KANSAS STATE UNIVERSITY | Department of Mathematics

Applied Matrix Theory - Math 551

Notes on orthogonal and orthonormal sets, and orthogonal projections.

Created by Prof. Diego Maldonado and Prof. Virginia Naibo

Let u and v be two vectors in $\mathbf{R}^m$. By means of the formula for their dot product

$$u \cdot v = \|u\|\|v\| \cos\theta,$$

where θ is the angle between u and v, we understand that the equality

$$u \cdot v = 0$$

carries a geometric meaning; namely, the fact that u and v are *orthogonal* to each other.

Next, we say that a set of vectors $\{v_1, v_2, \ldots, v_r\}$ is an *orthogonal set* if its elements are mutually orthogonal, that is,

$$v_i \cdot v_j = 0 \quad \text{for all } i, j = 1, \ldots, r \text{ with } i \neq j. \tag{1}$$

Among the nice properties of orthogonality we have that *orthogonality guarantees linear independence*. That is, if a set of non-zero vectors $\{v_1, v_2, \ldots, v_r\}$ is an orthogonal set, then it is a linear independent set. This can be seen quite easily by using the definition of linear independence: Suppose that there are real numbers $x_1, x_2, \ldots, x_r$ such that

$$\sum_{i=1}^{r} x_i v_i = x_1 v_1 + x_2 v_2 + \cdots + x_r v_r = 0 \in \mathbf{R}^m, \tag{2}$$

we will deduce that all the x_i's must be zero. In order to do so, let's fix any j (between 1 and r) and let's compute the dot product of v_j and the sum in (2), that is,

$$v_j \cdot \left(\sum_{i=1}^{r} x_i v_i \right) = (v_j \cdot 0) = 0 \in \mathbf{R}. \tag{3}$$

But

$$v_j \cdot \left(\sum_{i=1}^{r} x_i v_i \right) = \sum_{i=1}^{r} (v_j \cdot x_i v_i) = \sum_{i=1}^{r} x_i (v_j \cdot v_i). \tag{4}$$

Consequently,

$$0 = \sum_{i=1}^{r} x_i (v_j \cdot v_i).$$

But now we use (1) to see that $\sum_{i=1}^{r} x_i(v_j \cdot v_i) = x_j(v_j \cdot v_j)$, because (1) says that all the dot products $v_j \cdot v_i$, with $i \neq j$, vanish. Putting things together we get that

$$x_j(v_j \cdot v_j) = 0$$

and, since $v_j \cdot v_j = \|v_j\|^2 > 0$ (because we are assuming that none of the v_j's is the zero vector), we conclude that $x_j = 0$. Since j was arbitrary we get $x_j = 0$ for every $j = 1, \dots r$ and the vectors $v_1, v_2, \dots, v_r$ are linearly independent.

We say that the set of vectors $\{v_1, v_2, \dots, v_r\}$ is an *orthonormal set* if it is an orthogonal set (that is, its elements are mutually orthogonal) and if all its elements have norm equal to one. In other words,

Orthonormal set = Orthogonal set + All vectors have norm equal to one.

That is, an orthonormal set of vectors $\{v_1, v_2, \dots, v_r\}$ satisfies (1) as well as the equalities

$$\|v_j\| = 1 \quad \text{for all } j = 1, \dots, r. \tag{5}$$

Notice that, since $\|v_j\|^2 = v_j \cdot v_j$, we can rephrase (5) as

$$v_j \cdot v_j = 1 \quad \text{for all } j = 1, \dots, r. \tag{6}$$

Example 1. The simple Matlab function below ("testingorthnorm.m") tells us whether a set of vectors is orthonormal or not. Study it.

```
function out=testingorthnorm(V)
[m n]=size(V);
I=eye(n);
%that is, V contains, as its columns, n vectors in R^m
for i=1:n
    for j=1:n
        D(i,j)= dot(V(:,i),V(:,j));
    end
end

if D==I
    disp('The entered columns form an orthonormal set.')
    out=1;
else
    disp('The entered columns do not form an orthonormal set.')
    out=0;
end
```

Given a non-zero vector $u \in \mathbf{R}^m$, set $\mathcal{U} := \mathit{span}\{u\}$, that is, $\mathcal{U}$ is the straight line in the direction of u passing through the origin. We have seen how to project any vector $v \in \mathbf{R}^m$ onto $\mathcal{U}$ by introducing

$$\overline{v} := \left(\frac{v \cdot u}{u \cdot u}\right) u. \tag{7}$$

The vector $\overline{v}$ is then called *the orthogonal projection of v onto span$\{u\}$*. Also, setting $w := v - \overline{v}$ we see that $w \cdot u = 0$ and we have therefore decomposed v as

$$v = \overline{v} + w \tag{8}$$

where $\overline{v}$ belongs to $\mathcal{U}$ and w is orthogonal to $\mathcal{U}$, that is, $w \in \mathcal{U}^{\perp}$. Remember that, for any set $S \subset \mathbf{R}^m$, the linear subspace $S^{\perp}$ denotes the collection of all the vectors in $\mathbf{R}^m$ which are orthogonal to all the vectors in S.

The importance of the decomposition (8) is that $\overline{v}$ represents "exactly how much of v lives in the direction of u" or "exactly how much of u there is in v". Also, among all the vectors in $\mathcal{U}$, $\overline{v}$ minimizes the distance to v. Such shortest distance (or just "distance") amounts exactly to $\|w\|$.

We will now extend all this to higher dimensions. Let now $\mathcal{U}$ denote a linear subspace of $\mathbf{R}^m$ and let $\{u_1, u_2, \ldots, u_n\}$ denote an <u>orthogonal</u> basis for $\mathcal{U}$ (notice that we must have $n \leq m$). Let v be a vector in $\mathbf{R}^m$. The *orthogonal projection of v onto $\mathcal{U}$*, denoted by $\overline{v}$, can be computed by doing

$$\overline{v} = \sum_{j=1}^{n} \left(\frac{v \cdot u_j}{u_j \cdot u_j}\right) u_j. \tag{9}$$

Moreover, if $\{u_1, u_2, \ldots, u_n\}$ is an <u>orthonormal</u> basis for $\mathcal{U}$ we can simplify (9) to

$$\overline{v} = \sum_{j=1}^{n} (v \cdot u_j) u_j, \tag{10}$$

because in this case we also have $u_j \cdot u_j = \|u_j\|^2 = 1$ for every $j = 1, \ldots, n$. It's important to mention that the formulas (9) and (10) work for orthogonal and orthonormal bases, respectively, and they <u>do not work with arbitrary bases</u>. Also, a simple computation using the definition of transpose and matrix multiplication shows that we can recast (10), in matrix form, as follows

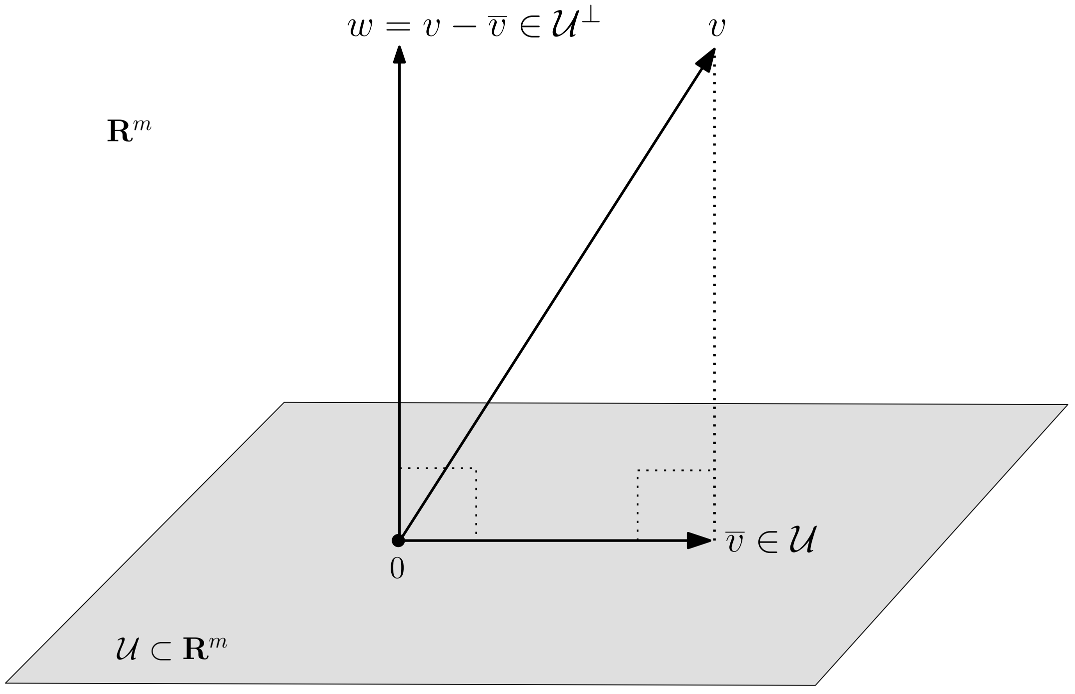
$$\overline{v} = UU^t v, \tag{11}$$

where the matrix U has, as its columns, the vectors $u_1, u_2, \ldots, u_n$. The orthogonal projection of v onto $\mathcal{U}$ is illustrated in Figure 1

Again, setting $w := v - \overline{v}$ we see that $w \cdot u = 0$ for every $u \in \mathcal{U}$ and we can decompose v as

$$v = \overline{v} + w \tag{12}$$

where $\overline{v}$ belongs to $\mathcal{U}$ and w is orthogonal to $\mathcal{U}$, that is, $w \in \mathcal{U}^\perp$. Also, among all the vectors in $\mathcal{U}$, $\overline{v}$ minimizes the distance to v and that *minimum distance equals* $\|w\|$. Similarly, $\overline{v}$ represents "exactly how much of v lives in $\mathcal{U}$".


$w = v - \overline{v} \in \mathcal{U}^\perp$
v
$\mathbf{R}^m$
0
$\overline{v} \in \mathcal{U}$
$\mathcal{U} \subset \mathbf{R}^m$

$\mathcal{U}$ is a subspace of $\mathbf{R}^m$ and let $\{u_1, u_2, \ldots, u_n\}$ be an orthonormal basis for $\mathcal{U}$. The orthogonal projection of v over $\mathcal{U}$, denoted by $\overline{v}$, is given by

$$\overline{v} = \sum_{j=1}^{n} (v \cdot u_j)\, u_j$$

Figure 1: The orthogonal projection of v onto $\mathcal{U}$ computed with an orthonormal basis of $\mathcal{U}$

At this point we start to understand the advantages of working with orthonormal bases as they allow for quite simple formulas for the objects of interest (such as $\overline{v}$). A good question is then, how do we get orthonormal bases? The answer is: *the Gram-Schmidt orthonormalization process*. This is a process that "straightens out" a given basis. That is, from a (non-necessarily orthonormal) basis the process yields an orthonormal basis for the same linear subspace. The Gram-Schmidt process will be discussed in the lectures and in the lab sessions, but here is the bottom line: the Matlab command "orth".

Example 2. Given the vectors

$$v = \begin{bmatrix} 2 \\ 1 \\ 1 \\ -1 \end{bmatrix}, \quad v_1 = \begin{bmatrix} 1 \\ 2 \\ -2 \\ 4 \end{bmatrix}, \quad \text{and} \quad v_2 = \begin{bmatrix} 3 \\ -1 \\ 6 \\ 5 \end{bmatrix},$$

find the distance from v to the subspace of $\mathbf{R}^4$ generated by v_1 and v_2.

Let's write $\mathcal{U} = span\{v_1, v_2\}$. Then the goal is to find the distance from v to $\mathcal{U}$. First off, if the vector v happens to actually be in $\mathcal{U}$ then we're done since the desired distance is zero! Now, by definition of $\mathcal{U}$, v belongs in $\mathcal{U}$ if and only if it is a linear combination of v_1 and v_2. Is that the case? Let's do

```
>> v=[2 1 1 -1]'
v =
     2
     1
     1
    -1
>> v1=[1 2 -2 4]'
v1 =
     1
     2
    -2
     4
>> v2=[3 -1 6 5]'
v2 =
     3
    -1
     6
     5
>> rref([v1 v2 v])
ans =
     1     0     0
     0     1     0
     0     0     1
     0     0     0
```

The third row of $rref([v_1\, v_2\, v])$ tells us that v is not a linear combination of v_1 and v_2, so we cannot take that shortcut. We need to find $\overline{v}$, then $w = v - \overline{v}$ and the desired distance will be $\|w\|$. We would like to use the formulas (9) or (10), but they only work with orthogonal or orthonormal vectors. Now the question is: Is the set $\{v_1, v_2\}$ orthogonal?

In order to answer that question we quickly do

```
>> dot(v1,v2)

ans =

     9
```

and realize that the vectors v_1 and v_2 are not orthogonal (let alone orthonormal...), which prevents us from using the formula (9) (let alone (10)). As promised, now enters the command "orth". Let's do

```
>> V=[v1 v2]

V =

     1     3
     2    -1
    -2     6
     4     5

>> U=orth(V)

U =

    0.3675    0.0883
   -0.0718    0.4455
    0.6480   -0.6376
    0.6632    0.6222
```

We claim that the columns of the matrix U above form an orthonormal basis for $\mathcal{U}$ (remember that $\mathcal{U} = span\{v_1, v_2\} = col(V)$). First, the orthonormality of the columns of U is easily seen from the following

```
>> U'*U

ans =

    1.0000    0.0000
    0.0000    1.0000
```

That is, $U^t U = I$, which tells us that the columns of U are orthogonal to each other (hence the 0's) and that they all have norm 1 (hence the 1's). Ok, orthonormality checks. But, do the columns of U form a basis for $\mathcal{U} = span\{v_1, v_2\} = col(V)$? We can quickly answer that by doing

```
>> rref([U v1])

ans =

    1.0000         0    1.5809
         0    1.0000    4.7435
         0         0         0
         0         0         0
```

and

```
>> rref([U v2])

ans =

    1.0000         0    8.3785
         0    1.0000   -0.8950
         0         0         0
         0         0         0
```

That is, both, v_1 and v_2, are linear combinations of the columns of U. Hence, the columns of U span both v_1 and v_2 and, therefore, they span all of possible linear combinations of them. That is, $col(U) = col(V)$. Finally, since the columns of U form an orthonormal basis for $col(U)$ and we just got that $col(U) = col(V)$, then the columns of U form an orthonormal basis for $col(V)$, as claimed.

Wonderful. Now, if we label the columns of U as u_1 and u_2, we can find $\overline{v}$ using formula (10) with the vectors u_1 and u_2! That is, we couldn't use V, we did $U = orth(V)$ and we're ready to use (10) with the columns of U.

Let's use (10) then

```
>> u1=U(:,1)

u1 =

    0.3675
```

```
   -0.0718
    0.6480
    0.6632

>> u2=U(:,2)

u2 =

    0.0883
    0.4455
   -0.6376
    0.6222

>> vbar=dot(v,u1)*u1 + dot(v,u2)*u2

vbar =

    0.1818
   -0.3306
    0.8264
    0.0331
```

By the way, notice that we can get to the same result by using the formula (11)

```
>> U*U'*v
ans =
    0.1818
   -0.3306
    0.8264
    0.0331
```

Finally, let's do

```
>> w=v-vbar

w =

    1.8182
    1.3306
    0.1736
   -1.0331
```

```
>> norm(w)

ans =

    2.4847
```

to find that the distance from v to the linear subspace generated by v_1 and v_2 equals 2.4847 (units of length).

An application to linear regression: best linear fit.

Consider the following matrix M whose columns x and y are vectors in $\mathbf{R}^{10}$

```
>> M=[x,y]

M =

    7.8025   25.8771
    3.8974   14.3985
    2.4169   10.8931
    4.0391   14.1482
    0.9645    4.9797
    1.3197    6.2972
    9.4205   31.5597
    9.5613   32.1475
    5.7521   20.5517
    0.5978    4.6952
```

Let's plot y vs. x by doing

```
>> plot(x,y,'*')
```

and producing the picture in Figure 2. We think of the data matrix M as containing 10 measurements of two parameters (x and y) and notice that the plotted measurements are "more or less" aligned. This indicates an "almost" linear dependence of y with respect to x.

If the parameter y were actually a linear (or affine) function of x, then we would have

$$y_j = m\,x_j + c \qquad \text{for every } j = 1, 2, \ldots, 10, \tag{13}$$

for some constants m (the slope) and c (the y-intercept). Let's introduce the vector

$$s := \begin{bmatrix} m \\ c \end{bmatrix}.$$

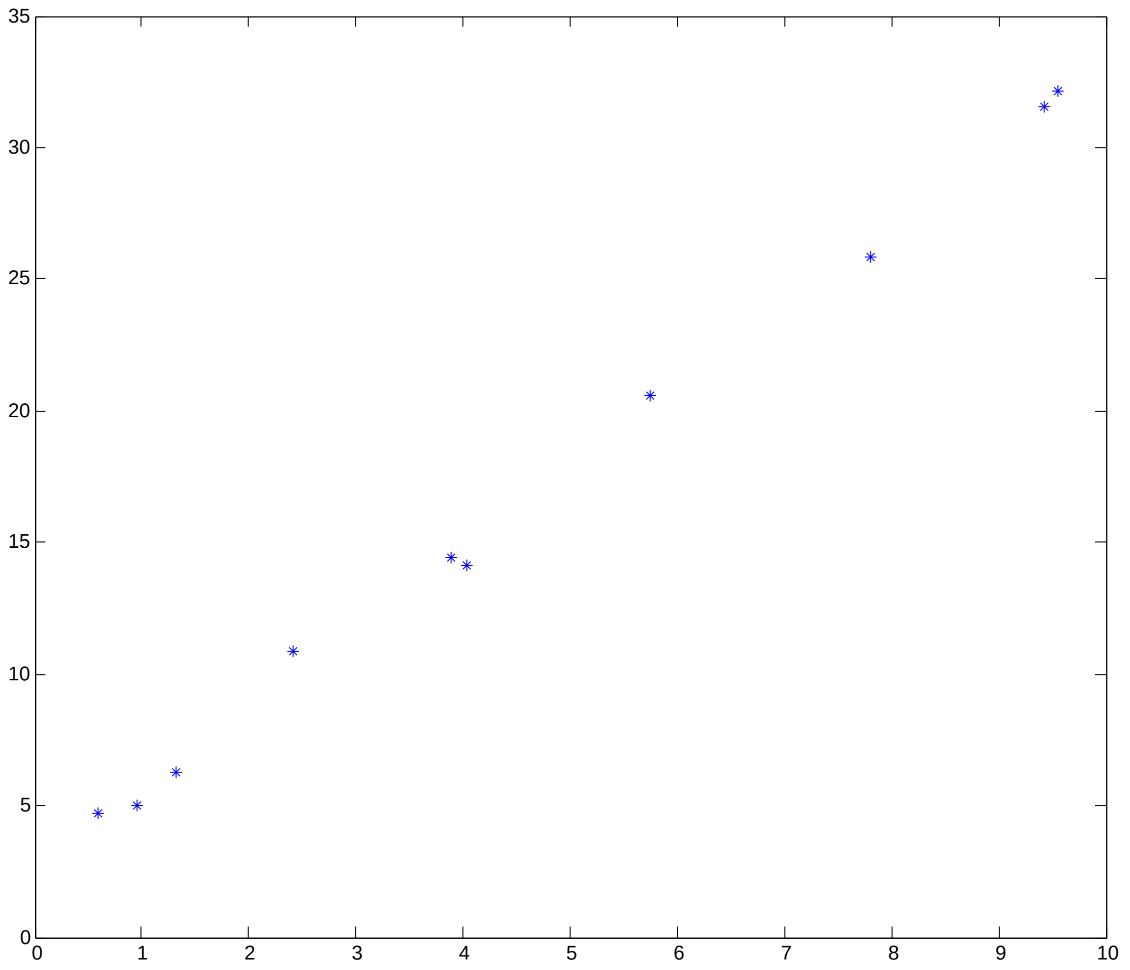


Figure 2: Plot of y vs x

Then, the equations in (13) can be written, in matrix notation, as

$$y = \begin{bmatrix} 25.8771 \\ 14.3985 \\ 10.8931 \\ 14.1482 \\ 4.9797 \\ 6.2972 \\ 31.5597 \\ 32.1475 \\ 20.5517 \\ 4.6952 \end{bmatrix} = \begin{bmatrix} 7.8025 & 1 \\ 3.8974 & 1 \\ 2.4169 & 1 \\ 4.0391 & 1 \\ 0.9645 & 1 \\ 1.3197 & 1 \\ 9.4205 & 1 \\ 9.5613 & 1 \\ 5.7521 & 1 \\ 0.5978 & 1 \end{bmatrix} \begin{bmatrix} m \\ c \end{bmatrix} =: A\,s, \tag{14}$$

where we have put

$$A = \begin{bmatrix} 7.8025 & 1 \\ 3.8974 & 1 \\ 2.4169 & 1 \\ 4.0391 & 1 \\ 0.9645 & 1 \\ 1.3197 & 1 \\ 9.4205 & 1 \\ 9.5613 & 1 \\ 5.7521 & 1 \\ 0.5978 & 1 \end{bmatrix}.$$

That is, (13) can be written as $As = y$ and then y is a linear (affine) function of x if and only if the system $As = y$ has a solution. But just looking at the matrix of coefficients A we realize that there are way too many equations for the unknowns m and c and we can expect inconsistency. Let's corroborate this by doing

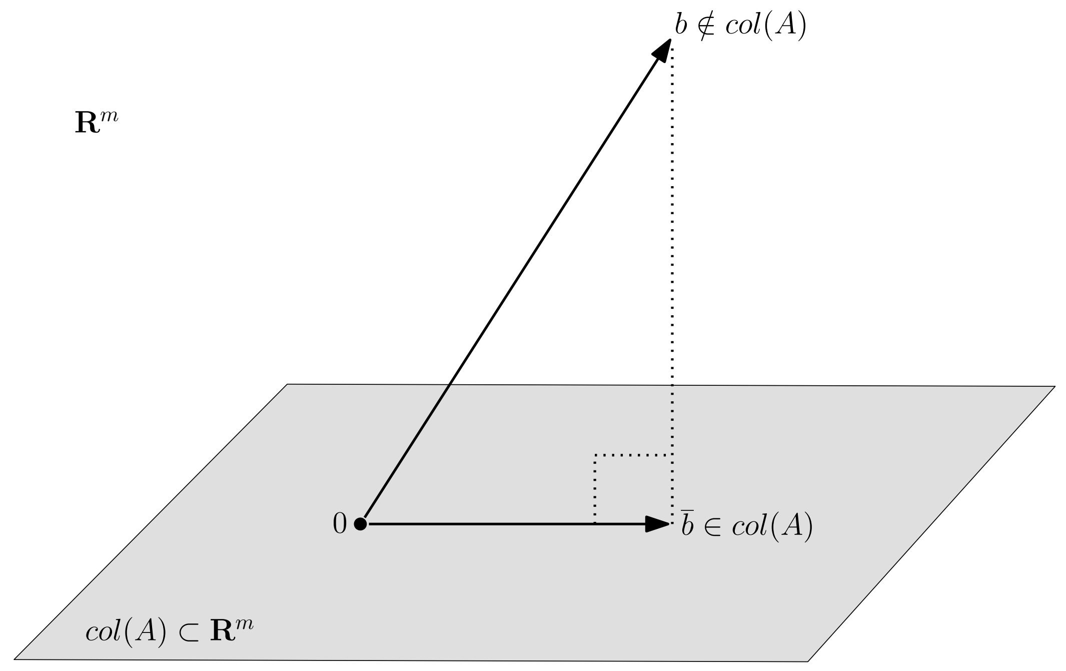
```
>> rref([A y])
ans =
     1     0     0
     0     1     0
     0     0     1
     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

where the third row tells us that the system $As = y$ is inconsistent, that is, y is not in $col(A)$.

Consequently, the parameter y is not a linear function of x (because there are no numbers m and c that verify (13)). That is, there is no straight line going through all the points in Figure 2, there is no perfect fit. Now, what is the second best thing? That is, among all the possible straight lines in the plane, which one is "closest" to all the points in Figure 2? In order to answer that question let's digress a bit.

"Almost solving" linear systems

Let's go back to our usual notation for linear systems, that is, $Ax = b$ (this x has nothing to do with the first column of the matrix M above, remember that we're digressing). Our basic matrix theory tells us that $Ax = b$ is a consistent system if and only if $b \in col(A)$. Suppose that we know before hand that $b \notin col(A)$, let's try to make sense of the concept "almost solving" $Ax = b$. The method is quite simple: We project b onto $col(A)$ to get $\bar{b}$ and then solve $Ax = \bar{b}$. This is illustrated in Figure 3



Recall that a system $Ax = b$ is consistent if and only if $b \in col(A)$.

With b as in the picture we cannot solve $Ax = b$, but we can certainly solve $Ax = \bar{b}$.

Among all the vectors in $col(A)$, $\bar{b}$ is the closest to b.

Figure 3: "Almost solving' an inconsistent system.

Obviously, the vector x obtained this way won't be a solution to $Ax = b$ (because those don't exist), but it will be "the best we can do" in the sense that we are solving with the right-hand side vector $\bar{b}$ that is closest to b.

Let's write the Matlab steps to "almost solve" $Ax = b$ with $b \notin col(A)$.

(a) Find an orthonormal basis for $col(A)$ by doing

```
>> U=orth(A);
```

(b) Find the orthogonal projection of b onto $col(A)$ by, say, using (11),

```
bbar=U*U'*b;
```

(c) Finally, solve $Ax = \bar{b}$,

```
>> rref([A bbar])
```

By the way, notice that if we apply the algorithm above to a vector b which actually belongs to $col(A)$, then, due to the fact that in that case $\bar{b} = b$ (b is its own projection), "almost solving" is the same as "solving". Moral: "almost solving" always works and it renders the same results as "solving" whenever "solving" is possible.

Back to linear regression

Let's go back to our data matrix M. We know that we can't solve $As = y$ because $y \notin col(A)$. Let's "almost solve" then, by doing

```
>> U=orth(A) %finding an orthonormal basis for col(A)

U =
   -0.4388    0.0807
   -0.2233   -0.2336
   -0.1415   -0.3527
   -0.2311   -0.2222
   -0.0614   -0.4696
   -0.0810   -0.4410
   -0.5281    0.2109
   -0.5359    0.2223
   -0.3256   -0.0843
   -0.0411   -0.4991
```

```
>> ybar=U*U'*y %finding the projection of y onto col(A)

ybar =
   26.4634
   14.4663
    9.9181
   14.9018
    5.4563
    6.5475
   31.4340
   31.8667
   20.1642
    4.3296

>> rref([A ybar]) %solving As=ybar, that is, 'almost solving' the inconsistent As = y

ans =
    1.0000         0    3.0721
         0    1.0000    2.4931
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
```

That is, the line

$$y = 3.0721\, x + 2.4931 \tag{15}$$

is *the best linear fit* to the data in the matrix M.

Let's do

```
>> t=[0:.01:10];
>> z=3.0721*t+2.4931;
>> plot(x,y,'*'); hold on, plot(t,z,'r-')
```

to draw the line (15) and the initial (x, y) data, see Figure 4.

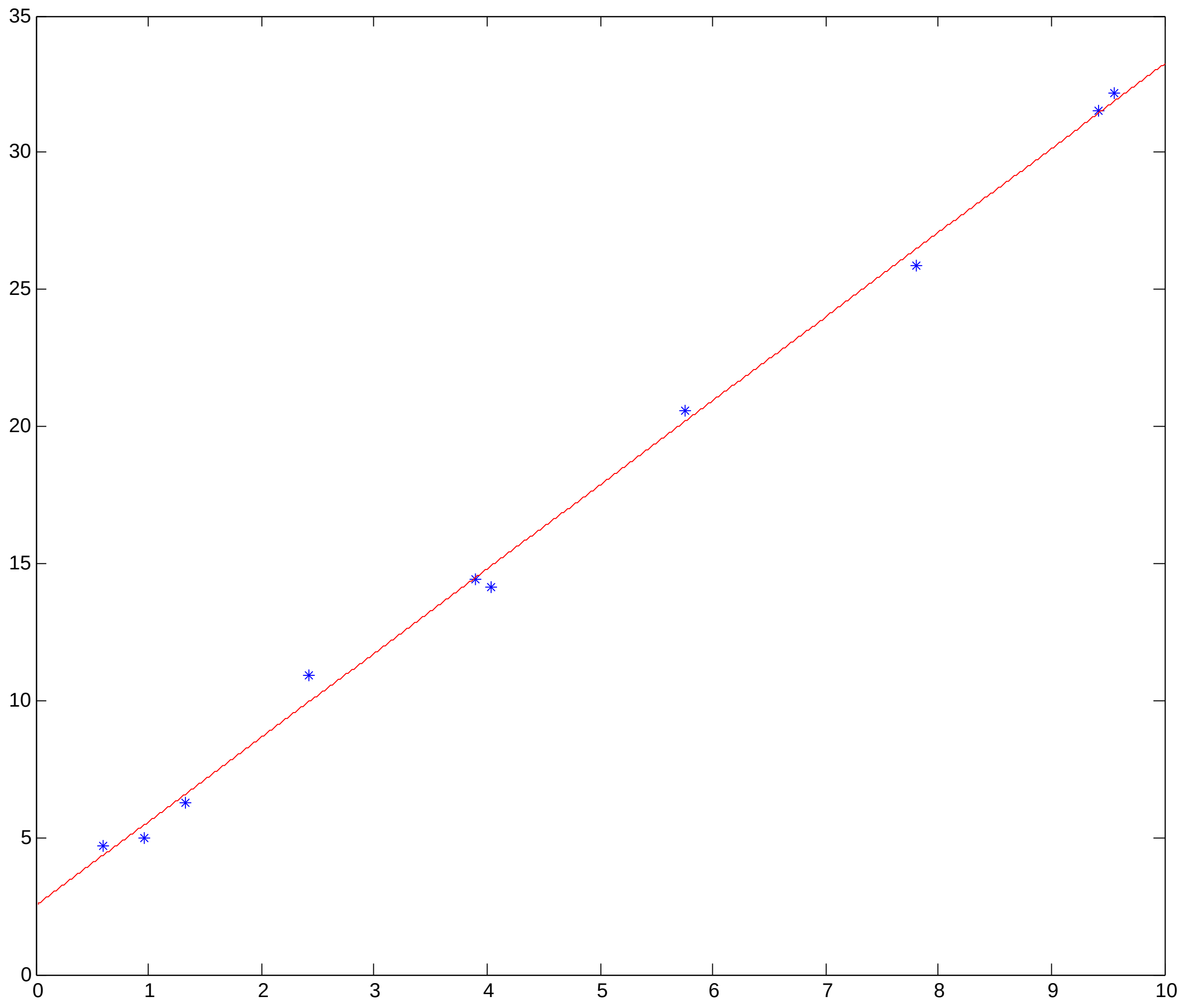


Figure 4: Best linear fit for the data matrix M.

What makes the linear function (15) the "best"? Let's zoom in on one of the (x, y) points in Figure 2, as shown in Figure 5, to appreciate the vertical distance between the data point (x, y) and the corresponding point in the best linear fit. For the j-th data point, that is, (x_j, y_j), that vertical distance equals

$$d_j := |y_j - (3.0721\, x_j + 2.4931)|$$

and it represents an error, how far off (vertically) the best linear fit is from passing through the data point. Let's now consider the sum of all the squares of all those vertical distances, that is,

$$\sum_{j=1}^{10} d_j^2 = \sum_{j=1}^{10} (y_j - (3.0721\, x_j + 2.4931))^2.$$

What makes (15) the "best" is that, among all the possible lines in the plane, (15) minimizes the sum of the squares of the vertical distances. In other words, for any other values for slope m and y-intercept c, we have

$$\sum_{j=1}^{10} (y_j - (m\, x_j + c))^2 \geq \sum_{j=1}^{10} (y_j - (3.0721\, x_j + 2.4931))^2. \tag{16}$$

The inequality (16) is sometimes expressed by saying that, among all linear functions, the best linear fit minimizes the quadratic (vertical) error with respect to the given data points. Also, this is why it is said that finding the best linear fit is an example of *the least squares method.*

Pearson's correlation coefficient

In the example for best linear fit described above, we first plotted the data points and just "by looking" we concluded that the data points were "almost" aligned. Is there a more precise way to quantify this? Maybe two different people can't agree on "how much" aligned the data points really are, maybe one thinks the points are "somewhat" aligned and the other one thinks they are "almost perfectly" aligned, and maybe a third person thinks they're "quite" aligned. So, we need a formula, a universal measure that everyone can compute (of course, people may later dispute the formula...). Anyways, a popularly accepted measure of "how aligned" the points are is given by Pearson's correlation coefficient and it goes like this.

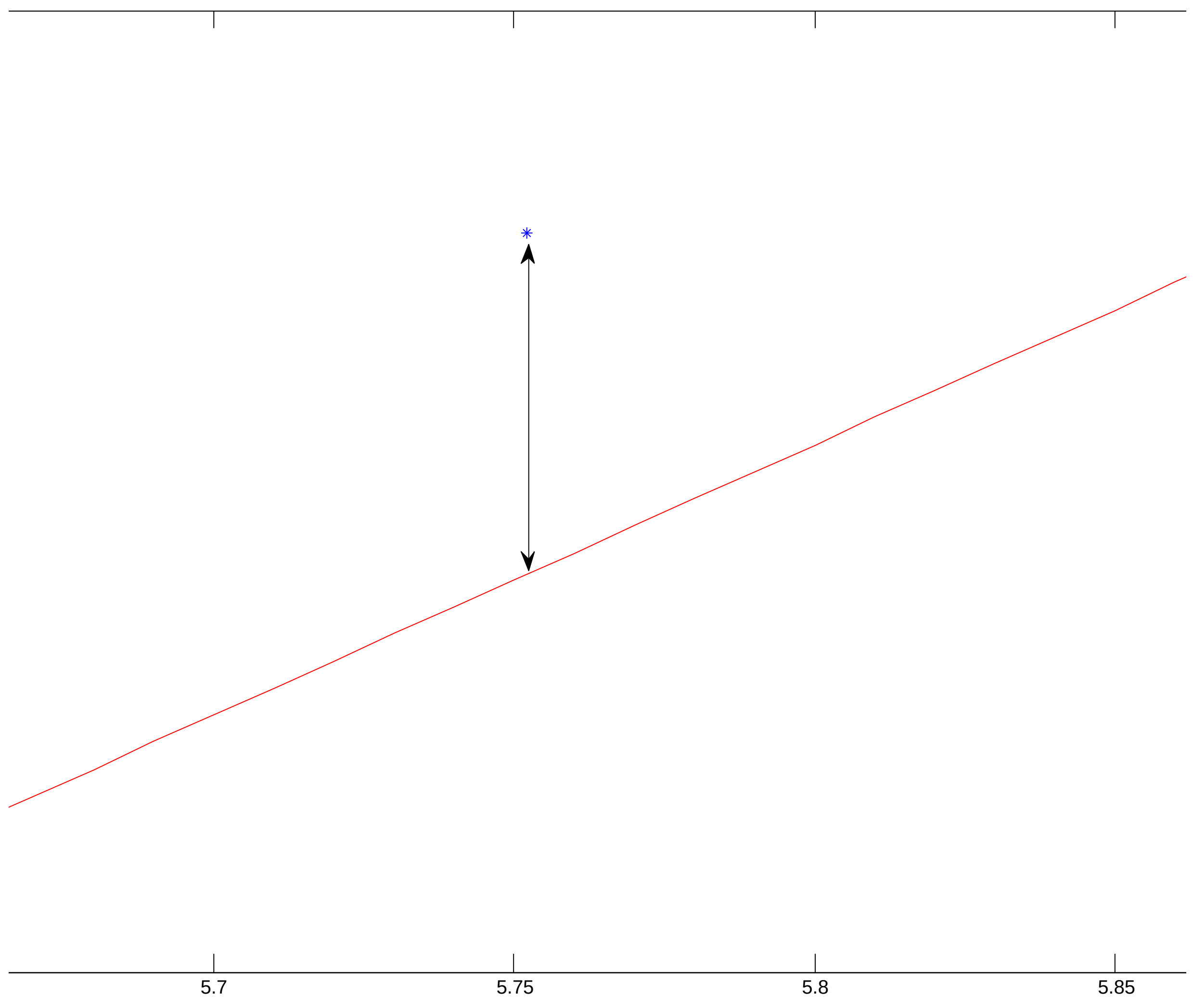


Figure 5: Zooming in on one of the data point in M.

Given the parameters vector x and y in $\mathbf{R}^n$, we first compute their averages, that is,

$$ave(x) := \frac{1}{n}\sum_{j=1}^{n} x_j \quad \text{and} \quad ave(y) := \frac{1}{n}\sum_{j=1}^{n} y_j.$$

Then we introduce the vectors a_x and a_y in $\mathbf{R}^n$ whose coordinates are all equal to $ave(x)$ and $ave(y)$, respectively. Finally, the *Pearson correlation coefficient* is defined by

$$r = \frac{(x - a_x) \cdot (y - a_y)}{\|x - a_x\|\|y - a_y\|}. \tag{17}$$

It turns out that the Pearson correlation coefficient r is always between -1 and 1 and it quantifies how much of a linear dependence exists between (the entries of) the parameter vectors x and y. There is rather well-established consensus in interpreting r as follows:

- The sign of r is the same as the sign of the slope of the best linear fit.
- The closer $|r|$ is to 1, the more of a linear dependence exists between x and y.
- The closer $|r|$ is to 0, the less of a linear dependence exists between x and y.
- If $|r| \geq 0.8$, there is a *high (linear) correlation* between x and y. The quadratic error in this case is expected to be "small".
- If $0.2 \leq |r| < 0.8$, there is a *moderate correlation*. The best linear fit will still minimize the quadratic error, but the error is expected to be "not so small".
- If $|r| < 0.2$, there is a *low correlation*. The quadratic error is expected to be big.

Let's follow the steps above to compute the Pearson correlation coefficient for the data in the matrix M discussed earlier. Here we go

```
>> avex=sum(x)/10

avex =

    4.5772

>> avey=sum(y)/10

avey =

   16.5548
```

```
>> ax=avex*ones(10,1)

ax =

    4.5772
    4.5772
    4.5772
    4.5772
    4.5772
    4.5772
    4.5772
    4.5772
    4.5772
    4.5772

>> ay=avey*ones(10,1)

ay =

   16.5548
   16.5548
   16.5548
   16.5548
   16.5548
   16.5548
   16.5548
   16.5548
   16.5548
   16.5548

>> r=dot(x-ax,y-ay)/(norm(x-ax)*norm(y-ay))

r =

    0.9987
```

That is, the Pearson correlation coefficient for the data in M equals 0.9987, pretty close to 1! In this case there is a positive, high correlation.