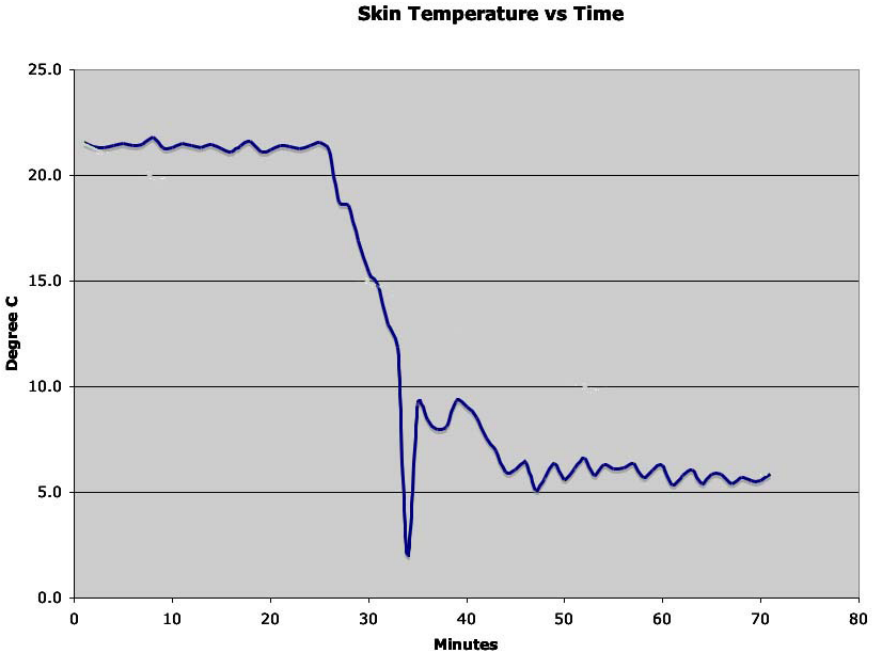
Lecture 1

INTRODUCTORY MATHEMATICAL IDEAS

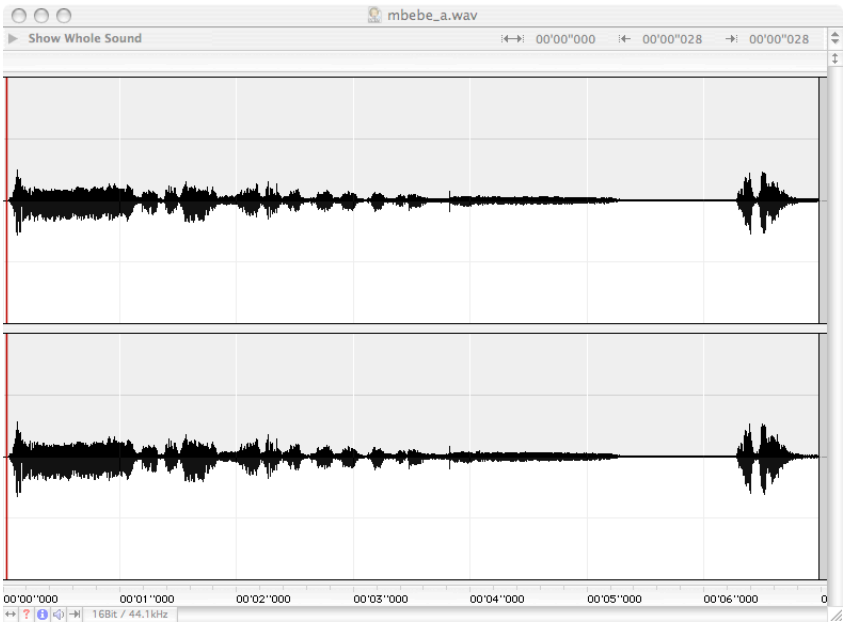
Signals

In this course, we'll be dealing primarily with *time series*, by which we shall mean functions of one variable, with time as an independent variable. The intuition is that at an instant of time of time t, we take a physical measurement m, and the pairs $(t,\ m)$ form a function. There are three major kinds of time series we consider, and we'll begin by giving examples of each.

Continuous Time Series: At each instant of time, we have a measured value. The pairs $(t,\ m)$ form a function $f(t)$, defined for all real numbers. Here are some examples, taken from '*real* ' applications.



Healthy young volunteers were isolated in a temperature-controlled room and the temperature suddenly reduced. The goal was to measure the speed of the subject's response to drastic temperature change.



Introduction to the song 'Mbube' by the Mahotella Queens. The vertical scale is in 'sound pressure level', measured above/below a standard pressure. See c1/signals/ mbube.wav on the CD.

Vanguard Energy VGENX See Fund Family

Data ▸▸

Performance more ▸▸

Growth of $10,000 08-31-04

	2000	2001	2002	2003	2004
Fund	36.4	-2.6	-0.6	33.8	16.4
+/- Cat	5.5	8.1	2.5	1.5	4.9
+/- S&P 500	45.5	9.3	21.5	5.1	15.9

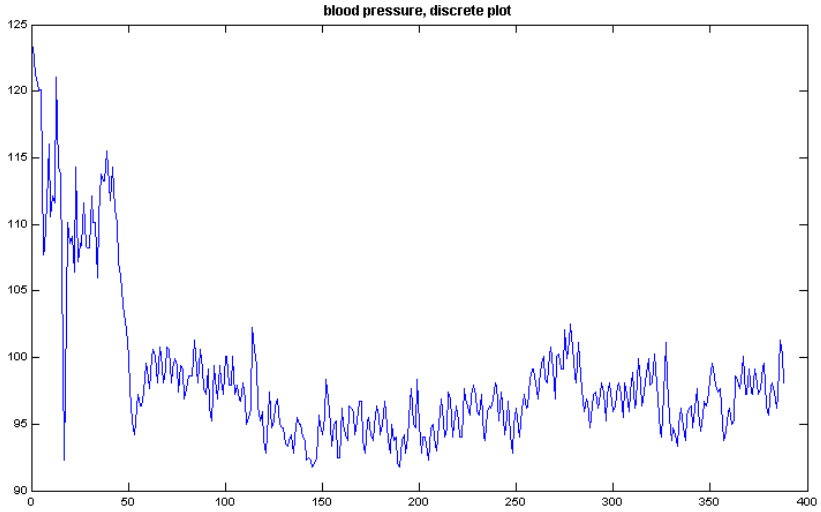
Trailing Returns % 09-24-04

	YTD	3 year	5 year
Fund	25.59	25.27	17.13

Key Stats more ▸▸

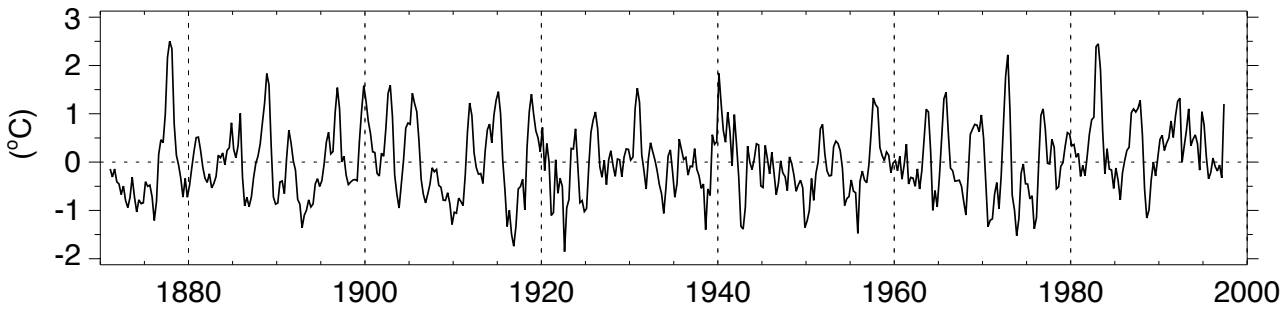
Morningstar Category Specialty-Natural Res	**Morningstar Rating** ★★★★★
NAV (09-24-04) $37.24	**Day Change** $0.37
Total Assets($mil) 4,113	**Expense Ratio %** 0.38
Front Load % None	**Deferred Load %** None
Yield % (TTM) 1.19	**Min Investment** $25,000

Long-term performance of a Vanguard mutual fund, showing steady growth. Note: to get in on this fund, you need to start with $25,000.



Systolic blood pressure (in millibars) as a function of time (seconds). Graph produced in Matlab, using the commands:

```
% load the file
        >load 'sbp.txt'
% file details
        >whos
        sbp 388x4
% the file 'sbp'  has 388 rows and
% 4 columns
%plot only the first column
        > plot(sbp(:,1))
```



The Nino3 sea surface temperature (SST) used as a measure of the amplitude of the El Nino–Southern Oscillation (ENSO). The Nino3 SST index is defined as the seasonal SST averaged over the central Pacific (5°S–5°N, 90°– 150°W). Data for 1871–1996 are from an area average of the U.K. Meteorological Office GISST2.3 (Rayner et al. 1996) Parker and T. Basnett, Hadley Centre for Climate Prediction and Research, UKMO). The data is on a 5° global grid, with monthly resolution from January 1871 to December 1994.
121.75 days) using a least-squares fit.

The data is from Torrence *et. al.*, or on the CD at c1/signals/torrence.pdf.

Power level vs. time, April 26, 1985 in reactor 4, Chernobyl, USSR. It is rare to see a physical system with a vertical asymptote; often it denotes a sudden catastrophic failure of a system.

Periodic Time Series: These are time series which repeat. The primal example is planetary observations: as the earth rotates about the sun, the positions of the stars, and distances of planets to the the sun, all repeat. If time is measured in days, $f(x + 365) = f(x)$ (Actually this is not quite correct, a true solar year is a bit longer than 365 days.

The primal periodic functions are $\sin x,\ \cos x$; their repitition is expressed by formulae such as $\sin(x+2\pi) = \sin(x)$. Because of this relationship, the values of f between 3π and 4π have no information for us that isn't already contained in the values between π and 2π. For this reason, we only need to define $\sin x,\ \cos x$ on the interval $[0,\ 2\pi]$ or, sometimes, $[-\pi,\ \pi]$.

There's a very nice 3D Java applet of planetary orbits at the NASA site.

Discrete Time Series: We remarked earlier that we'd show examples of some '*real*' time series. Why the 'quoties'?

The examples were actual data, taken from the work of real scientists, mathematicians and engineers. The unreality comes in the fiction that **At each instant of time, we have a measured value**. In fact, that would require an infinite number of measurements, and an infinite amount of storage space. The idea that a time series is a function defined on the real line $(-\infty,\ \infty)$ is a convenient fiction, but it doesn't represent what any scientist or engineer does in their work.

The best any real time series can be is just a *sample* of all the infinite amount of data out there. The Vanguard mutual fund prices were recorded at the end of each trading day; we say the *sampling rate* for Vanguard is one sample per day. The El Nino data was taken once a month (sampling rate T = one sample/month). The systolic blood pressure was taken once every $1/256$ seconds (sampling rate $T = 1/256$ seconds).

There's another way traditionally used to describe the sampling process: the number of samples per second, referred to as the *sampling frequency*. Then, if the sampling rate T is measured in seconds, the sampling rate ν is measured as samples/second, and is $\nu = 1/T$. In these units, the systolic blood pressure has a sampling frequency of 256 samples/sec. And the music sampling rate is determined by industry standards for music recorded on CD's: 44100 samples/second.

The sampling frequency, measured in samples/second, is used so often that it is an international unit, the *Hertz*, abbreviated Hz or hz. By definition, one Hz = one sample/second. And CD music is sampled at 44100Hz.

We're going to use a unit-free way of writing sampled time series. If $f(t)$ is the physical quantity we're measuring, and our sampling rate is T, then the n^{th} sample will be written as

$$a_n = f\left(\frac{n}{T}\right)$$

Checking the units, n is the sample number; T is defined as $samples/time$, whence n/T has units of time. This is the correct unit for the domain of f. One of the consequences of this notation is that while the original function f was defined on the real line, or possibly on just $[0,\ 2\pi]$, the a_n form a function on the integers, $\mathbf{Z}$. We will also use the notation $\mathbf{Z}_K$ to denote a function defined for $0 \leq n \leq K-1$.

Downsampling Lab As we said above, if we tried to measure a quantity for every instant of time, we'd get an infinite amount of information. Sometimes even sampled data is too much data; we want to compress it. Certainly the easiest method of compression is *downsampling*: we simply discard every other data point.

Fortunately, this is really easy to do in Matlab. To demonstrate the technique, start with a simple vector:

```
y=[1:8]
y = 1    2    3    4    5    6    7    8
```

Now let's skip every other element

```
>> z= y(1:2:8)
z = 1    3    5    7
```

We've downsampled the music. For this lab, you're you use the linked file, queen.wav (Queen Latifah, from the Chicago soundtrack; c1/signals/queen.wav on the CD). The music has been sampled at 44100 Hz., converted to mono, and saved in the wav music format. You can use the command

```
x=wavread('queen.wav');
```

to get the sound into Matlab. Now downsample by a factor of five, then ten, and write out the results as new wav files. Play the original and the downsampled versions. What's the difference in the sounds? In particular -- this is more relevant for the sound reduced by a factor of five -- note the different effects of downsampling on the instruments compared to the voice.

For now, it's enough note that sampling rate affects the quality of a signal. To do any serious work, we'll need to quantify "quality" and we'll need to understand precisely the trade off between quality and sampling rate.

Other Signals: The examples we've seen are all time series, but these are not the only kinds of signals. We'll also be looking at some three dimensional signals, generated from pictures. The linked file, tea.jpg, or c1/signals/tea.jpg on the CD, is a jpeg file of size 315 pixels high by 440 pixels wide. It should easily be representable not as a single series, but as a 315 by 420 matrix. The value of the matrix *m(i,j)* ought to tell you the color at pixel *(i, j)*. Read the jpeg into Matlab via the commands:

```
a=imread('tea.jpg', 'jpg');
>> whos
a      315x420x3
```

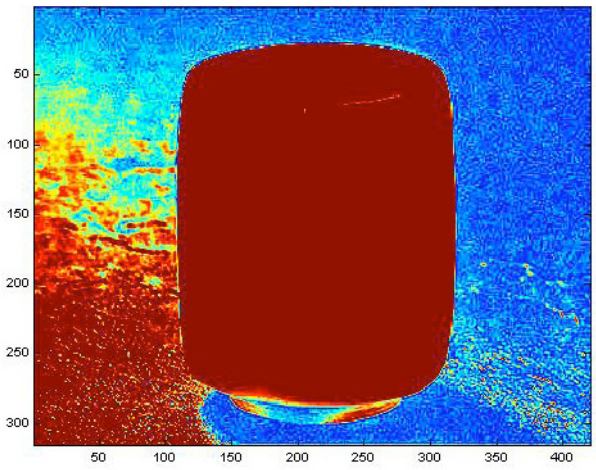
which produces something unexpected: we don't get a two-by-two matrix, specifying one color for each point in a plane; we get a --- an object that isn't a matrix at all; it's some kind of tensor.

Matlab re-interprets it as an image via the command:

>> image(a)

whereas if we create an actual two by two matrix,

>> b=a(:,:,1);
>>image(b)



It's not hard to guess what's happened: we assumed color is one dimensional; in fact, most color models are at least three-dimensional. This jpeg uses the RBG color schema, and Matlab therefore has used the a(-,-, 1), a(-,-, 2), a(-,-, 3), channels to store the red, blue and green information in the picture. This isn't quite what we thought it ought to be, but it is a signal, though not a time-varying signal.

Lab Problem Read the image tea.jpg (or c1/signals/tea.jpg on the CD) into Matlab, downsample it, and save the downsampled image as a jpg.

Images, one sees, tend to be large files: they two dimensions to specify the position of a pixel on the page, and three more to specify the color. A compression scheme might deal with this by trying to reduce the number of dimensions of colors, using two main ideas. First, the human eye doesn't see all colors equally well. A psycho-visual compression scheme would rank the colors in terms of their visibility, and assign less "weight" to the less important colors. Second, the eye does not detect rapid changes in hue and saturation of a color as well as it detects intensity.

This suggests a change of basis for color information, from RGB to a system that is in better sync with what properties are important to the eye. The basis used in JPEG is the luminance-chrominance basis, referred to as YCbCr. Y represents the luminance, or brightness of the image, while Cb and Cr are two parameters representing color, or chrominance. Like all good changes of bases, the map between RGB and YCbCr is linear:

Like any good change of basis, the transformation from RGB space to YCbCr space is linear:

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.0813 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Lab Problem: Take the 'tea' jpeg and transform the RGB info to YCbCr. Then, as we did with the red tea cup above, display each of the channels Y, Cb, Cr.

Our source on all things chroma tells us,

> . . . the luminance information contains most of the detail perceived by the human eye, while the overlying chrominance color information can be fuzzy without causing any serious image degradation. The chrominance information really just defines a tint overlaid on a grayscale image.
>
> This means that JPEG doesn't necessarily need to keep all the chrominance information, and the specification allows for a feature known as "decimation" in which part of the chrominance information is simply thrown away. For example, compression can be increased by only sampling every other horizontal pixel in a chrominance block, which cuts the number of chrominance bits in half. This is known as "horizontal decimation", using a decimation factor of 2, and results in one decimated 8x8 chrominance block for every two luminance blocks.
>
> Even more compression can be achieved by only sampling every other horizontal and vertical pixel in a chrominance block, which cuts the number of chrominance bits to a fourth. This is known as "horizontal and vertical decimation" using a factor of 2, and results in one decimated 8x8 chrominance block for every four luminance blocks.

Lab Problem: Decimate the chroma channels you obtained in the previous computation. Then use the inverse of the above matrix to change the decimated picture back to RGB space. How does it compare to the original?

One thing you'll notice right away, if you type a whos: the size of the decimated & converted image is still the same; there hasn't been any real compression. JPEG doesn't in fact achieve its compression this way; the compression comes when the numbers defining the image are arranged. At that point, a scheme like Huffman entropy encoding is used, and the strings of zeros we put there when we decimated can be encoded using less space. That's the first trick JPEG uses; we'll discuss other tricks in later application notes.

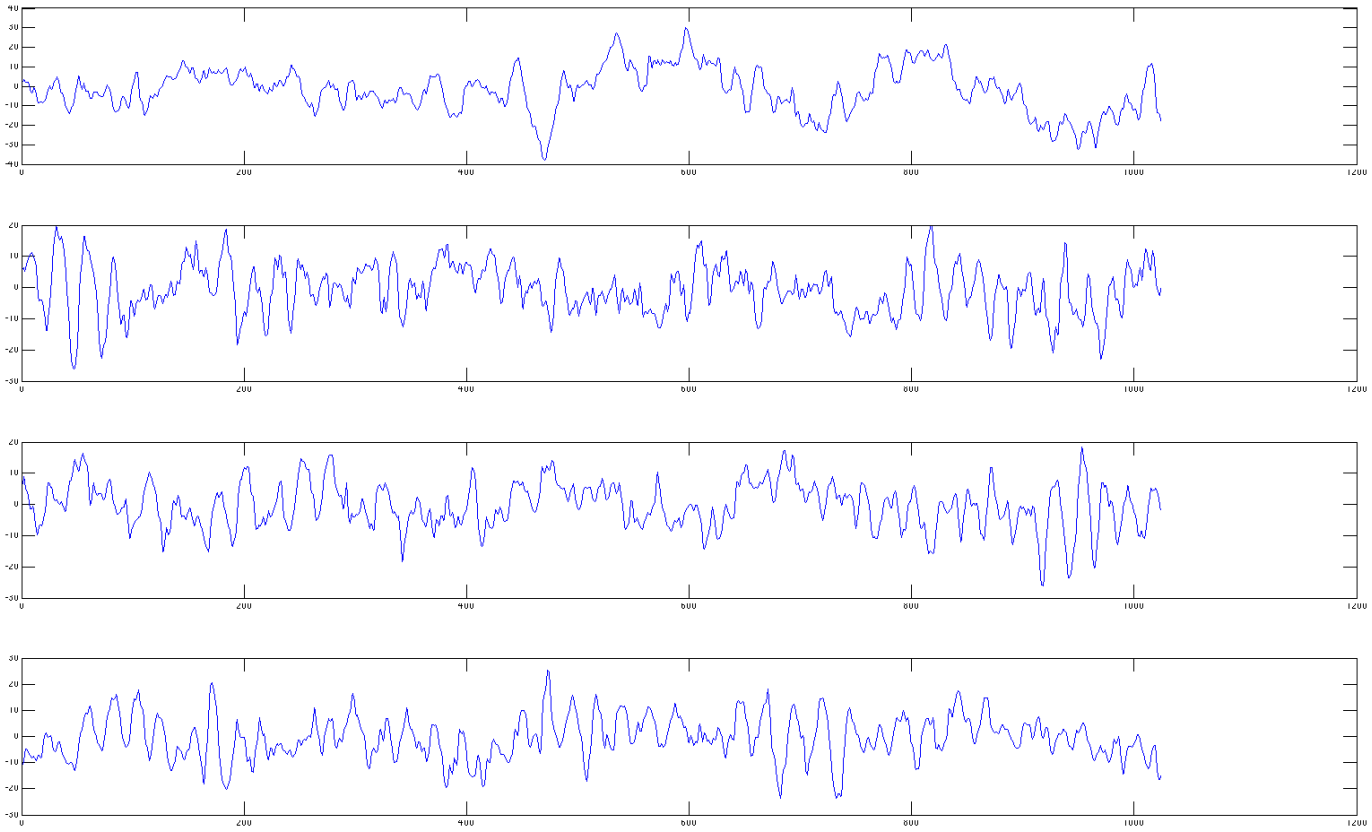
EEG Signals: The EEG, or electroecephalogram, records electrical signals on the scalp. The hope is that the signals detected have some relation to the processes occurring in the brain, thus providing a non-invasive method to determine brain function.

Given the size of the brain and the weakness of electrical impulses reflecting nerve firing, the EEG does not carry a lot of information, and much of that is information necessarily averaged out over large regions of the brain. Nonetheless, one uses what one has, and the goal of electroencephalography is to squeeze as much information as possible from the EEG.

The example below is taken from online material for one of our reference books, *Bioelectrical Signal Processing in Cardiac and Neurological Applications*. The data here is taken from a study of epilepsy. As the authors describe it,

Patients with suspected epilepsy usually undergo an EEG investigation which involves photo stimulation. This technique is used to trigger epileptic potentials by repeated light flashes (possibly resulting in a repetitive spike-wave pattern which occur about 3 times per second if the patient suffers from epilepsy). The lamp, which is positioned approximately 15 centimeters from the patient's face, flashes at a rate that varies from 10 to 30 Hz. In the present project, a one-channel EEG is first recorded without photo stimulation, and then followed by recordings obtained at stimulation frequencies of 10, 20 and 30 Hz; thus, four recordings were made in total. All recordings have a length of four seconds and were sampled at a rate of 256 Hz with an amplitude resolution of 1 microVolt. The electrodes were positioned such that cortical center which processing visual information is beneath (posterior part of the parietal lobe).

Here are the four channels, plotted in Matlab. It's clear that the flashing light is having some kind of effect on the brain ...



Lab Project: Load the file eeg.mat into Matlab, and reproduce the above plot, but with proper titles for each graph, and with each graph a different color.

By the way: the universal source for heart or brain signals online is Physionet.org. Check it out!

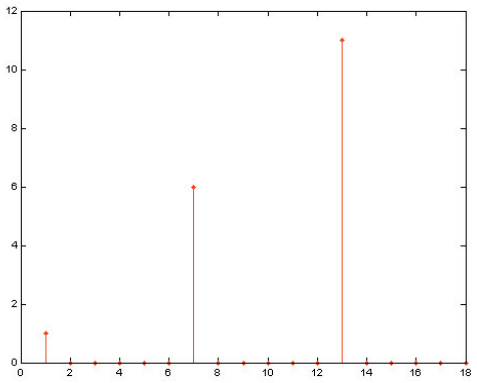
Upsampling Lab Upsampling should be the opposite of downsampling, but in practice, downsampling throws same data away, and that data can't be recovered. It can, however, be replaced with other data; this lab explores how.

We begin with some Matlab constructs: first start with a time series *a* and downsample:

```
>> a=[1:15];
>> b=a(1:5:15)
b = 1     6    11
```

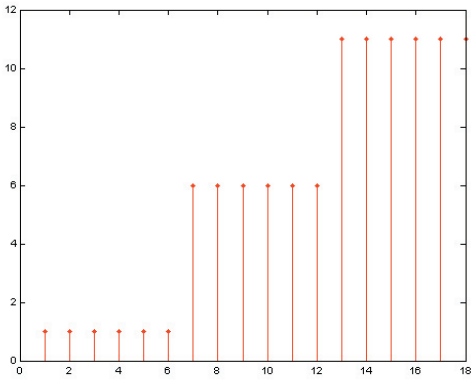
In the first example,we pad this out with zeros:

```
>> f=ones(1,5)

f =  1     1     1     1     1

 >> c=[ b(1) 0*f b(2) 0*f b(3) 0*f];
>> stem(c, '.r')
```



Now pad the empty spaces out with the last b-value

```
>> c=[ b(1) b(1)*f b(2) b(2)*f b(3) b(3)*f];
>> stem(c, '.r')
```



In engineering terms, this last is called a 'zero-order sample-and-hold' scheme. It turns out to be very easy to implement with circuits, so is applied widely. Your first task is to implement each of these schemes with the music file 'queen.wav' you downsampled earlier. Of course you'll need to find a substitute for the construction

```
c=[ b(1) b(1)*f b(2) b(2)*f b(3) b(3)*f].
```

There's also linear upsampling; you'd have to do a linear interpolation between b(1) and b(2), say, but stretch that out over four samples. Implement this as well; how does it sound? Which of the two upsampled files has better sound?

One of the goals of the course is to give a quantitative measure of which sound is better, as a prelude to understanding why. This in turn gives us the ability to improve the sound quality, with new techniques.

Lecture 1

INTRODUCTORY MATHEMATICAL IDEAS

Vector Spaces

As defined in class, the collection of functions $l^2(\mathbf{Z})$, $l^2(\mathbf{Z^2})$, $L^2(\mathbf{Z_K})$, $L^2(\mathbf{R})$, $L^2[0,\ 2\pi]$ are all vector spaces (and in fact inner product spaces).

Homework Problem Show that $l^2(\mathbf{Z_K})$ is finite dimensional, by finding an orthonormal basis. Show that neither of $L^2(\mathbf{R})$, $L^2[0,\ 2\pi]$ is finite dimensional, by finding an infinite number of orthogonal vectors in each. Why does this prove it is not finite dimensional?

The question here is what good a vector space structure does us. For example; if we have $p, w \in l^2(\mathbf{Z^2})$, with each of p and q representing a picture, what in the world could $p+q$ represent? Who would want to add pictures?

We'll start with something simpler: sounds. Notice that if each of p and q are vectors in $l^2(\mathbf{Z^2})$, with $p(n)$ representing sound pressure at time n, then $p+q$ is defined as $(p+q)(n) = p(n) + q(n)$ – that is, the two sound pressures are added at the same time. If $p,\ q$ come from music, $p+q$ represents two pieces of music being played at the same time. And forming linear combinations $\alpha p + \beta q$ is a process musicians call *mixing* tracks. Here's an example, based on Apple's Garageband mixer. Notice there are four 'tracks', each one a named instrument. This means we have four vectors, $p,\ q,\ r,\ s$. The volume of each instrument (circled in red for two of them) represents the coefficient in the linear combination. Here; the volumes are all pretty much equal, at half way, so we'll call the coefficient $\frac{1}{2}$. Then

$$\text{Shufflin}' \text{ Guitar Blues } = \frac{1}{2}p + \frac{1}{2}q + \frac{1}{2}r + \frac{1}{2}s$$

It helps to hear the sound: here, or as c1/vectors/blues_a.wav on the CD.

I want to change the tune: less drum, less bass and more harmonica. Set drum to about a tenth; bass to a third, guitar a half and harmonica to nine-tenths. Then

$$'\text{Toots}' \text{ Davis Blues } = \frac{1}{10}p + \frac{1}{3}q + \frac{1}{2}r + \frac{9}{10}s$$

Hear it here; notice how the bass and drums are almost gone, but now the harmonica stands out. Good mixers are well-paid professionals; they can make or break a song. And it's all in the linear algebra (well, some).

Music seems a very special case; after all, we don't see two images at once, so we can't add them. It turns out this is misleading. The next example will seem simple and very artificial; we'll see later it has a great deal of depth and is very general.

I have a 1997 car; I bought it with a warranty, but eventually (like in another year) it is going to start breaking down. Ignition, transmission. The dreaded Timing Belt – we've all been there. The trick is in deciding when to sell and get a newer car. Here's a very simple rule my friend Kate uses: sell when the monthly cost of repairs is bigger than the cost of new car payments.

Easy enough; you can look up new car payments online, but how do you estimate the monthly cost of repair? The last visit to the garage was $710, but that was my 75,000 check-up, and doesn't count. I don't pay that every month.

The tricky word I used above is *estimate*; I mean it in the sense of statistics – I want a method of predicting what my future expense will be. A typical statistical estimator would be an average; I could take the average of my monthly costs at the end of each year. The disadvantage is that if my costs go sky-high in the middle of the year, I won't know until six months later.

I need to keep on top of those expenses, which means I should probably compute my average every month. Here's what I'll do: at the end of every month, I'll average the cost of repairs in the preceeding four months. As soon as that gets too large, I'm off to the car dealer.

This computation is called *a moving average*, and it has a very easy formula. If c_n is my repair cost in month n, the moving average for month n is

$$a_n = \frac{1}{4}c_n + \frac{1}{4}c_{n-1} + \frac{1}{4}c_{n-2} + \frac{1}{4}c_{n-3}$$

This doesn't look very like a linear combination, but I can fix that. Define new vectors as

$$d_n = c_{n-1};\ e_n = c_{n-2};\ f_n = c_{n-3}$$

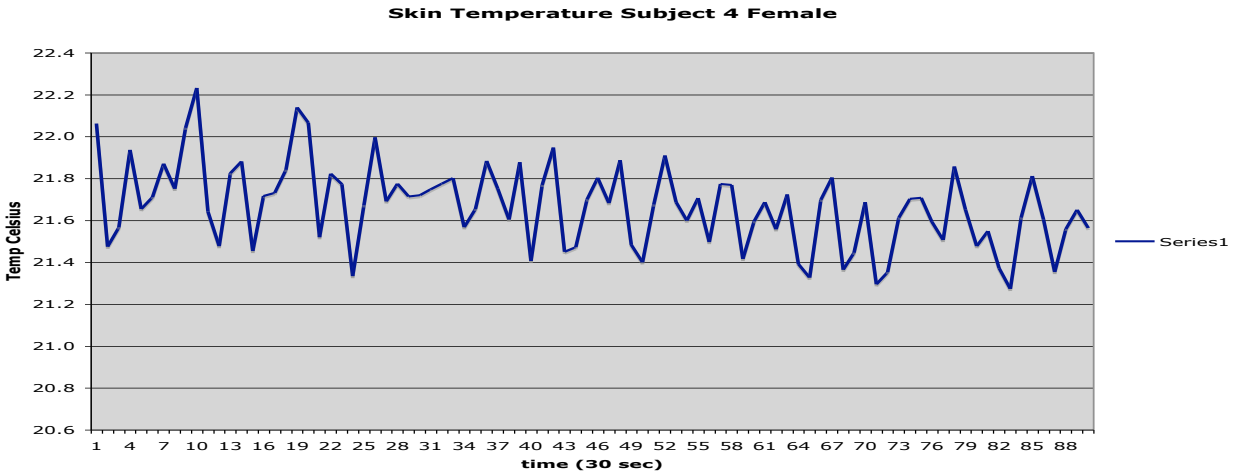
Then

$$a = \frac{1}{4}c + \frac{1}{4}d + \frac{1}{4}e + \frac{1}{4}f$$

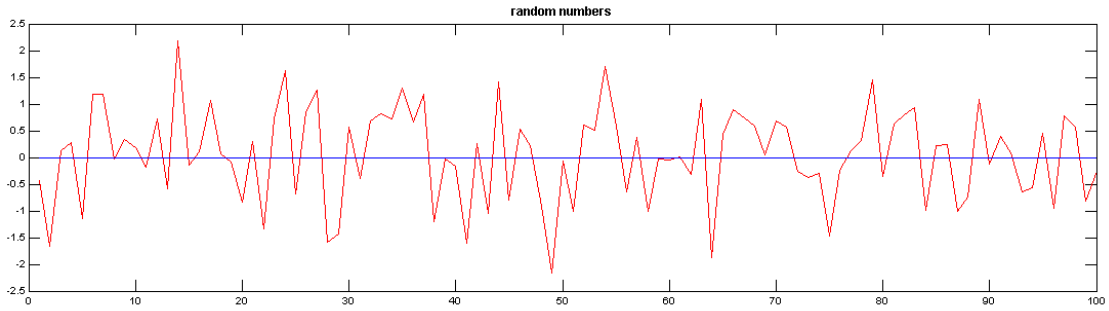
Side Remark: Notice that $d_1 = c_0$, and, as there is no c_0, d is going to have a hard time existing. In fact, I can't compute the moving average until at least four months have gone by. This little nuisance is an example of an *edge effect*, and it will be an issue throughout the course. The easiest way to fix this one is to pad c with zeroes from $-\infty$ to ∞. Oddly enough, this method is called *zero padding*.

We said this is also important. Why? Here's another actual example, from our consulting work.

A volunteer is seated in a temperature controlled room, and her skin temperature is recorded every thirty seconds (.0333 Hz sampling frequency). Here's what the temperature graphs as:



Compare with a graph of a uniformly distributed random variable:



All data collection has a certain randomness to it. Even in a temperature controlled room, there are air currents; if a subject sweats the electrode/skin contact can weaken; if a subject shivers, the electrodes can move. All these effects are lumped together under the term *noise*; they are in contrast to the true body temperature, which we call *the signal*.

Conceptually, we model this as

$$recorded\ data\ =\ signal + noise$$

Two points to note here:
i) We've used the vector space structure of signals, to perform the addition
ii) Averaging preserves the sum:

$$\text{Average}(recorded\ data)\ =\ \text{Average}(signal) + \text{Average}(noise)$$

This is significant because probability theory tells us that as the number of points in the moving average increases, the term Average(*noise*) tends to zero. This allows us, in theory anyway, to recover a signal that's been corrupted with noise.

Lab Example: The goal here is to take a signal (spell.wav), add a random noise, and then use a moving average to reduce the noise.

This produces a random signal:

```
>> r = randn(1,100);
```

And this produces a length-four moving average:

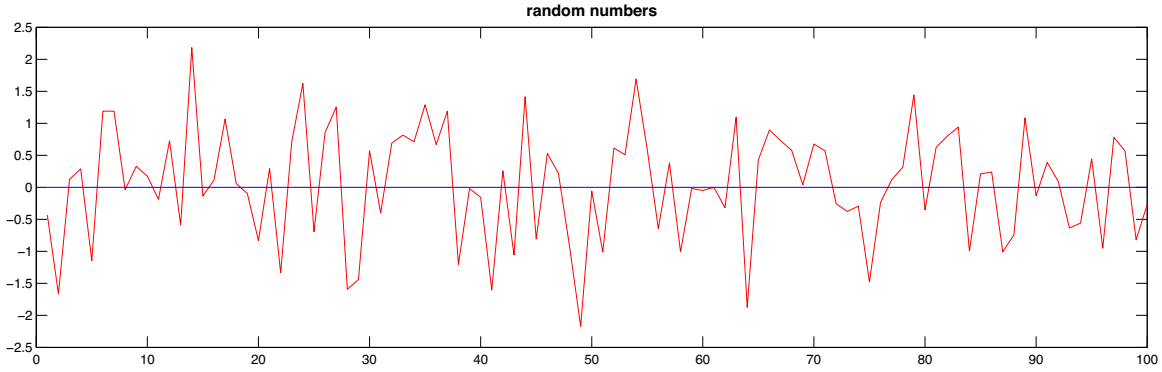
```
>> a=filter(.25*[1 1 1 1], [1], r);
```

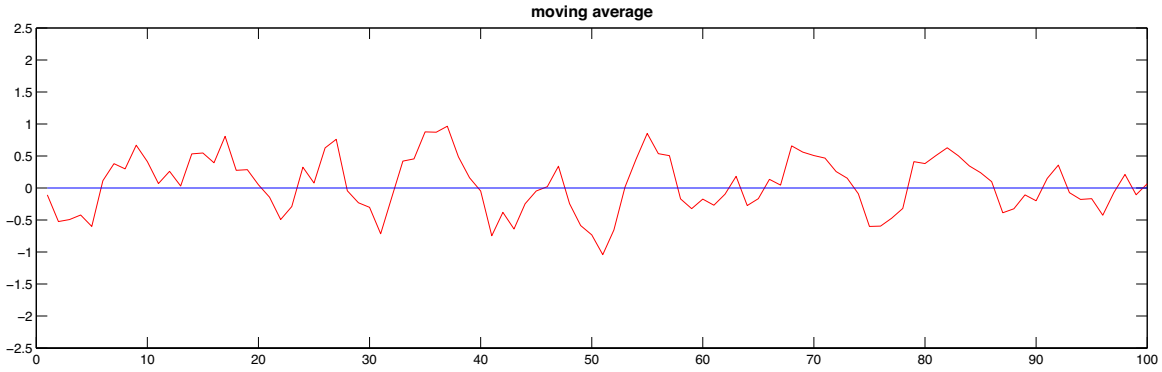
Note: if you don't have the Matlab Signal Processing Toolbox installed, you'll have to program the moving average on your own.

The following code produces a stacked graph of the two signals:

```
>> subplot(2,1,1), plot(r, 'r'), title('random numbers')
>> hold
Current plot held
>> subplot(2,1,1), plot(zeros(100,1), 'b'), title('random numbers')
>> a=filter(.25*[1 1 1 1], [1], r);
>> subplot(2,1,2), plot(a, 'r'), title('moving average')
>> hold
Current plot held
>> subplot(2,1,2), plot(zeros(100,1), 'b'), title('moving average')
```

And here are the graphs:



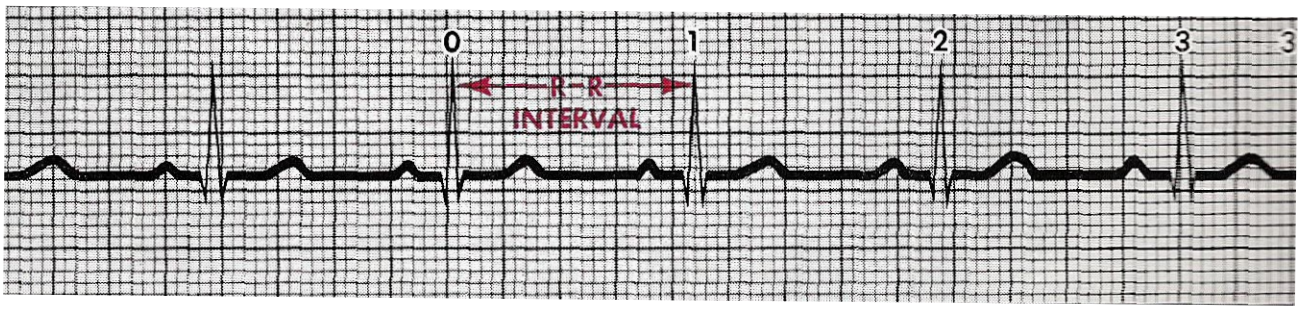


How would you describe the difference between the two graphs?

Your task: read spell.wav into Matlab, add noise, use a moving average to denoise, and save the noisy and denoised version. Small hint: '*r*' is way too noisy, as you can check by trying it. *r*/50 might work better.

The moving average did a good job of removing noise, but if you have a good ear, you'll notice that the singer's voice was changed a bit as well; some of her highs notes have been muted. That's the downside of moving averages; they reduce noise, but they take a bit of signal with them. Here's an example, again from our research, where moving averages don't help much.

The graph below is the data from an electrocardiogram (ECG or EKG). Electrodes are attached to a person's chest, and volage differences are measures as the heart beats. Since the muscles of the heart operate by movement of ions, the beat itself can be detected when there are large currents between electrodes. In the picture below, the tiny upward bump is called the P-wave; it corresponds to contraction of the muscles in the atrium of the heart. The large spike is called the R-wave; it corrreesponds to contraction of the muscles in the ventricles of the heart; they do most of the pumping; in fact the atrium serves as a kind of fuel injection system for the heart. However, the atrium can become dysfunctional and all you may notice is a slight shortage of breath. If the ventricles mis-function, you may die within a minute. Medically, one wants to track the ventricular contractions, and the distance between R-wave spikes measures the number of beats per minute.



The EKG above was taken from a patient lying down in a hospital. If you've ever seen *ER* that's what you expect a good clean EKG to look like. The problem comes when you want to measure EKG in order to monitor the health of people with heart problems, who aren't in the hospital, but are going about their daily lives. Then muscles other than the heart give off electrical signals, and corrupt the EKG. Here's an example of an EKG from an otherwise healty young woman who just happens to be exercising:

Which peak is the actual beat?

To finish the lab, download the file ekg.txt, plot it, and then compute a moving average and plot that. Has the moving average helped in picking out the R-wave spikes?

Note: to read the ekg into Matlab, use

```
>> load 'ekg.txt' -ascii
```

We want to finish with a few remarks about moving averages.
i) The moving average is a linear map $T : l^2(\mathbf{Z}) \to l^2(\mathbf{Z})$. (The linearity isn't very challenging; can you prove that the range of T is contained in $l^2(\mathbf{Z})$?
ii) The map T is called a *filter*. It's easy to see why – filters are supposed to remove the bad stuff – noise in this case – and keep the good stuff, the signal. We said T was a filter – we didn't say it was a great filter, and we are going to be trying to find better filters. One thing we'll find, though: no filter is perfect.
iii) The map T is *time-invariant*. We'll hold off until later on a precise definition, but roughly it means this. The filter does the same kind of operation to a signal at one instant as at the next. If you'd started the whole sequence a half-hour earlier, the map T would produce the same kind of output – except that you'd have to look a half-hour early in the output.

A good portion of this course will be concerned with the properties of linear, time-invariant filters of one kind or another.

Lecture 1

INTRODUCTORY MATHEMATICAL IDEAS

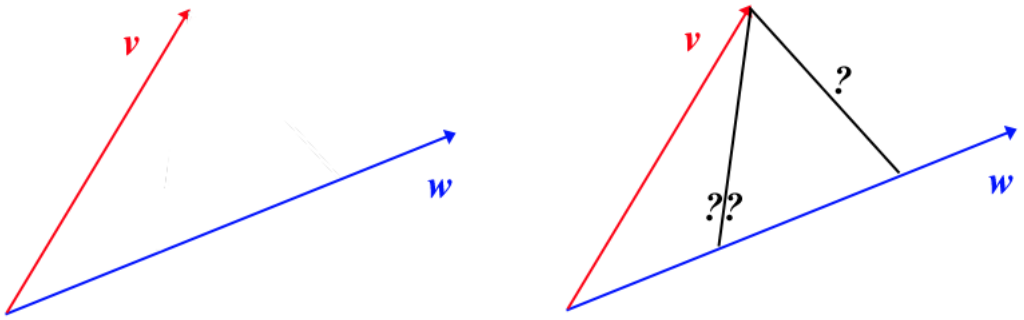
Projections

In this section, we will relate compression and filtering to the theory of vectors and matrices, or, in a basis-free way, to vector spaces and linear transformations.

Start with a vector space V– it could be the collection of all music signals, or our spaces l^2 or L^2 or just plain old three-dimensional Euclidean space $\mathbf{R}^3$. Now take a subspace W. It's supposed to represent something like all the music that's been compressed, or all the signals where the high frequencies have been filtered out. We'll think of the process of compression as a mapping $P : V \to W$. This isn't so obviously a linear map, at least not in all examples. In mp3, for example, the compression depends on the psychoacoustic structure of the ear, and this is NOT linear. However, there are good examples – say filtering – where the map P is linear. If P simply eliminates high frequencies, you can do that just as well to $s_1 + s_2$ together as to s_1, s_2 separately. So we shall assume in this section that P is linear.

Now, how should I compress? To answer the question, I have to have some measure of what good compression is – I need a metric, and it might as well be a norm $||v|| = \sqrt{(v,\ v)}$ coming from the inner product. One way to think about compressing a signal v, then, is to think of sending it into a $w \in W$, where w has lost as little information as possible about the original v. In terms of the metric, we want $||w||$ to be as close as possible to $||v||$; another way of saying that is that we want to find the $w \in W$ that minimizes $||v - w|| = ||v - Pv||$. This is a calculus problem . . .

Let's look at a simple case, $V = \mathbf{R}^2$, two-dimensions, and W is a subspace generated by one vector, w. Then a generic element of W is of the form rw, where r is a real number. My calculus problem is this: given a vector $v \in V$, find the number t that minimizes $||v - tw||$. Here's the set-up, on the left; on the right a couple of really bad guesses are drawn.



We see that $||v - tw||$ is minimized when $v - tw$ is perpendicular to w. Then it's easy to solve for t. As in the picture below, tw is the adjacent side of a right triangle with hypotensuse $||v||$ and angle θ between v and w. Then

$$\cos\theta = \frac{\text{adjacent}}{\text{hypotenuse}} = \frac{||tw||}{||v||}$$

whence

$$t = \frac{||v||\cos\theta}{||w||}$$

Since $Pv = tw$, we get

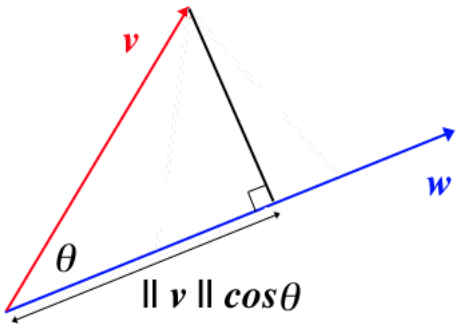
$$Pv = ||v||\cos\theta\ \frac{w}{||w||}$$

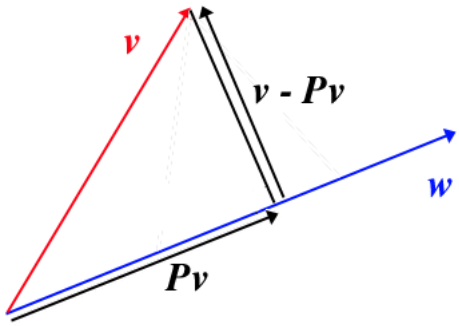
But $(v,\ w) = ||v||\ ||w||\cos\theta$, hence

$$Pv = \left(v,\ \frac{w}{||w||}\right)\frac{w}{||w||}$$

Oops – we should have taken w to be a unit vector; we knew, if we'd thought about it, that Pv doesn't depend on the scale of w – after all, $-\infty < t < \infty$ pretty much eliminates scale from tw.

So: if w is a unit vector, then $Pv = (v,\ w)w$. We also learned that the minimum occurs when $v - tw$ is perpendicular to w.





Homework Problem: Now try it in general – no pictures. Use the fact that the t which minimizes $||v - tw||$ is the same t which minimizes $||v - tw||^2 = (v - tw,\ v - tw) = ||v||^2 - 2t(v,\ w) + t^2||w||^2$. Now use calculus (on t) to show the minumum value of t is $(v,\ w)/||w||^2$, and that therefore $Pv = (v,\ w)w$. Notice that we get, for free, that projection P is a linear map! And, now you're done with that – why not show that $v - Pv$ is perpendicular to w? This property gives its name to P; it is called "the orthogonal projecton of v onto W".

Lab Problem: Take the file 'blues.wav' and do a projection onto the orthogonal complement of 'harmonica.wav'. This should eliminate the harmonica sound completely. It's a silly sort of computation, because it can't be used in the real world to eliminate an instrument.

Yet.

It's interesting to see what all this looks like in some examples. First, let $V = \mathbf{R}^3$ with the standard Euclidean inner product, and let $w = (0, 1, 0)$. Then $P(a_1,\ a_2,\ a_3) = (0,\ a_2,\ 0)$, and P is given by the 3-by-3 matrix

$$P_A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

You'll also notice that $v - Pv = (a_1, 0,\ a_3)$, and that this too defines a subspace of V which is perpendicular to W; call it $W^\perp$. Then any $v \in V$ can be written as $v = Pv + (I - P)v$ where $Pv \in W$ and $(I - P)v \in W^\perp$. It also doesn't take much fiddling to see that for our projection operator P, $P^2 = P$, hence $P(I - P) = 0$. Then P acts as projection onto W, and $I - P$ is 'projection' onto the two-dimensional subspace $W^\perp$.

This works generally in $V = \mathbf{R}^n$. Let $\{v_1,\ v_2, \ldots,\ v_n\}$ be an orthonormal basis of $\mathbf{R}^n$. Let $S \subseteq \{1,\ 2,\ \ldots,\ n\}$. Then

$$W = \{ \sum_{j \in S} a_j v_j \mid a_j \text{ is real} \}$$

is a subspace of V, and the orthogonal projection $P : V \to W$ is easily seen to be

$$P(\sum_{j=1}^{n} a_j v_j) = \sum_{j \in S} a_j v_j$$

If $S = \{1,\ 2,\ \ldots,\ k\}$ and $T = \{1,\ 2,\ \ldots,\ n\} - S$, then the projection matrix is

$$P_A = \begin{array}{c} \\ S \\ T \end{array} \begin{array}{c} \begin{array}{cc} S & T \end{array} \\ \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \end{array}$$

Homework Problem: Assume V is any finite dimensional inner product space, and W is any subspace. Let $P : V \to W$ be an orthogonal projection. . . $P^2 = P$. Is it true that you can always find a basis of V such that

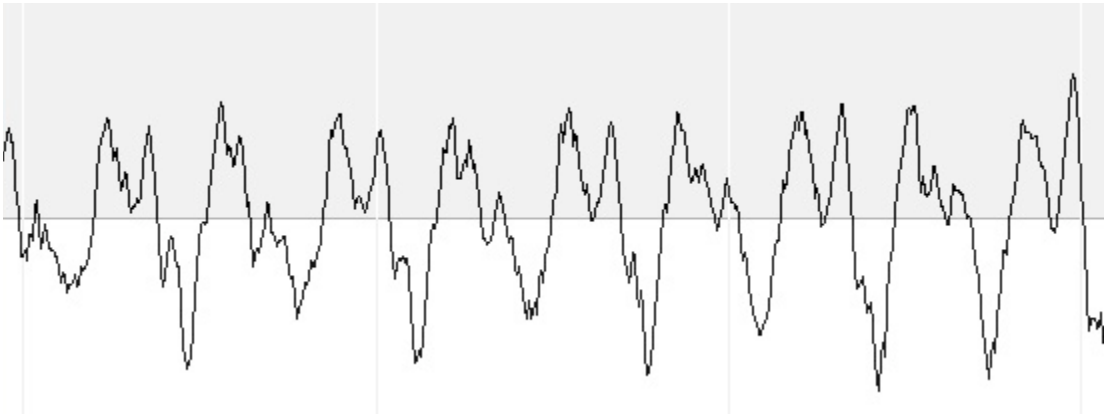
$$P_A = \begin{array}{c} \\ S \\ T \end{array} \begin{array}{c} \begin{array}{cc} S & T \end{array} \\ \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \end{array} \ ?$$

Lab Problem: Show that mp3 compression is not linear, in the following way. Load ‘estoy.wav’ into Matlab. Then create a random noise of the same length, using .02*rand. Compress each signal, then add them. Finally, add the signals then compress them.

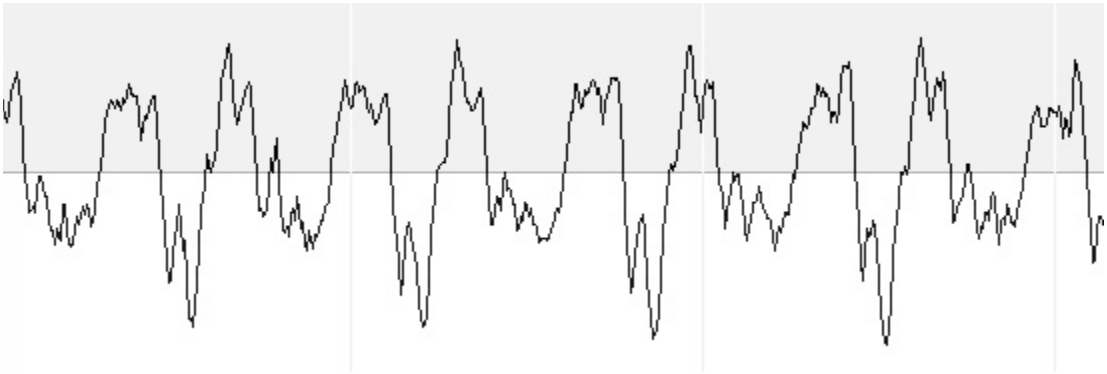
How will you check that the two different proceedures are not the same?

Note: mp3 compression is non-linear because, essentially, it uses non-linear characteristics of human hearing. The most important for mp3 is *masking* -- a loud sound will drown our a small sound. You can hear this for yourself in your Estoy + noise file. We chose Shakira’s Estoy Aqui because the first half is very quiet, and added noise stands out. The song then brings in a beat and gets loud; at this point, the noise is harder to hear. Mp3 encoding makes use of this to eliminate some of the noise. You can see it in the comparison below:

In the top clip, Estoy+noise has been mp3-compressed. Since this is the region where the song is louder than the noise, mp3 tends to reduce the noise part, as it won’t be heard anyway.



Below, Estoy and noise have each been compressed, then added. Since there is no Estoy to overwhelm the noise, mp3 keeps much of the noise. You can see that the bottom signal is a bit jaggier than the top.



Lecture 1

INTRODUCTORY MATHEMATICAL IDEAS

Correlations

What is a 'random' signal? How can we test randomness? If a time series is truly random, then the past and future are unrelated. We know how to determine the extent to which two signals are related – we project one onto another. To check randomness, we could take a segment of the past, and project it onto present or future. We expect the component of past to future to be zero and present onto present, past onto past ought to be large.

To implement this, projections and components are given by dot products; if $a = (a_1,\ a_2,\ \ldots,\ a_n)$ and $a = (b_1,\ b_2,\ \ldots,\ b_n)$, the component of a in the b direction is

$$\frac{a \cdot b}{\|b\|} = \frac{a_1 b_1 + a_2 b_2 + + a_n b_n}{\|b\|}$$

The *auto-correlation* of a is defined as

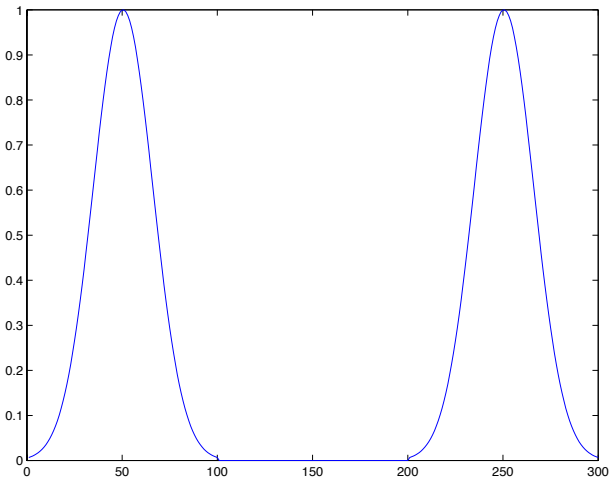
$$c_k = a_1 a_{k+1} + a_2 a_{k+2} + + a_n a_{k+n}$$

and the *cross-correlation* of two signals $a,\ b$ is defined as

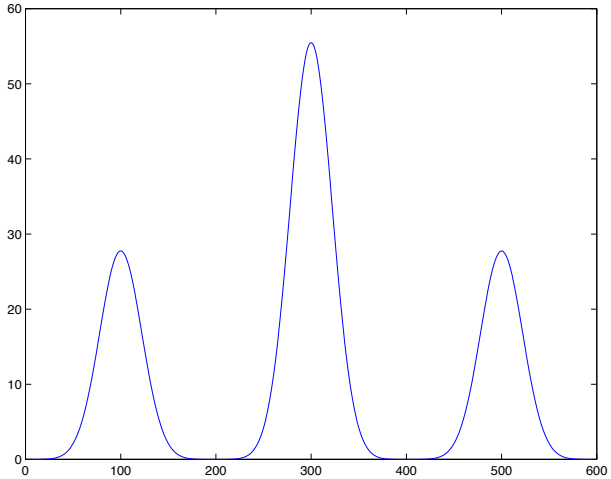
$$c_k = a_1 b_{k+1} + a_2 b_{k+2} + + a_n b_{k+n}$$

You can see that the terms like $b_{k+1},\ b_{k+2},\ b_{k+2}$ move b into the future if $k > 0$ and to the past if $k < 0$. So the cross and auto-correlations determine how much in common past and present and future have.

We'll compute an autocorrelation by hand. On the left, a function with two bumps. We compute by sliding, multiplying, and integrating this against itself. When a bump on one copy overlaps zero on the other – the projection is zero. You only get serious projection when one bump overlaps another. You'll get even larger projection when two bumps in each overlap two in the other– that's when the two align completely, present to present.

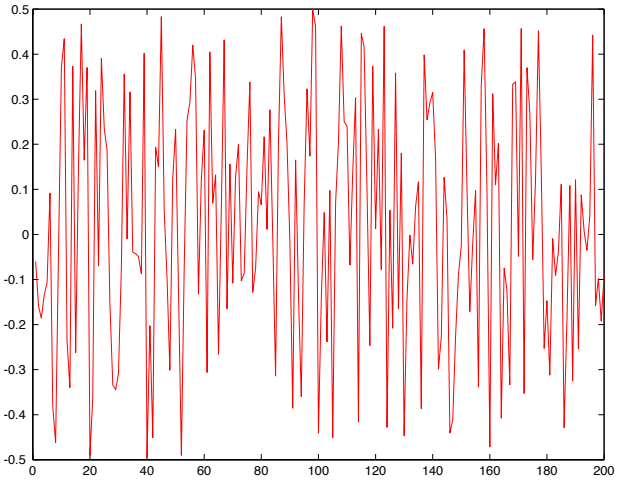


time series

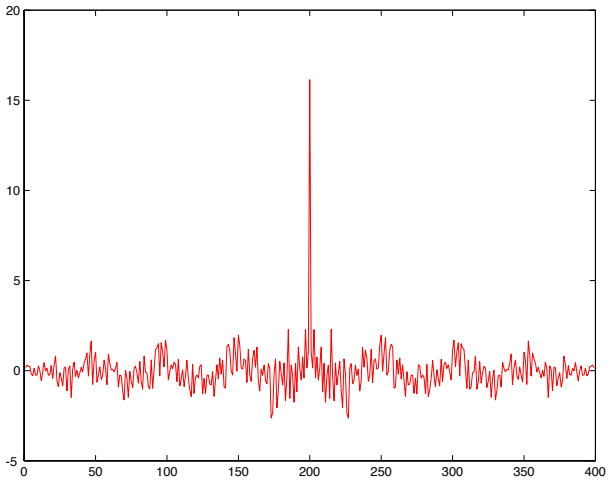


autocorrelation

Below left, a simulated random sequence, and at right, its autocorrelation. Unlike the previous example, we only get a strong correlation in the middle: present on present. This suggests that the past and future are unrelated, or, as one says, *uncorrelated*.



random time series



autocorrelation

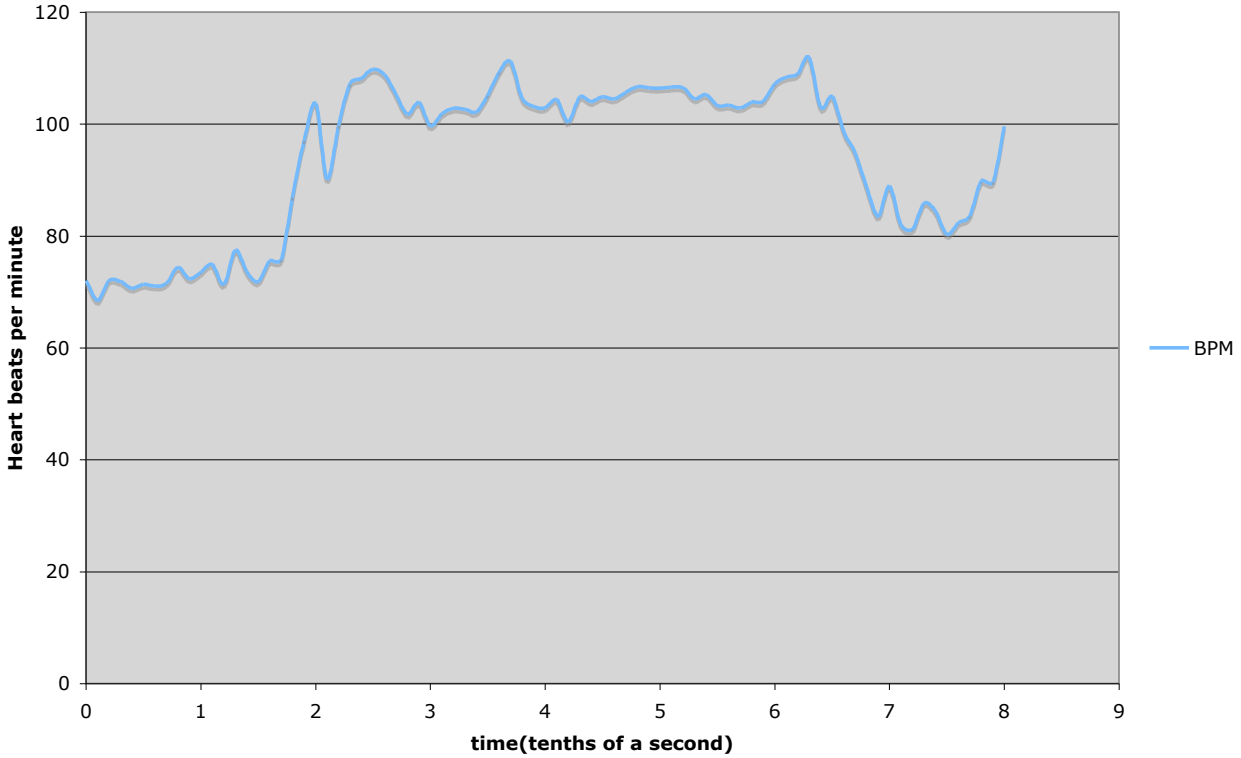
Lab Project Construct a sine wave sampled at 44100hz and a square wave of the same frequency. plot the cross-correlation and interpret it.

The following Matlab code might help:

```
>> x=linspace(0, 10, 10*44100);
>> y= sin(2*pi*x);
>> z=sign(y);
>> cor=xcorr(y,z);
>> plot(cor)
```

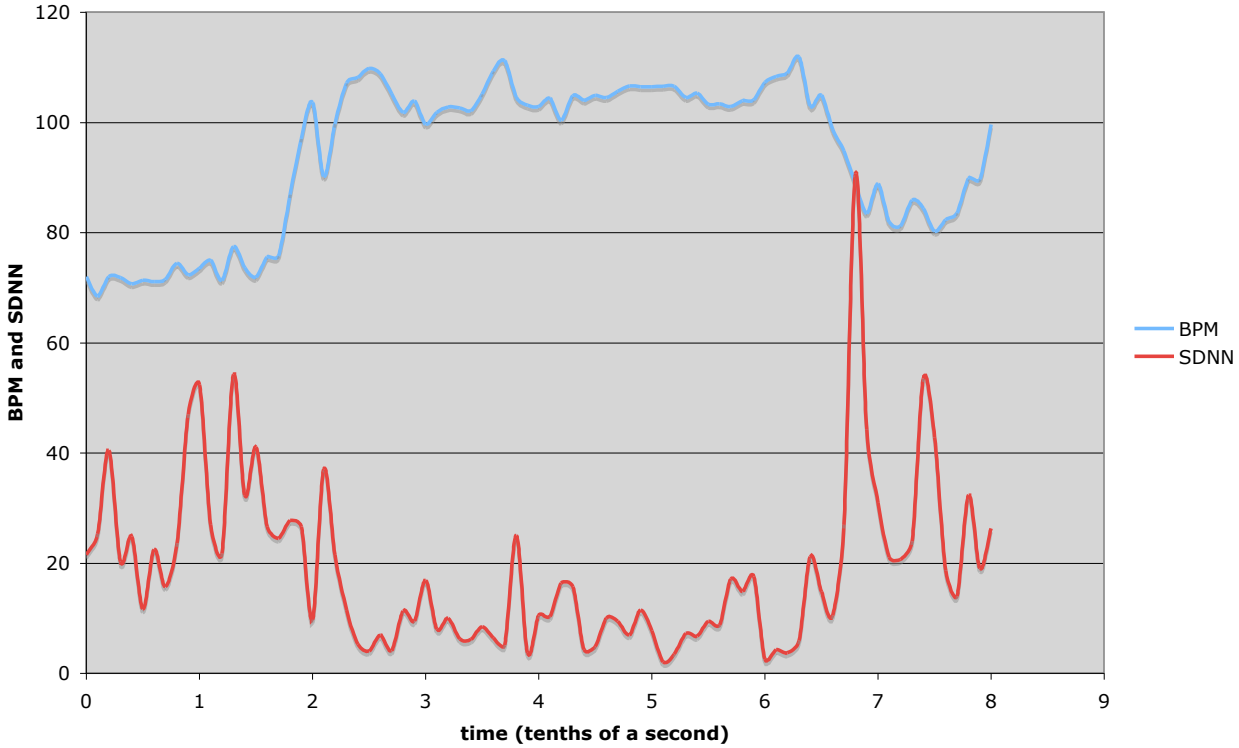
Everything we've said so far is designed to be intuitively true, and because of that, none of the examples are surprising or even very useful. Next we want to do an auto-correlation which arose in our research.

The experimental subject is a healthy young male human volunteer. He is put in an environmentally controlled room, and the temperature is suddenly dropped. He is then asked to move about, do light exercise, during which his internal temperature and heart rate are monitored. Of course, as exercise begins, the heart rate jumps. This is plotted on the next page:

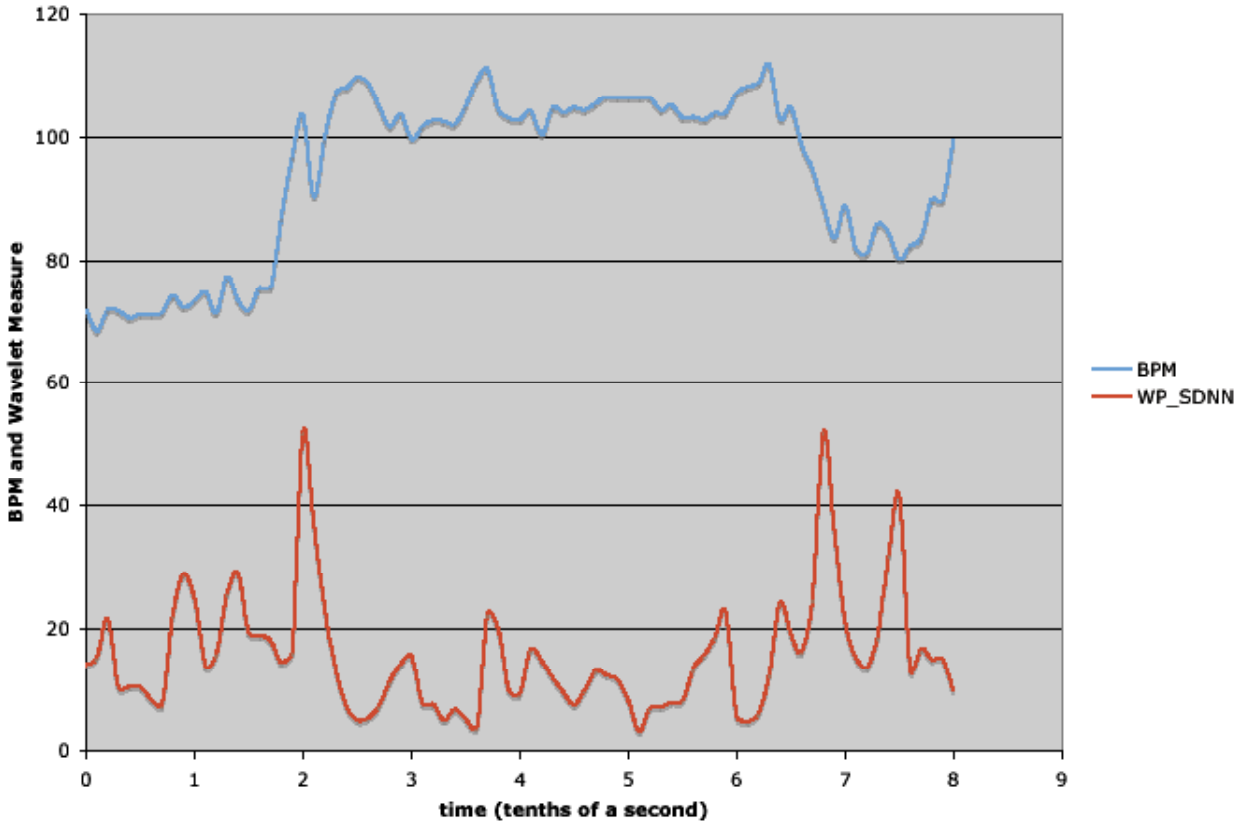


The goal of the research project is to determine the onset of the changes in heart rate and the physiologic causes. The heart is a complex feedback system, designed to maintain blood pressure and body temperature automatically. Two of the main regulatory components of the heart are the sympathetic, or adrenergic system, and the parasympathetic, or cholonergic or vagal system. They act in concert, the adrenergic (think adrenalin) to increase blood flow, and the vagal, to decrease it. Here's a diagram, or go to the CD at ch1/correlations/heart.pdf.

In this experiment, we used two different quantities to try and detect the onset of the change in heart rate. The first is well-known in the medical research literature, it is called SDNN, and is based on standard deviations. Here it is, plotted with the BPM:



The red signal has peaks which match well with the decrease or increase in beats per minute. The point of our research is to develop a new method, based on wavelet transforms, that will, we expect, separate out the contributions of the sympathetic from the vagal parts of the control system. Here's what that looks like:



The SDNN and the wavelet measure look very similar. Thaty's good; it serves as a partial validation that our approach is as good as the established approach in the literature. But: how similar? Do their peaks match up?

Lab Project Download the Excel spreadsheet, bpm.xls, and load it into Matlab using the xlsread command. The time channel is in the first column; the BPM in the second; WPT is third and SDNN is fourth.

Do a cross-correlation of WPT and SDNN. What does it tell you about the two?

Side remark: our wavelet measure is the sum of an adrenergic term and a cholenergic term. That sum reproduces the SDNN, but the point of our research is the development of the independent terms, not shown here.

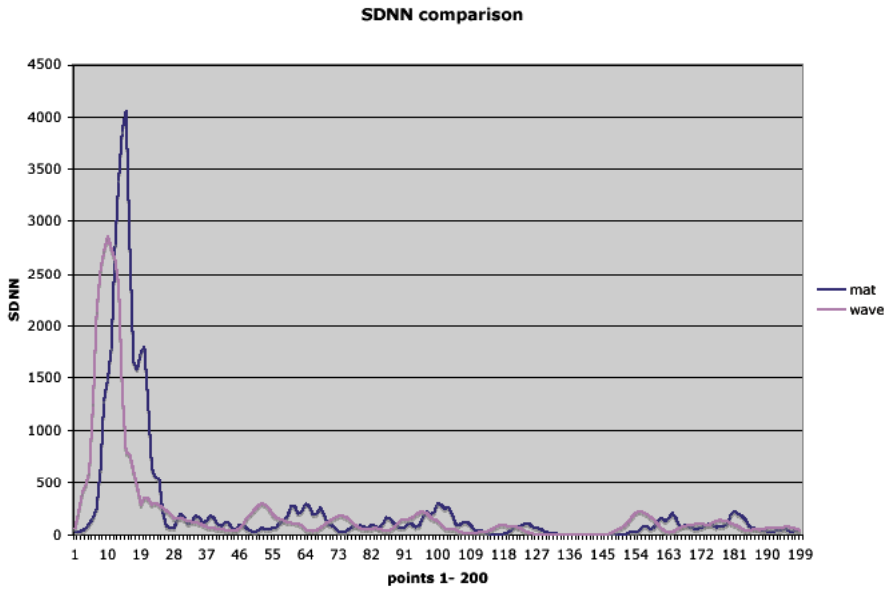
Look at the example above, where a random signal defined on N points is correlated with itself. We get a signal defined on 2N points, but, because the signal is random, the autocorrelation is largest when the signal exactly overlaps itself -- at N.

If two signals are shifted in time with respect to each other, the correlation can detect that time shift. Try it with the signals y1.txt and y2.txt. Note that xcorr(32) is NOT the maximum of the cross-correlation; the position of the maximum should instead detect the time shift between the two signals (which you can see most clearly by plotting one ontop of the other).

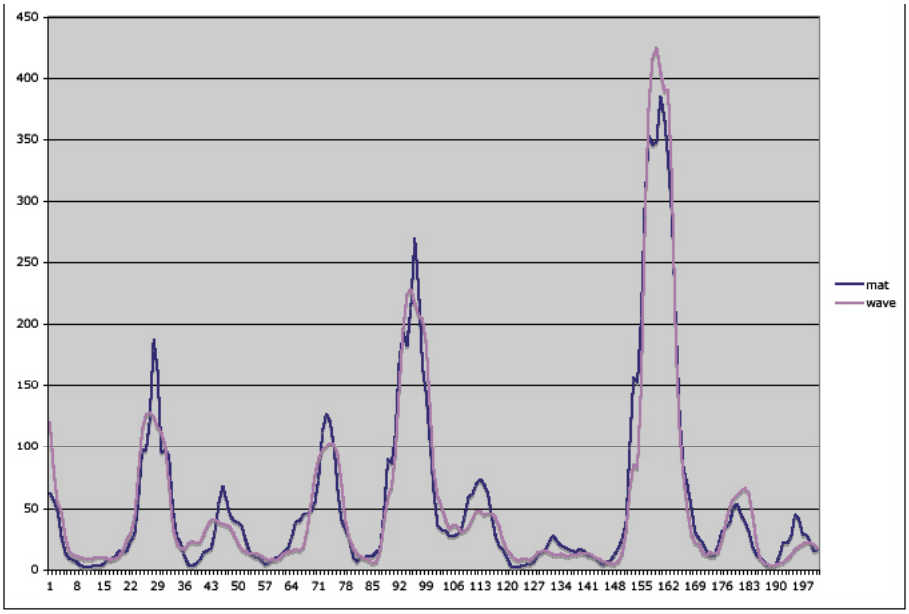
For our last example, more data from "real life" -- our biomedical research. Again we're looking at heart signals, and we're using Matlab and the *wavelet packet transform* to compute a quantity like SDNN. The difference here is that SDNN can only identifyheart rate changes a minute or two after they've happened. The quantity we're developing useing wavelets could, in principle, detect those changes within five seconds.

That would have an enormous impact in cases where fast response is necssary to save lives -- in war zones, for example (the actual problem we're looking at is called hemmorhagic shock).

Our Matlab program works great; we call it's output sdm.txt. Now we've asked our lead programmer (a very experienced and expensive C++ expert) to code it so it can be run without Matlab, eventually perhaps on medical devices. The the C++ program is called wavetest, and its output is sdw.txt. If you graph the two together, you see there's a phase problem:



After a phase adjustment, we get graphs like these:



Close, but not exact. The boss says, 'this isn't good enough: I'm not convinced by the pictures.' Our response is to compute a cross-correlation. Or -- well

Close, but not exact. The boss says, "not good enough: I need more than pictures." Our response is to compute a cross-correlation. Or – well – a related quantity.

If $a = (a_1,\ a_2,\ \ldots,\ a_n)$ and $a = (b_1,\ b_2,\ \ldots,\ b_n)$, are vectors, the quantity

$$\frac{a \cdot b}{\|a\|\ \|b\|}$$

is the cosine of the angle between a and b. Moreover, we know that the cosine is $+1$ when when the angle is zero, and -1 when the angle is π.

So, what we proposed to do is to compute the cross correlation between sdm.txt and sdw.txt, but divided by $\|a\|\ \|b\|$. We'll then plot the quantity, and see where it peaks. If the peak is near one, that will tell us that the two really are similar, with the exception of a phase difference. Moreover, the phase difference will be N - peak.

So, that's your next project: what's the report back to the boss?