ELECTRONOTES
1016 Hanshaw Road
Ithaca, NY 14850

APPLICATION NOTE NO. 385
September 5, 2012

LOOKING AT PRINCIPAL COMPONENT ANALYSIS

Introduction to the Difficulties Here

Coming at PCA from a "Signal" Background

This introduction is long (4 pages) and is followed by a rather prosaic presentation of theory. I think it is necessary. It is followed by a rather prosaic presentation of theory, and then an attempt to make sure the time-sequencing is clarified. But I don't expect anyone to read it right now. Instead, you will be better motivated and better prepared once you try to use PCA and then find yourself in need of more theory. Here, you might want to jump to page 18, or run the Matlab code ***rs.m*** for a start. Then you may come here and find what is bothering you. But an introduction does come first. So here it is.

The readers of our newsletters and application notes are likely mostly "Signal People". We are familiar with signals, usually occurring as a function of time (t), like music and speech. We can have more than one signal of course, like signals $x_1(t)$, $x_2(t)$ and $x_3(t)$. Or just as likely, they are functions of a time index (n) so we would have $x_1(n)$, $x_2(n)$ and $x_3(n)$ where we assume that proper sampling is in place. We recognize that while multiples signals in quite large numbers, occurring simultaneously, are possible, much of our experience is with perhaps half a dozen such signals at most. We move signals around, patching them through different modules, for example.

First of all: samples. A sample might mean one number. But "samples" might mean a collections of numbers or a collection of signals. In the first sense we often use it to mean something like x(5) = 3.22. This indicates that at a time t=5T, where T is the reciprocal of the sampling frequency f_s, the values of a signal x(t) was sampled (captured) as 3.22. None the less we are not uncomfortable when someone refers to a "sample" as being what we might also call an "audio clip", like a bassoon $F^{\#}$ of duration 1.6 seconds. A large collection of such "samples" are useful for studying the nature of bassoon sounds, or perhaps as a basis of signals for "synthesis". These are time signals in parallel. Analysis tools and models using single time signals are familiar (Fourier analysis, linear predictive coding, wavelet analysis, etc.). [And of course, statisticians often mean "sample" to mean data collected from a subset of a larger population.]

Not all the numbers we crunch are simple time signals. Of course, we are familiar with signals that are really images, two spatial dimensions. Or video that is a signal consisting of images as a function of time. Academics routinely crunch array of numbers – students, and the corresponding scores awarded for their assigned tasks. And so on.

Along comes the notion of "Principal Component Analysis" or PCA for short. Where does this fit in. While sometimes classed as a topic in signal theory, it is probably more properly thought of as a child of the discipline of statistical analysis. As such, we approach it through a fog of often unfamiliar terms and data sets that are not exactly what we are familiar with. There are probably a dozen excellent tutorials on PCA, and many of these can be enjoyable read or watched and leave you with the notion that you understand it. But not unlikely, you don't - yet. You ask yourself "how do I really do this?" and "what does it mean?" A failure to clearly get you mind around the nature of the input data is perhaps the most common stumbling block.

Time Signals, Variables, Vectors, and Functions

Let's begin with the familiar notion of a vector, or let's say three vectors, x_1, x_2, and x_3. Suppose we see them denoted precisely as:

x_1 = [1 2 3 4 5 6 7 8 9 10]

x_2 = [3.1 2.7 2.1 1.4 0.6 -0.4 -0.9 -1.2 -0.3 0.7]

x_3 = [77 72 91 88 87 81 68 68 93 81]

These three vectors all have the same length, but don't look at all like the "same thing". The vector x_1 looks like an index, perhaps it represents time, an independent "variable". Perhaps it is time, and perhaps x_2 is a corresponding set of values of a time sequence, a dependent variable. The vector x_2 even looks relatively smooth – we could plot x_2 as a function of x_1. The vector x_3 looks more like it could be students' scores on an exam. But what would be the independent variable in this case? Student ID numbers perhaps? If we had a vector of student ID's, would it make sense to plot the scores against the ID numbers? Of course not. Tabulate YES – plot NO. Perhaps however x_3 is a function of x_1, where x_1 is the indexing number of a weekly quiz – now more or less a time sequence, and x_3 is the scores of just one particular student. Another student would have 10 different scores. Thirty students would be 30 vectors in parallel. Academics deal with such records in tables (often processed as matrices) all the time.

But vectors are just arrays of numbers. They don't automatically correspond to anything at all.

We mentioned that x_1 might well be an (independent) time-indexing variable, and x_2 a corresponding dependent variable of samples – perhaps the $F^{\#}$ of the bassoon (well - part of it anyway). We can well imagine a large set of additional vectors similar to x_2 corresponding to other bassoon tones, like G, $G^{\#}$, A., etc. We could plot these against x_1, and perhaps Fourier analyze them, etc. It would not however seem to make any sense to plot one tone against the other (it might be useful or at least fun, but it is not familiar). But, we might well be receptive to a proposal to subject our set of bassoon tones to PCA analysis. Perhaps PCA will tell us something basic and useful. This does not mean we know how to do it, or what a result might mean, but we can visualize the nature of the input data. Output? Well, we would hope, as always, to find meaningful patterns for our efforts.

Statisticians are interested in the patterns in time series of course (stock market prices, proxy temperature data for historical climate, the progress of a student's test scores, etc.). Such an analysis may yield meaningful parameters, and may even be predictive. Here we might hope PCA is a new tool in our toolbox of signal analysis techniques.

So it is perhaps disconcerting when first examples of PCA are apparently not actually time series. Or, aside from some familiar notions such as eigenvalues and eigenvectors, we get lost in statistical jargon. As always, nothing beats having a program and running data through it. We put something in, and something comes out. So that's what it actually does! This will be our goal here. Understanding through familiarity.

Tutorials

First, the problem with the tutorials. They do generally plot sequences (vectors) or functions against one another. We don't mind doing this in a general sense. Lots of engineering is of this nature. In our synthesizer circuits, we plot, for example, the output current of an exponential converter versus the input voltage. But that's kind of the whole story. It's according to theory, or perhaps it fails to some degree in some range. But we aren't looking for anything new to emerge. Such a plot is a visualization to help us.

So in a tutorial, we might find a plot of student scores (exactly this in one excellent video). On this plot, each student is a single point in a two dimensional plot with one axis the grade in one subject, and the other the grade in a different subject. Perhaps something like physics versus math. Note that time is not involved here. And this is very different from a plot of a time series. How would you "substitute" time back in? It doesn't make any obvious sense.

So the axes are scores – time is not involved here – and we probably guess exactly what the goal of the plot is. We are probably trying to find out if students who are good (or poor) in math are likewise good (or poor) in physics. We seek correlation (or not). We would suspect that such scores would have a fair to excellent degree of correlation in general. Basically a straight line at 45° with some scatter. But we could also look for a correlation with a variable that was not a score in a related subject (or any score at all). We might expect a positive correlation with the number of hours spent studying, and perhaps an inverse correlation with hours of TV watched. So we don't mind if our vectors are in different units and of a different nature. We might expect no correlation at all with other variables, such as for example, the students' heights. But how do we get time back in – or should we be trying? Not if it's not actually an independent variable.

So, sticking first with scores. The successive quiz scores of any one student constitutes a time series (one vector), and we are generally interested in monitoring progress of the student and perhaps making predictions. But aside from such occurrences as a general "mid-term slump", we aren't really expecting much correlation with time. The scores of a group of students are accordingly multiple, parallel time series (many vectors). We might well expect these vectors to show some correlation – some quizzes may be easier or harder than intended. But for the group, time as such, (other than a convenient index) seems not too important.

We also usefully plot final (end of term) scores for many students across subjects, and we expect to find correlations. But time is irrelevant in this case too. Correlations with other factors (study effort, TV watched, beer consumed, height, number of siblings, etc.) can be examined. All these are facts about the student independent of time, and can only be useful if a set of students is in the data. The only requirement is an index that assures that the position of a number in a data vector always corresponds to the same student. The students must always be in the same order. But that index is not in general time, and the specific numerical value of the index does not matter, except as it is automatically tracked by the position within the vector. So, the position within the vector is like time, and may even correspond exactly to time. Yet this "index" is often not involved in PCA.

* * * * * * * * * * * * * * * * * *

So what is the input data structure for PCA. It's a matrix. It is useful for signal people to think of the rows of the matrix as the signals (the data vectors) and the columns as the parallel array of values. And we can't have an input vector – there must be at least two rows minimum. There are no principal components for one signal.

What does the output of PCA mean? Well, we want to take what is generally a very large data set (vectors arrayed in a matrix) that is difficult to analyze and find its essence. In this case, the term "Principal Components" describes exactly what we hope to get out.

Example 1: Scores in Physics and Math

Here we will do a first example in detail, introducing the procedure as we go. This will be the suggested case of comparing final scores of a group of students in math and in physics. Let's assume there are 10 students total. Each student has two scores. Let's say the scores are:

M = [90 87 77 65 71 93 85 83 86 86]

P = [94 84 87 71 77 85 88 79 87 90]

Thus the first student has a score of 90 in math, and 94 in physics, and so on. The index along the vector is just that, an index. The only thing that is necessary is that we keep the scores for the same student in the same relative position. We have no compelling reason for a separate indexing vector, or a vector of student names, or student numbers. Time is not involved here, but it is perhaps helpful to put it in in the following way. Assume that these are the scores of the students of one advisor who have scheduled appointments every 15 minutes. So "time" in this sense runs left to right. But, if the students agreed to switch their appointment times, this makes no difference to the results below. [One further thing: we might also consider examining the separate distributions of the math grades and of the physics grades, like plotting histograms, but we are not doing this here.]

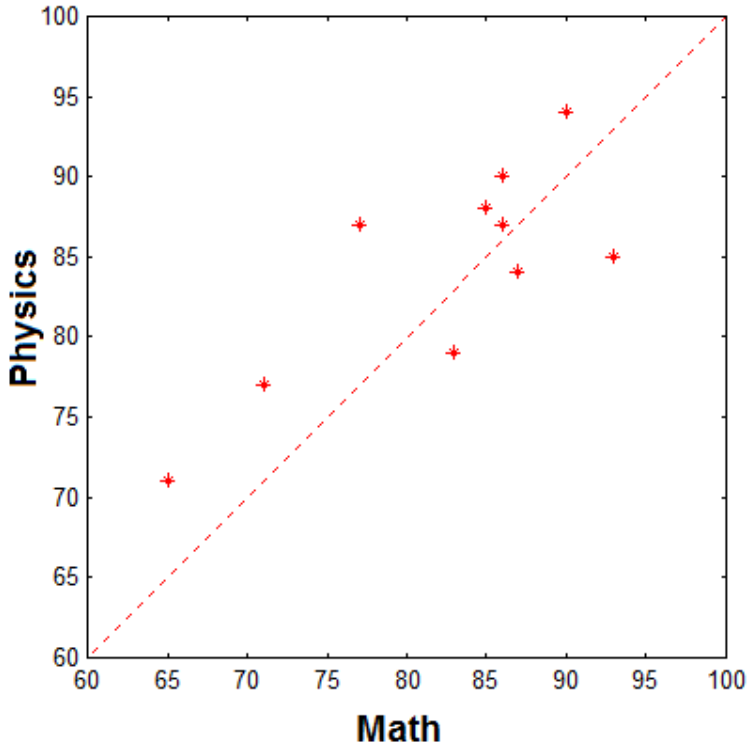


Fig. 1

Physics and Math scores. Based on this plot alone, we guess that there is a fair to good correlation between the scores, and that the scores in physics were a bit higher.

What we can do is plot the math grades against the physics grades – each student contributing one point in a two dimensional "score space". This is sometimes called a "scatter-plot". This we show in Fig. 1. Each student is one of the 10 points. Note that you can't tell from the plot the ordering along the data vectors. That's gone.

Already our eyes find the interesting patterns. If the scores were perfectly correlated, they would all be on a straight line. They aren't – but they do appear in an elongated cluster sloping upward. So from this we guess that students who do well in one course tend to do well in the other, with some exceptions. Further, we have plotted a dashed red line that corresponds to having both grades the same. No student is on this line, but 7 students are above it and three below. This tells us that for whatever reason or reasons, the physics grades were a bit higher. All this we see from Fig. 1. Not all data will be so obvious, and this is where PCA comes in.

The first step in PCA will be to organize the data vectors into a data matrix, x, and "normalize it". It is probably safest to consider the normalization of the vectors first, followed by the combining of the vectors into the matrix. (Note that program code often does it the other way – leading to the exact same result, but perhaps leaving the code more flexible. Matlab stands for Matrix-Laboratory and "likes" to do operations on whole matrices at once).

The second step is to form the covariance matrix corresponding to x. This will be a 2x2 matrix so it is clear that information is going to be lost (or in a more positive perspective, the essential data is compressed to fewer numbers). The Principal Components directions are then (third step) the eigenvectors of this covariance matrix. There are only two of them here, and they are two-dimensional.

The calculations shown on page 7 (top) show the normalization of the data. This is first the subtraction of the mean (average) from the data vectors individually, and then the data vectors are divided by their standard deviations. Recall that the standard deviation is the RMS (Root Mean Square), the square root of the mean of the squares of the samples. To compute it, you square all the samples in the data vector, add them, divide by the number of samples (caution!) and take the square root. The caution is that it is often not N (10 here) that we divide by, but (N-1), (9 here) for obscure reasons. It usually makes little difference. The square of the standard deviation is the "variance" which will reappear in the covariance matrix. [Incidentally, we note that the mean and standard deviation of test scores seem to be an obsession with students – as though their entire future depended on these two numbers!].

COMPUTATION OF PRINCIPAL COMPONENTS

We begin by forming the 2x10 matrix x as:

$$x = \begin{bmatrix} 90 & 87 & 77 & 65 & 71 & 93 & 85 & 83 & 86 & 86 \\ 94 & 84 & 87 & 71 & 77 & 85 & 88 & 79 & 87 & 90 \end{bmatrix} \quad (1)$$

The mean of the first row is 82.3 while the mean of the second row is 84.2.

Subtracting these gives:

$$x = \begin{bmatrix} 7.7 & 4.7 & -5.3 & -17.3 & -11.3 & 10.7 & 2.7 & 0.7 & 3.7 & 3.7 \\ 9.8 & -0.2 & 2.8 & -13.2 & -7.2 & 0.8 & 3.8 & -5.2 & 2.8 & 5.8 \end{bmatrix} \quad (2)$$

The standard deviation of the first row [equations (1) or (2), before or after subtracting mean makes no difference] is 8.731 while that of the second row is 6.779. we can divide the rows by their standard deviations to give:

$$x = \begin{bmatrix} 0.8819 & 0.5383 & -0.6070 & -1.9814 & -1.2942 & 1.2255 & 0.3092 & 0.0802 & 0.4238 & 0.4238 \\ 1.4456 & -0.0395 & 0.4130 & -1.9472 & -1.0631 & 0.1180 & 0.5606 & -0.7671 & 0.4130 & 0.8556 \end{bmatrix} \quad (3)$$

We will at some point be interested in what happens when we don't do the normalizations completely or correctly, but for now, the data as in equation (3) is ready. Step one is done.

Next we need to compute the "covariance matrix" of x, which is the matrix $x \cdot x^T$ where x^T is the "transpose" of x (flipping upside down and twisting 90 degrees) as shown

$$\text{covar} = \begin{bmatrix} 0.8819 & 0.5383 & -0.6070 & -1.9814 & -1.2942 & 1.2255 & 0.3092 & 0.0802 & 0.4238 & 0.4238 \\ 1.4456 & -0.0295 & 0.4130 & -1.9472 & -1.0621 & 0.1180 & 0.5606 & -0.7671 & 0.4130 & 0.8556 \end{bmatrix} \begin{bmatrix} 0.8819 & 1.4456 \\ 0.5383 & -0.0295 \\ -0.6070 & 0.4130 \\ -1.9814 & -1.9472 \\ -1.2942 & -1.0621 \\ 1.2255 & 0.1180 \\ 0.3092 & 0.5606 \\ 0.0802 & -0.7671 \\ 0.4238 & 0.4130 \\ 0.4238 & 0.8556 \end{bmatrix} \quad (4)$$

$$= \begin{bmatrix} 9.0000 & 7.0351 \\ 7.0351 & 9.0000 \end{bmatrix} \quad (5)$$

Once the data is properly normalized, we form the "covariance matrix" by multiplying the data matrix x by its transpose. This is illustrated in the lower portion of the calculations on page 7. We don't doubt that the reader knows how to do matrix multiplication, but we have written it down for specificity and so that we can comment on it. This being a 2x2 matrix, half of it, the main diagonal, is variance (same vector) and not covariance, off diagonal (different vectors). The elements of the covariance matrix are just "dot products". Both data vectors have the same variance (9 here – see comment above) because they were divided by their standard deviation in the normalization process. It is further evident that the matrix is symmetric, and why.

Now for the calculation of the eigenvalues and eigenvectors. We are not going to do this calculation here. Most readers are (or were at one time) familiar with the procedure. Briefly recall that we are looking for eigenvectors **e** and eigenvalues λ such that for a matrix A, A**e** = λ**e**. It's just linear algebra. Remember solving the characteristic equation det(A- λI)=0 for the eigenvalues and then simultaneous linear equations for **e**. We want the eigenvectors of the covariance matrix for our principal component directions, and the corresponding eigenvalues as a means of assessing the "strength" or influence of the particular PC.

Using Matlab's ***eig*** function we find the eigenvalues to be 16.0351 and 1.9649, with corresponding eigenvectors [0.7071 0.7071] and [0.7071 -0.7071]. Shockingly these are vectors at angles of ±45°. Well, we drew a line in Fig. 1 at 45°, suggesting that that direction had a particular significance (perfect correlation). True enough (Fig. 2), this vector at 45° is the <u>first PC</u> as it has the largest eigenvalue (16.0351), and the second eigenvector is perpendicular to it (eigenvalue 1.9649), indicating less significance. The math and physics grades are correlated.

But wait a minute. We said that in Fig. 1 we had evidence of better grades in physics. If we had free-hand drawn a straight line as an approximation to the correlation, wouldn't it have swung away from the 45° angle? What's going on? This is considered below.

Comparing Fig. 1 and Fig. 2, we see that Fig. 2 (normalized) is a somewhat better fit to the data, and the fact that the scatter-plot is rotated is formalized by the principal components. The two eigenvectors are in fact orthonormal. This suggests that a better basis would be possible by rotating the whole thing 45° so that the first PC is horizontal. This is shown in Fig. 3. Rotating the data 45° is a matter of multiplying each data point, (each student here), as a vector, by a matrix for a 45° rotation. This is not at all hard, we just need the right 2x2 rotation matrix. We already have it – it's the matrix of eigenvectors – see Equation (5). Note that the second PC is at +90 to the first in Fig. 2, so in plotting PC2 as a positive "y-axis", the plot flips vertically. Now each student is <u>approximately</u> represented in a reduced data set of one dimension, along the horizontal axis.

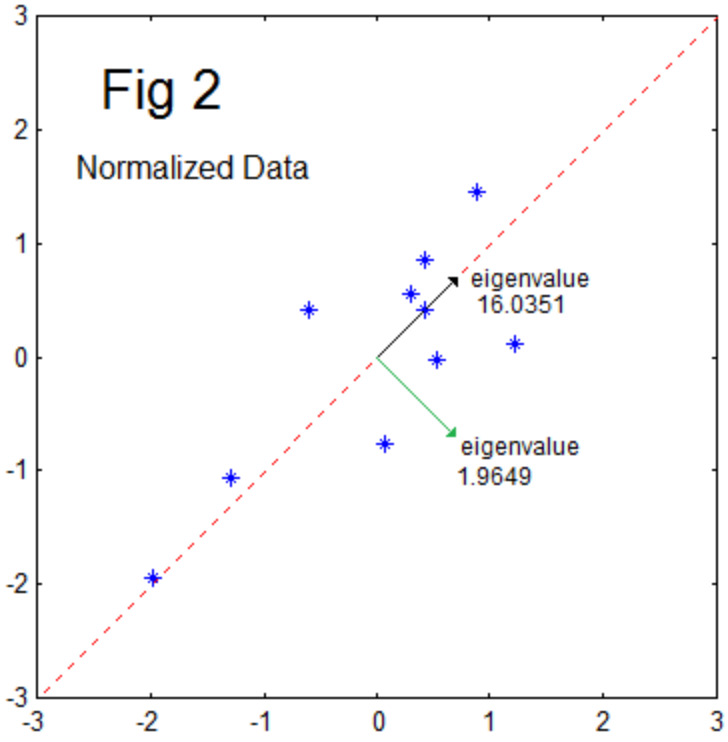


Fig 2
Normalized Data
eigenvalue
16.0351
eigenvalue
1.9649
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

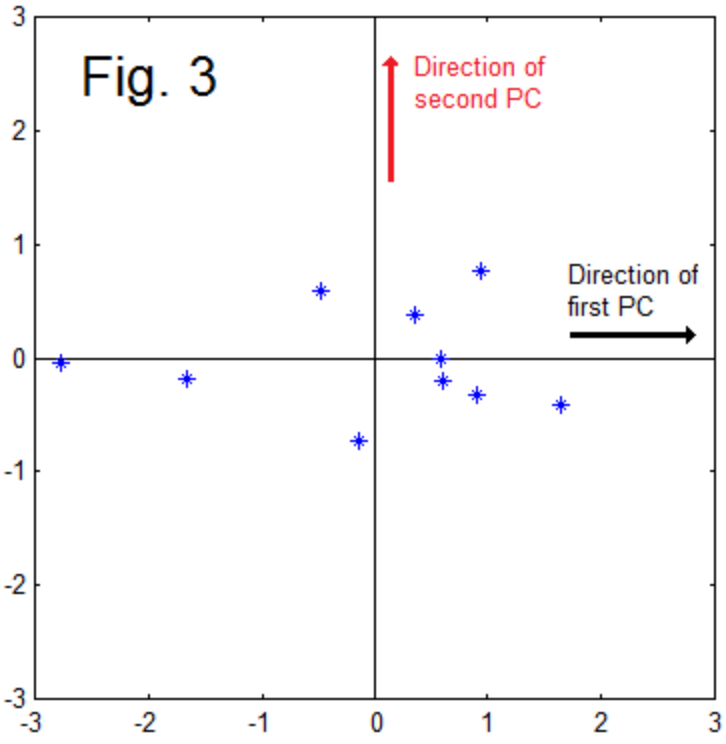
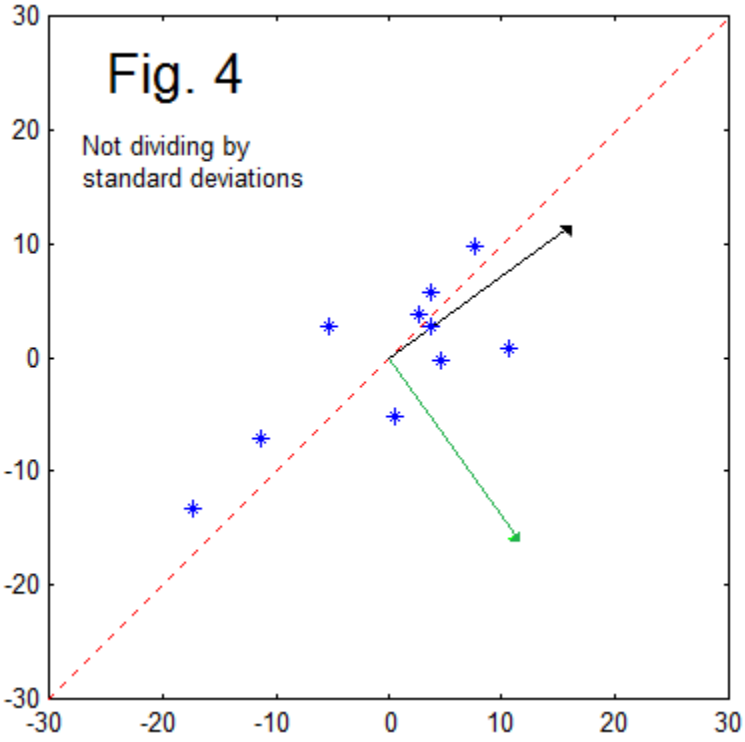


Fig. 3
Direction of
second PC
Direction of
first PC
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

$$\begin{bmatrix} 0.7071 & 0.7071 \\ 0.7071 & -0.7071 \end{bmatrix} \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (5)$$

Does this make sense? To judge this, let's look at a couple of students. First, the point furthest to the left in Fig. 3. Looking back at Fig. 1, this is the student with M=65 and P=71. Note that the point is quite close to the axis, reflecting the fact that the PCA procedure has adjusted for the generally better grades in physics across the 10 students. Another student is the one plotted in Fig. 3 at about 0.6 on the horizontal axis. This is the M=86, P=87 student from Fig. 1. Here the PCA has decided that these two scores are exactly the same, adjusting for the slightly better physics grades overall. These first two students are thus quite consistent across courses. Students with more spread are off the horizontal axis, and would need the second PC (now vertical) to be better represented. The largest PC2 in Fig. 3 is for M=93, P=85. While only the second largest spread, it also represents a reversal of the Physics > Math trend.

Finally we come to the question of why the angle is exactly 45°. Well Fig. 4 shows the case where we did the exact same PCA-type calculations, but did not divide the data by the standard deviation. The angle of the first PC is now more like 36 degrees. In fact, it could well be considered a better fit (there would be five students on each side) relative to the hand drawn 45° angle in Fig. 1.

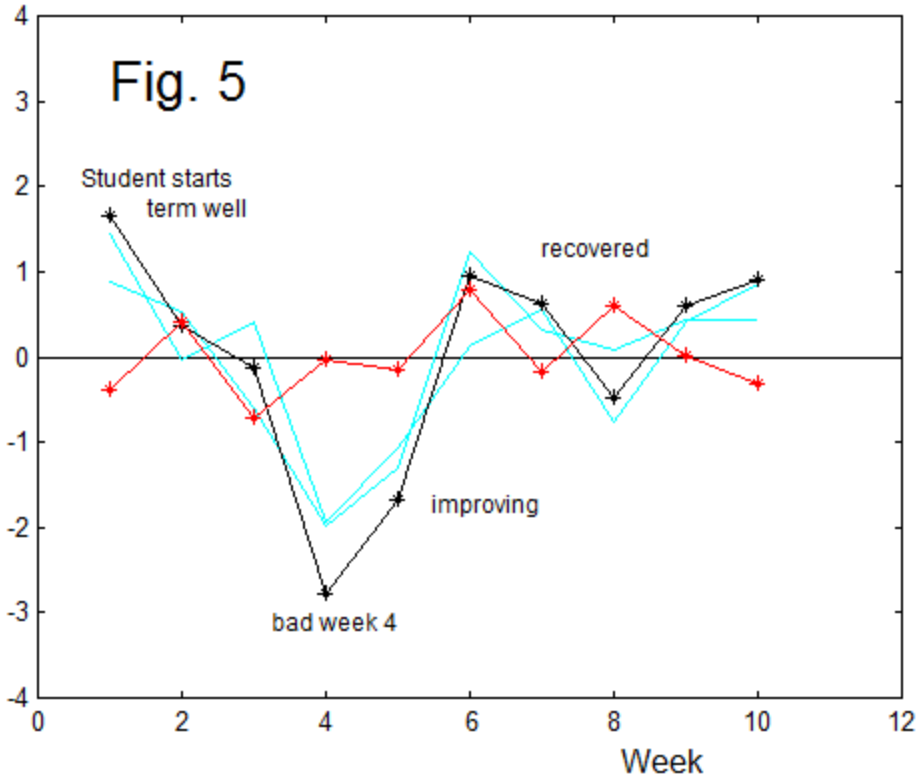


Bring Time Back In – Weekly Quiz Scores For One Student

Examples like the one involving student grades abound in PCA tutorials, and as we have said, the notion of time seems to have been dropped. We did just manage to hold onto time by the ruse of postulating that students dropped in at regular intervals to report their grades. But that supposition was not actually used (or discernible from the plots).

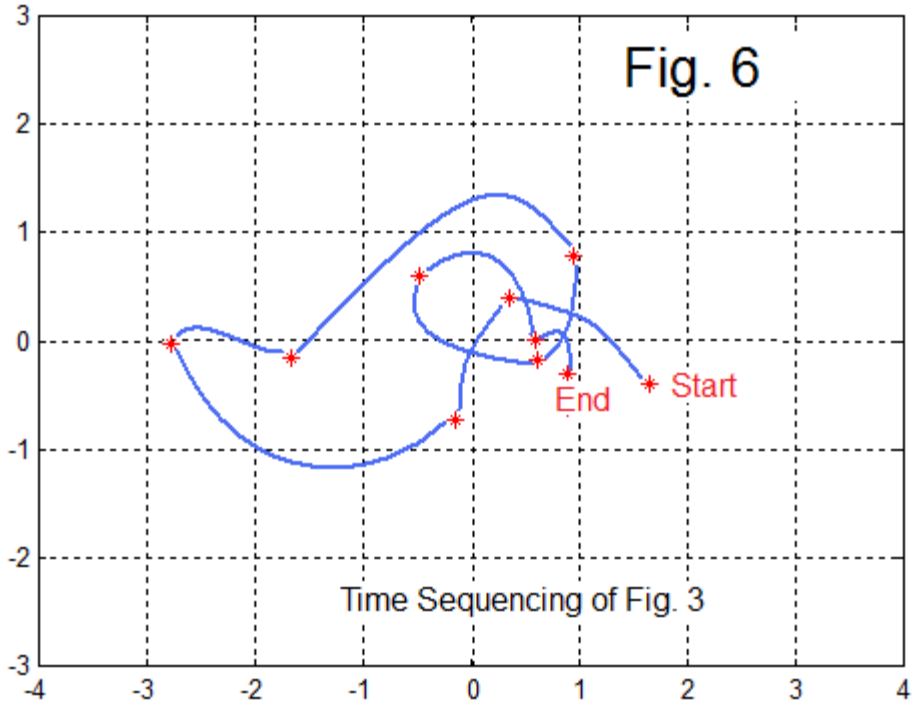
Perhaps more alarming than the loss of time is the loss of the actual length 10 data vectors. Instead of data vectors, we get some principal components. In fact, we took two data vectors of length 10 (10 students), twenty numbers, and ended up with two principle component vectors of length 2 (four numbers).

Another way to put time back in would be to suppose that the data is not for 10 students, but the weekly quiz scores for just one student. In this case, instead of the scatter-plots we have been using above, a conventional time series plot is used (Fig. 5).



In Fig. 5, we monitor the student as the term progresses. Note that the student starts out very well, goes into a general dive (well, for two courses) in the fourth week, and then recovers.

But where did Fig. 5 come from? Well, it is the projection of the data on to the PC basis (r in the Matlab code below). This is how we got length 10 information back. The step is to multiply the matrix of eigenvectors by the normalized data. Now, we already did this, believe it or not. Fig. 5 is Fig. 3 as a time plot (see also Fig. 6, which is Fig. 3 redrawn with blue lines showing the sequencing). Please study Fig. 5 and Fig 6 carefully. If you understand the relationship, you will have avoided a good deal of the confusion about the ways PCA is presented.



Time for Some Code

We need to look at this in our familiar Matlab code. There exist many useful programs and variations on them. Below we give an example ***pcax.m*** that is the principal component analysis of x, x being a data matrix. As with many programs, there is a core of operations, and just as much (or much more) that is for plotting and displays which usually requires custom adjustments according to the situation. Here is the example code on page 13.

```
% pcax.m
% Principal Components of a Data Matrix x
% x contains N signals (Rows) of length M (Columns)
[N,M]=size(x)

% NORMALIZATION remove Mean and Divide by Standard Deviation
y=x;  %reserve copy
for k=1:N
  x(k,:)=x(k,:) - mean(x(k,:));
  x(k,:)=x(k,:) / std(x(k,:));
end

% PCA
covx=(x*x');
[Evec,Eval]=eig(covx); % Eigen-Analysis
Eval=diag(Eval)';
% Re-Order
[Eval,ind]=sort(Eval);
Eval=fliplr(Eval)      % print e-values
ind=fliplr(ind);
Evec= Evec(:,ind)
% Projections
r=Evec*x

% Displays
figure(1)
plot([-1 M+1],[0 0],'k')
hold on
for k=1:N;  plot(x(k,:),'c');end
plot(r(1,:),'k');plot(r(1,:),'k*');  % first PC
plot(r(2,:),'r');plot(r(2,:),'r*');  % second PC
axis([0 M+1 -4 4])
hold off

if N==2
  figure(2)
  plot(x(1,:),x(2,:),'*r')   % scatterplot if N=2
  axis equal
end
```

ABOUT THE CODE

The top lines of the program do the data vector normalizations. This is followed by two lines forming the covariance matrix and doing the eigen-analysis. Then there are five lines just to put everything in descending eigenvector order. That's it. Well, except for the generation of **r**: the projection of the normalized data to the eigenvector space. We did this in going from Fig. 2 to Fig. 3, in the manner of Equation (5). We saw that this was a rotation, so perhaps r stands for rotation here. Exactly what happens is given in more detail, for the student scores example, in equation (6).

Dot Products

$$\begin{bmatrix} 0.7071 & 0.7071 \\ 0.7071 & -0.7071 \end{bmatrix} \begin{bmatrix} 0.8819 & 0.5383 & -0.6070 & -1.9814 & -1.2942 & 1.2255 & 0.3092 & 0.0802 & 0.4238 & 0.4238 \\ 1.4456 & -0.0295 & 0.4130 & -1.9472 & -1.0621 & 0.1180 & 0.5606 & -0.7671 & 0.4130 & 0.8556 \end{bmatrix}$$

$$= \begin{bmatrix} 1.6458 & 0.3598 & -0.1372 & -2.7779 & -1.6662 & 0.9500 & 0.6150 & -0.4857 & 0.5917 & 0.9046 \\ -0.3986 & 0.4015 & -0.7213 & -0.0242 & -0.1641 & 0.7831 & -0.1777 & 0.5991 & 0.0076 & -0.3053 \end{bmatrix} \quad (6)$$

So the matrix product represents the dot product of the eigenvectors with the columns of the normalized data matrix. It is clear here that the dot product with the upper eigenvector (the matrix is transposed but this is the same in this case) relates to the sum of the two scores in math and physics, while the lower eigenvector relates to the difference between the scores. Because we noted that we expected (and found) the scores somewhat correlated, the magnitude of the differences are small in general. Note that the answer in equation (6) is the data plotted in Fig. 3, Fig. 5, and Fig. 6. Basically a mechanical operation, but Equation (6) shows what the matrix multiply really does.

The bottom lines of code are for display, and these need to be manipulated for a given application, but that's true of most programs. We also need to remove semicolons at times from the end of statements to let the result of that line print.

Summary to This Point

(1) Thinking in terms of signals, it makes no sense to speak of principal components of one signal. The signal has a spectrum, a wavelet-decomposition, etc., but not principal components.

(2) Still thinking of signals, a set of two or more signals can have an expression in terms of principal components. Usefully we think of such signals as vectors of samples. This set of vectors has its numbers ordered by an automatic index (position along the vector). This may be time (as for our usual signals), but it's just a position along the vector in general.

(3) It is the normal procedure to "normalize" the data by subtracting the mean and dividing by the standard deviation of each vector individually.

(4) If we want to plot the data, we could do it as multiple curves, one for each row, as we would normally do for time signals. When there are only two data vectors, we can plot the two rows against each other, what is called a scatter-plot. With three dimensions, we can sometimes understand the three-dimensional scatter-plot projected on a flat page. Beyond three, it is really tough. Many tutorial examples have two data vectors only.

(5) Plotting multiple curves is most natural for time signals, and larger numbers of data vectors. Scatter-plots are most useful for small numbers of data vectors (like two!), where the index along the vector is not clearly time, and where we are looking for a quick notion of correlation. Indeed we often seek correlations between only two variables.

(6) We think of our vectors as row vectors of a data matrix, x, that contains the entire set. This matrix is in general not square. It has N rows of length M.

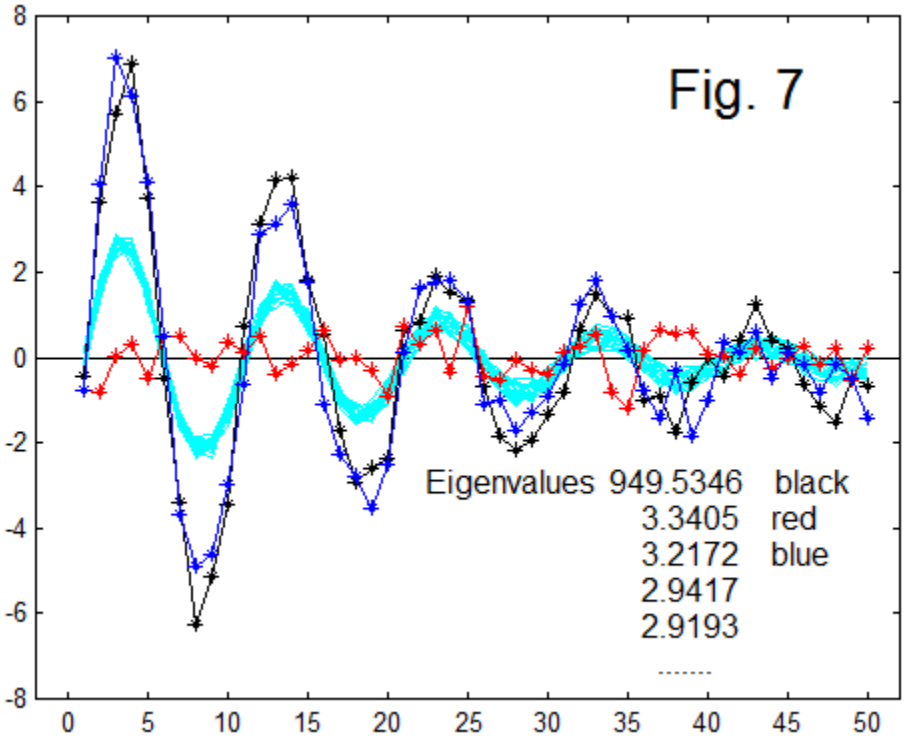
(7) There are several ways to do PCA, including the covariance method (here) or "Singular Value Decomposition" (SVD). There is related jargon which may be unfamiliar, if not misleading.

(8) The covariance matrix is $x*x^T$ and is an NxN matrix. We take the eigen-analysis of this, and call these the eigenvalues and eigenvectors, the terms probably already familiar to signal folks. The eigenvectors are sometimes called the PCs, or at least the orthogonal basis of the PCs. The corresponding eigenvalues tell us how important a particular PC is.

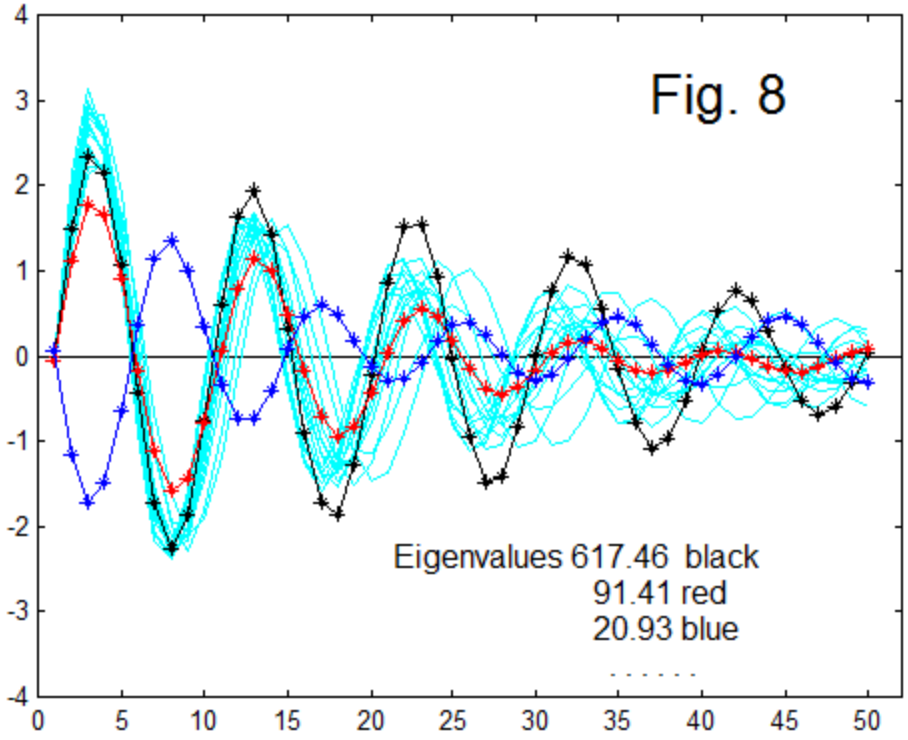
(9) Once we have the eigenvalues, we can choose the eigenvectors corresponding to the largest eigenvalues, and project the normalized data into this basis set. These results, sometimes called signals or scores, seem to also be loosely called the PCs. Notice that these projections have the same length as the original data vectors. The length of the eigenvectors is generally less, often much less.

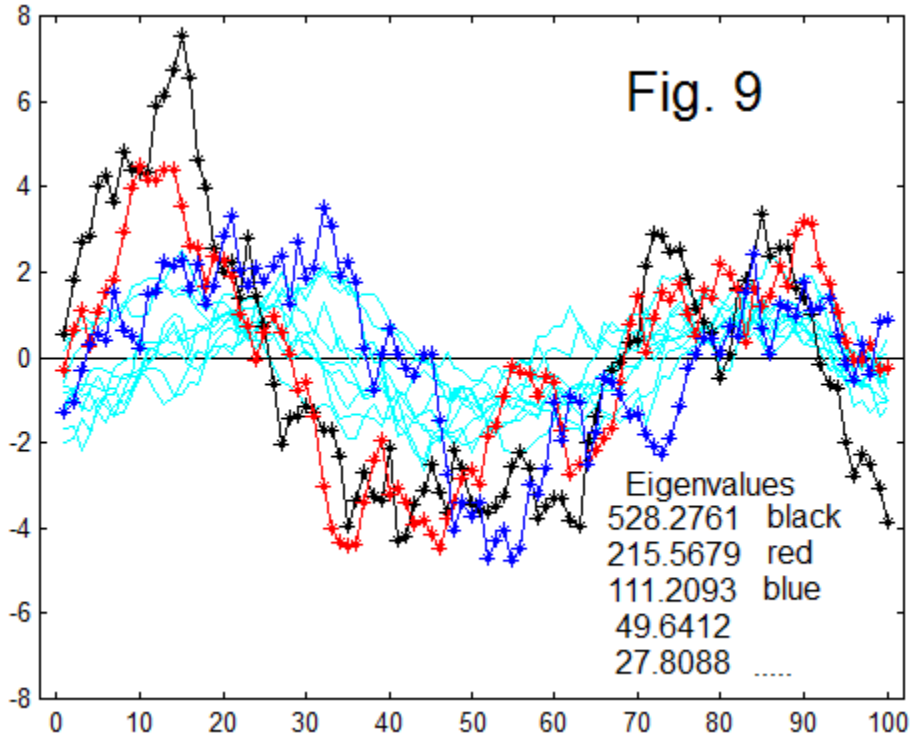
Examples of Sets of Time Signals

At this point we have enough idea about how PCA maybe applied to sets of signals (time sequences) that we can run some experiments. Here we will take certain sets of individually assembled signals and plot the results of PCA using the ***pcax.m*** program or similar. Fig. 7 shows the case were we have 20 exponentially decaying sinewaves, each with added noise. These are plotted in light blue so the result is the blur we expect from the added noise. There the first principal component projected on the basis is the black sequence. Also shown are the 2nd and 3rd PCs (red and blue respectively). However, we also list the eigenvalues, so we see that the first PC is far stronger than any of the others. This is probably just what we expect given the input data set.



A second example is seen in Fig. 8 where we have 15 exponentially decaying sinewaves but there the frequency and the decay constant are varied just a bit (frequency by 2%, decay by about 4%). These are again plotted in light blue so we see the variations, which as we expect, increases as time progresses. Again we have plotted the first PC (black) and the second and third PCs (red and blue). The first (largest) eigenvalue is nearly seven times larger than the second largest, but nowhere as dominant as the one in Fig. 7.





By far the most interesting graph so far is seen in Fig. 9. The input to the PCA here was a hand-selected (eye selected) collection of red noise signals (see AN-384) where we looked at the sequences and choose those that has a significant M-shape. That is,

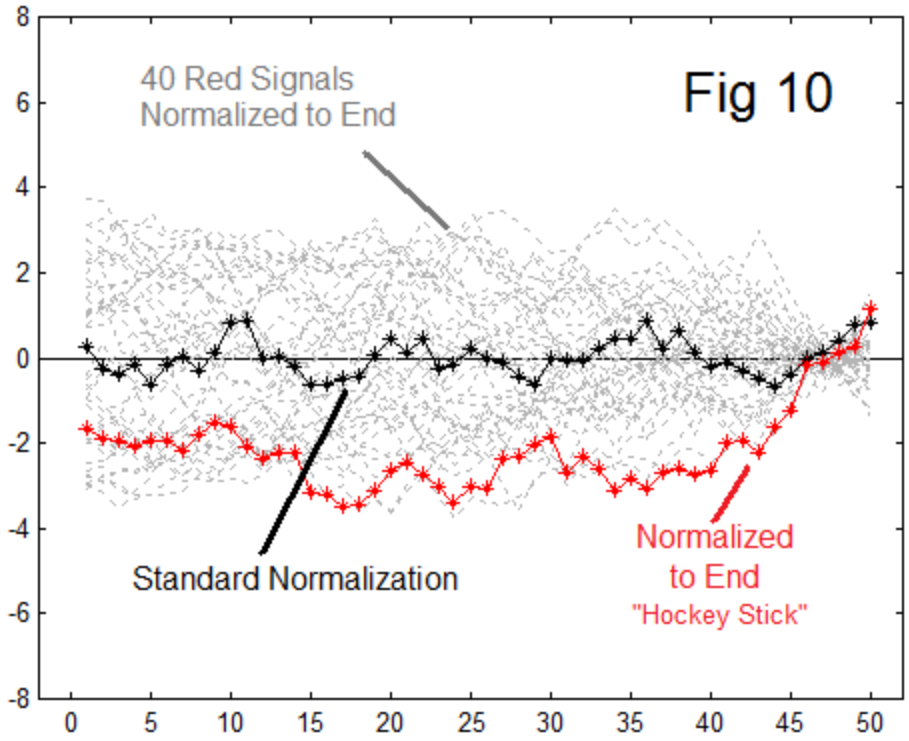
we looked for upward bumps in the first third and the third third, with a dip in the middle. We chose 10 such signals from about 400 total as they went by. Again the light blue in the figure shows the 10 red sequences, and the black, red, and blue PCs as before, are plotted. All three shown do have somewhat of an M-shape, and note that the second eigenvalue is half the largest, so is still very significant.

We might complain that the result is not very "clean" in that we might have hoped for a procedure that somehow figured out that we were looking for an M-shape. Not so. The input set still has considerable variability, and the PCA is trying to account for all the variability in the set and express it in a lower dimensional basis. In many ways, Fig. 9 tells us most of what we need to know about the PCA process. No – it's not telling us how to recover the original data. It's telling us what was in the data with less information.

The Controversial Case of "Short-Centering" PCA

In AN-384 [4], in the context of the "Climate Change" controversy we discussed red-noise signals. We noted the fact that when we truncated a red sequence, it being a "random walk", it could end up far from zero. In consequence, it was fairly essential to remove the mean of the (necessarily truncated to finite length) sub-sequences. We further argued that red-noise signals were a reasonable random test signal here when we wanted to try a procedure on some sort of random noise, to see if the procedure itself had artifacts. If we got out a discernible features from a random input, it is quite essential that similar outputs from supposed real data must be considered bogus.

The climate dispute [1-3] involved whether or not it was proper to normalize the data by subtracting not the mean of the entire sequence (as in our examples and code above), but rather the mean of only part of the sequence (notably the end 10% or so – so called "short centering"). In AN-384, we showed that using this flawed notion of a mean "pinched" the red-noise curves in the region where the mean was taken, in much the same manner that all the red-noise curves were pinched at the very start (starting at zero). This has to make a difference in the PCA result, and it does. It is of course perfectly possible that the mean of the last samples could be very close to the mean of the entire sequence, just by chance. So we can't expect a very noticeable effect in all cases. Remember, the claim is that the procedure itself generated hockey stick shapes. Here there is no selection of input data (as we had in Fig. 9). The output results shown, while selected as noted, are demonstrably typical of perhaps1/3 of the cases. It is also true that in all the cases examined, short-centering had a observably different first PC from full averaging.



In Fig 10, and in all the other examples here, we begin with a set of 40, length-50 red-noise sequences. These we generate with the program such as ***redsignals.m*** from AN-384. The standard PCA is performed with the program such as ***pcax.m*** in this note. The two programs combined are illustrated in the program ***rs.m*** printed below. Note in the code the use of parallel procedures, xs for the standard removal of the mean and ys for the non-standard removal of the mean. Fig. 10 is the result of the very first iteration of ***rs.m*** – and yours may well be the very same (since the seeds of the random generator in Matlab are often common). It is not only the fist curve, but an excellent example. The grey "cloud" in the background is the 40 red signals, normalized to the last 5 samples. Note the pinching of all the curves at the end. The black curve is the resulting first PC if the standard normalization (subtracting the mean of all 50 samples, not just the last five) is used. The red curve is the PCA for the non-standard procedure, and it shows a pronounced hockey stick. This hockey stick is an artifact of the procedure, as advertized by the global warming "skeptics".

To be fair we of course look at more examples, and the second iteration (Fig. 11) is disappointing perhaps, although it does show the pinching at the end, and the great difference between the methods. We need to look at more examples, and at this point we can safely stop plotting the grey cloud in the background, which is almost the same in all trials (way too much detail averaged).

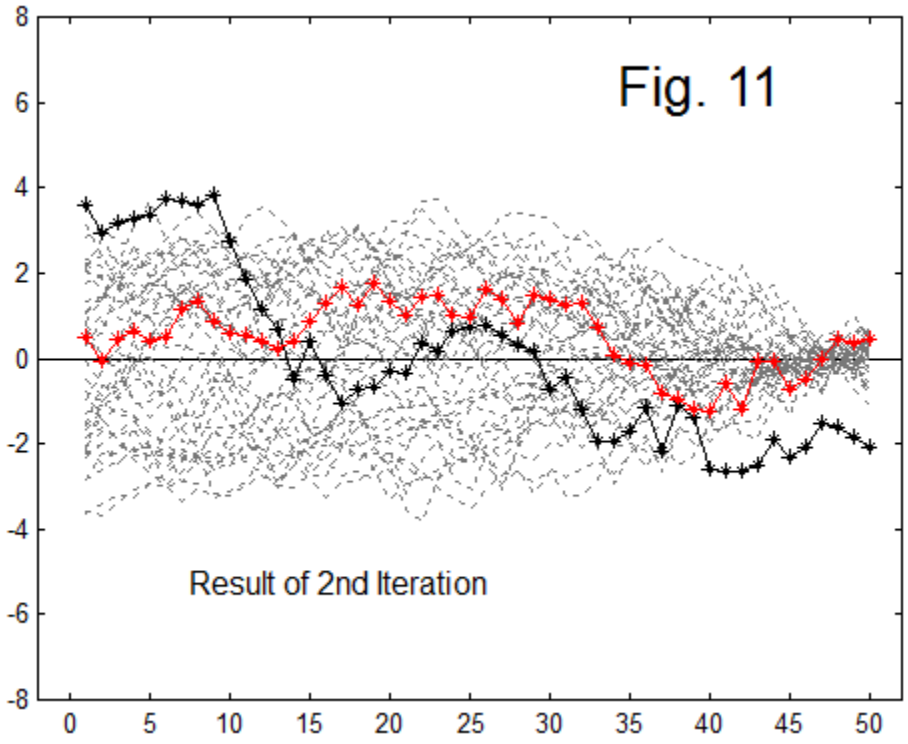


Fig. 11
Result of 2nd Iteration
8
6
4
2
0
-2
-4
-6
-8
0
5
10
15
20
25
30
35
40
45
50

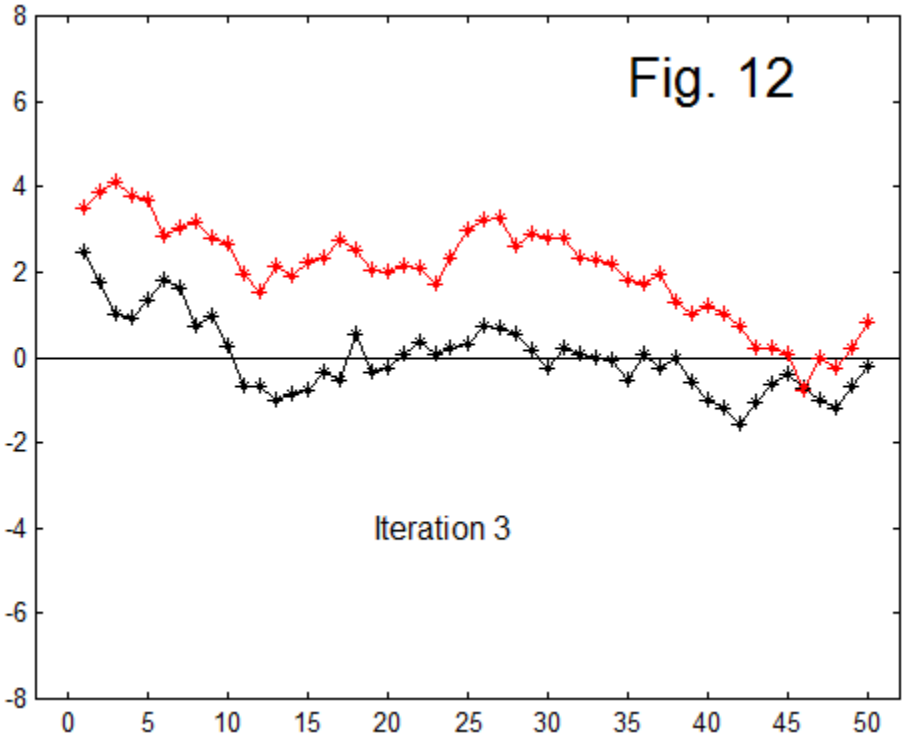


Fig. 12
Iteration 3
8
6
4
2
0
-2
-4
-6
-8
0
5
10
15
20
25
30
35
40
45
50

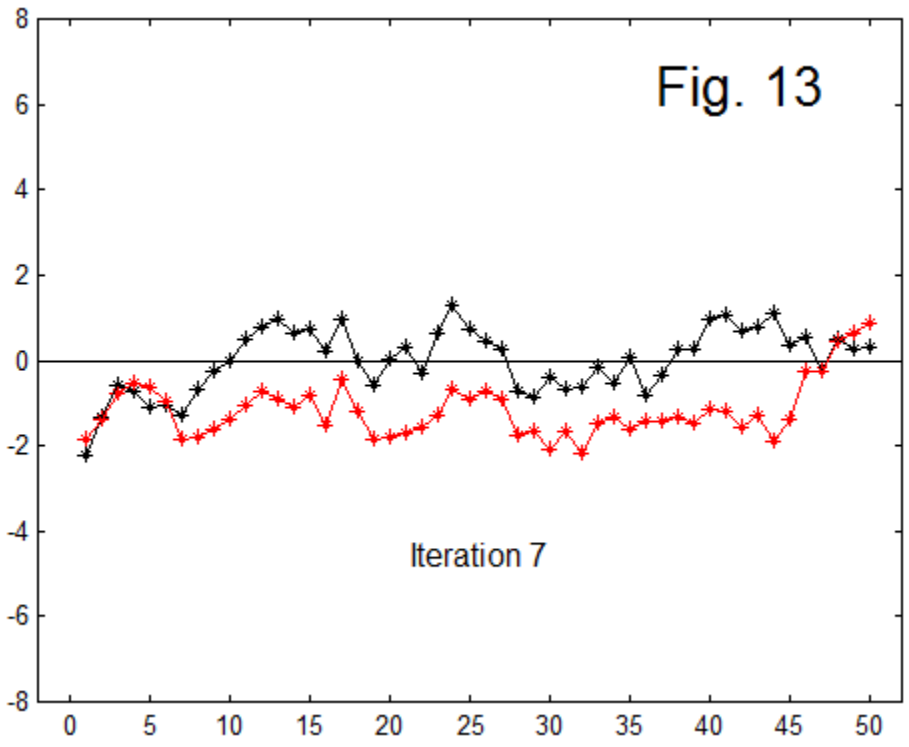


Fig. 13
Iteration 7
8
6
4
2
0
-2
-4
-6
-8
0
5
10
15
20
25
30
35
40
45
50

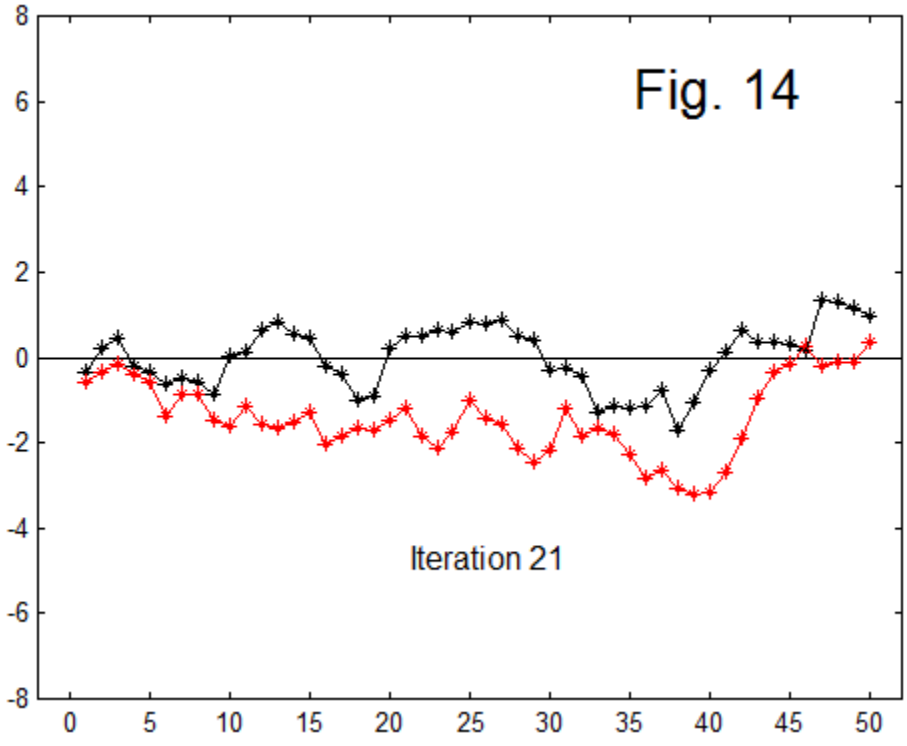


Fig. 14
Iteration 21
8
6
4
2
0
-2
-4
-6
-8
0
5
10
15
20
25
30
35
40
45
50

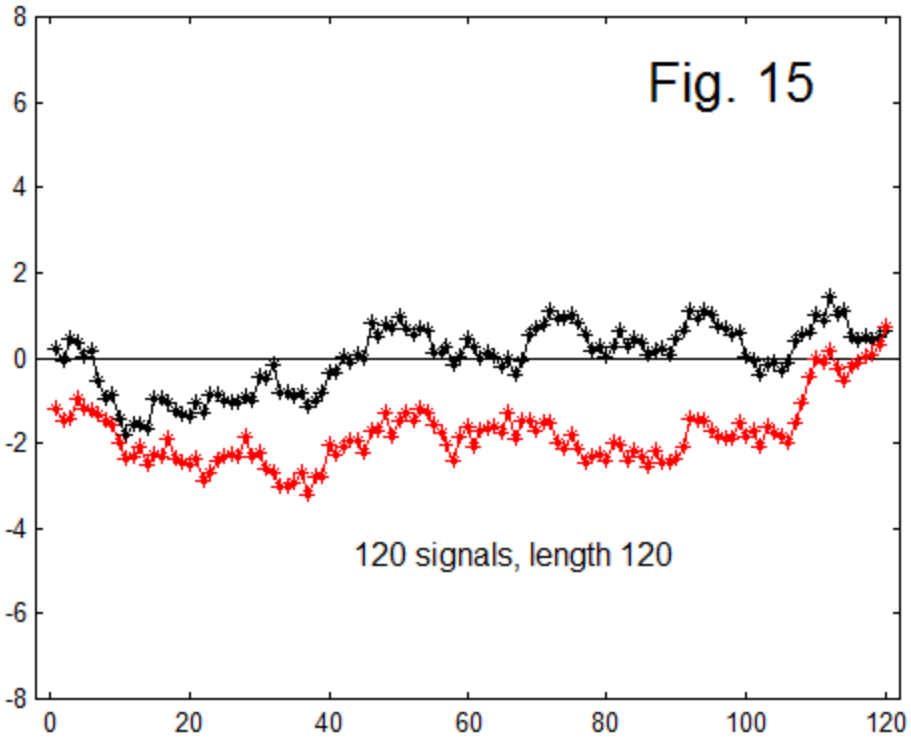


Fig. 12, the third iteration also shows a hockey stick, this time bending down, which does not matter as the PC's are vectors indicating a direction. Many PC's in the literature are upside down. Note as well the upward hook at the left side of the black curve. Much is due to chance here.

Instead of just showing the whole series, we at this point jump to Fig. 13 and Fig. 14, iterations 7 and 21. Fig 13 is an excellent example. Fig. 14 is remarkable in that it is very reminiscent of the actual hockey stick shape, showing a slight decline for most of its length, followed by a blade that even has a slight wiggle. All due to chance. Fig. 15 shows a good example where here we have changed to 120 signals of length 120, with the mean taken over the final 10 samples.

Here we have used only random input – not one sample of actual climate data. Neither have we tried adding to the set artificial hockey stick shapes to "prime" the pump. The reader is invited to run the ***rs.m*** code to fully appreciate that first of all, the chosen method or removing the mean always makes a significant, and usually a large difference. Secondly, while examples without a hockey stick are not uncommon, as we expect, we do not have to wait more than a few iterations for some indication of a hockey stick, and perhaps only ten iterations for a very good example. It really happens.

The above is not new by any means. Here we may have contributed a few useful additions. We reviewed the PCA basics, aiming it at “signal people”, and we have attempted to relate the common tutorials to the necessary time-domain perspective needed here. We have also attempted to add more insight into the actual cause of the biases caused by the “short-centering” – the pinching at the far end shifting the curve, producing HS shapes that are then favored in the PCA. Previously this has been described mainly in words. A few example curves are an immense help in seeing how the shift greatly enhances the variance from the mean. Finally, we (as is our custom) provide the simplest possible demonstrations and the actual code we used, in Matlab.

We have not used any real climate data, nor attempted, as McIntyre and McKitrick did, to match the autocorrelation of real data. They [3] used something called ARFIMA (Autoregressive Fractionally-Integrated Moving Average). I have not examined this, but assume it is related to using a feedback constant less than 1 (Fig. 1 of AN-384) and some FIR feedforward. It would seem that this calculation could be a matter of inverting the power spectrum and fitting the corresponding filter. In any event, it is perhaps useful to avoid complications; because with the use of a fine-tuned setup may come the impression that the phenomenon here is also fine tuned. This is not true, it is a robust phenomenon, hence the use of the simplest of conditions.

This result, as we have certainly tried to suggest, is about the mathematical procedure. It describes the inadequacy of the tool being used. It does not demand that there is no temperature signal in the proxy data. Nor does it claim that there is no hockey stick in the signal. Indeed, the finding is that it enhances any such HS, even to the extent of mining any such signals that are there, even as only normal random fluctuations. The claim that any algorithm would do such a thing perhaps seems unlikely at first. So, experience with our own programs first of all convinces us that it is <u>really happening</u>. The explanation in terms of the shift from the mean, as a visual illustration of the mechanism, is perhaps the key factor in seeing why this was, quite simply, the inevitable technical consequence.

REFERENCES:

[1] Montford, Andrew W., *The Hockey Stick Illusion, Climategate and the Corruption of Science,* Stacey International (2010).*

[2] McIntyre, Stephen and Ross McKitrick, "Hockey sticks, principal components, and spurious significance", *Geophysical Research Letters*, Vol. 32, L03710, doi:10.1029/2004GL021750, (2005):
http://climateaudit.files.wordpress.com/2009/12/mcintyre-grl-2005.pdf

[3] McKitrick, Ross, "What is the Hockey Stick Debate About?", APEC Study Group, Australia April 4, 2005:
http://www.uoguelph.ca/~rmckitri/research/McKitrick-hockeystick.pdf

[4] Hutchins, B., "Fun with Red Noise", *Electronotes Application Note AN-384 Sept 1, 2012*
http://electronotes.netfirms.comAN384.pdf

* This book is rapidly becoming a classic. It is extremely well-documented to the point of being of first-rate academic quality, and is very tightly organized and presented. Indeed, a snapshot of the Hockey-Stick affair. Possibly this was not that hard to do because the material was all more or less available right at the computer. The marvel was that Montford was able to get his mind around all the details. And even more amazing, that McIntyre and McKitrick got their minds around it all in the first place, particularly given the opposition.

Matlab Code for PCA of Red Noise Signals

```
%  rs.m
%  Compare Standard vs. Custom Mean Removal
%  This program generates some of the figures in AN-385

% GENERATE SIGNALS
% 40 length 50 signals - RED Noise
N=40    % number of signals
M=50    % length of signals
n=5     % length of end region
addys=1 % to plot ys with PCs

xorig=2*(rand(N,M)-.5);  % White

% Now Redden
xs=xorig;
for kk=1:N       % rows
  for mm=2:M      % columns
    xs(kk,mm)=xs(kk,mm-1) +xs(kk,mm);
  end
end


% Save copies zs and ys
zs=xs ;   % reserve copy
ys=xs ;   % to be mean adjusted non-standard

%
% ****************************************
% Standard Removal of Mean - xs
mn = mean(xs,2);
for k=1:N
  xs(k,:)=xs(k,:)-mn(k);
end
%
```

```
%
%****************************************
% Modified Removal of Mean - ys
for k=1:N
  rowk=ys(k,:);
  rowk=rowk-mean(rowk(M-n:M));
  ys(k,:)=rowk;
end



% Fig. 1-4 Display
figure(1)
plot([-2 M+2],[0,0],'k')
hold on
for k = 1:N
  plot(xorig(k,:),'b:')
end
hold on
axis([-5 M+5 -2 2])
hold off
title('WHITE')
%
figure(2)
plot([-2 M+2],[0,0],'k')
hold on
for k = 1:N
  plot(zs(k,:),'b:')
end
hold on
axis([-5 M+5 -10 10])
hold off
title('RED as Generated')
```

```
%
figure(3)
plot([-2 M+2],[0,0],'k')
hold on
for k = 1:N
   plot(xs(k,:),'b:')
end
hold on
axis([-5 M+5 -10 10])
hold off
title('RED with Standard Mean Subtracted')
%
figure(4)
plot([-2 M+2],[0,0],'k')
hold on
for k = 1:N
   plot(ys(k,:),'b:')
end
hold on
axis([-5 M+5 -10 10])
hold off
title('RED with Modified Mean Subtracted')


% Divide xs and ys by STD
for k=1:N
   xs(k,:)=xs(k,:) / std(xs(k,:));
   ys(k,:)=ys(k,:) / std(ys(k,:));
end
figure(5)
plot([-2 M+2],[0,0],'k')
hold on
for k = 1:N
   plot(xs(k,:),'b:')
end
hold on
axis([-5 M+5 -5 5])
hold off
title('RED with Standard Mean Subtracted, Divided by STD')
```

```
%
figure(6)
plot([-2 M+2],[0,0],'k')
hold on
for k = 1:N
   plot(ys(k,:),'b:')
end
hold on
axis([-5 M+5 -5 5])
hold off
title('RED with Modified Mean Subtracted, Divided by STD')
%


%  PCA for xs
covxs=(xs*xs');
[Evecxs,Evalxs]=eig(covxs);

Evalxs=diag(Evalxs)';

[Evalxs,ind]=sort(Evalxs);
Evalxs=fliplr(Evalxs);
ind=fliplr(ind);
Evecxs= Evecxs(:,ind);

rxs=Evecxs*xs;


% PCA for ys
covys=(ys*ys');
[Evecys,Evalys]=eig(covys);

Evalys=diag(Evalys)';

[Evalys,ind]=sort(Evalys);
Evalys=fliplr(Evalys);
ind=fliplr(ind);
Evecys= Evecys(:,ind);

rys=Evecys*ys;
```

```
figure(7)
plot([-2 M+2],[0 0],'k')
hold on
if addys==1
   for k=1:N
   plot(ys(k,:),'c:')
   end
end

plot(rxs(1,:),'*k')
plot(rys(1,:),'*r')
plot(rxs(1,:),'k')
plot(rys(1,:),'r')


axis([-2 M+2 -8 8])
hold off

Evalxs(1:5)
Evalys(1:5)
```