# **Projections.**

****

**Example 1**: we want to determine the projection ***p*** of vector ***b*** onto vector ***a***. Both vectors belong to **R3**.We start by noticing that the point ***p*** is the point of the line on which ***a*** resides, that is closest to point ***b***. Clearly, ***p*** is some fraction *x* of ***a***: ,and the vector ***e*** = ***b*** *-* ***p*** from ***p*** to ***b*** is perpendicular to ***p*** (and of course ***a***)**.** Therefore, we can write the following matrix relationships:

***a*T*e*** = ***a*T**(***b*** *-* ***p***)= ***a*T**(***b*** - x***a***)= **0**

***a*T*b*** - ***a*T**x***a*** = ***a*T*b*** - x***a*T*a*** = **0**

x***a*T*a*** = ***a*T*b***

is often referred to as the *best possible scale* of to . It follows that:

Notice that the numerator of the fraction is a 3 x 3 matrix, while the denominator is a number. Altogether, the fraction represents a matrix known as the *Projection Matrix*, and ***p*** *=* ***Pb***:

**Example 2**: we want to find the projection of a vector ***b*** onto a plane ***A*** going through the origin. Both ***b*** and any vector residing on the plane belong to **R3**, that is, we are in 3D space.

We start by noticing that the plane ***A*** represents a *subspace* of **R3** defined by a basis of 2 vectors. We can pick any two vectors ***a*1** and ***a*2** as the basis for this subspace: they don't need to be orthogonal, but they must be linearly independent.

Therefore, we can represent the plane as a matrix, ***A***, with ***a*1** and ***a*2** 'spanning' its *column space*:

A = [1 3 5;2 1 1]', rank(A)

If ***b*** is outside the plane it means that ***b*** is not in the column space of ***A***,

b = [2 2 7]' , rank([A b])

and that there exists a vector ***p*** (the projection of ***b*** on ***A***) such that ***p*** + ***e*** = ***b***, where ***e*** = ***b*** - ***p*** is a vector perpendicular to the plane.

First, we notice that ***p*** is in the column space of ***A***, as it is formed by some linear combination of ***a*1** and ***a*2**; we can write or in matrix notation:

Since ***e*** = ***b*** *-* ***p*** = ***b*** *-* ***Ax*** is perpendicular to the plane, it is also perpendicular to both ***a*1** and ***a*2**. Thus, we can write:

Expanding we obtain:

which is known as the *normal equation* because it represents the concept that is perpendicular to the column space of ***A***. We also know that the projection is :

from which we derive a *projection matrix* ***P***:

This matrix has two important properties:

1. ***P*** is *symmetric*: recalling that the product of a matrix by its transpose is always a symmetric matrix, and that the inverse of a symmetric matrix is also always a symmetric matrix, we obtain:

2. ***P*** is *idempotent*:

Now we notice something very important: since ***e*** is perpendicular to the plane, it can't be in the *column space* of ***A***, **C(*A*)**. But we have learned that, out of the 4 fundamental spaces of a matrix, the *left null space* **LN(*A*)** is perpendicular to **C(*A*)**. Therefore:

In conclusion, if we have any vector ***b*** and any space defined by a basis ***A*** we can calculate two projections of ***b*** that are orthogonal to each other:

1. Projection of ***b*** onto **C*(A)*** ⇒ vector of **C*(A)*** that '*looks*' most similar to ***b***

2. Projection of ***b*** onto **LN*(A)*** ⇒ shortest vector from the subspace **C*(A)*** topoint ***b***

We have completed the derivation of 3 fundamental operations of linear algebra. They are:

**CHANGE OF BASIS**

**SIMILARITY TRANSFORMATION**

**PROJECTION**

With these three tools at hand we are now ready to tackle a number of important problems and applications in the biochemical sciences.

Corollary 1:

We know that the matrix equation ***Ax*** = ***b*** represents a systemof linear equations.Unfortunately, it is not always possible to find a solution to this system.

For example, if ***A*** is the basis for a *plane* subspace in **R3** and ***b*** is a vector outside that plane (as shown in the figure above), obviously there is no solution to the matrix equation ***Ax*** = ***b*** because there is no combination of any two vectors residing on the plane that could possibly give ***b***. However, in the derivation of a general formula for projection we have learned that, *provided the product is invertible*, there exist a solution for the *normal equation*:

or equivalently, substituting for ***x*** as :

In this context, the projection matrix is often called the *hat matrix*.

Thus, now we know that if the matrix equation has no solution, provided the product is invertible, we can at least find a solution for the system , which is the same as finding a solution for the system , in which ***p*** is the *'most similar'* vector to ***b*** that allows a solution.

Corollary 2:

If is a projection matrix, then also , where is the identity matrix, is a projection matrix. In fact since both and are symmetric, their sum or difference is symmetric. Furthermore,

and thus is also idempotent. We also notice that and are orthogonal to each other; in fact:

which means that they *project* on spaces that are orthogonal to each other. Therefore, if projects onto the *column space* of ***A***, projects onto the *left null space* of ***A***. Likewise, if projects onto the *row space* of ***A***, projects onto the *null space* of ***A***.

**PRACTICE**

**1.** Find the projection matrix ***Pc*** onto the *column space* of ***A***:

**2.** Find the projection matrix ***Pr***onto the *row space* of ***A***.

**3.** Multiply ***B = PcAPr***. Your answer ***B*** will be a little surprising: can you explain it? Hint: since any projection matrix **P** is symmetric:

**4.** Find the projection matrices ***Pn***and ***Pln***that project onto **N(*A*)** and **LN(*A*).**

**5.** Carry out the product ***Ax*** = ***b*** for ***x*** = [7 1 11]' by calculating separately the projections of ***x*** onto the row and the null space and by multiplying those projections by ***A***.

**6.** Compound X absorbs between 300 and 400 nm. We have collected several absorbance spectra in this region of the spectrum from a highly purified sample of the compound at the concentration of 1 mM: we have averaged all these independent determinations in order to have a noise-free 'reference' spectrum (spectrum A) that can be used to determine accurately the concentration of the same compound in other experiments. In a different experiment we are measuring the amount of X formed over time in an enzymatic reaction. Using a diode array spectrophotometer we collect a full spectrum of the product in the 300 to 400 nm range every second. Spectra B and C show the spectrum of the product at 5 and 10 seconds, respectively. The three spectra are contained in the file 'DATABASE/Spectra.txt': the 1st column of the array contains the wavelengths.

Import the spectra:

filename = 'Spectra.txt'; dataArray = dlmread(filename);

Allocate imported array to column variable names and clear temporary variables:

wl = dataArray(:, 1); A = dataArray(:, 2); B = dataArray(:, 3); C = dataArray(:, 4);

clear filename dataArray;

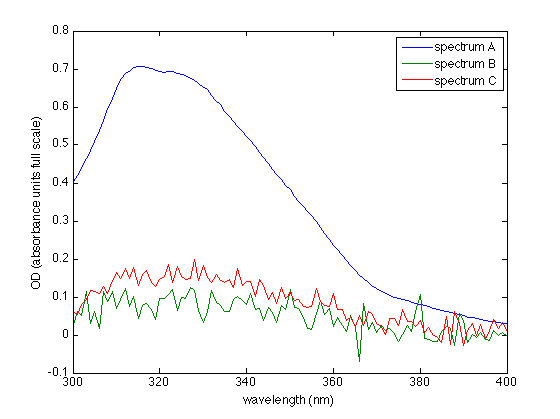
Plot spectra against the wavelength:

figure;plot(wl,[A B C])

legend('spectrum A','spectrum B','spectrum C')

xlabel('wavelength (nm)'); ylabel('OD (absorbance units full scale)'); hold on

The spectra at 5 (spectrum B) and 10 seconds (spectrum C) clearly show much less compound than 1 mM. However, due to the fact that the spectra are collected in only 1 second, they show different background trends and much higher levels of noise. For this reason it would be unwise to simply take the OD value at the absorption peak at 5 and 10 s. Instead, we want to 'see' how the A spectrum would look if it was scaled to spectrum B and spectrum C: in this way we will get information not just from the absorption peak, but from all the wavelengths. In other words we want to find the best fit of the 'shape' of spectrum A to spectrum B and spectrum C. We need to calculate 2 scale factors, *x*1 and *x*2:



⇓

but **,** however

⇓

⇓

⇓

where ***P*** is a projection matrix, and ***PB*** and ***PC*** are the projections of ***B*** and ***C*** onto ***A.***

Based on these considerations:

a. Determine the absorption wavelength and extinction coefficient for compound X and the concentration of this compound at 5 and 10 seconds based on this extinction.

b. Calculate the new spectrum B and spectrum C projected onto spectrum A and then plot the three spectra.

c. What are the 'scale factors' required to correct spectrum A? What is their relationship to the concentration of compound X at 5 and 10 s?

d. Calculate the projections of spectrum B and spectrum C onto the *left null space* of spectrum A. Can you tell what these projections represent? Since you now know the 'scale factors' required to correct spectrum A to make it similar to spectrum B and spectrum C, make them a little bigger and smaller and apply the modified scale factors to the original spectrum A. Can you see what's happening? This should give you a clue to the meaning of the 'projection' operation.

Solution to Problem 6.

a. Determine the absorption wavelength and extinction coefficient for compound X and the concentration of this compound at 5 and 10 seconds based on this extinction.

[ec,ec\_ind] = max(A)

B\_conc = B(ec\_ind)/ec; C\_conc = C(ec\_ind)/ec

b. Calculate the new spectrum B and spectrum C projected onto spectrum A and then plot the three spectra.

P\_cA = A/(A'\*A)\*A'

B\_to\_A = P\_cA\*B; C\_to\_A = P\_cA\*C;

plot(wl,B\_to\_A,'.g',wl,C\_to\_A,'.r'); legend({'spectrum A','spectrum B','spectrum C','fit B','fit C'})

c. What are the 'scale factors' required to correct spectrum A? What is their relationship to the concentration of compound X at 5 and 10 s?

scale\_B = (A'\*A)\A'\*B scale A to B

scale\_C = (A'\*A)\A'\*C scale A to C

or simply:

scale\_A\_B = (A'\*A)\A'\*[B C]; scale\_A\_B = A\[B C]

B\_conc\_fit = B\_to\_A(ec\_ind)/ec; C\_conc\_fit = C\_to\_A(ec\_ind)/ec

d. Calculate the projections of spectrum B and spectrum C onto the *left null space* of spectrum A. Can you tell what these projections represent?

[nrows,ncols] = size(P\_cA)

P\_lnA = eye(nrows) - P\_cA

B\_to\_A\_er = P\_lnA\*B; C\_to\_A\_er = P\_lnA\*C

B\_er = B - B\_to\_A; C\_er = C - C\_to\_A

for i = 1:ncols

plot([wl(i) wl(i)], [B\_to\_A(i) B\_to\_A(i)+B\_er(i)], ':k')

plot([wl(i) wl(i)], [C\_to\_A(i) C\_to\_A(i)+C\_er(i)], ':k')

end

legend({'spectrum A','spectrum B','spectrum C','fit B','fit C','error B','error C'})

What is the meaning of the 'projection' operation?

B\_plus\_er = B - 1.01\*scale\_B\*A; B\_minus\_er = B - 0.99\*scale\_B\*A;

B\_er'\*B\_er; B\_plus\_er'\*B\_plus\_er; B\_minus\_er'\*B\_minus\_er

Conclusion: the projections, ***PB*** and ***PC***, of B and C onto Arepresent a *least squares* fit of spectrum A to spectrum B and C, respectively!