ELEC3004: Tutorial 1

Part 1: Signals as Vectors

In the discrete world signals are represented as vectors. We are going to investigate this concept, and learn how vector math is applied to signal processing.

Univariate signals

First lets look at a few simple vectors.

Open SuppMat1.m in the matlab editor.

In section one we establish three signals. An impulse, a step, and a piece of music. Run section one to place these signals in the workspace. We are going to use these signals as inputs into a simple system, and discuss what happens.

In the case of the piece of music, we are dealing with a time series. Is this also the case for the other two? can you think of examples of non-time series data where you would observe this?

Task one: Build a Low pass filter

You are going to build and apply a low pass filter. build this within the YOUR CODE HERE section of the script, and avoid overwriting the input signals. save each signal to output1, output2 and output3. you have 15 minutes.[1]

[1] Hint: There are no rules on how you build this filter, but you may want to use the matlab functions fir1 and filter. fir1 to design the filter and filter to apply it to the data. Read the documentation to find out more. Bonus points if you can generate the filter using only the functions zeroes(), ones() and basic arithmetic

Questions

fir1 (or the equivalent)
What does fir1 give us?
It is also a vector, would we therefore consider this a signal?
What is the relationship between each scalar value, IE how does LPF(1) relate to LPF(2)

filter
What is the function filter doing?
Can you give me the mathematical formula for this operation?

output1
this should be very closely related to the filter, How?

output2
what is another name for this output?
how has the filter modified the input signal?

output3
use the function sound(output3,Fs) and compare to sound(input3,Fs). Is there an audible difference?

Homework for tutorial 2: what does Matlab do to make filter more efficient?

Convolution

Calculate y[n] where:

$$y[n] = x[n] * h[n] \tag{1}$$

Where x[n] = [1, 2, 3, 4], and h[n] = [0, 1, 1, 0], by hand. The star is the convolution operator.

Multivariate Signals

How do we extend the concept of a signal to the multivariate case?

Open SuppMat2.m in the matlab editor

This peice of software generates a simple model of a ball rolling across the ground, and a set of noisy observations of the ball. In a future tutorial we will implement a filter to estimate the motion of the ball from the noisy observations, via a kalman filter. For now, look carefully at section one, and observe how the objects motion in two variables, the X and Y axis, are expressed.

How is the multivariate input signal expressed?

This concept can be extended to 100's of variables for controlling chemical plants and flying jumbo jets, but the foundations of the math are extensions of the univariate case. We will come back to multivariate processing and the Kalman filter around week 8.

Part 2: Linear algebra revision

Surya alteratively refers to ELEC3004 a linear systems course. This is accurate, every topic covers some aspect of linear systems, whether it be modelling, filtering or control. The foundations of how we manipulate linear systems lie in the mathematics of linear algebra. Hopefully this is revision, and in future weeks we will look at the application of linear algebra to signals systems and control.

Dot product

The dot product, alternatively the inner product, is one of the basic operations in linear algebra. $< v_1, v_2 >$ is the symbolic representation of the dot product of vectors v_1 and v_2. It is found as follows:

$$\sum_{i=0}^{N} v_1(i) \cdot v_2(i) \tag{2}$$

Find the dot product of the following vector pairs.

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 3 & 2 \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} 1 & 0 & 5 \end{bmatrix}, \begin{bmatrix} 3 & 4 & 0 \end{bmatrix} \tag{4}$$

Basis

If we google the definition of the word 'basis', we get: "the underlying support or foundation for an idea, argument, or process". In mathematics, a basis is similar. It is an underlying structure of how we look at something. It is similar to a coordinate system, where we can choose to describe a sphere in either the Cartesian system, the cylindrical system, or the spherical system. They will all describe the same thing, but in different ways. And the reason why we have different systems is because doing things in specific basis is easier than in others. For example, calculating the volume of a sphere is very hard in the Cartesian system, but easy in the spherical system.

When working with discrete time-series signals, each consecutive element of the vector represents a consecutive measurement. This is the most obvious basis to look at a signal. Where if we have the vector [1, 2, 3, 4], then at time 0 the value was 1, at the next sampling time the value was 2, and so on, giving us a ramp signal. However, there are also other basis for representing discrete signals, and one of the most useful is the Discrete Fourier Transform. When we apply this transform, each consecutive element of the result represents the contributuion of a sinusiod of a particular frequency to the original vector. In this space, we express our data not by the individual values of the data, but by the summation of different frequencies of sinusoids, which make up the data. It's like the continuous Fourier transform you should be familiar with from ELEC2004, but here the sinusoidal functions are discrete. We will look at the DFT in the next tutorial, so please review the Fourier transform over the next two weeks.

A more 'mathsy' definition of a basis is:
"Any set of linearly independent vectors. For a basis of R^n space, any n-dimensional vector can be exactly reproduced by a linear combination of the basis." Also, it is worth reviewing some important properties of

bases. First, an 'Orthogonal' basis is a basis where all vectors in the basis are orthogonal to each other. An easy example of this is the Identity matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

In this case, the three vectors are going in the x, y, and z directions respectively. Each vector is perpendicular to all the other vectors and therefore it is an orthogonal basis. We may test orthogonality by taking the dot product of two vectors. If $< v_1, v_2 >= 0$ then v_1 and v_2 are orthogonal.

Another important type of basis is the 'Orthonormal' basis. Any matrix which is orthonormal is orthogonal, and each vector in it has unit length.[2] The useful properties of orthonormal matrices is that for matrix A, $A^{-1} = A^T$. That is, its inverse is equal to its transpose. Also extremely useful is the fact that when you transform between two orthonormal basis, the length and angles are maintained.

[2]Read: Length = 1, so that when we change basis everything doesn't get bigger.

Exercise: Are the following matrices orthonormal, orthogonal, or neither?

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{6}} & \sqrt{\frac{2}{3}} & -\frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \tag{9}$$

You don't really need to check that last one. It **is** orthonormal, it's just more of an exercise to show that complicated matrices can also be orthonormal.

Bases are fundamental to the implemetation of a number of signal processing algorithms. In some cases we may be able to apply a basis transform to a multivariate dataset with hundreds of variables, and reduce the problem to say 3 or 4 of the most influential components that are actually generating that dataset. In the next section we will show you a simple example. Also, Linear Transforms can be expressed as a basis. This can help in seeing how the transform operates, as we will see next week when we derive the Basis for the DFT. It can also show us where redundant calculations are occuring and guide us in the development of a more efficent algorithm, as is the case when we look at the DFT and see the structure that the Fast Fourier Transform exploits to speed it up.

Transforms

A Transform maps a vector from one space to another. for example you may wish to map the vector describing the position of a spacecraft from one frame of reference, maybe one centred on the earth, to another frame of reference, such as one centred on the sun. The way you do this, as you may recall from MATH1051 or 1052, is you multiply your vector by the transform matrix.

Take for example the rotation of a 2 dimensional vector by some angle θ. This is done like so:

$$\begin{bmatrix} x\prime \\ y\prime \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{10}$$

Here the point $\begin{bmatrix} x\prime \\ y\prime \end{bmatrix}$ will be the same distance from the origin as $\begin{bmatrix} x \\ y \end{bmatrix}$ but will have rotated θ degrees clockwise around the origin. By varying theta any rotation around the origin can be computed. Once the rotation matrix has been computed, the rotation can be applied to a 2 dimensional dataset of arbitrary length as follows

$$A = \begin{bmatrix} cos(20) & sin(20) \\ -sin(20) & cos(20) \end{bmatrix}, X = \begin{bmatrix} x(1) & x(2) & \cdots & x(n) \\ y(1) & y(2) & \cdots & y(n) \end{bmatrix} \tag{11}$$

$$X' = \begin{bmatrix} x\prime(1) & x\prime(2) & \cdots & x\prime(n) \\ y\prime(1) & y\prime(2) & \cdots & y\prime(n) \end{bmatrix} = AX \tag{12}$$

here X represents a two dimensional variable sampled n times. X' expresses the same data as X rotated about the origin by 20 degrees. a possible use case is a system detecting a robots position in 2 dimensional space. If the position was being measured from two separate sensors, there will likely be a rotational offset between them. Once that offset is known the data from one sensor may be recomputed to allign with the other sensor, and averaging used to reduce sensor noise or redundancy used to reduce missed detections.

Lets apply this rotation to our Multivariate data example.

Go to the supplementary materials 2 m file. In the your code here section, do the following:

Generate a 2x2 rotation matrix.
Take the Observation data and use the rotation matrix to rotate the data by 20 degrees.

0.1 Discussion: Transforms

Depending on which way you rotated the matrix, the input is now restricted to lie only on the x dimension. Could this simplify the kalman filter?

The rotation is performing what is known as a *Dimensionality reduction*. An input that required two dimensions to describe in the first case was reduced to a single dimension. What does this tell us about the number of free parameters of the system we are observing?

We will see in the next tutorial how the Fourier Transform, both discrete and continuous, are often used to obtain a dimensionality reduction. The purpose is reduce the number of parameters required to describe a signal. Can you name any processes which employ this concept?

Part 3: The Nyquist Principle

Hopefully you have heard of the Nyquist frequency before this point. The Nyquist frequency is half of the sampling rate of a discrete signal processing system. When a signal of higher freuqency is sampled, it will *alias* at some frequency below the nyquist frequency. The freuqency at which it will alias is known, in other words if you give me the sampling rate of a system, and a high frequency signal I can tell you the exact signal that will be returned by the system. However, the process is not reversible, given an output from the same system, I cannot tell you if the samples came from a signal that was below the nyquist rate, or have been generated by a high frequency signal aliasing. When building discrete systems, an understanding of the nyquist frequency is essential. Generally we aviod situations where inputs, either intentional or noise, may alias, but there are occasions where exploiting aliasing can be useful.

Here we are going to look at a short example of what happens when we exceed the nyquist limit, and therfore generate aliased signals (un)intentionally. We also introduce the spectrogram function, which can be a nice way to visualise signals that change in frequency over time. This technique belongs to an advanced subsection of signal processing, called time-frequency analysis. Chirps are an excellent example of a signal that is described very efficently when looked at using time-frequency analysis.

Open SuppMat3.m in the matlab editor.

run the script and observe the spectrogram of each signal. Press space after each sound has finished playing to observe the next signal.

For the second Chirp we have downsampled the signal. Here the sampling rate is 4096 samples per second, corresponding to a nyquist rate of 2048Hz. as the chirp goes to 3000Hz we find that once 2048Hz is exceeded the high frequency signals **alias** to a lower frequency. we can hear this as a descending tone in the audio and observe this in the spectrogram. The third reduces the sample rate even further, to 2048 samples per second.

The final signal is just for fun and shows you how to play an audio file in matlab and the corresponding spectrogram. If you would like to see an example of real time audio processing in a future tutorial let the tutors know.

A Practical Nyquist Example from Team Project II

So last years TP2 project had the following paragraph in the project specification:

> "Your system is required to be able to simultaneously display two input signals in the range 0-250kHz and you must build low-pass anti-aliasing and other conditioning circuitry to ensure that the input signals can be properly sampled... ...Additionally, you need to be able to operate your scope such that it can simultaneously process a single 650kHz-750kHz bandpass signal on channel A. To digitise this signal you are required to implement bandpass sampling. This involves passing the signal into suitable bandpass filters, then sampling it at suitable rates and then processing the samples appropriately." – TP2 Project Specification

To give some background, the 'system' is an oscilloscope which samples at 1MHz. One channel is required to have a mode, which when active, allows that channel to display signals from 650kHz to 750kHz, up from the normal range of 0-250kHz.

What type of filter is required before the sampling stage, in order for us to sample signals at this high a frequency?

Also, it was required to plot on a display, a reconstruction of the sine wave from only these samples. What is the process required on the software side to draw the reconstructed signal from the few samples we have.