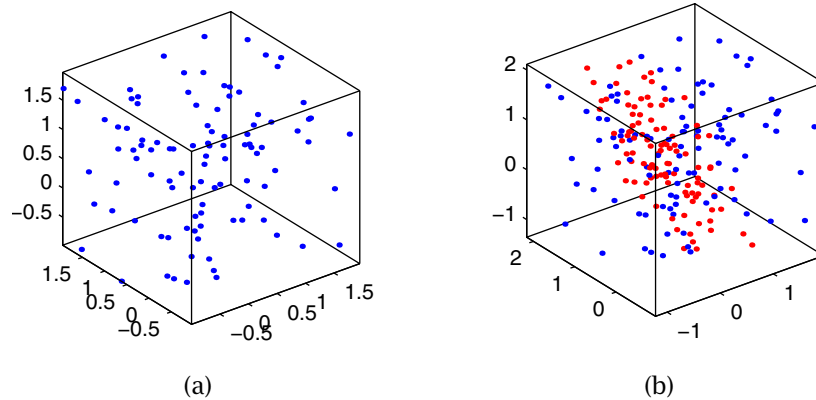
```
>> plot3(x,y,z,'b.')  
>> box on
```

Turning on the box frame adds a bit of depth. Let's project the points in [Figure 6a](#) onto the column space of

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

First, enter the matrix  $A$ .

## Least Squares



**Figure 6** Projecting points in space onto the column space of  $A$ .

```
>> a1=[1;1;0];a2=[0;1;1]
>> A=[a1,a2];
```

Next, use **Proposition 10** to calculate the projection matrix.

```
>> P=A*inv(A.'*A)*A.'
P =
    0.6667    0.3333   -0.3333
    0.3333    0.6667    0.3333
   -0.3333    0.3333    0.6667
```

Again, multiplying matrix  $X$  by the projection matrix  $P$  will project each column of the matrix  $X$  onto the column space of matrix  $A$ .

```
>> U=P*X;
```

Strip of the  $x$ ,  $y$ , and  $z$ -values.

```
>> ux=U(1,:);
>> uy=U(2,:);
>> uz=U(3,:);
```

Now, plot the original points and their projection. The following command was used to generate the image shown in **Figure 6b**.

```
>> plot3(x,y,z,'b.',ux,uy,uz,'r.')
>> box on
```

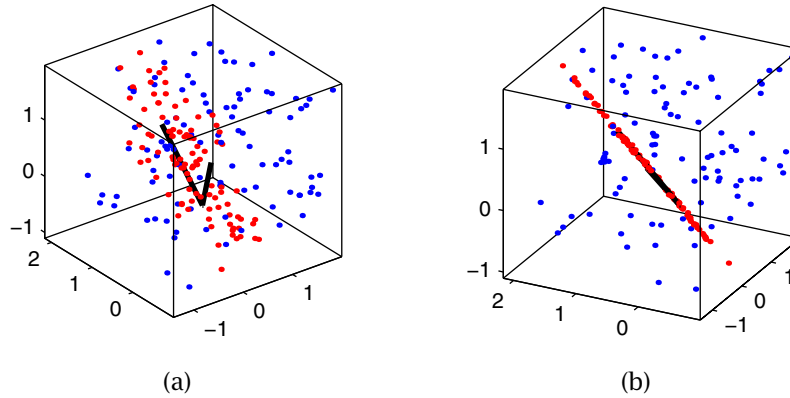
Again, we add a bit of depth by turning on the frame box.

Now, did it work? It's difficult to tell from **Figure 6b**. However, there are two things you can do to convince yourself that the projection was indeed onto the plane spanned by  $\mathbf{a}_1 = (1, 1, 0)^T$  and  $\mathbf{a}_2 = (0, 1, 1)^T$ . First, draw the vectors  $\mathbf{a}_1 = (1, 1, 0)^T$  and  $\mathbf{a}_2 = (0, 1, 1)^T$  on your plot with the following commands.

```
>> line([0,1],[0,1],[0,0],'linewidth',2,'color','k')
>> line([0,0],[0,1],[0,1],'linewidth',2,'color','k')
```

The `line` command allows you to add a plot to your graph without refreshing the figure window. The vectors  $\mathbf{a}_1$  and  $\mathbf{a}_2$  are shown in [Figure 7a](#).

Now, if you press the rotation tool in the figure window, you can experiment with different rotations. In [Figure 7b](#), we used the mouse to rotate the figure into position having azimuth  $-63$  and elevation  $26$ . You can set this without the mouse by using the command `view([-63, 26])`. Note that the vectors  $\mathbf{a}_1$  and  $\mathbf{a}_2$  have disappeared in [Figure 7b](#), covered by the projected points in the plane. This is convincing visual evidence that we have projected onto the plane spanned by the vectors  $\mathbf{a}_1$  and  $\mathbf{a}_2$ .



**Figure 7** Rotating the figure to see the plane spanned by the vectors  $\mathbf{a}_1$  and  $\mathbf{a}_2$ .

## 2 Least Squares

It is not an uncommon occurrence in the laboratory that you are called upon to fit some sort of line or curve to a set of collected data. Because measurements in the lab are always approximations at best, you really don't expect the fitted curve to pass through each of the data points. Rather, the scientist is usually content with finding a line or curve that passes as close to the data points as possible. Let's start with a small, albeit contrived, example.

### Example 1

Find a line of "best fit" in a least squares sense for the data in [Table 1](#).

$x$	0	1	2
$y$	6	0	0

**Table 1** A collection of data points.

You probably have two immediate questions that you would like to ask. First, just what is a line of "best fit," and secondly, what does the phrase "in a least squares sense" mean? Be patient, as both of these questions will be addressed in this example.

First, plot the data given in [Table 1](#), as shown in [Figure 8](#).

Well, it's pretty clear that the points in [Figure 8](#) are not collinear, so we won't be able to draw a line through them. But let's make an attempt to do so anyway.

The equation of a line in the plane is well known. It is  $y = mx + b$ . Substitute each of the points in [Table 1](#) into this equation.

## Least Squares

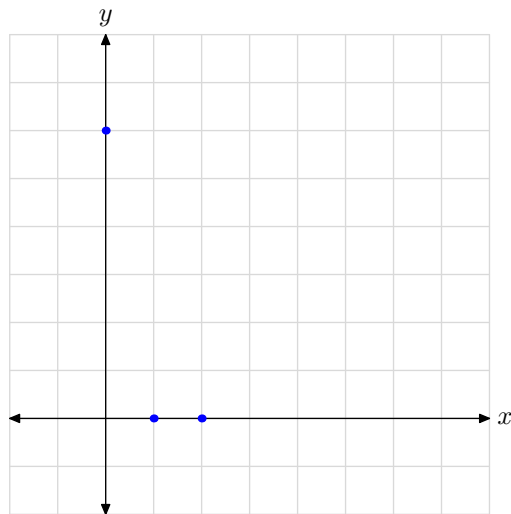


Figure 8 Plotting the data from Table 1.

$$6 = m(0) + b$$

$$0 = m(1) + b$$

$$0 = m(2) + b$$

This system of equations is *overdetermined* — there are more equations than there are unknowns. It is a rare occurrence when an overdetermined system has solutions, simply because there are too many constraints on the variables. But, let's try to solve the system anyway. Set up the augmented matrix and reduce.

```
>> M=[0 1 6;1 1 0;2 1 0]
```

```
M =
```

```
    0    1    6
    1    1    0
    2    1    0
```

```
>> R=rref(M)
```

```
R =
```

```
    1    0    0
    0    1    0
    0    0    1
```

Sure enough, since the third row of  $R$  represents the equation  $0m + 0b = 1$ , the system is inconsistent and has no solutions.

The system can be written nicely in matrix form.

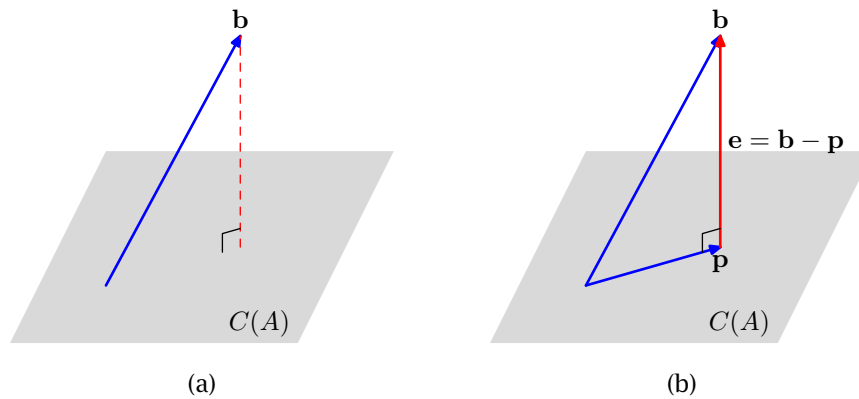
$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix}$$

Note that the system now has the form  $A\mathbf{x} = \mathbf{b}$ , where

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} m \\ b \end{pmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix}$$

## Least Squares

Because the system has no solution, this means that  $\mathbf{b}$  cannot be written as a linear combination of the columns of  $A$ . That is,  $\mathbf{b}$  does not reside in the column space of  $A$ . For emphasis, we repeat an earlier set of pictures in **Figure 9**.



**Figure 9** Projecting vector  $\mathbf{b}$  onto the column space of  $A$ .

We are currently faced with the situation depicted in **Figure 9a**. Our vector  $\mathbf{b}$  does not lie in the column space of  $A$ , so the system  $A\mathbf{x} = \mathbf{b}$  has no solutions. What are we to do? **Figure 9b** holds the answer to our question. Since  $\mathbf{b}$  does not reside in the column space of  $A$ , there is no hope of finding a solution of  $A\mathbf{x} = \mathbf{b}$ . However, the projection  $\mathbf{p}$  vector *does* lie in the column space of  $A$ , so we *can* find a solution of  $A\hat{\mathbf{x}} = \mathbf{p}$ . Furthermore, this solution will minimize the “error vector”  $\mathbf{e} = \mathbf{b} - \mathbf{p}$ .

Since  $A\mathbf{x} = \mathbf{b}$  does not have a solution, we will “settle” for a solution of  $A\hat{\mathbf{x}} = \mathbf{p}$ . Remember, we know that the error vector  $\mathbf{e} = \mathbf{b} - \mathbf{p}$  is orthogonal to every vector in the column space. Again, this means that  $\mathbf{e} = \mathbf{b} - \mathbf{p}$  is orthogonal to each of the columns of  $A$ , which, in turn, means that  $\mathbf{e} = \mathbf{b} - \mathbf{p}$  is orthogonal to each of the rows of  $A^T$ . Thus,

$$\begin{aligned} A^T(\mathbf{b} - \mathbf{p}) &= \mathbf{0}, \\ A^T(\mathbf{b} - A\hat{\mathbf{x}}) &= \mathbf{0}, \\ A^T\mathbf{b} - A^TA\hat{\mathbf{x}} &= \mathbf{0}, \\ A^TA\hat{\mathbf{x}} &= A^T\mathbf{b}. \end{aligned}$$

The system of equations  $A^TA\hat{\mathbf{x}} = A^T\mathbf{b}$  are called the *normal equations*. Note that it is easy to generate the normal equations. Simply take the equation that has no solution,  $A\mathbf{x} = \mathbf{b}$ , multiply both sides on the left by  $A^T$ , then replace  $\mathbf{x}$  with  $\hat{\mathbf{x}}$ .

We can find  $\hat{\mathbf{x}}$  by solving the system  $A^TA\hat{\mathbf{x}} = A^T\mathbf{b}$ . Matlab can do this easily. Start by entering the matrix  $A$  and the vector  $\mathbf{b}$ .

```
>> A=[0 1;1 1;2 1]
A =
     0     1
     1     1
     2     1
>> b=[6;0;0]
b =
     6
     0
     0
```

Set up the augmented matrix  $[A^TA, A^T\mathbf{b}]$ .

## Least Squares

```
>> M=[A.'*A,A.'*b]
M =
     5     3     0
     3     3     6
```

Reduce.

```
>> R=rref(M)
R =
     1     0    -3
     0     1     5
```

Thus, the solution of  $A^T A \hat{\mathbf{x}} = A^T \mathbf{b}$  is  $\hat{\mathbf{x}} = (-3, 5)^T$ . Note that  $\hat{\mathbf{x}}$  can also be computed with the computation  $\hat{\mathbf{x}} = (A^T A)^{-1} A^T \mathbf{b}$ .

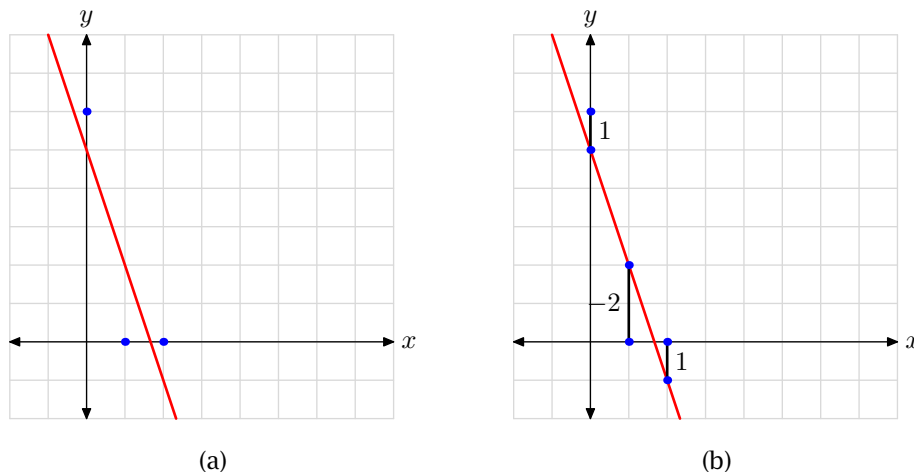
```
>> xhat=inv(A.'*A)*A.'*b
xhat =
   -3.0000
    5.0000
```

Since no solution of  $A\mathbf{x} = \mathbf{b}$  exists, we can accept  $\hat{\mathbf{x}}$  as an approximate solution of the system  $A\mathbf{x} = \mathbf{b}$ , if we keep in mind that  $A\hat{\mathbf{x}}$  actually equals the projection vector  $\mathbf{p}$ . Thus,

$$\mathbf{x} \approx \hat{\mathbf{x}},$$

$$\begin{pmatrix} m \\ b \end{pmatrix} \approx \begin{pmatrix} -3 \\ 5 \end{pmatrix}.$$

Of course, this gives us the equation of our line. Because  $m = -3$  and  $b = 5$ , the equation of the line of “best fit” is  $y = -3x + 5$ . It is instructive to superimpose the graph of  $y = -3x + 5$  on the plot of the data from [Table 1](#). This is shown in [Figure 10a](#).



**Figure 10** Estimating with the line of “best fit.”

It is equally instructive to examine the error made in approximating the data with the line of best fit. This is shown in [Figure 10b](#). The  $y$ -values of the original data points are 6, 0, and 0. These  $y$ -values are paired with the  $x$ -values 0, 1, and 2, and are also contained in the vector  $\mathbf{b} = (6, 0, 0)^T$ . Note, however, the points on the line with the same

## Least Squares

$x$ -coordinates. These points have  $\hat{y}$ -values 5, 2, and  $-1$ , respectively. Use Matlab to compute the projection vector  $\mathbf{p} = A\hat{\mathbf{x}}$ .

```
>> p=A*xhat
p =
    5.0000
    2.0000
   -1.0000
```

Thus,  $\mathbf{p}$  contains the values of  $\hat{y}$ , the values predicted by the line of best fit.

But there's even more! Note the errors made. At  $x = 0$ , the data point has  $y = 6$ , and the point on the line with  $x = 0$  has  $\hat{y} = 5$ . The error is  $y - \hat{y} = 6 - 5 = 1$ , which is clearly marked in [Figure 10b](#). Similarly, at  $x = 1$ ,  $y - \hat{y} = 0 - 2 = -2$ , and at  $x = 2$ ,  $y - \hat{y} = 0 - (-1) = 1$ . These errors are easily calculated by noting that  $\mathbf{e} = \mathbf{b} - \mathbf{p}$ .

```
>> e=b-p
e =
    1.0000
   -2.0000
    1.0000
```

This is precisely why we have been calling  $\mathbf{e}$  the “error vector.” However, one question still remains. Why have we been calling this technique a “least squares” fit? We need to examine the total error to answer this question.

Obviously, when fitting data with a line, we would like to keep the total error to a minimum. But what happens when you sum the errors made in [Figure 10b](#)?

```
>> sum(e)
ans =
    0
```

The total error is zero? This is no good. Intuitively, the total error is not zero. If that were the case, then you would think that the line would fit the data perfectly, which is definitely not the case in [Figure 10b](#). Perhaps we should take the absolute value of each error and then sum to find the total error.

```
>> sum(abs(e))
ans =
    4
```

That's better, but it is not what we've done in this activity, nor does it indicate why our method is called a “least squares” fit. What scientists and mathematicians actually do is this. They square each error before summing. This will ensure that the total error is not zero and give a good indication of the total error.

```
>> sum(e.^2)
ans =
    6
```

That is, the total squared error is  $(1)^2 + (-2)^2 + (1)^2 = 6$ . But this computation should set alarm bells ringing in our heads, as this is the square of the magnitude of  $\mathbf{e}$ .

## Least Squares

$$\begin{aligned}\|\mathbf{e}\|^2 &= \mathbf{e} \cdot \mathbf{e} \\ &= (1)^2 + (-2)^2 + (1)^2 \\ &= 6\end{aligned}$$

In **Figure 9b**, note that our technique ensures that the length of  $\mathbf{e}$  is kept to a minimum, because  $\mathbf{p}$  is the orthogonal projection of  $\mathbf{b}$  onto the column space of  $A$ . But the length of  $\mathbf{e}$  is the sum of the squares of its components, which are the errors made at each data point. Thus, our technique ensures that the sum of the squares of the errors is kept to a minimum. Hence the name, “least squares” fit.

### 3 Examples

Matlab frees the user from the intensive calculations required to find a line of best fit. Therefore, larger data sets are no problem, and Matlab’s graphics can provide nice illustrations for your laboratory assignments.

#### Example 2

Imagine you are working in the physics lab, hanging masses from a spring, then measuring the distance the spring stretches from its equilibrium position for each mass. You collect the data in **Table 2**, where  $m$  is the mass (grams) of the object suspended from the spring, and  $d$  is the distance (cm) that the spring stretches from its equilibrium position.

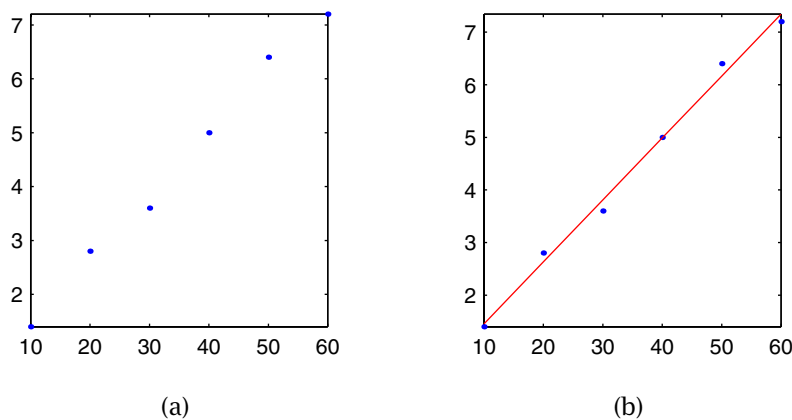
$m$	10	20	30	50	40	60
$d$	1.4	2.8	3.6	5.0	6.4	7.2

**Table 2** Spring-mass data.

Use Matlab to find a plot a line of best fit for the data in **Table 2**.

First, load and plot the data in Matlab. The following commands were used to generate the image in **Figure 11a**.

```
>> m=(10:10:60).';  
>> d=[1.4, 2.8, 3.6, 5.0, 6.4, 7.2].';  
>> plot(m,d,'*')
```



**Figure 11** Fitting a line to the spring-mass data.

## Least Squares

Note the use of the transpose operator to form column vectors. Also note that the data points in **Figure 11a** appear to follow a linear pattern.

We will fit the data to a line having equation  $d = a + bm$ . First, substitute each data point into the equation.

$$1.4 = a + b(10)$$

$$2.8 = a + b(20)$$

$$3.6 = a + b(30)$$

$$5.0 = a + b(40)$$

$$6.4 = a + b(50)$$

$$7.2 = a + b(60)$$

Set the system of equations in matrix form.

$$\begin{pmatrix} 1 & 10 \\ 1 & 20 \\ 1 & 30 \\ 1 & 40 \\ 1 & 50 \\ 1 & 60 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1.4 \\ 2.8 \\ 3.6 \\ 5.0 \\ 6.4 \\ 7.2 \end{pmatrix}$$

Note that this last matrix has the form  $A\mathbf{x} = \mathbf{d}$ , where

$$A = \begin{pmatrix} 1 & 10 \\ 1 & 20 \\ 1 & 30 \\ 1 & 40 \\ 1 & 50 \\ 1 & 60 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} a \\ b \end{pmatrix}, \quad \text{and} \quad \mathbf{d} = \begin{pmatrix} 1.4 \\ 2.8 \\ 3.6 \\ 5.0 \\ 6.4 \\ 7.2 \end{pmatrix}.$$

The vector  $\mathbf{d}$  is already entered. The second column of matrix  $A$  holds the mass data which is already stored in the vector  $\mathbf{m}$ .

```
>> A=[ones(size(m)),m]
A =
     1     10
     1     20
     1     30
     1     40
     1     50
     1     60
```

The normal equations are  $A^T A \hat{\mathbf{x}} = A^T \mathbf{d}$ . These are easily solved in Matlab. Simply set up the augmented matrix and reduce.

```
>> M=[A.'*A, A.'*d]
M =
  1.0e+003 *
    0.0060    0.2100    0.0264
    0.2100    9.1000    1.1300
>> R=rref(M)
```

## Least Squares

```
R =
    1.0000    0    0.2800
         0    1.0000    0.1177
```

Thus,  $\hat{\mathbf{x}} = (0.2800, 0.1177)^T$  and  $\mathbf{x} = (a, b)^T \approx (0.2800, 0.1177)^T$ . We can use this intercept and slope to plot the equation of the line of best fit. These commands were used to generate the line of best fit in [Figure 11b](#).

```
>> yhat=0.2800+0.1177*m;
>> plot(m,d,'*',m,yhat,'r')
```

### Example 3

In a second lab activity, a toy rocket is fired into the air. The height of the rocket at fixed times is recorded in [Table 3](#). The time  $t$  is recorded in seconds, the height  $s$  in meters.

$t$	5	10	15	20	25	30
$s$	722	1073	1178	1117	781	102

**Table 3** Rocket data (time and height)

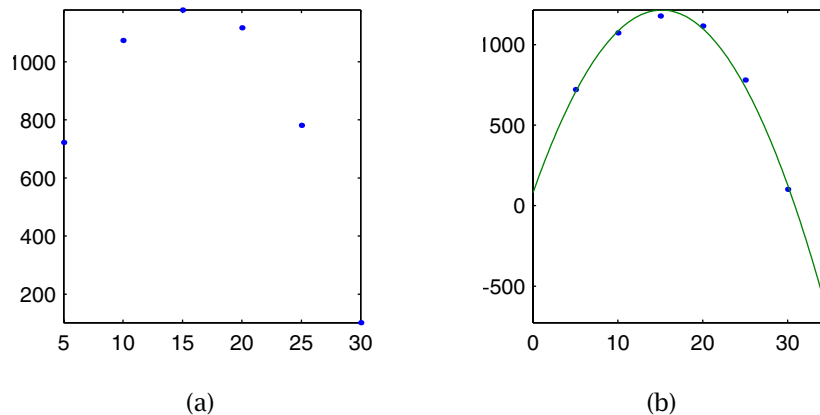
Examine the data then decide on an appropriate model to fit to the data. Use the least squares technique to minimize the error.

Start by entering the data in column vectors  $\mathbf{t}$  and  $\mathbf{s}$ .

```
>> t=(5:5:30).';
>> s=[722, 1073, 1178, 1117, 781, 102].'
```

Note that the transpose operator is used to generate column vectors. The image in [Figure 12a](#) was generated with the following command.

```
>> plot(t,s,'.')
```



**Figure 12** Fitting a parabola to the rocket data of [Table 3](#).

Note that the data forms an outline for an inverted parabola. Let's try to fit the data to the equation  $s = a + bt + ct^2$ . Substitute the data from [Table 3](#) into the equation  $s = a + bt + ct^2$ .

## Least Squares

$$\begin{aligned}
 722 &= a + b(5) + c(5)^2 \\
 1073 &= a + b(10) + c(10)^2 \\
 1178 &= a + b(15) + c(15)^2 \\
 1117 &= a + b(20) + c(20)^2 \\
 781 &= a + b(25) + c(25)^2 \\
 102 &= a + b(30) + c(30)^2
 \end{aligned}$$

Set this system of equations in matrix form.

$$\begin{pmatrix} 1 & 5 & (5)^2 \\ 1 & 10 & (10)^2 \\ 1 & 15 & (15)^2 \\ 1 & 20 & (20)^2 \\ 1 & 25 & (25)^2 \\ 1 & 30 & (30)^2 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 722 \\ 1073 \\ 1178 \\ 1117 \\ 781 \\ 102 \end{pmatrix}$$

Note that this last equation has the form  $A\mathbf{x} = \mathbf{s}$ , where

$$A = \begin{pmatrix} 1 & 5 & (5)^2 \\ 1 & 10 & (10)^2 \\ 1 & 15 & (15)^2 \\ 1 & 20 & (20)^2 \\ 1 & 25 & (25)^2 \\ 1 & 30 & (30)^2 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \quad \text{and} \quad \mathbf{s} = \begin{pmatrix} 722 \\ 1073 \\ 1178 \\ 1117 \\ 781 \\ 102 \end{pmatrix}.$$

The vector  $\mathbf{s}$  is already entered. The second column of matrix  $A$  holds the time data which is already stored in the vector  $\mathbf{t}$ . The third column of the matrix  $A$  contains the square of each entry in the vector  $\mathbf{t}$ .

```

>> A=[ones(size(t)), t, t.^2]
A =
     1     5    25
     1    10   100
     1    15   225
     1    20   400
     1    25   625
     1    30   900
    
```

The normal equations are  $A^T A \hat{\mathbf{x}} = A^T \mathbf{s}$ . These are easily solved in Matlab. Simply set up the augmented matrix and reduce.

```

>> M=[A.'*A, A.'*s]
M =
      6      105      2275      4973
     105     2275     55125     76935
     2275     55125    1421875    1417125
>> R=rref(M)
R =
      1         0         0      80.2
      0         1         0     149.78
      0         0         1    -4.9385
    
```

## Least Squares

Thus,  $\hat{\mathbf{x}} = (80.2, 149.78, -4.9385)^T$  and  $\mathbf{x} = (a, b, c)^T \approx (80.2, 149.78, -4.9385)^T$ . We can use these coefficients to plot the parabola of best fit. A parabola is a smooth curve. Consequently, we will want to use a lot of points to draw the graph of the parabola. We were also curious about two things: (1) what was the rocket's initial height, and (2) at what time did the rocket return to ground level. This inspired us to extend the time interval to take these points into consideration. With these thoughts in mind, the following commands were used to generate the image in [Figure 12b](#).

```
>> tt=linspace(0,35);  
>> shat=80.2+149.78*tt-4.9385*tt.^2;  
>> plot(t,s, '.',tt,shat)
```

## 4 Exercises

Here are a number of exercises designed to reinforce the ideas presented in this activity.

1. Provide a detailed proof for each part of [Proposition 1](#).
2. Provide a detailed proof for the Cauchy-Schwarz inequality in [Proposition 3](#). *Hint.* Use [Proposition 5](#).
3. Provide a detailed proof for each part of [Proposition 4](#).
4. Find the matrix  $P$  that projects all points in the plane onto the vector  $\mathbf{a} = (1, 2)^T$ . Provide an image similar to that in [Figure 4b](#) as visual evidence that you've selected the correct projection matrix.
5. Find the matrix  $P$  that projects all points in  $R^3$  onto the vector  $\mathbf{a} = (1, 1, 1)$ . Provide an image, similar to that in [Figure 4b](#), but in  $R^3$ , that shows that your projection matrix projects points in  $R^3$  onto the line passing through the origin in the direction of the vector  $\mathbf{a}$ .
6. Find the matrix  $P$  that projects all points in  $R^3$  onto the plane spanned by the vectors  $\mathbf{a}_1 = (1, 0, 0)^T$  and  $\mathbf{a}_2 = (1, 1, 1)^T$ . Provide an image, similar to that in [Figure 7b](#), that demonstrates that your projection matrix works as advertised.
7. Find a line of best fit for the data set in [Table 4](#).

$x$	5	10	15	20	25	30
$y$	28	39	48	65	72	82

**Table 4** Data for [Exercise 7](#)

Show all of your work on college ruled paper, setting up the equations, the matrix form of the system, etc. Then provide a Matlab plot of the data and the fitted line.

8. Find a parabola of best fit for the data set in [Table 5](#).

$x$	2	6	10	14	18	22
$y$	286	589	749	781	563	282

**Table 5** Data for [Exercise 8](#)

## Least Squares

Show all of your work on college ruled paper, setting up the equations, the matrix form of the system, etc. Then provide a Matlab plot of the data and the fitted parabola.

9. If each equation in your system is linear, then linear algebra can be applied to find a least squares fit. At first, trying to fit an exponential equation  $y = ae^{bx}$  to the data in **Table 6** seems doomed to failure.

$x$	1	2	3	4	5	6
$y$	128	149	214	269	336	434

**Table 6** Data for **Exercise 9**

However, taking the natural logarithm of both sides of this equation gives the following result.

$$y = ae^{bx}$$

$$\ln y = \ln ae^{bx}$$

$$\ln y = \ln a + \ln e^{bx}$$

$$\ln y = \ln a + bx$$

- Prepare a plot of  $\ln y$  versus  $x$  that reveals the linear relationship between  $\ln y$  and  $x$ .
  - Use the technique of this activity to fit a line to the transformed data in part (a).
  - Use the slope and intercept of the line in part (b) to find an exponential equation  $y = ae^{bx}$  that fits the original data. Prepare a plot containing the original data and the graph of this exponential equation.
10. Fit a power function of the form  $y = ax^b$  to the data in **Table 7**.

$x$	1	2	3	4	5	6
$y$	117	385	920	1608	2518	3611

**Table 7** Data for **Exercise 10**