Time Series Analysis: Introduction to Signal Processing Concepts

Liam Kilmartin

Discipline of Electrical & Electronic Engineering,

NUI, Galway

Aims of Course

- To introduce some of the basic concepts of digital signal processing
  - Where possible to avoid getting too mathematical!
  - In many cases looking at concepts you have already studied from a slightly different perspective
- To complete some workshop exercises using MATLAB to gain an basic appreciation of what MATLAB offers
- Examine applications in a number of application areas

Workshop Syllabus

- The DSP topics which we will cover include some basic topics:
  - Sampling waveforms
  - Time domain:
    - Digital filters
    - Energy estimation
    - Periodicity estimation
  - Frequency domain analysis
    - Fourier Analysis
    - Wavelet Analysis
    - Coherence and Synchrony
  - Applications

Background Material

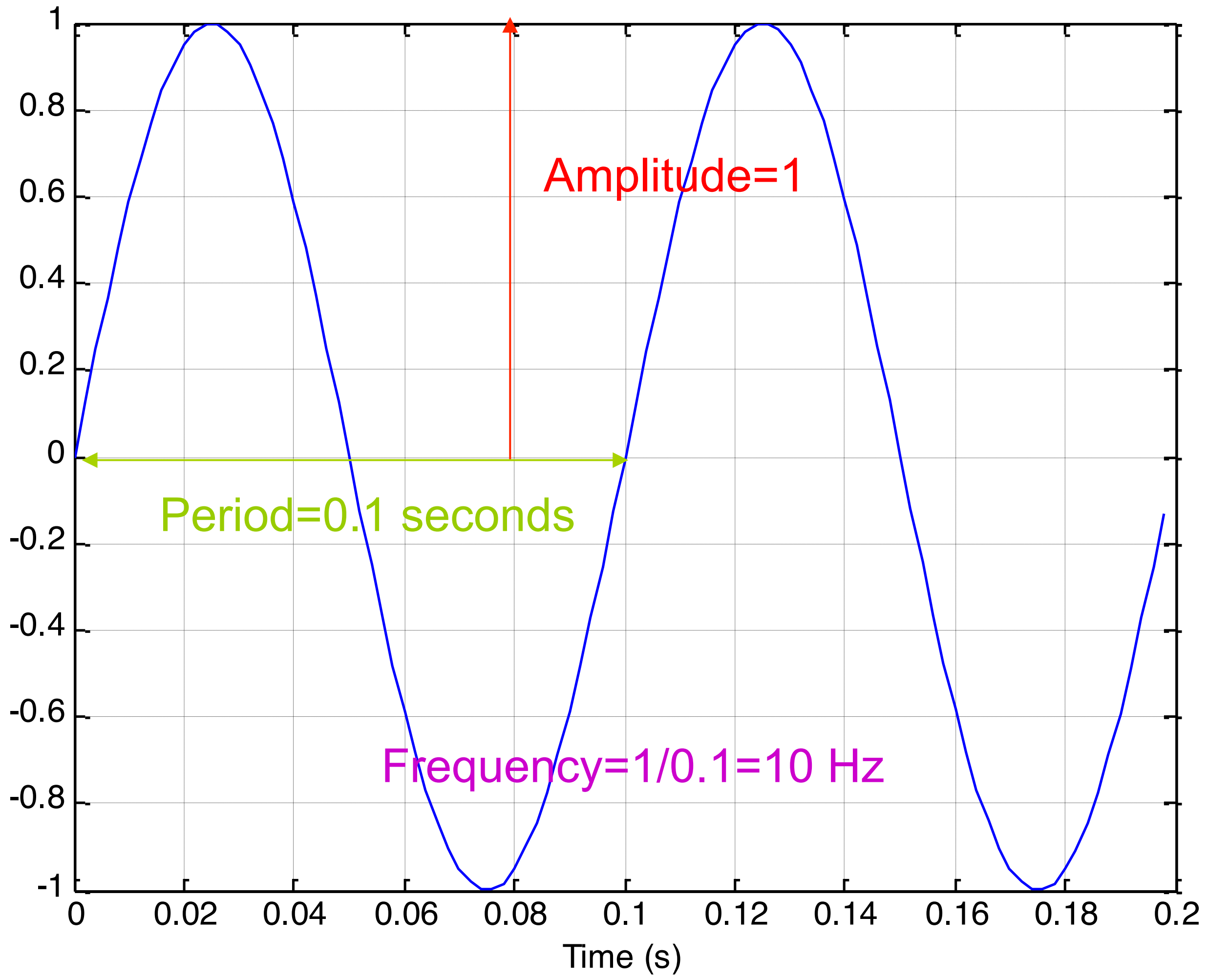
Where to start ?

- Before we actually start looking at signal processing, we need to familiarise ourselves with some basic terminology
- As will be seen presently, sinusoidal waveforms are very important in signal processing so firstly a primer on these waveforms

What is a Sine Wave ?

- A sine wave has three basic parameter that define its behaviour
  - Amplitude (A) (the "height" of the oscillation)
  - Frequency (f)
    - The signal "repeats" itself exactly over and over.
    - Its frequency is the "number" of complete oscillations/cycles completed in a single second. Measured in Hertz (s^{-1})
    - Directly related to the Period (T) of the waveform which is the time it takes to complete one full oscillation
      - Frequency=1/Period ($f=1/T$)
    - An alternate measure termed **angular frequency** (ω) measured in radians per second is sometimes used ($\omega=2\pi f$)
    - But in some application areas, e.g. financial time series, talking about frequencies in units of Hz is nonsensical
      - Consider time series of closing prices of a exchange index (1 "sample" per day!)
  - Phase (ϕ)
    - The relevance of this quantity is often misunderstood

Sine Wave Parameters



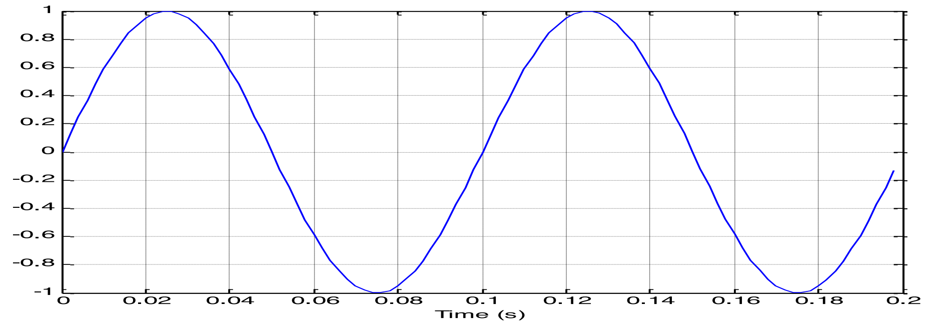
Phase of a Sine Wave

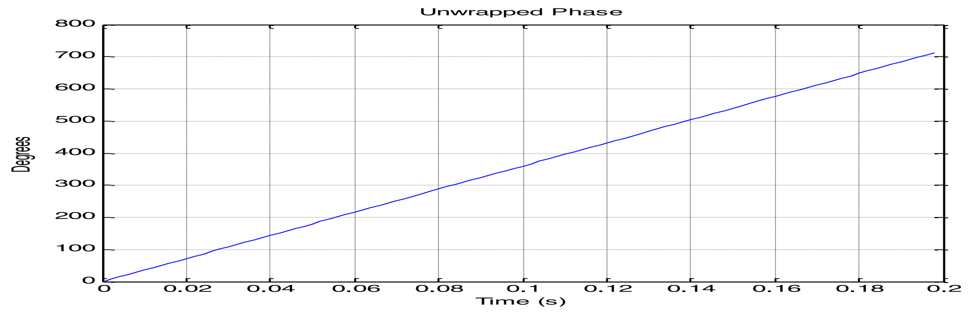
- Phase is an angular measurement and hence can be quantified in units of either degrees or radians
  - Remember 360° is the same as 2π radians and this represents one full cycle of the waveform!
- The term **phase** is often used quite loosely
- It can have two distinct meanings
  - When you are referring to a single sine wave then you can refer to the **instantaneous phase**
    - This reflects the current point that the sine wave is at in its cycle at a specific moment in time
  - When you are referring to two sine waves (of the same frequency) then you can talk about a **phase difference**
    - This can be measured by measuring the time difference between the points where each sine waves passes through the axis. This value is then represented as a phase angle by comparing it to the period (The period is viewed as being the equivalent of 360°)

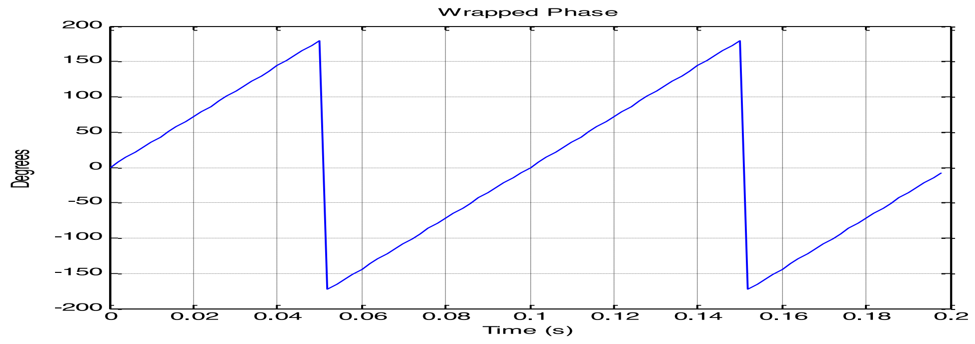
Instantaneous Phase

- When you draw a graph of a single sine wave, it typically has a value of 0 at the start of the graph (i.e. time=0)
- In such a case, the instantaneous phase at time=0 will be 0˚ (i.e. the “start” of a cycle)
- When the waveform hit its first positive peak value (i.e. one quarter way through the cycle), this is deemed to be the time where the instantaneous phase is 90˚
- When the waveform returns to zero next (i.e. half way through the cycle), this is deemed to be when the instantaneous phase is 180˚
- And so on until the moment when one complete cycle has been completed which is deemed to be the time where the instantaneous phase in 360˚
  - What happens after this point when the “next” cycle starts?
- One option:
  - The measure of phase continues on past 360˚ (i.e. 450˚, 540˚ etc.) which is termed **unwrapped phase**. However, such a measure is unbounded and hence difficult to represent graphical
- So instead we more commonly use **wrapped phase**
  - Once 360˚ is reached we start again at 0˚ **or** once 180˚ is reached we start again at -180˚
    - The latter range is more commonly used
    - Wrapped phase is more commonly used simply because it makes drawing graphs of phase more manageable)

Graphs of Instantaneous Phase



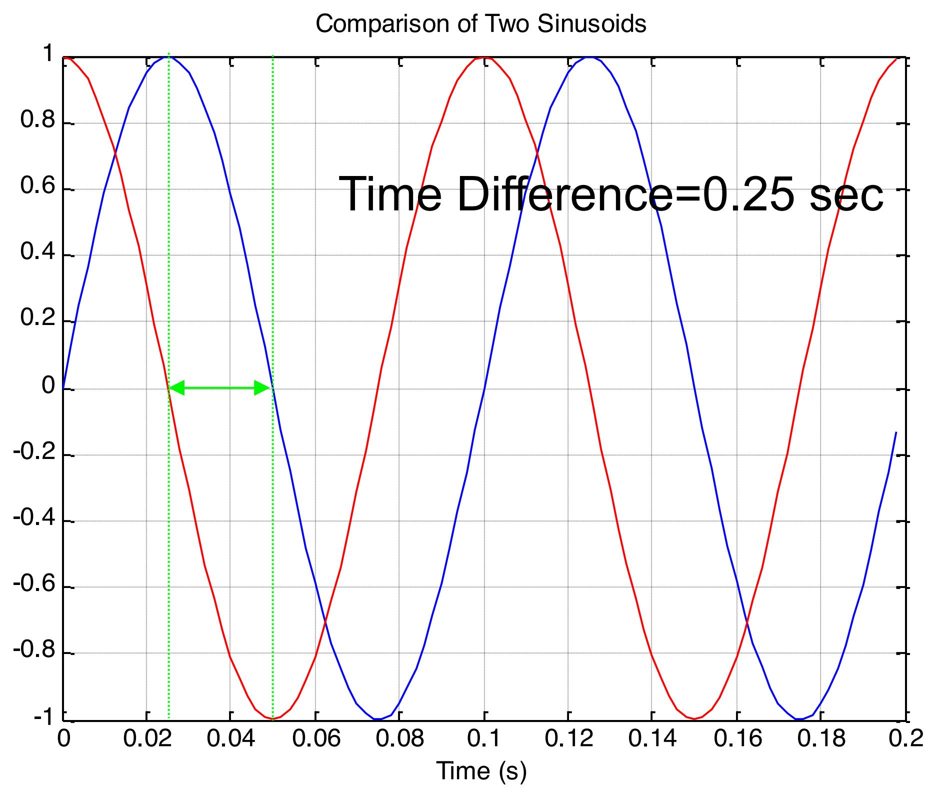




Phase Difference

- Typically (though not exclusively) used to quantify the “timing difference” between two sine waves at the same frequency
  - To complicate matters this measure is often applied to only a single sine wave where the “second” sine wave is assumed to be an implied “reference” sine wave
- Two sine waves are said to be “in phase” if their phase difference is 0° and completely “out of phase” if their phase difference is 180°

Graphs of Phase Difference

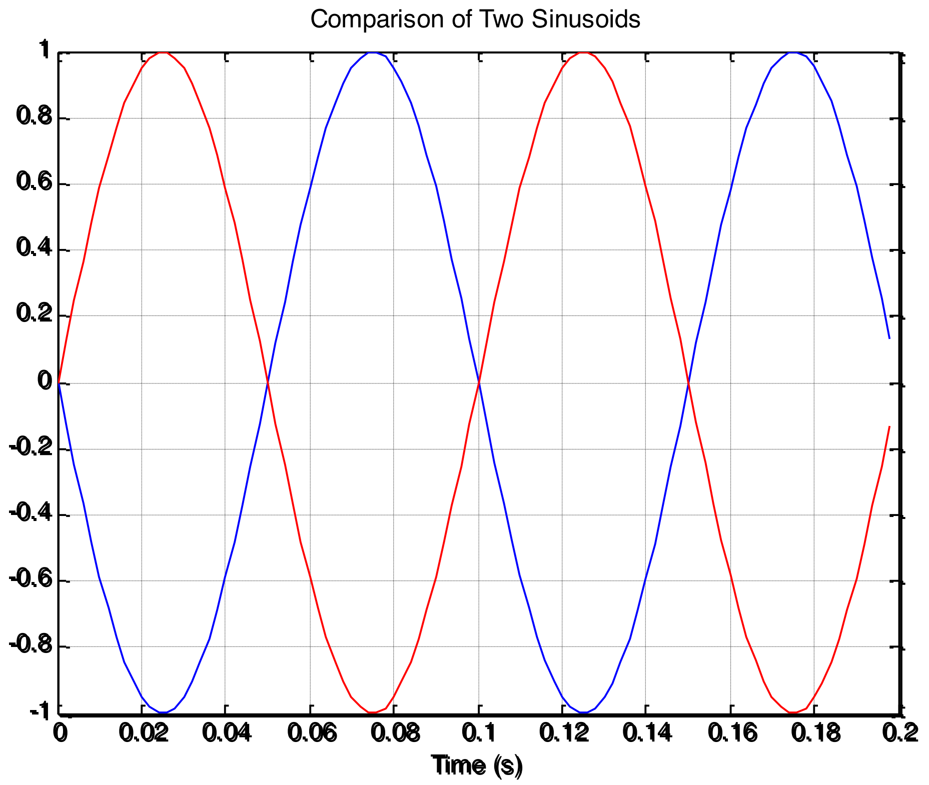


Phase Difference=0.025/0.1*360 degrees=90 degrees

Phase Difference of red wave relative to blue wave is +90 degrees

Phase Difference of blue wave relative to red wave is -90 degrees

Phase of a Single Sine Wave



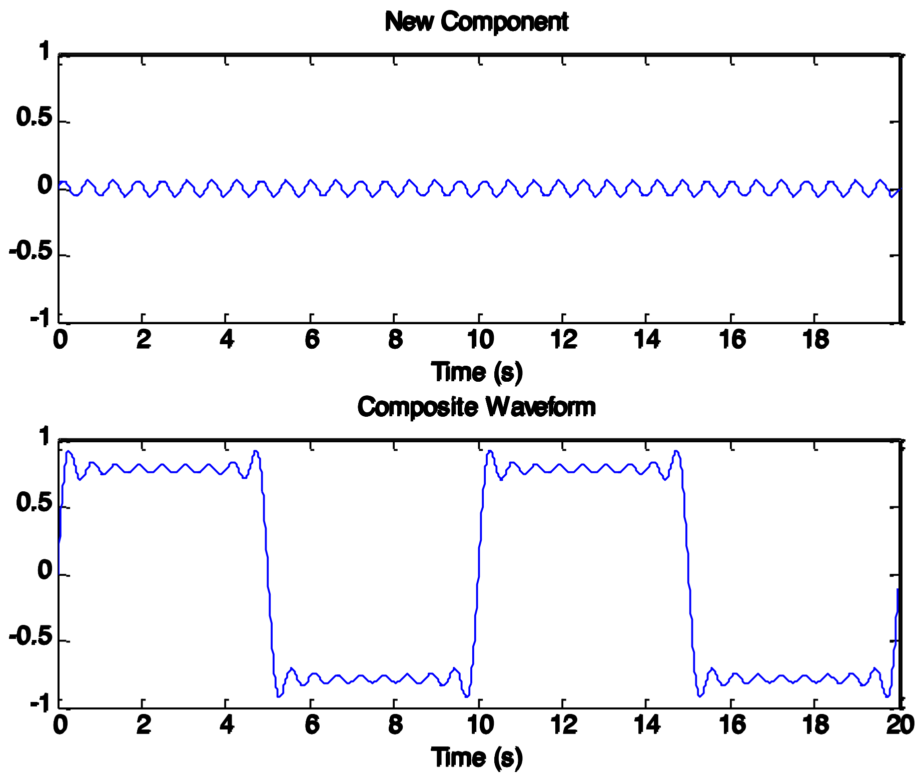
Assumption is that phase is being measured relative to a similar sinusoid whose value at time 0 seconds was 0

Time Difference in this case is clearly 0.05 seconds and hence phase relative to this notional second (red) sine wave is 0.05/0.5*360=180 degrees

So Why the Emphasis on Sine Waves ?

- A basic understanding of the terminology of sine waves is essential because mathematically all real world waveform can be viewed as being formed by adding together a carefully selected sequence of sine waves
  - Each sine wave having a carefully selected amplitude and phase difference
  - This approach is known as Fourier Analysis
- For real world signals, it offers an analysis methodology rather than a synthesis approach
  - By viewing any signal as being a sum of sine waves, we can analyse or alter the signal by considering these individual sine wave components rather than the complex complete signal
    - There is loads of mathematical theory and techniques for processing/analysing sine waves

Illustration of Fourier Analysis



We will illustrate how a more complex waveform (square wave) can be formed by adding together specific sine waves

With each component that is added the composite waveform is getting "nearer" to the desired square wave

Comments

- The approach just outlined is a nice illustration of how a more “complex” signal can be formed by adding together a series of simple “basis” signals (sinusoids)
- But some points to note
  - The “complex” signal that we are try to form must be periodic
    - very few real world signal exhibit anything approximating this behaviour

Comments (2)

- This approach only produces an approximation
  - The more (higher frequency) basis signals we added the closer the approximation gets to the desired complex signal
- The basis signals as all periodic with a frequency that is an integer multiple of the frequency of the desired complex signal
  - Harmonics

- Consider notionally what happens to this approach if we consider “real world” signals that are not strictly periodic ?

Sampling Signals

Typical DSP Configuration

"Real World" Signals

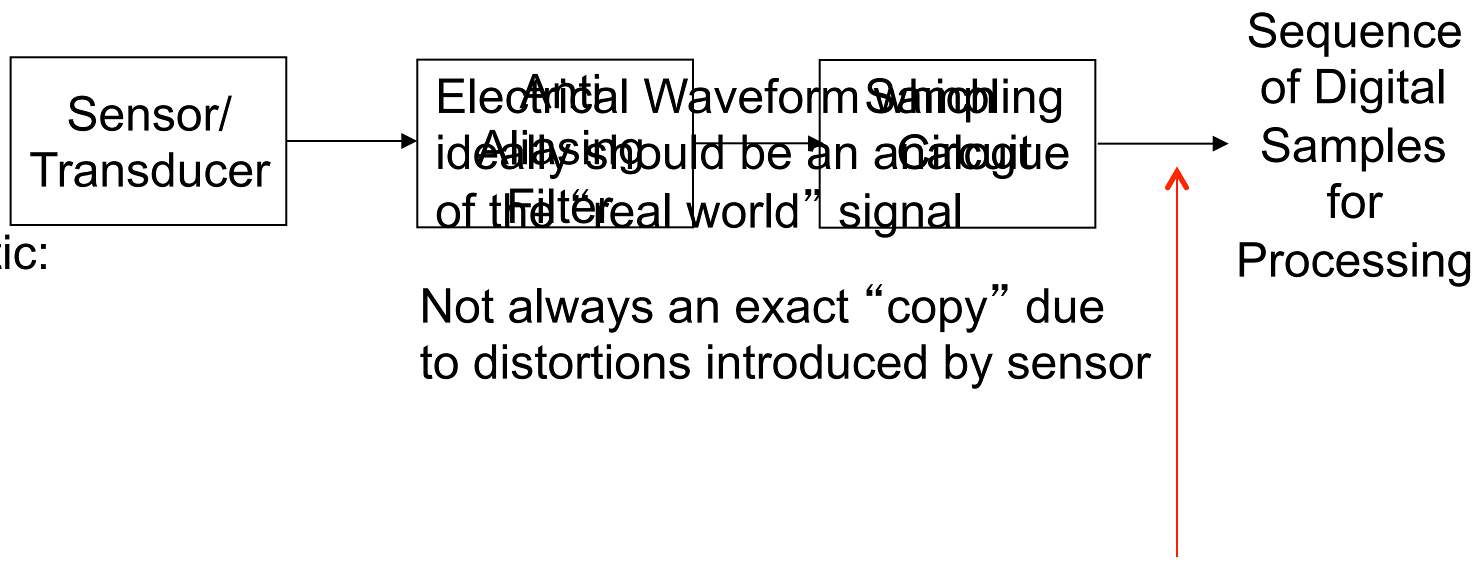
Pressure:
- Speech
- Music
- SONAR

Bio-Electric:
- EEG
- ECG

Electromagnetic:
- Radio
- RADAR

Images:
- Camera
- MRI

Other:
- Seismic


Sensor/ Transducer
Anti Aliasing Filter
Sampling Circuit
Electrical Waveform which ideally should be an analogue of the "real world" signal
Sequence of Digital Samples for Processing

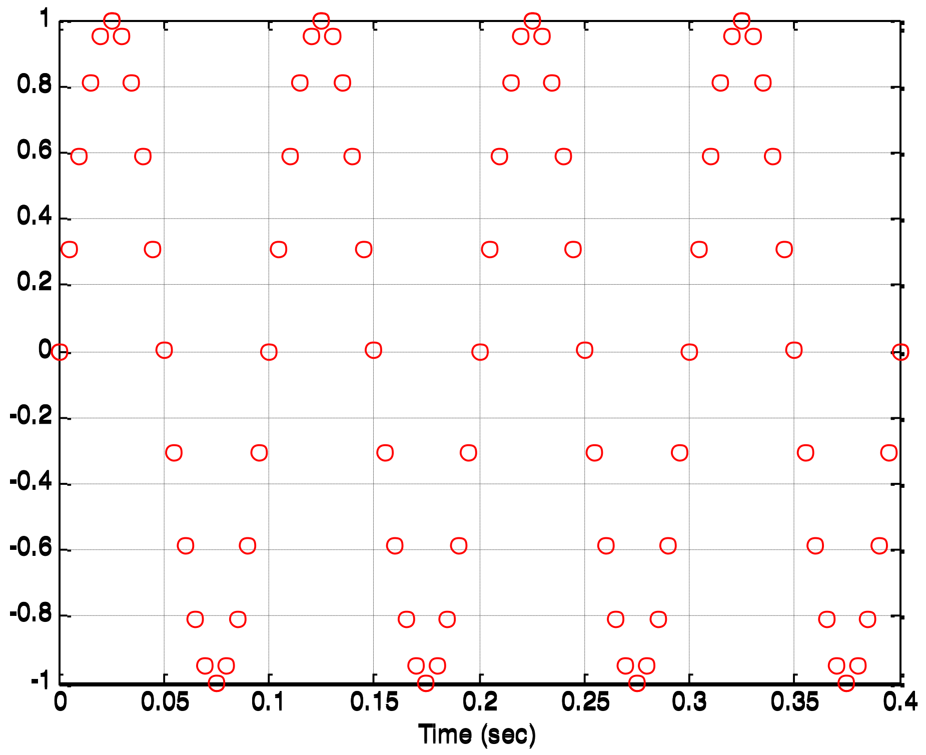
Not always an exact "copy" due to distortions introduced by sensor

Financial\Economic "Signals"

What is Sampling ?

- Sampling Circuits (also known as Analogue to Digital Converters [ADC])
  - Instantaneously measure the current value of the electrical voltage waveform (e.g. output from a physiological electrode or sensor, a micro-phone etc.)
  - Represents this voltage by a **number**
    - This number reflects the voltage level
      - If the voltage is positive then the number should be “positive”
      - If the voltage is negative then the number should be “negative”
      - If the voltage is large then the number should be “large”… etc.
- This process is repeated over and over again at a fixed interval known as the **sampling period** (T_s)
  - The number of times that this sampling process occurs every second is known as the **sampling frequency** (f_s)
- One important point:
  - The real electrical voltage waveform can have ANY voltage level in a continuum between its maximum and minimum value
  - However, there will be a finite set of “numbers” (defined by the number of bits in the ADC) by which the voltage measures can be represented
    - Thus this process may/will introduce an error (known as **quantisation error**) in the digitised representation of the analogue waveform but this is unavoidable!
    - The quantisation error can be made smaller by using an ADC which has a larger number of bits but this has cost/storage(?) implications

Example of Sampling



Let’s look at sampling this sine wave with a frequency of 10 Hz

In this example the sampling frequency of 200 Hz is used (Ts=0.005 sec)

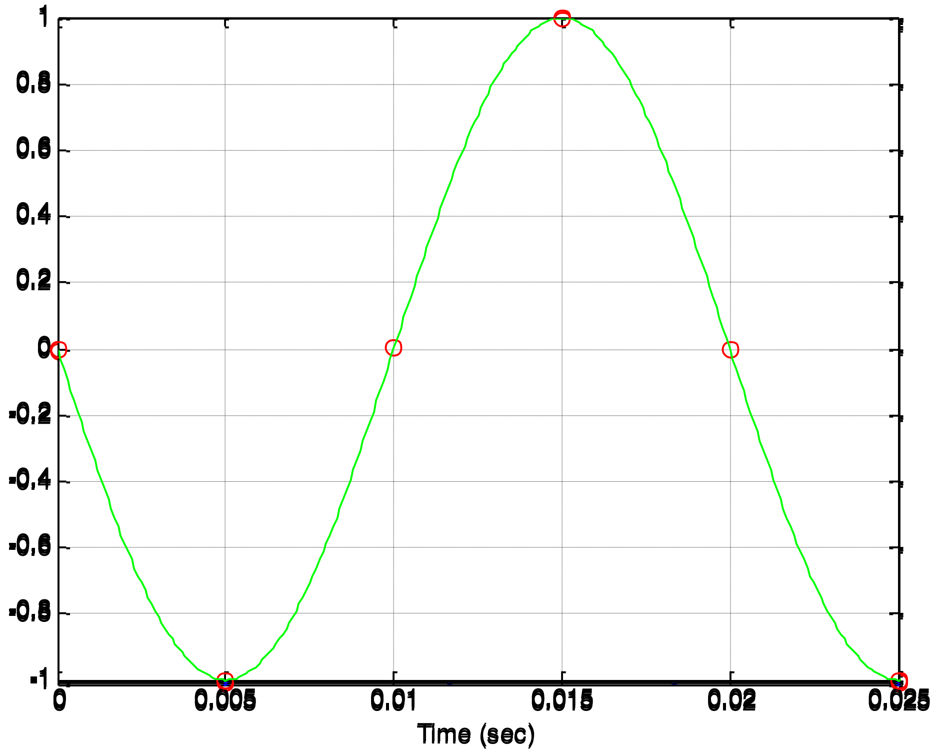
How do you choose the Sampling Frequency?

- There is a criteria that must be satisfied when selecting a sampling frequency to use
  - Nyquist Criteria or Shannon's Theorem
- The sampling frequency (f_s) must be AT LEAST TWICE (a factor of 2.5 is sometimes used as a rule of thumb!) the highest frequency present in the waveform being sampled
  - Remember real world signals can be viewed as being formed by a sum of sine waves of different frequencies
- However … in some application spaces…. It is impossible to quantify as there is no "analogue" signal to measure or, arguably more correctly, it is not practical to measure the underlying signal
  - e.g. Financial data

Aliasing

- What happens if this criteria is NOT adhered to ?
- If there are frequency components present in the waveform being sampled which have a frequency greater than $f_s/2$ (**the Nyquist frequency)** then a distortive effect known as **aliasing** will occur
- Aliasing results in a sine wave whose frequency is greater than the Nyquist frequency (say $f_s/2+a$) appear as if it were a completely different frequency which is less than the Nyquist frequency (say $f_s/2-a$)
- Thus, once a signal is sampled it only makes sense to talk about frequencies in the range of 0Hz up to $f_s/2$ Hz
- The function of the **anti-aliasing** filter/circuit is to ensure that this criteria is met
  - It is an electronic circuit which “blocks” all frequencies above $f_s/2$
  - Well, approximately blocks them – in reality it attenuates them heavily

Example of Aliasing



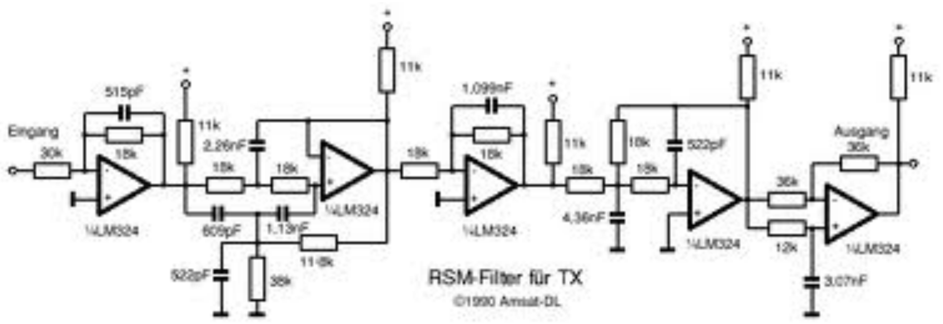
This time we consider sampling a 150 Hz sine wave with a sampling frequency of 200 Hz – Clearly the Nyquist criteria is NOT met!

The original 150 Hz sine wave now actually appears like a 50 Hz sine wave!
Note: It “aliases” to 50 Hz = 150Hz-200Hz/2

Some Notation

- “Real World” signals are termed **continuous** waveforms
  - They have a value at all moments in time
  - They are generally represented by notation in the form x(t) or y(t)
    - The “t” indicates it exists at all times!
- The process of sampling results in the continuous waveform now only existing at the specific sampling points $t=0$, $t=T_s$, $t=2T_s$, $t=3T_s$….. $t=nT_s$
  - Therefore we now have a sequence of samples $x(t=0)$, $x(t=T_s)$, $x(t=2T_s)$, $x(t=3T_s)$ …. $x(t=nT_s)$
  - This is more commonly noted as x[0],x[1],x[2],x[3] or x[n] in general
    - n is known as the **sample index**
- Consider a continuous sine wave the equation of which is:
  - $x(t)=\sin(2\pi ft)$
- The equivalent equation for the sampled sinusoid is got by replacing t with nT_s
  - $x[n]=x(nT_s)= \sin(2\pi fnT_s)= \sin(n2\pi fT_s)= \sin(n2\pi f/f_s)= \sin(n\theta)$
  - The term $\theta=2\pi f/f_s$ is known as **digital frequency**
  - Since f (analogue frequency) can only have values between 0 and $f_s/2$, hence θ can only have a value between 0 and $2\pi f_s/2/f_s=\pi$

Digital Filters



What is a filter ?

- In some applications, the terms “filter” and “model” are often synonymous
- Most introductory DSP courses have a far more restrictive definition of a filter
  - Operates on a sequence of input samples x[n] and generates a sequence of output samples y[n] by means of a static ***difference equation***
  - These “basic” filters are
    - Linear
    - Causal
    - Time invariant

Why use a digital filter ?

- Can be used to model an existing data set but, initially, we will use them to process a sequence of data samples that have either been previously recorded and stored or perhaps process them in “real time” (i.e. as they are sampled)
- The aim of time domain filtering is typically to enhance certain frequencies that may be present in x[n] (the “signal”) and/or to attenuate other frequencies that may be present in x[n] (“noise”)
- Of course there are cases where it may not work that well!

Difference Equation

- So how is y[n] produced from x[n]?
  - Difference Equation
- A difference equation is an iterative equation which defines how the samples of y[n] are calculated
- Examples:
  - $y[n]=0.5x[n]+0.5x[n-1]$
    - Current output sample y[n] is got by adding 0.5 times the current input sample x[n] and 0.5 times the previous input sample x[n-1]
  - $y[n]=0.2x[n]+0.8y[n-1]$
    - Current output sample y[n] is got by adding 0.2 times the current input sample x[n] and 0.8 times the previous output sample y[n-1]
- We will restrict the types of filters that we will look at:
  - Time invariant (the difference equation does NOT change over time)
  - Linear (difference equation will NEVER have a term of the form x[n-1]y[n-3] or x[n]x[n-5] or anything like this!)
- Have you seen this before ?
  - Moving Averages !!!
- Difference Equations clearly explain how to calculate y[n] from x[n] but you cannot tell what the filter does in terms of frequency content

FIR and IIR Filters

- There are two classes of digital filters
- Finite Impulse Response (FIR) have difference equations which only contain terms with x[n] and previous x[n] samples
  - $y[n]=0.25x[n]+0.25x[n-1]+ 0.25x[n-2]+ 0.25x[n-3]$
  - Also called non-recursive filters
- Infinite Impulse Response (IIR) have difference equations which contain BOTH terms with x[n] and previous x[n] AND previous y[n] samples
  - $y[n]=0.2x[n]+0.8y[n-1]$
  - Also called recursive filters
- What's the difference ?
  - Different characteristics that make one type preferable to the other under certain scenarios

Calculation of Difference Equation – Non-Recursive

n	x[n]	-0.5x[n-1]	2x[n-3]	y[n]
0	0.5	0	0	0.5
1	1.0	-0.25	0	0.75
2	-0.2	-0.5	0	-0.7
3	-0.8	0.1	1	0.3
4	-1.2	0.4	2	1.2

Difference equation: $y[n]=x[n]-0.5x[n-1]+2x[n-3]$

Calculation of Difference Equation - Recursive

n	x[n]	-2x[n-1]	0.5y[n-2]	y[n]
0	0.5	0	0	0.5
1	1.0	-1.0	0	0
2	-0.2	-2	0.25	-1.95
3	-0.8	0.4	0	-0.4
4	-1.2	1.6	-0.975	-0.575

Difference equation: $y[n]=x[n]-2x[n-1]+0.5y[n-2]$

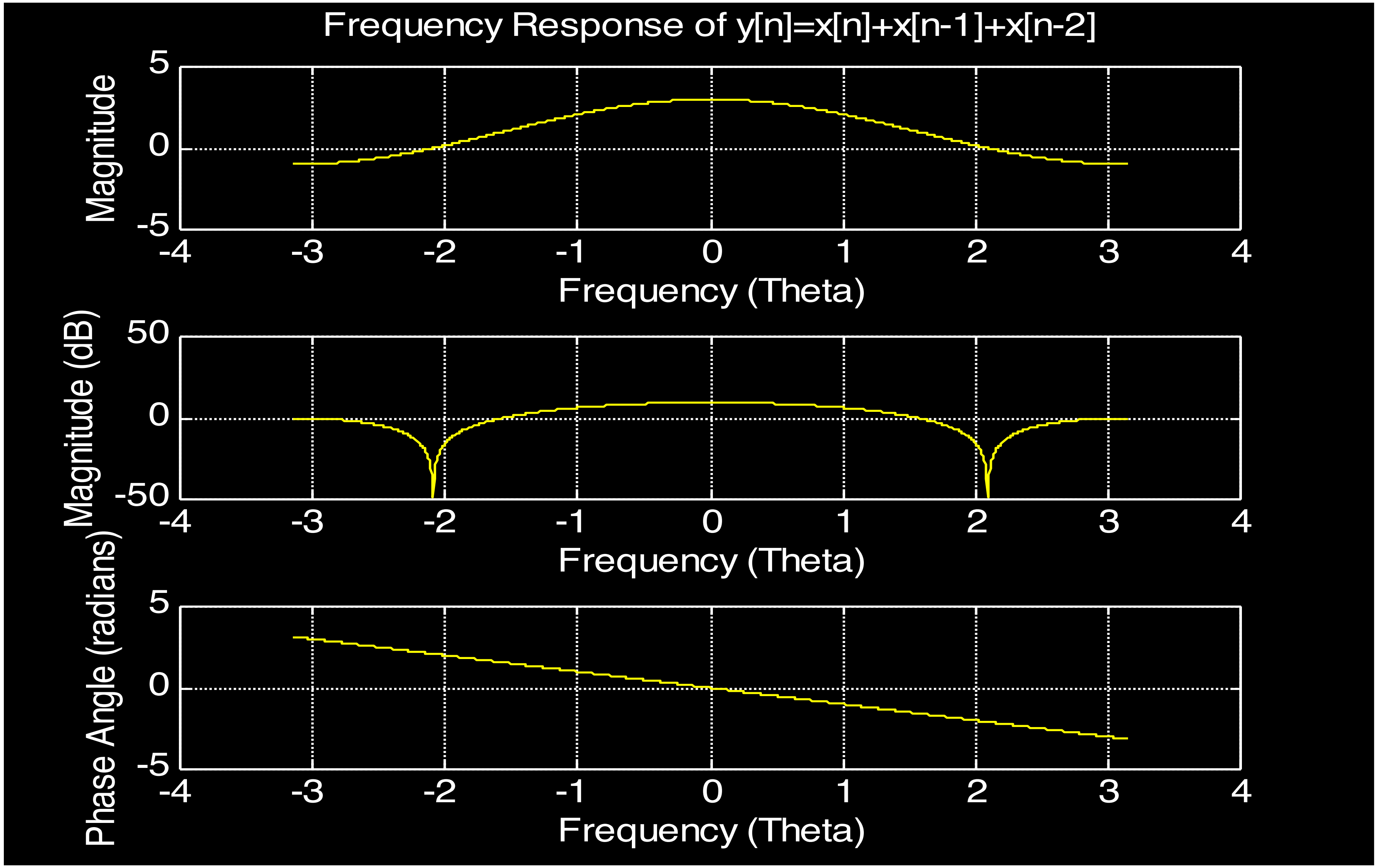
Frequency Response

- So, knowledge of a difference equations allows you to write a program to implement the filter
  - But how are these difference equations determined to start with ?
- A small number of basic filters are commonly known
  - Moving average FIR filters!
- All other digital filters have to be **designed**
- There are a variety of techniques which can be used to generate a difference equation which implements a desired filter
- “Desired” is what sense ?
  - The filter will implement a desired “frequency response”
    - It will attenuate the desired frequency range(s) and/or enhance a different desired frequency range
- So the **Frequency Response** is another alternate way of describing the behaviour of a filter
  - It is can be determined as an equation (and engineering student do indeed learn how to calculate such an equation from knowing the difference equation)
  - More commonly (and more usefully) the frequency response is shown graphically

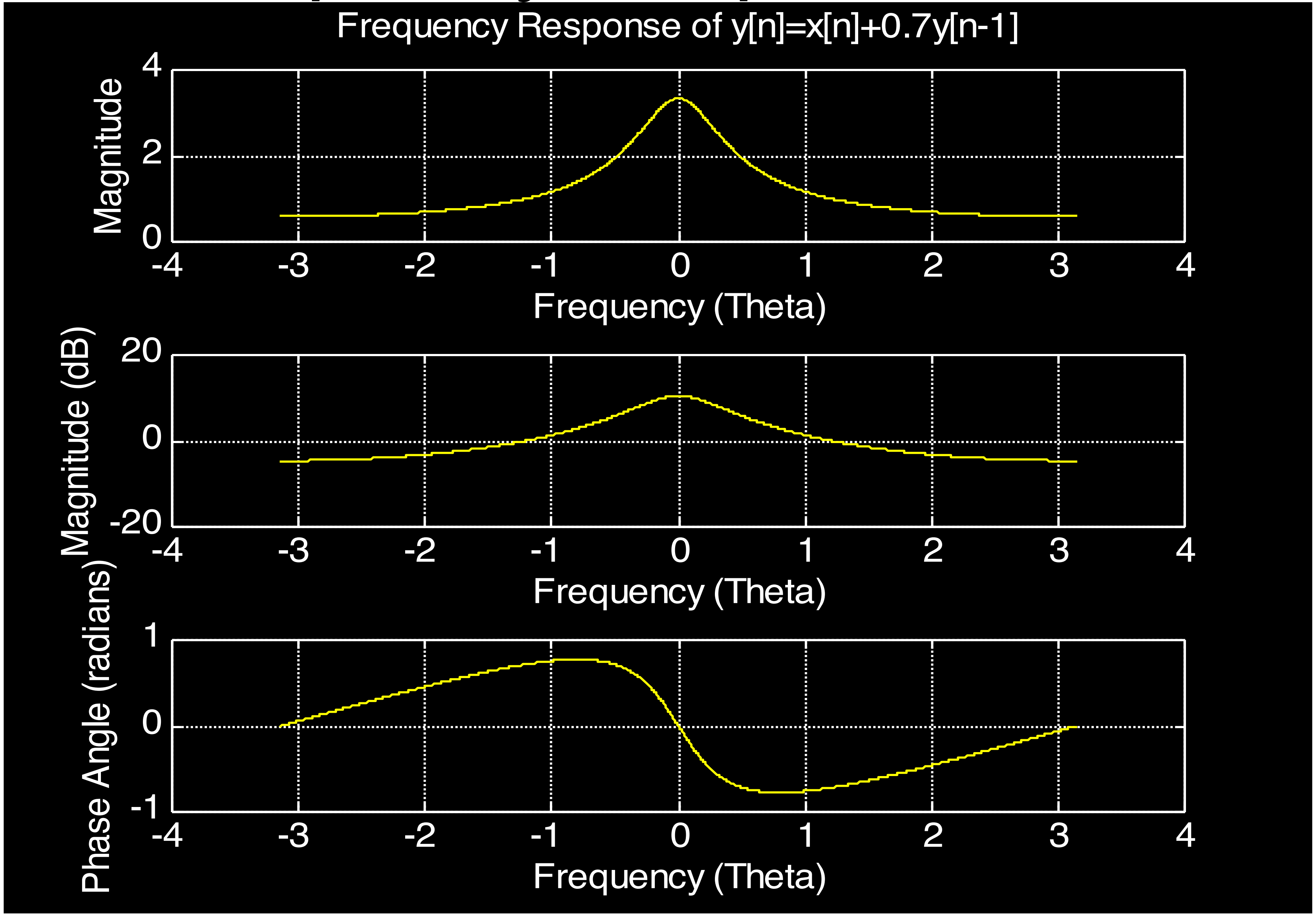
Frequency Response

- The two graphs which constitute a frequency response are:
  - Magnitude response
  - Phase response
- If $x[n]=\sin(n\theta)$
  - Input to the filter is a sampled sine wave
- Then $y[n]=G(\theta) \sin(n\theta+\phi(\theta))$
  - Output of the filter is a sampled sine wave at the same frequency but whose amplitude has been scaled and which has undergone a phase change
    - More correctly this is the “stable” output after a certain amount of time after the input was applied!
- The amplitude scaling factor is the magnitude response for that frequency whilst the phase change is the phase response for that frequency
- The magnitude response is often quantified in **decibels**
  - 20log(Magnitude Response) = decibels (dB)

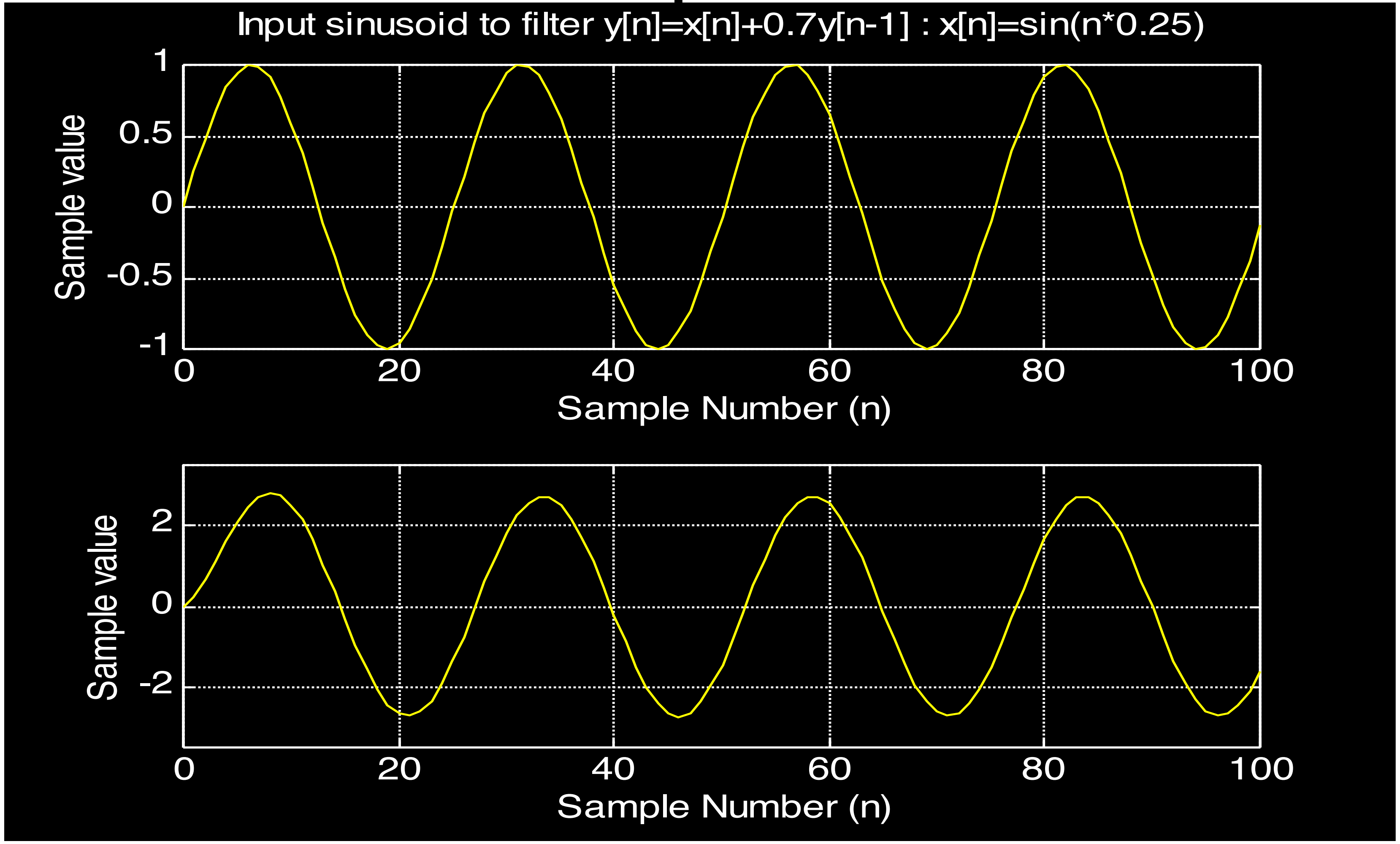
Frequency Response - FIR



Frequency Response - IIR

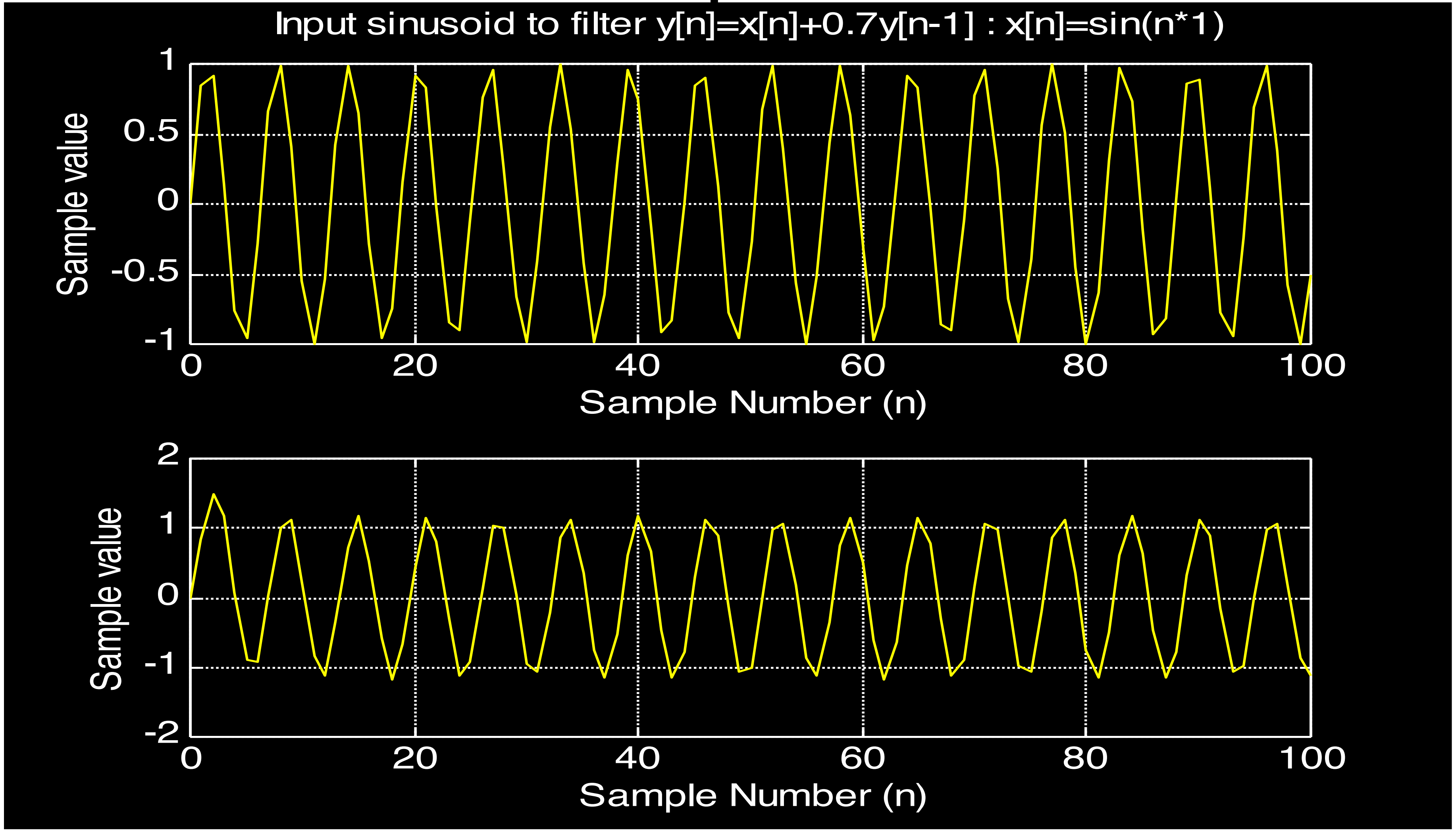


Filter's Response to Sinusoidal Input



Note : Scaling and phase shift of output signal

Filter’s Response to Sinusoidal Input



Note : Scaling and phase shift of output signal and short “transient” signal

Importance of Frequency Response

- Clearly, the magnitude response of a filter is VERY important as it indicates how the filter “attenuates” or “enhances” particular frequencies
- In many applications, the phase response of a filter is not that important
  - However, if the signal being filtered is composed of several different frequencies (as most “real world” signals are) then the phase response of a filter may be important
  - Also filters introduce a “delay” which is directly associated with this phase response

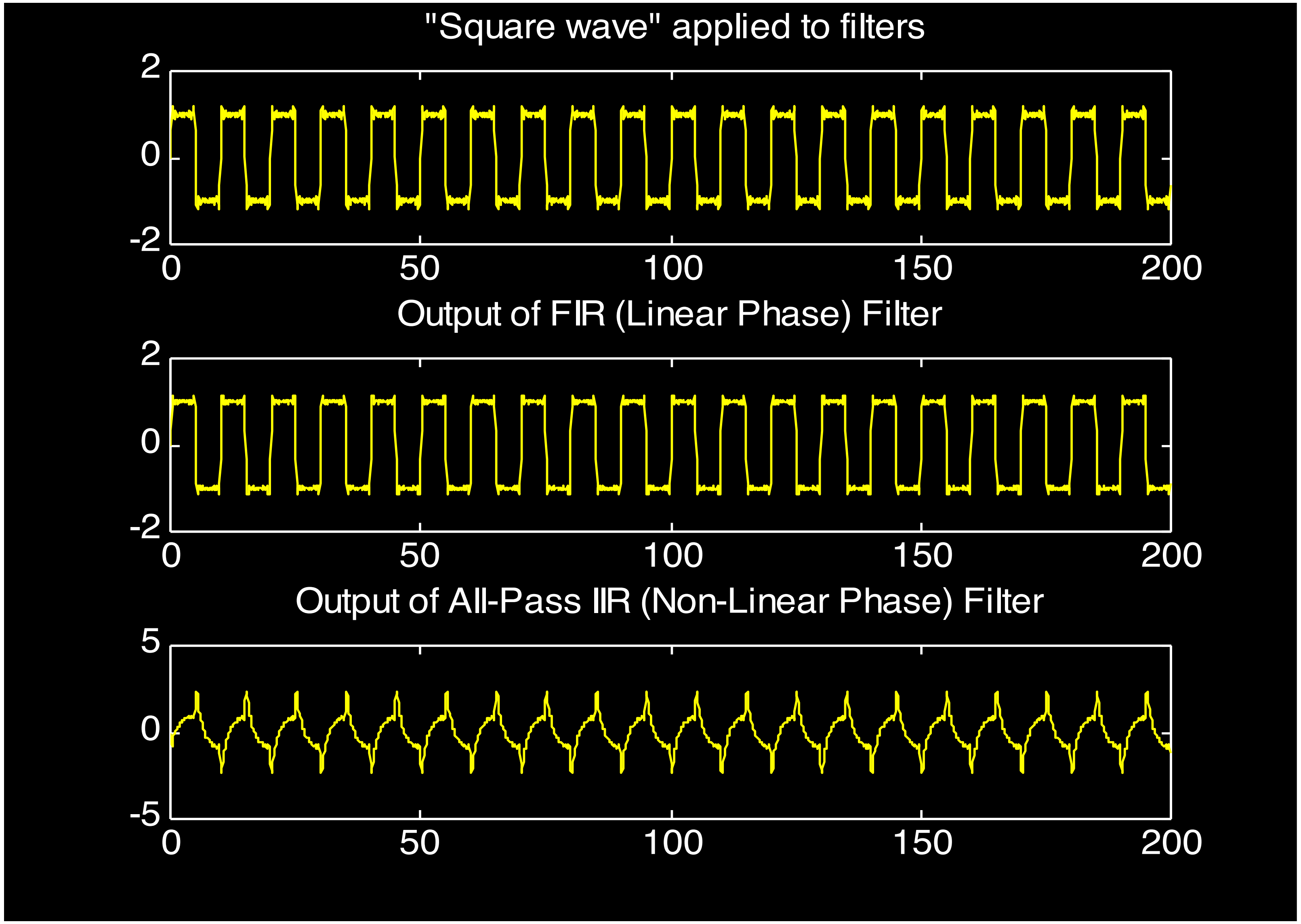
Phase Response

- The phase response indicates the phase shift which a sinusoid at a particular frequency suffers
- If the phase response of a filter is a straight line then it is said to have a **linear phase response**
  - Linear phase responses mean that another quantity known as **group delay** is constant for all frequencies
- This **may be** important when a signal composed of many different frequencies is applied to the filter

Linear Phase Filters

- If each of the frequencies present do NOT suffer the same group delay then the signal will be distorted in the time domain event if the filter does NOT alter their respective amplitudes
- Only certain FIR filters have a linear phase responses
- The are certain applications where this type of distortion is important and others where it is not!

Distortion due to Non-Linear Phase Response



Designing Filters

- Usually it is a case that you will have some idea of the desired frequency response of the filter you need
  - Low Pass Filter (LPF)
  - High Pass Filter (HPF)
  - Band Pass Filter (BPF)
  - Band Stop Filter (BSF)
- Many “design tools” allow you to specify the desired frequency response characteristics and they generate the coefficients of a difference equation that delivers this frequency response

Correlation as a Signal Processing Tool

- Correlation is an often used in a signal processing setting to investigate temporal relationships between two signals
- Consider an application like radar or sonar
  - Sonar emits an acoustic pulse (containing a number of cycles of a pure tone)
  - After travelling through the medium (e.g. water) the acoustic wave will bounce off an object and travel back to the sonar unit
    - Getting smaller and corrupted by “noise” along the way

Correlation

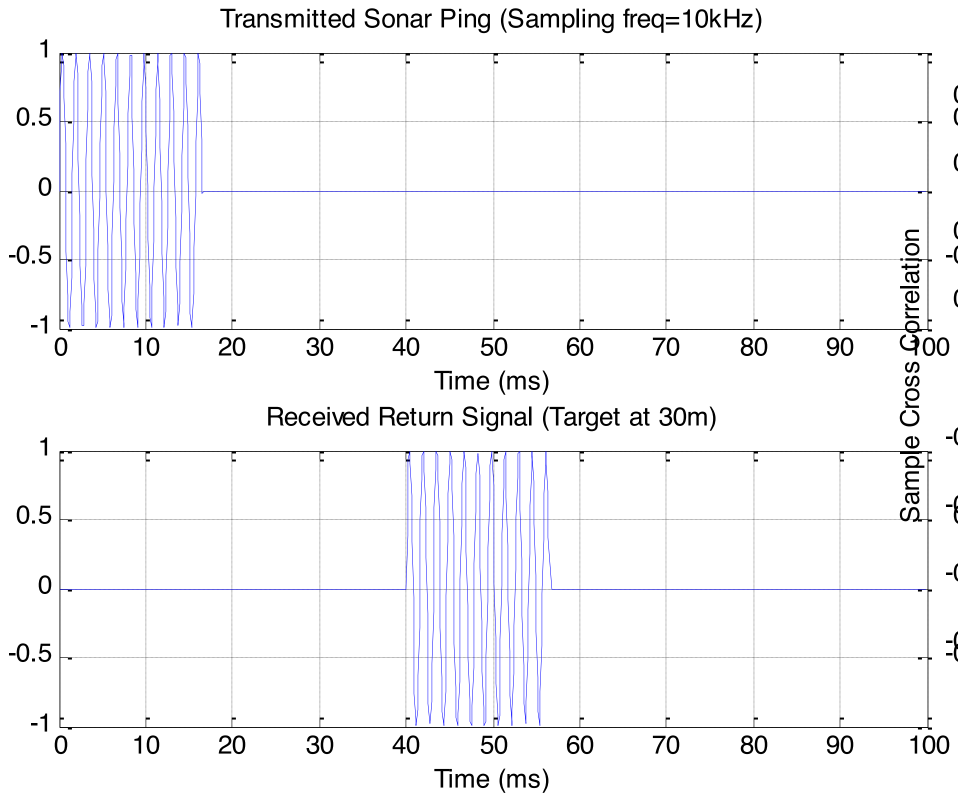
- We want to determine the distance of the target from the sonar device by reliably estimating the time difference between the “ping” and the “echo” back
  - More generally, we have two sampled signals (the “ping” and the “echo”) that we know\fell are “similar” except for a time “lag”

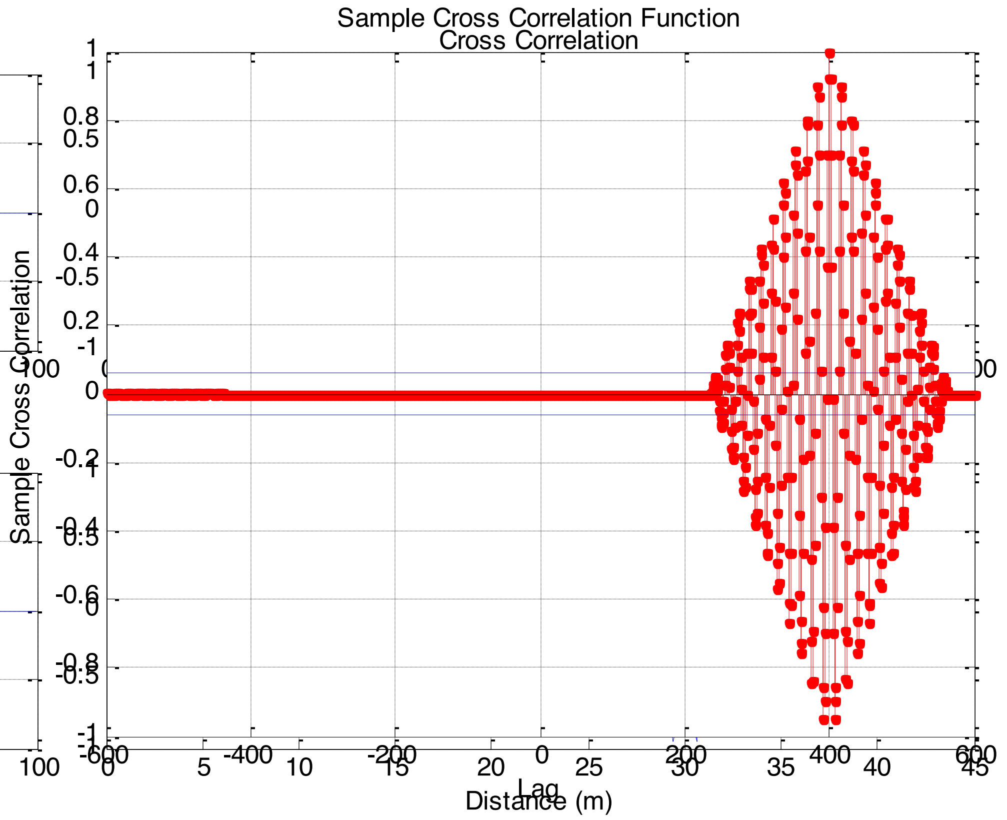
The solution!

$$\emptyset(\tau) = \sum_{n} x[n]y[n+\tau]$$

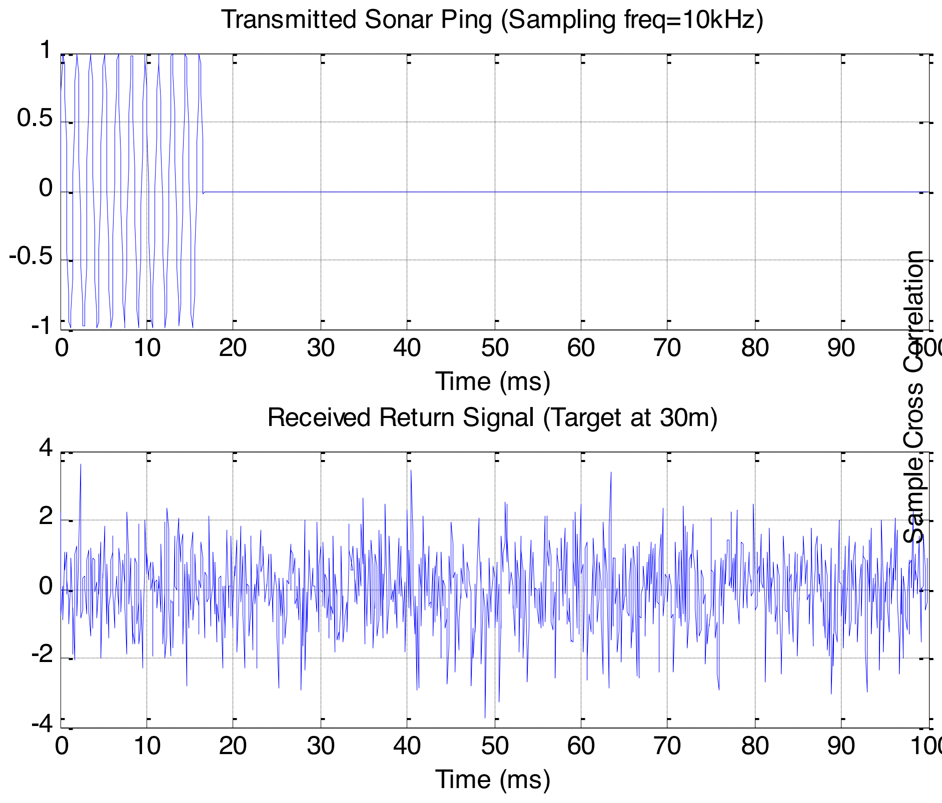
- Calculate the (cross-)correlation of the “ping” samples (y[n]) and the “echo” samples (x[n]) at a number of sample “lag” or delay values (τ)
- Search for the “lag” which has the largest (positive or negative) cross-correlation value

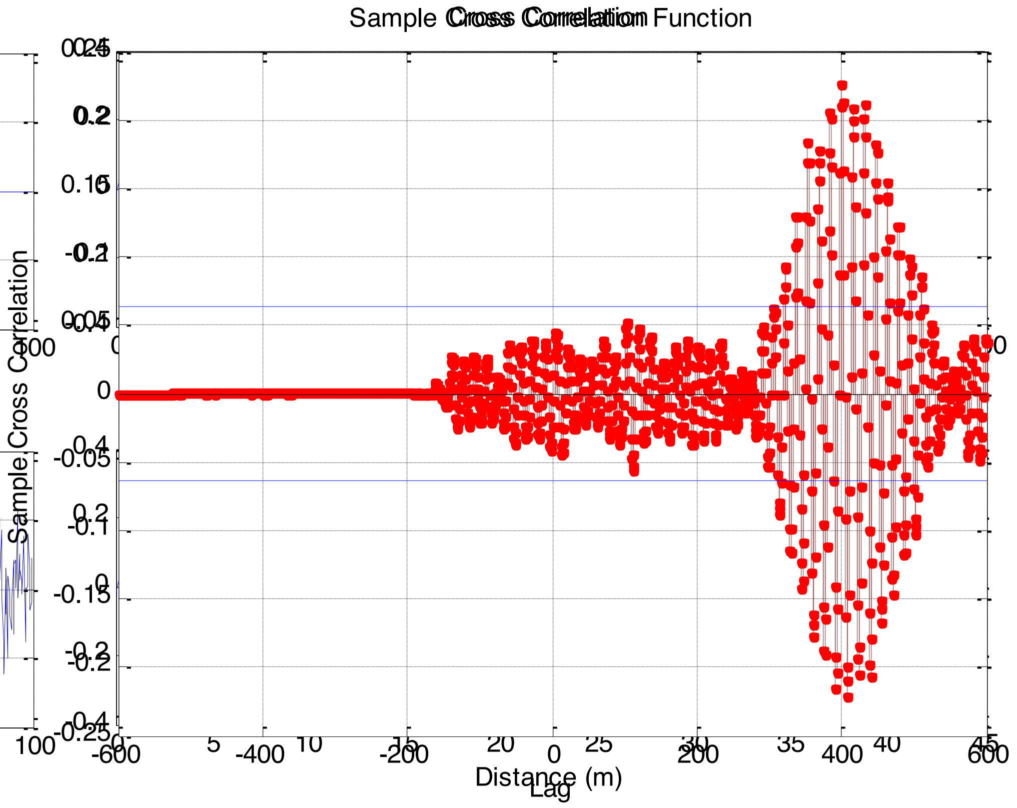
Simple problem ?





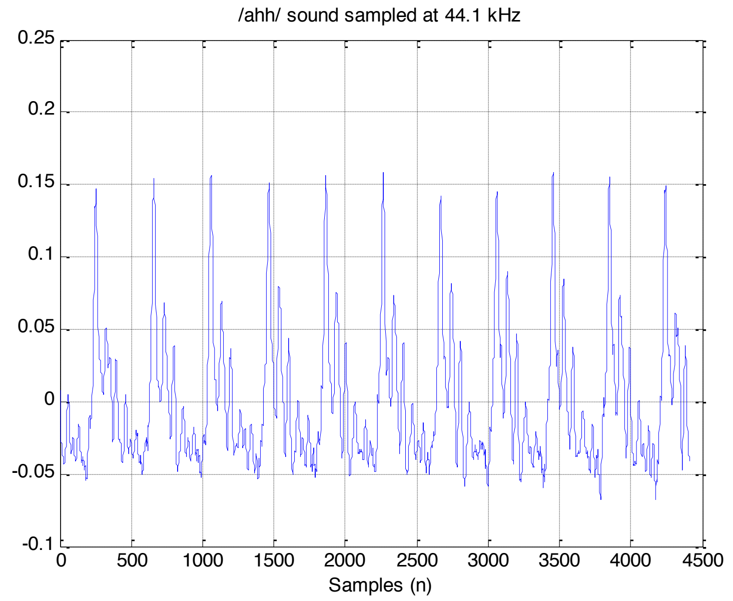
Bit more challenging in the real world though!

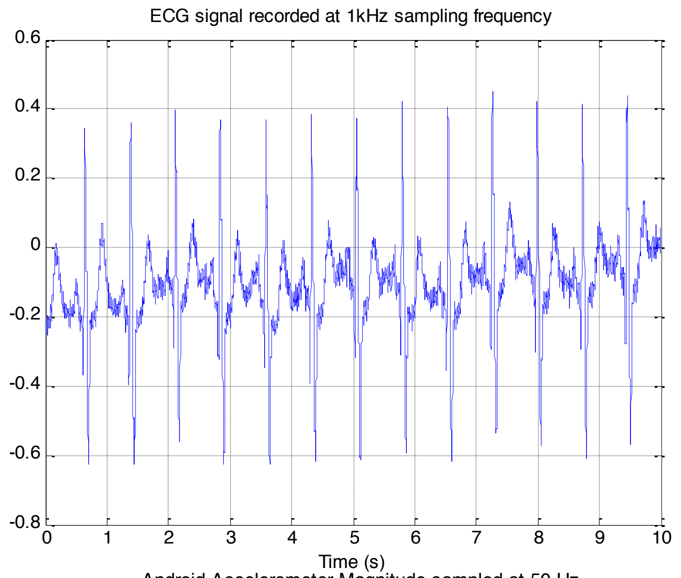


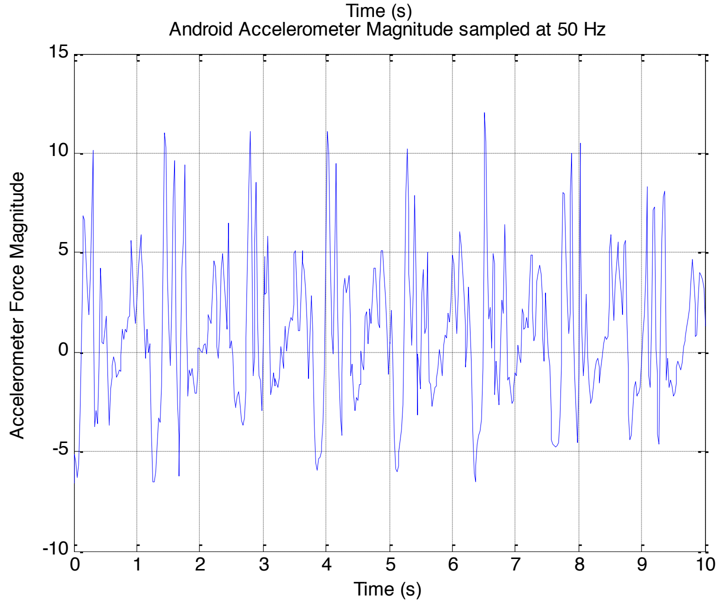


Noise !

Periodicity Estimation







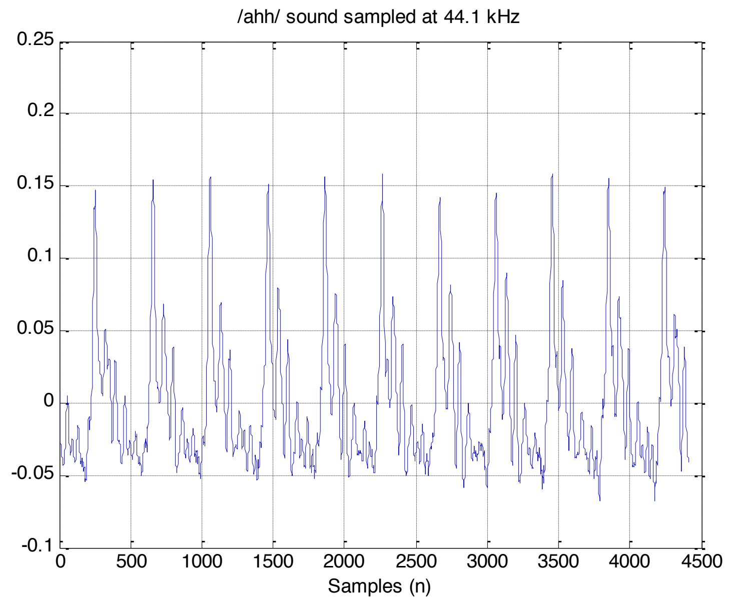
- Periodicity estimation is a multi-domain problem
- (Auto)-correlation and “related” algorithms offer elegant and computationally “simple” means of getting robust estimates

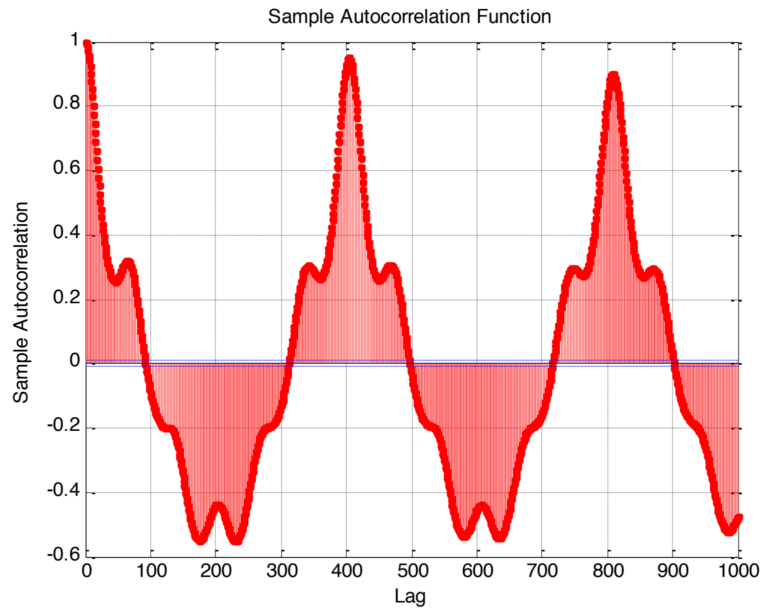
Use Autocorrelation

$$\emptyset(\tau) = \sum_{n} x[n]x[n+\tau]$$

- Calculate the autocorrelation function for various lag values
- Maximum (positive!) value in autocorrelation should happen when the lag is equal to the period of the signal

Example

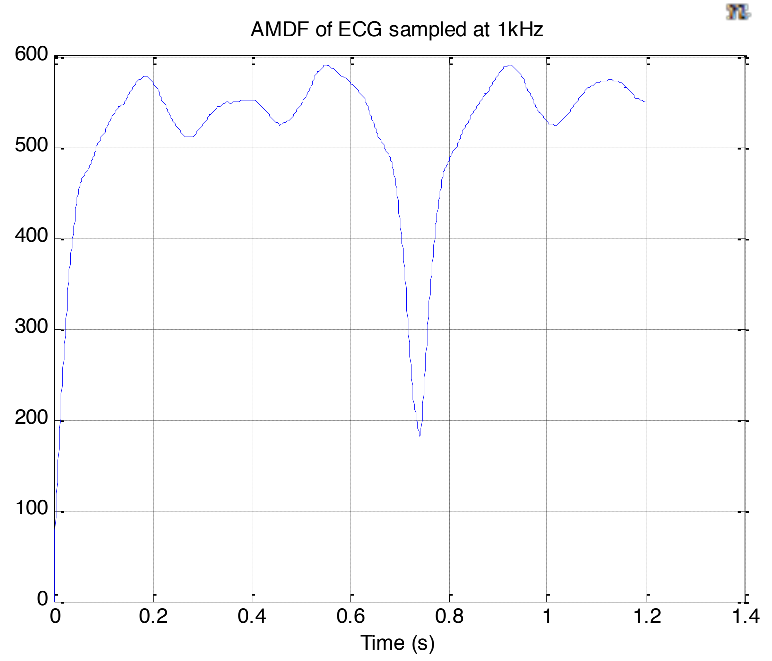


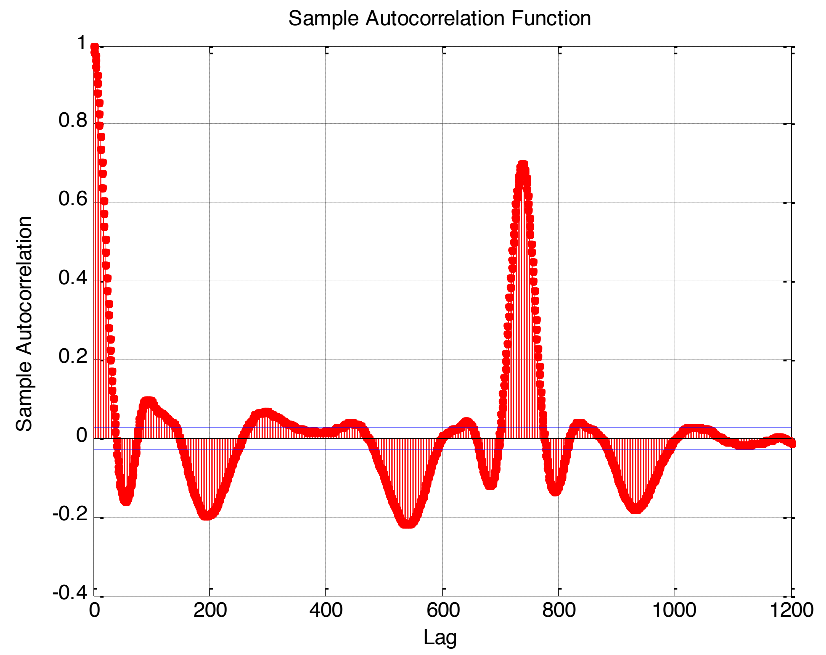


- The largest peak in this case occurs at a time lag of 410 samples
- Thus the period is 410/44100= 9.3 ms or 107 Hz

AMDF – Alternative to Autocorrelation

$$AMDF(\tau) = \sum_n \left| x[n] - y[n+\tau] \right|$$





•Average Magnitude Difference Function (AMDF) may offer a "simpler" and more obvious means to autocorrelation for period estimation

•Often will perform better in cases where "harmonics" of the "fundamental" frequency can cause problems for autocorrelation

Some Caveats!

- Carefully avoided some of the more challenging aspects of developing a robust period estimator
- Many (perhaps most) signals, signal periodicity will be an intermittent property
  - speech – only when “voiced” sounds are being produced
  - accelerometer – only when user is actually walking/running/ cycling
- If fact ECG is probably the only application where the signal will be periodic most of the time
  - At least you hope so if its your ECG!
- Application specific detection algorithms to identify time intervals containing periodic behaviour are required
  - Along with custom heuristics for accepting valid/sane results from AC or AMDF

Frequency Domain Analysis

Frequency Domain Representation

- So far, we have introduced the concept of a frequency domain representation of a digital filter’ s operation
  - Frequency Response
- However, it would be very useful if we had a tool for estimating the frequency content (i.e. both magnitude and phase) of any sequence of samples
- The Discrete Fourier Transform (DFT) is such a mathematically tool

Discrete Fourier Transform

- The DFT is an equation which allows use to analyse a block or **frame** of samples in order to estimate the component present at ANY **particular** digital frequency (θ)
  - Mathematically speaking, the output of this equation is a **complex number** which can be represented by a **magnitude** and a **phase** value
  - The magnitude is the "amplitude" of the component at that frequency
  - The phase is the "phase angle" of the component at that frequency
  - As noted previously, only digital frequency values between 0 and π have meaning (equivalent to 0 Hz to fs/2 Hz)
- However, typically you will be interested in knowing such information for more that just one frequency

Fast Fourier Transform

- Typically, you will want to quantify the frequency components is a signal for a whole set of discrete frequencies in the range of possible digital frequencies
- This would require you to use the basic DFT equation repeatedly for EACH frequency
  - Computationally very demanding!
- The Fast Fourier Transform (FFT) addresses this issue by carrying out the DFT calculation for a certain specific set of frequencies in a computationally efficient manner
  - FFT is NOT a different type of analysis to DFT – just an efficient way of calculating the DFT for a specific set of frequencies

Properties of FFT

- The FFT uses as its input a block or frame of N consecutive samples
  - For an FFT there is a restriction that $N=2^k$ where k is some integer value!
    - Why ? This is a “restriction” results in the mathematical equations that calculate the FFT being much “faster” to complete
    - N is termed the FFT “order” (e.g. “512 point” (2^9) FFT)
  - However, if you can use frame sizes which do not meet this criteria but before the FFT is calculated the frame is **padded** with extra zero samples to meet this criteria
    - This does NOT effect the result in ANY way!

Properties of FFT

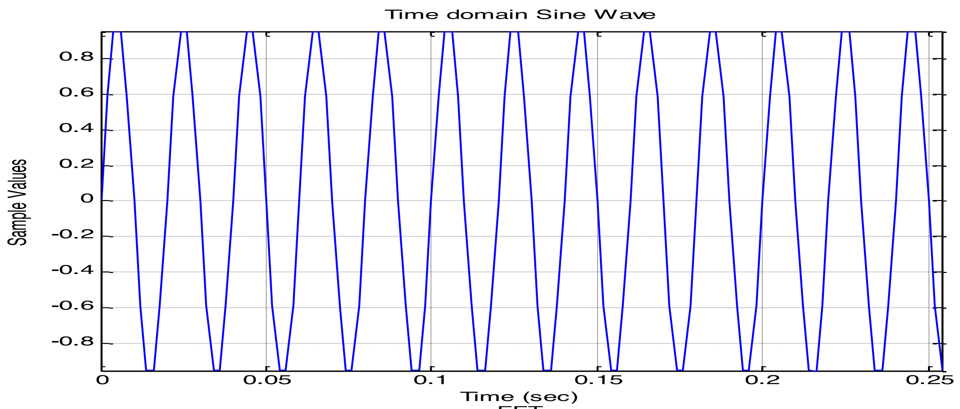
- The output of the FFT will be exact same result that you would have got if you calculated the DFT for these specific set of N digital frequencies
  - In other words, it tells you information on the components present at these N frequencies
  - Just much more “quickly”!
  - N points in time $\rightarrow$ N frequency points after the FFT

FFT Frequency Resolution

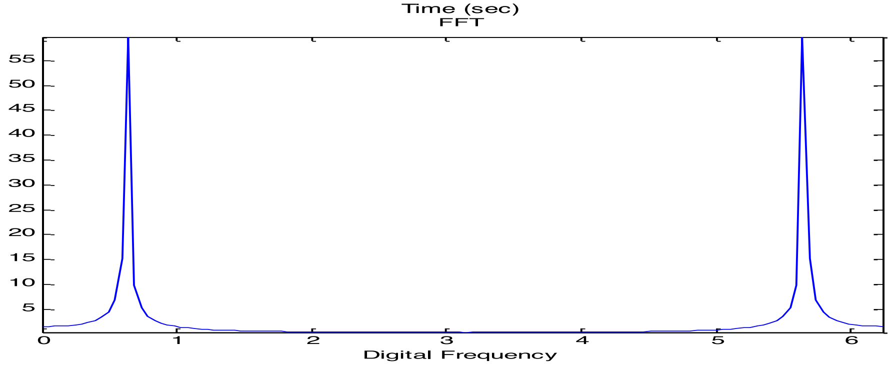
- What are these N specific frequencies ?
- For mathematical reasons, this set of digital frequencies cover the range of $\theta=0$ (0 Hz) to $\theta=2\pi$ (f_s Hz)
- The set of N frequencies start at $\theta=0$ (i.e. 0 Hz) and are spaced $2\pi/N$ (f_s/N) apart
  - Therefore there will only be N/2 of these points covering frequencies within the standard digital frequency range of $\theta=0$ (0 Hz) to $\theta=\pi$ ($f_s/2$ Hz)
    - Starting at 0 Hz with the last one at $f_s/2-f_s/N$ Hz
- Allow you to view the “spectrum” of the signal

Example of FFT

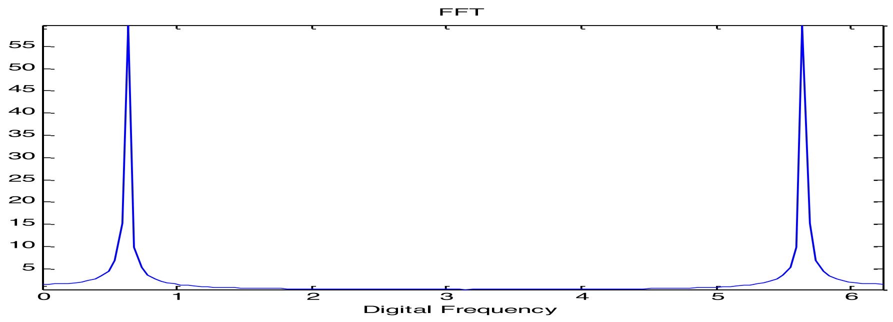
Time Domain Waveform



FFT Result (θ)
(Magnitude Only)

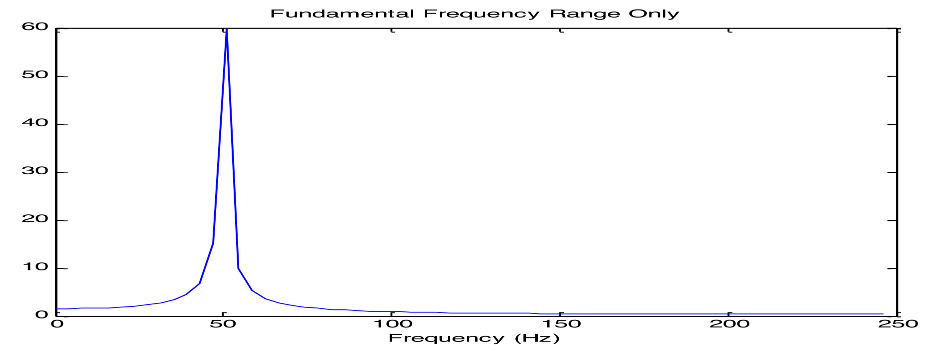


FFT Result (Hz)
(Magnitude Only)

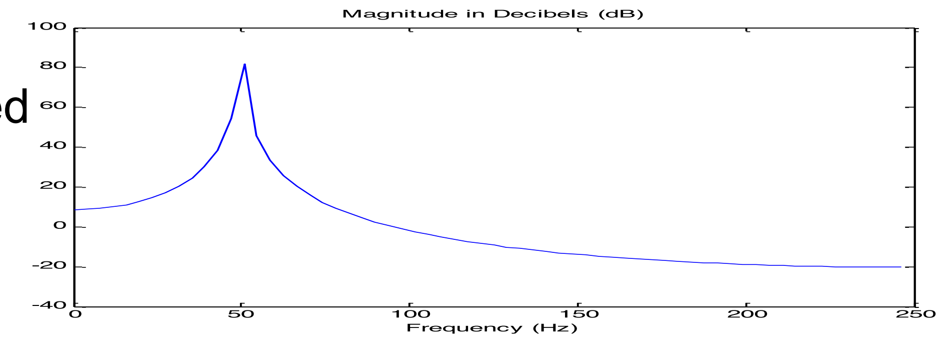


Example of FFT

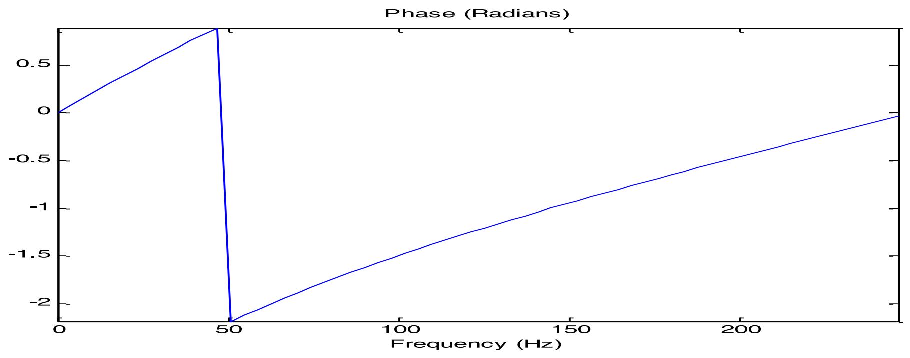
FFT for Fundamental Frequency Range Only



Magnitude Displayed In Decibels



FFT Phase Result



What if I want more resolution ?

- If you want more “points” in the output of the FFT (i.e. you want to see what is going on at finer spaced frequencies)
  - Increase the FFT order (N)
  - Does not mean that you need more time domain points to do this !!!
  - Be clear in your mind of the difference between the FFT order (and how it can be increased by padding) and the “frame size” of signal samples you are analysing
    - You can take a frame of 300 samples of a signal and pad them to 512 points in order to complete a 512 point FFT!

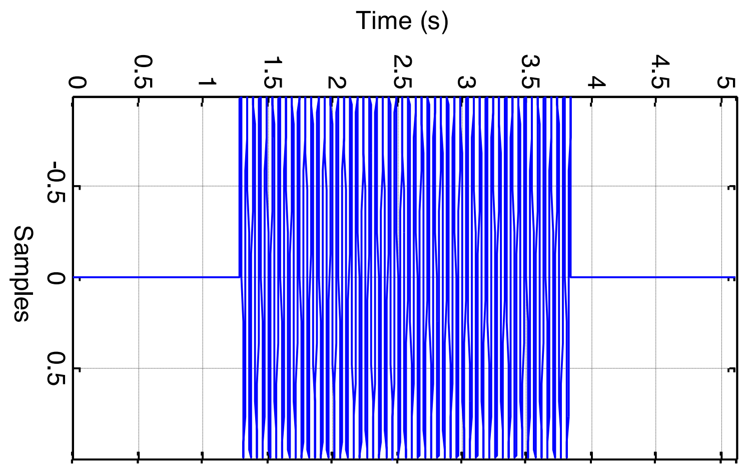
Some Caveats about FFT!

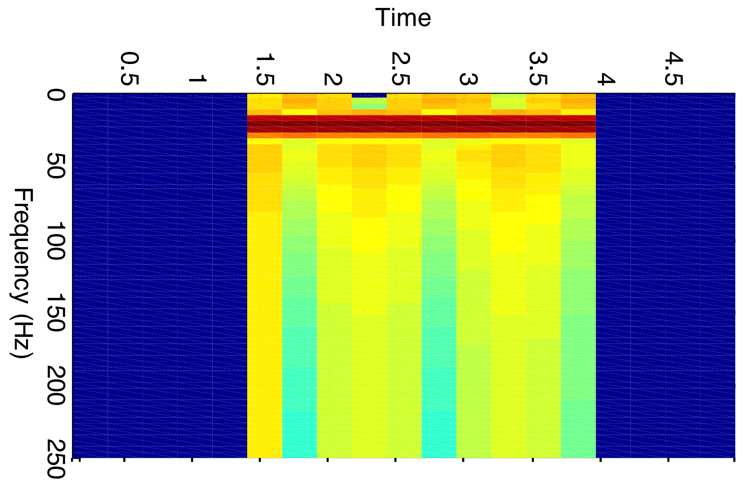
- Often your signal samples will span quite an extended period of time so you may want to break up this time period into a number of smaller individual frames and complete and FFT on each the smaller frames
  - Rather than just doing one FFT for a complete EEG duration
- There is a complete loss of time resolution capabilities in the FFT output
  - You can tell if a frequency component was present in the frame but not when it was present in the frame!
- Mathematically speaking the frequency content/spectrum should be **stationary** for the duration of the frame
  - In other words, the frequency content should not be changing during the frame duration
  - Not always the case!

Spectrogram

- A spectrogram is used to show how the FFT magnitude evolves over time
  - Or more specifically for each frame of N samples (in which case the FFT is often referred to as calculating the "Short Term Fourier Transform (STFT) of each block)
- Remember each block of N samples provides N/2 frequency points (of interest)
- Spectrogram displays these over time with the colour of the plot being used to indicate the "size" of each component

Spectrogram Example





Segments the time domain signal into short term "blocks" or "frames" with FFT used to calculate spectrum of each frame

Presents a "time-frequency" representation of the signal

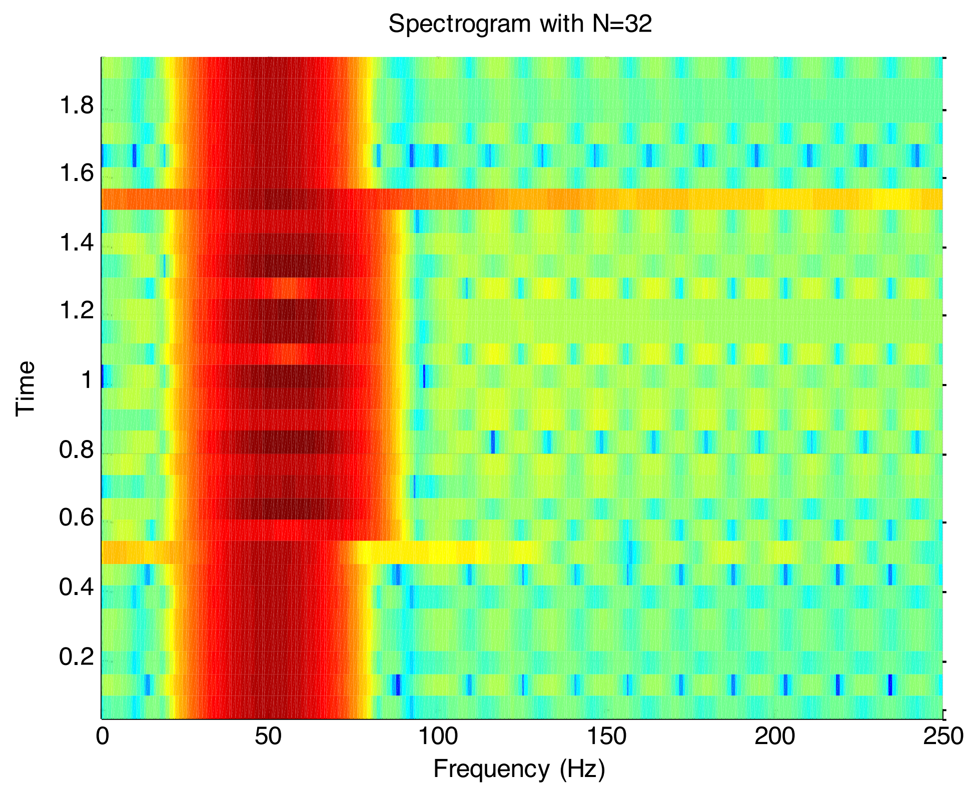
Impact of Frame Size

- The issue of time resolution can be addressed by making the frame size (N) smaller
  - This however has a negative impact on the ability to resolve information in the frequency domain – more later!
- The issue of the signal being stationary during the N samples can also have a impact of what can be deduced from the FFT output

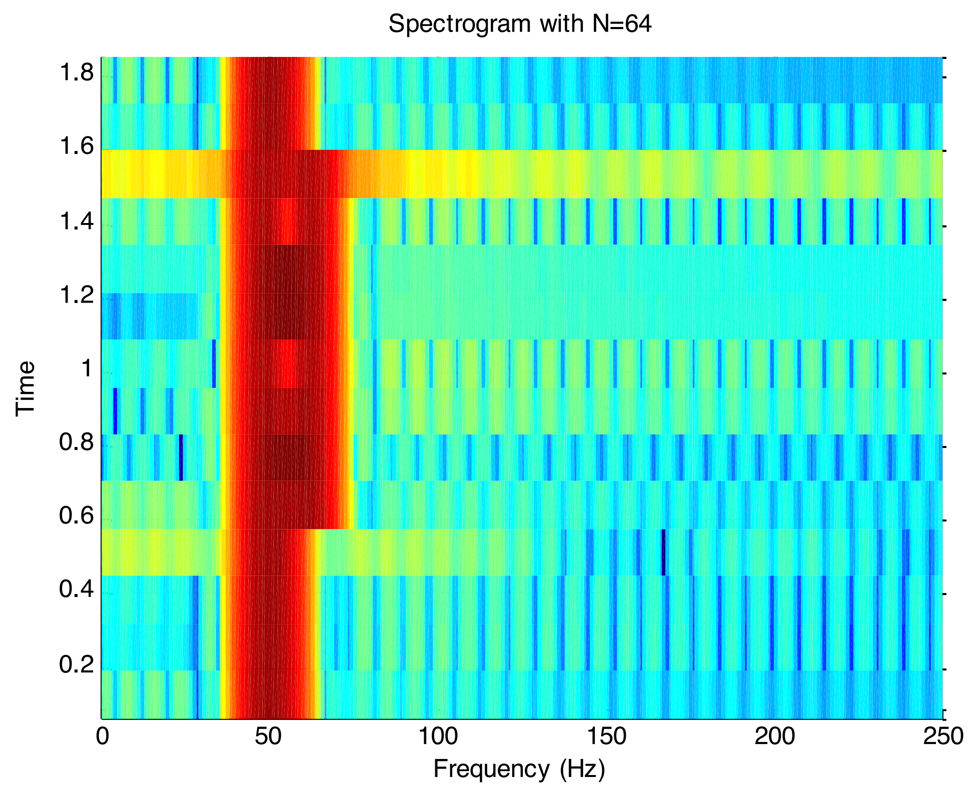
Example Impact of Frame Size

- Consider a signal made up of TWO sine waves (sampled at 500 Hz)
  - First wave is at 50 Hz and starts at time t=0
  - Second wave is at 60 Hz is present only for a burst starting at t=0.25 sec and ending at 0.75 sec
- Consider spectrogram for different frame sizes
  - Watch out for:
    - Time Resolution
    - Frequency Resolution

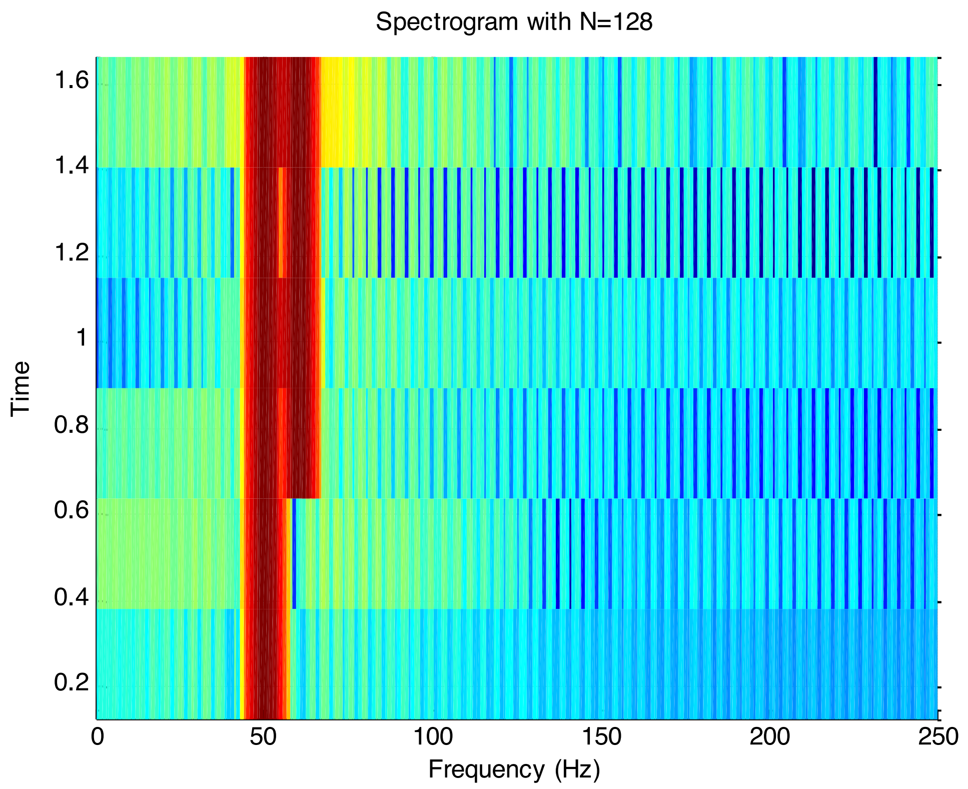
Spectrogram N=32



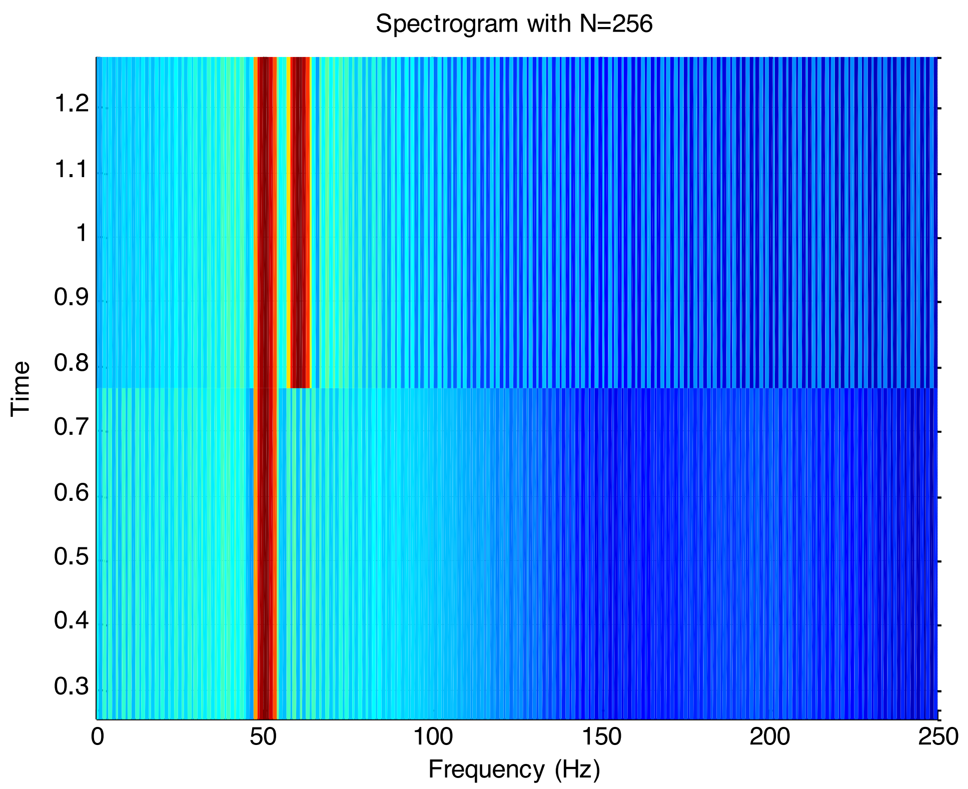
Spectrogram N=64



Spectrogram N=128



Spectrogram N=256



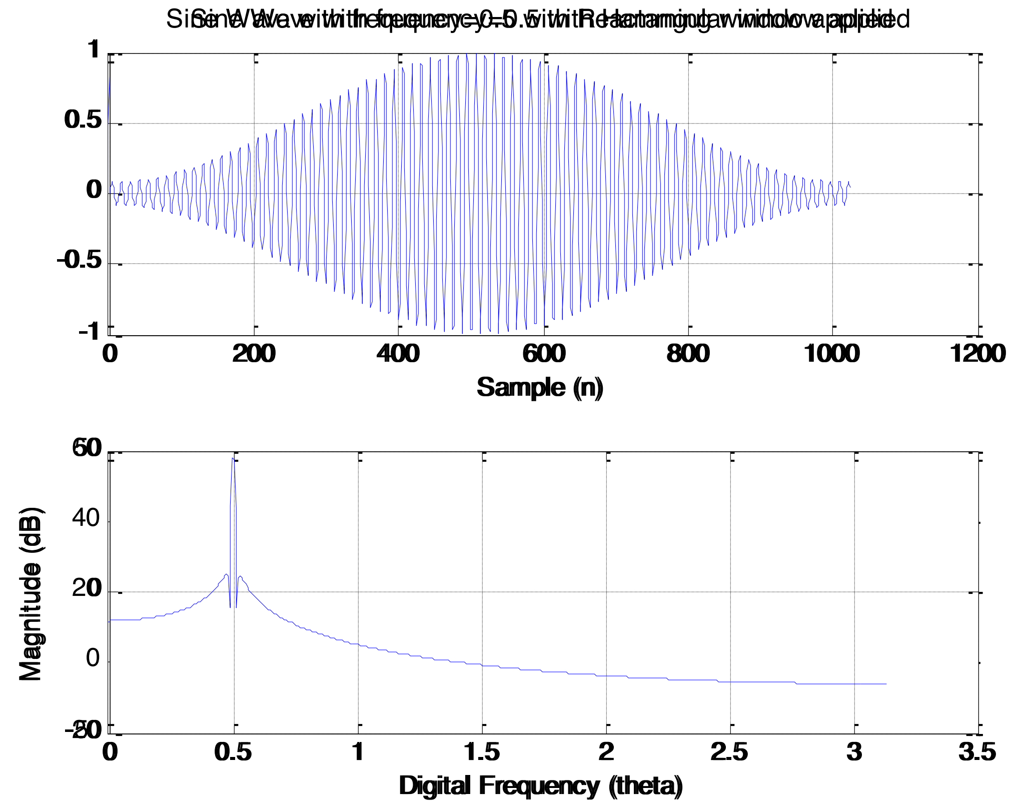
Windows

- There is a trade-off between achieving time and frequency resolution with the FFT
- The more accurate the time resolution the less accurate the frequency resolution
- Mathematically, this is due to the actual process of breaking the overall set of samples in frames of N samples
- This process is termed **windowing**
- In the simple case, a window of N samples moves along the overall signal in “jumps” of N samples for each time the FFT is calculated
- The window itself has a “non-perfect” frequency content and this distorts (“smears”) the frequency content of the signal being analysed
  - The longer the window (larger N) the less frequency distortion occurs
  - The shorter the window (smaller N) the more frequency distortion occurs

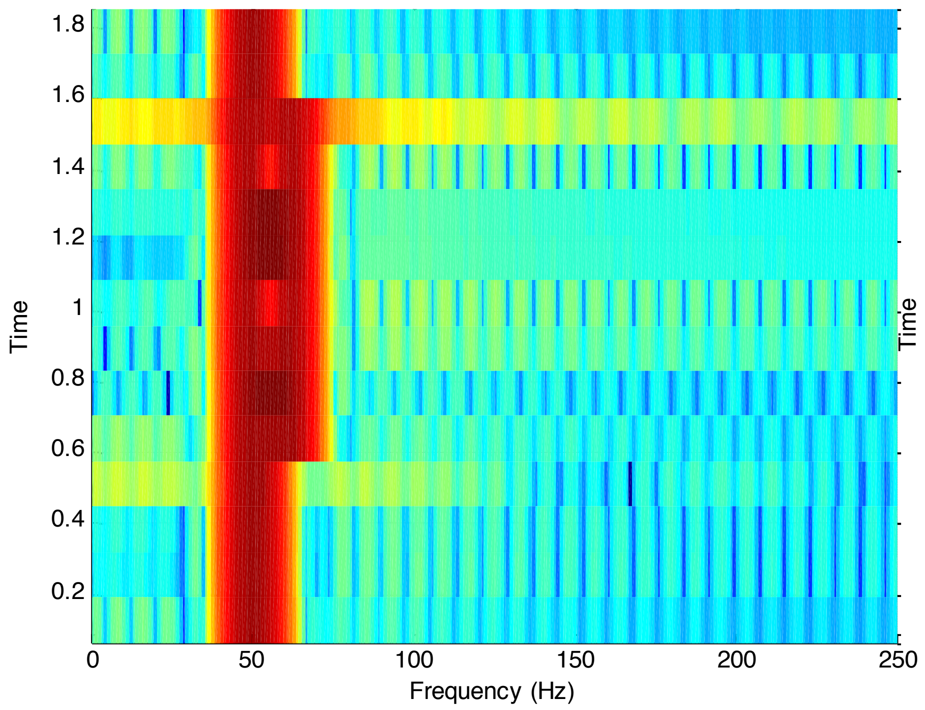
Other Windows

- The example of windowing previously shown uses what is known as a **rectangular window** of size N
  - The frame can be viewed as the multiplication of the original complete set of signal samples multiplied by an equal length **window** signal
  - This window signal has samples which are all 0 except for N samples centred around the current frame of interest
- There are other windows more commonly used that offer less “distortion”
  - Hamming Window
  - Hanning Window
  - Kaiser Window
  - Etc.

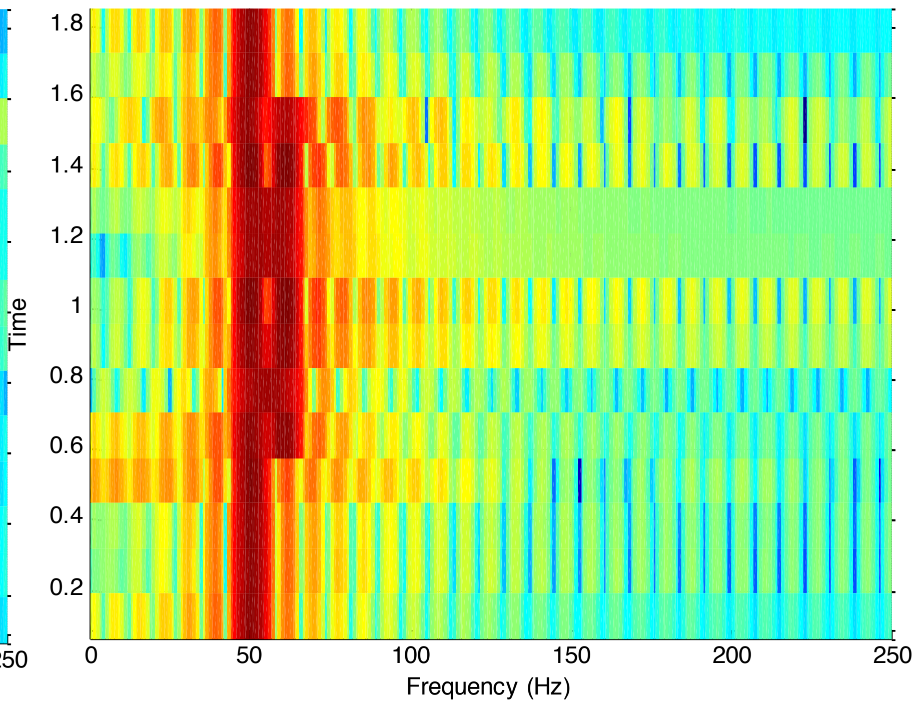
Impact of Window Type on frequency resolution



Ability to Identify Frequency Components



FFT for N=64 with Rectangular Window (Two Sine Waves F=50Hz and f=60Hz)



FFT for N=64 with Kaiser Window (Two Sine Waves F=50Hz and f=60Hz)

Power Spectral Density

- The Power Spectral Density (PSD) is closely associated with the FFT
- Theoretically it can be related to either a stochastic signal’s autocorrelation function or its Fourier Transform\Frequency Spectrum
- It offers a more statistically valid means of representing where signal power is distributed in the frequency spectrum
  - Typically at the expense of frequency resolution

PSD Calculation

- Common methods for calculating PSD use the following steps:
  - Subdivide the frame of N signal samples into P frames with M samples in them (M<N) and (P>=N/M)
  - For each of these P frames of M samples
    - Calculate the FFT of the frame of M samples
    - Square the magnitude of the resulting frequency points in the FFT (and “weight” them)
  - Average (perhaps with some weighting factor or using overlapping frames) corresponding frequency points in the P frames

Comments on PSD

- Does the PSD tell you anything new that you would not “know” from looking at the FFT?
  - Probably not!
- Does the PSD offer a “more valid” measure of the frequency make-up of a signal ?
  - Yes and certainly to journal paper reviewers!

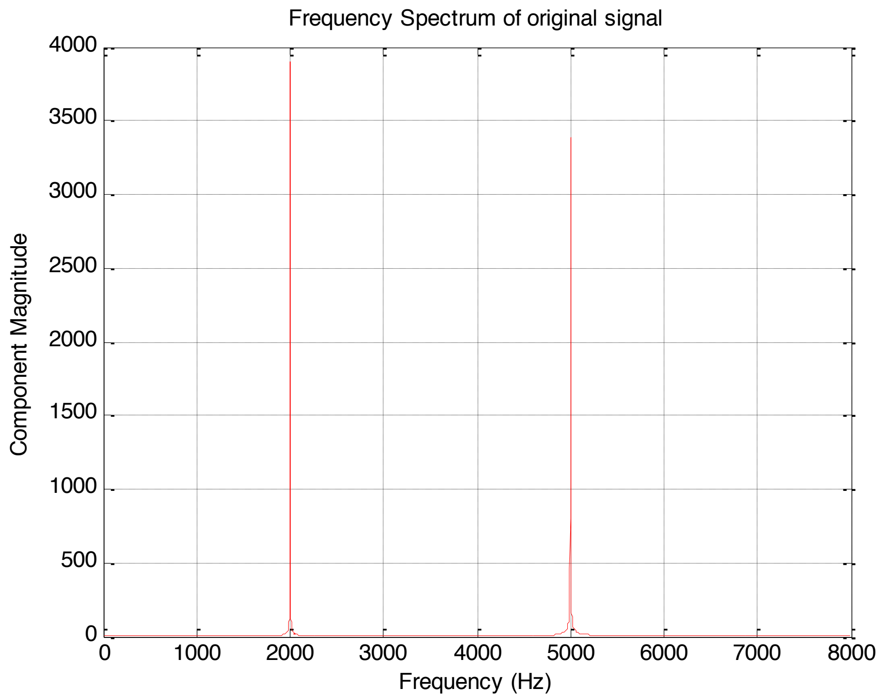
Interpolation and Decimation

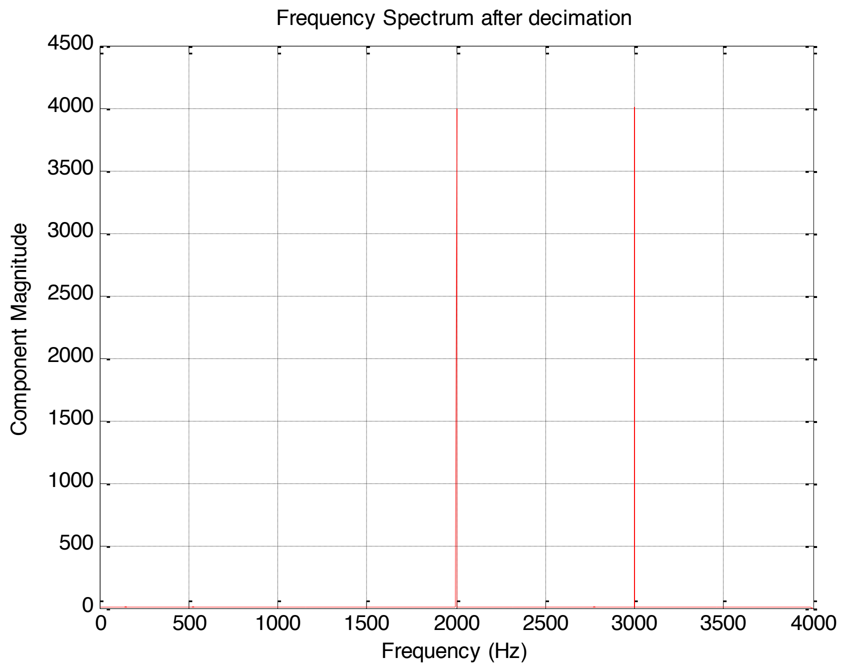
- Sometimes you may feel that you have “too” many or “too” few sample points after you have sampled your signal
- In these cases you may want to **decimate** or **interpolate** the samples
  - Reduce the number of samples/sampling frequency by a factor of N
  - Increase the number of samples/sampling frequency by a factor of N

Decimation

- Intuitively this may seem trivial
  - e.g. If you need to drop33 half the samples then just select every second sample
- Partially correct but there may be an issue with “aliasing”
  - In dropping every 2nd sample in this example we have effectively reduced the sampling frequency by 2
  - Aliasing will occur if there were (significant) frequency components in the original signal above half the **NEW** sampling frequency

Undesired Aliasing Effect





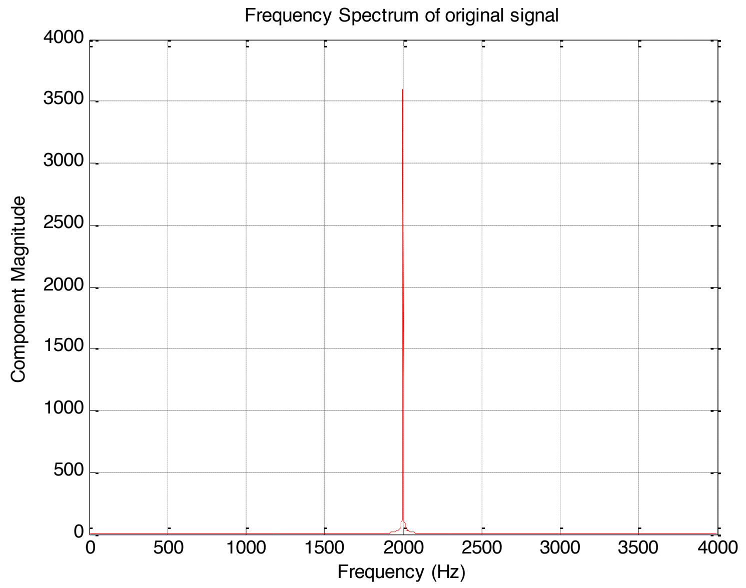
- Two (sine wave) components at 2 kHz and 5 kHz in the original signal sampled at 16 kHz
- When we decimate by a factor or 2, we can only have a signal with components up to 4 kHz (i.e. half the NEW sampling frequency)
- However, it now looks like we have a 2nd “phantom” component at 3 kHz as well as the “valid” component at 2 kHz
- We should have filtered (LPF) the original signal before decimation!

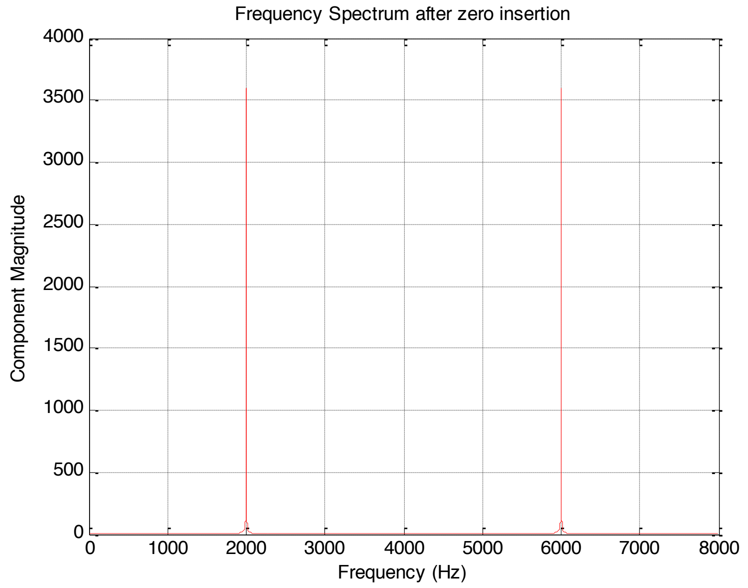
More Advanced Topics

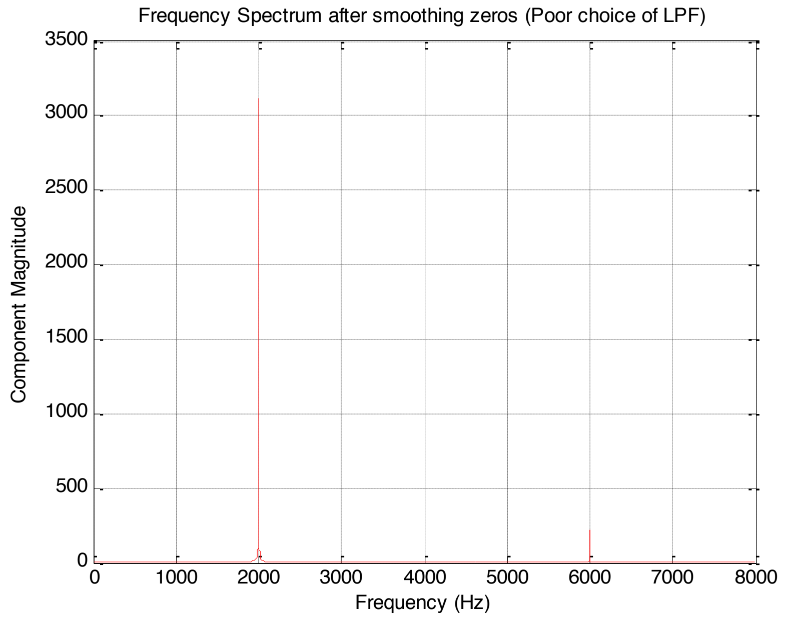
Interpolation

- There may be cases where we need to increase the number of samples\sampling frequency (Say by 2)
- Again an intuitive solution would be to insert “zero values” between each current sample and “smooth” the resulting signal
- But “smooth” really means that we apply a digital filter
  - So what filter is suitable and what happens if we don’t use one?

Interpolation Steps







A poor LPF smoothing filter was deliberately apply to illustrate how an interpolation “artefact” can remain in the frequency spectrum after smoothing

Time-Frequency Resolution Trade-off

- So with the FFT you can localise in time (Δt) OR localise in frequency (Δf) but not both at the same time

$$\Delta t \, \Delta f = \text{constant}$$

  - Short time windows offer better time resolution
  - Long time windows offer better frequency resolution
  - But whichever we choose – it’s applicable to all frequencies which often is not desirable!

Wavelet Transform

- In some applications however, it may be desirable to trade-off time and frequency resolution to different degrees at difference frequencies
- A quite common requirement would be to want
  - Better time resolution at “higher” frequencies at the expense of frequency resolution
  - Better frequency resolution at “lower” frequencies at the expense of time resolution

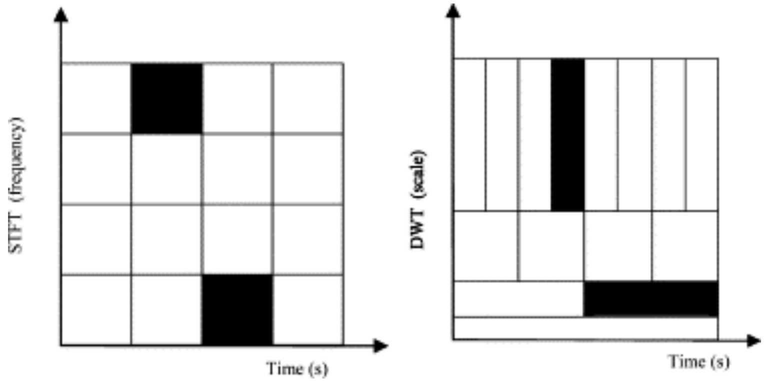
Limitation of Fourier Transform

- The limitation of the (Short Term) Fourier Transform relates back to its mathematical foundations
  - Sampled signal could (in theory) be formed by summation of an infinite number of **equal length** sine waves each with a different amplitude scaling factor (magnitude of the STFT) and each with a different phase shift (phase of the STFT)
  - All these “basis functions” have the same length and hence time resolution!

Wavelet Transform (WT)

- Wavelet analysis has a similar foundation except that the length\duration of the “basis” functions can be different at difference frequencies (called “scales in WT parlance)
  - Short “duration” basis functions used at “higher” frequencies
  - Longer “duration basis functions used at “lower” frequencies

Comparing STFT and WT for Time-Frequency Analysis



- Highlights how wavelet analysis can achieve a trade-off in time-frequency resolution compared with traditional STFT analysis

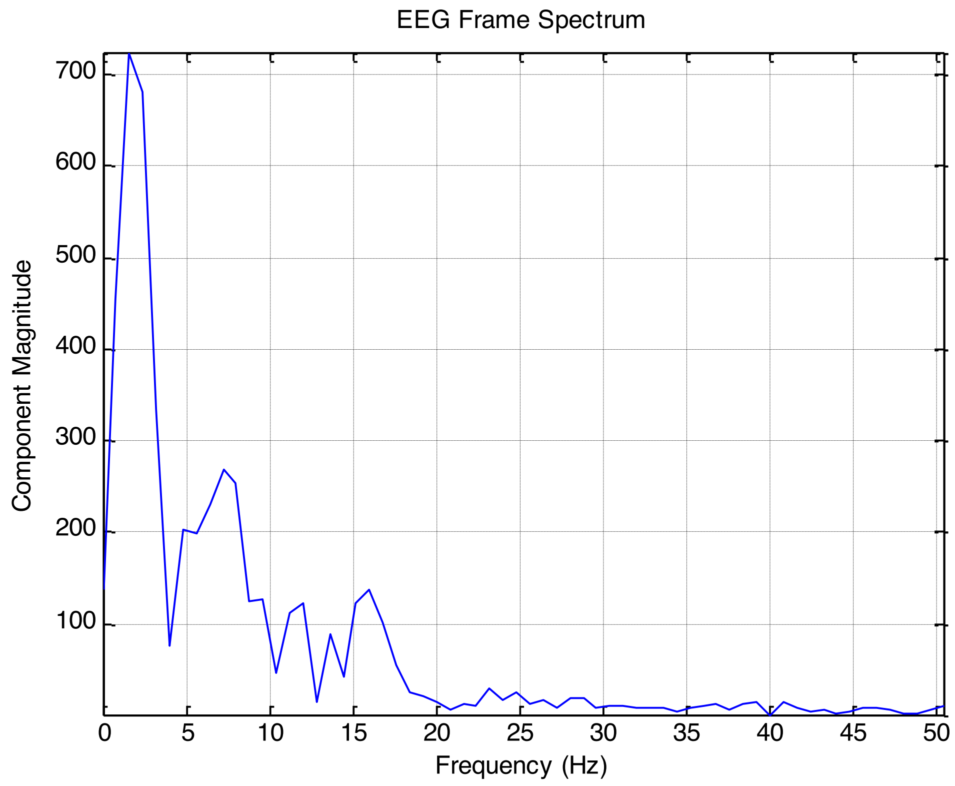
When to use WT and what needs to be done ?

- Wavelet analysis may prove a useful tool if you feel that STFT is failing to “accurately” capture localised characteristic either in frequency (low frequency spectral separation) or time (high frequency temporal resolution)
- A huge array of difference wavelet “families” have been used and it is often a question of “trial and error” to determine which is best suited to an application
- But before you start with WT – you need to ask yourself whether there is evidence in STFT analysis to make you think that WT will uncover something new in your signal

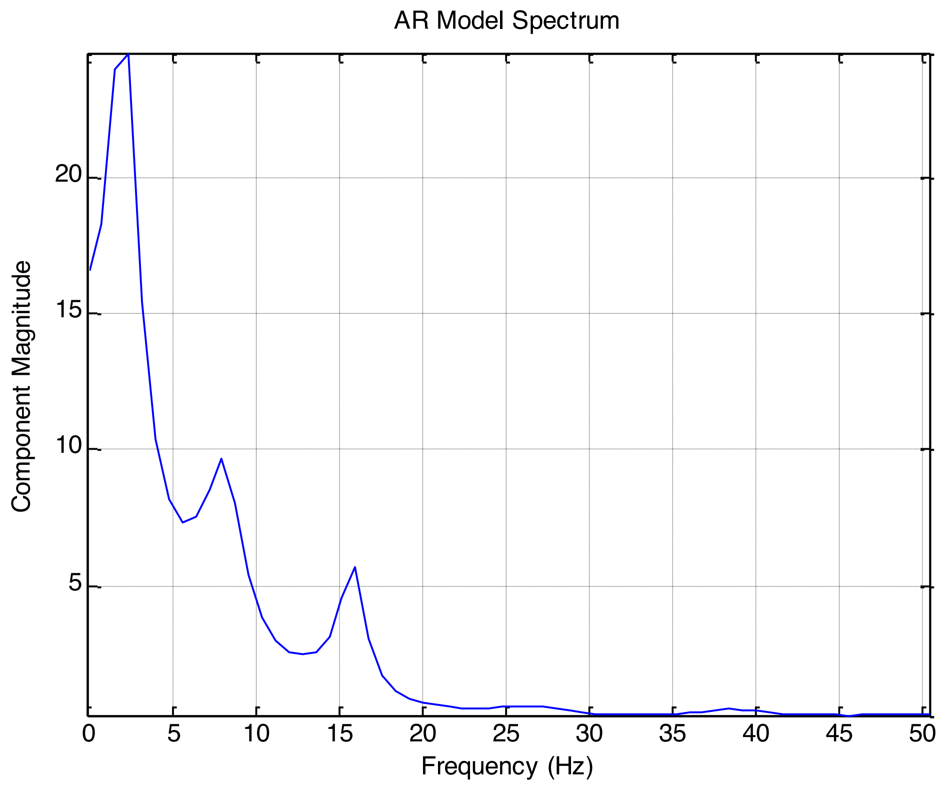
AR(MA) Prediction Models

- AR(MA) models have already been examined in detail as linear predictors in the time domain
- These techniques also provide a complimentary analysis techniques to STFT
- The sampled signal is segmented into frames of N samples (during which the signal is assumed to be stationary)
- AR(MA) models produce a set of coefficients for a (filter) difference equation
  - AR models result in a difference equation in x[n] and y[n-k] terms
  - ARMA models result in addition terms in the difference equation contains terms in x[n-l]
- Like all filters\difference equation, we can examine the frequency response of the model

Frequency Response of an AR Model of an EEG signal



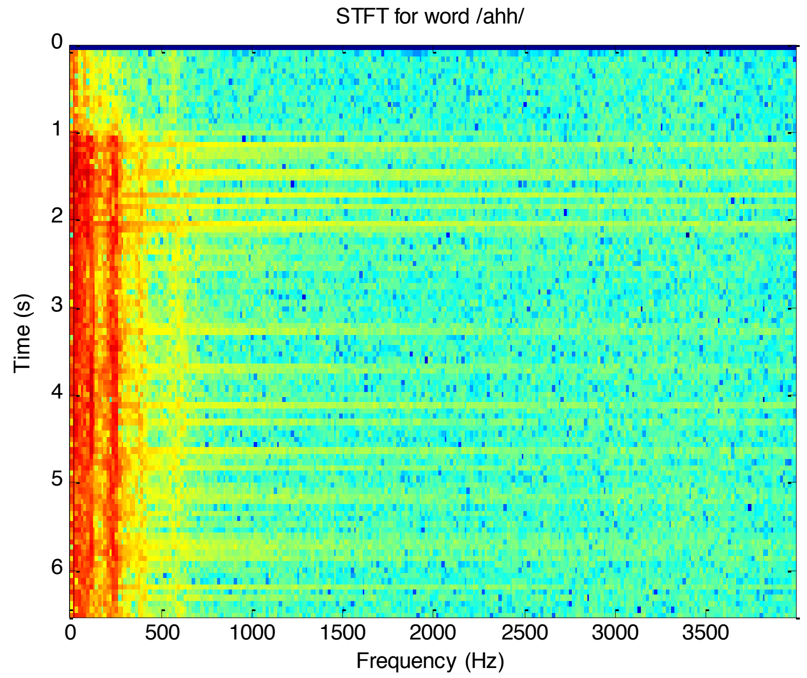
STFT Magnitude

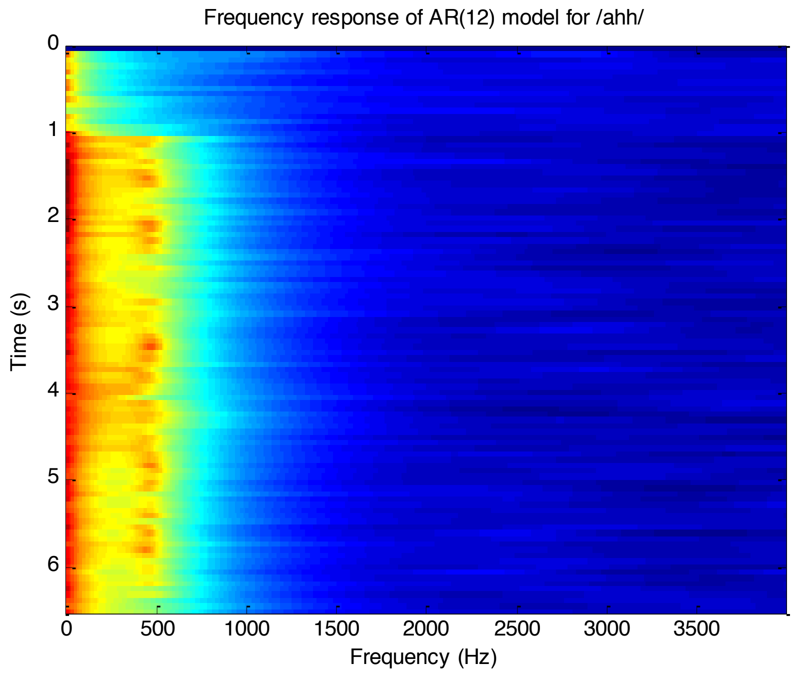


AR(p=12) Model
Magnitude Response

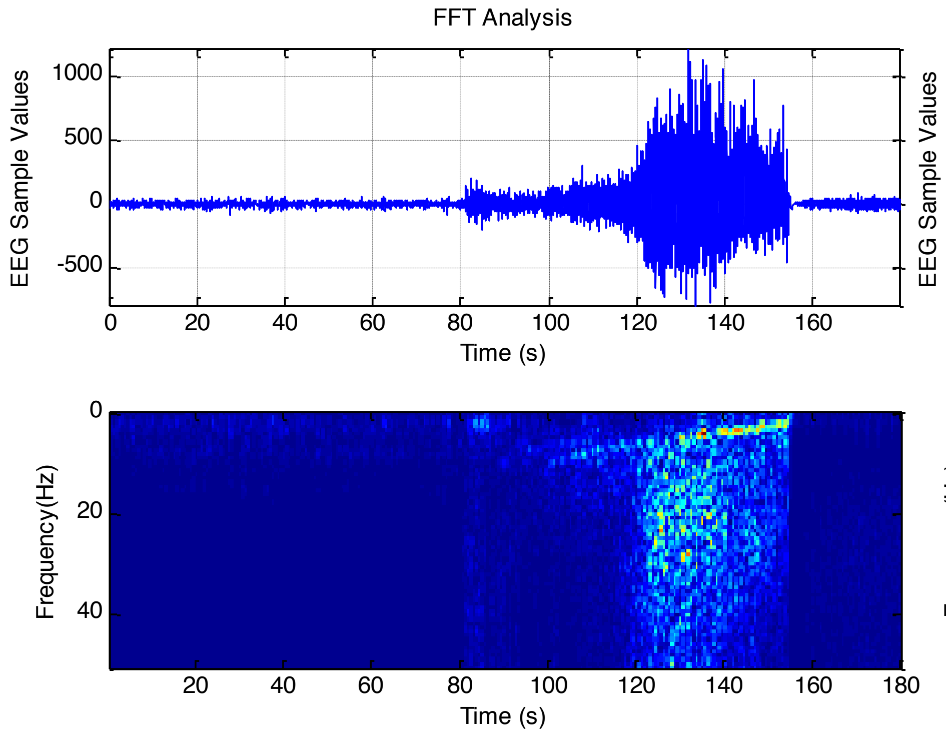
AR(MA) Model Frequency Response

- The frequency magnitude response of an AR(MA) model is effectively a smoothed approximation of the STFT magnitude response
- As such, it highlights the main "resonant" frequencies of the signal rather than the finer detail of the STFT

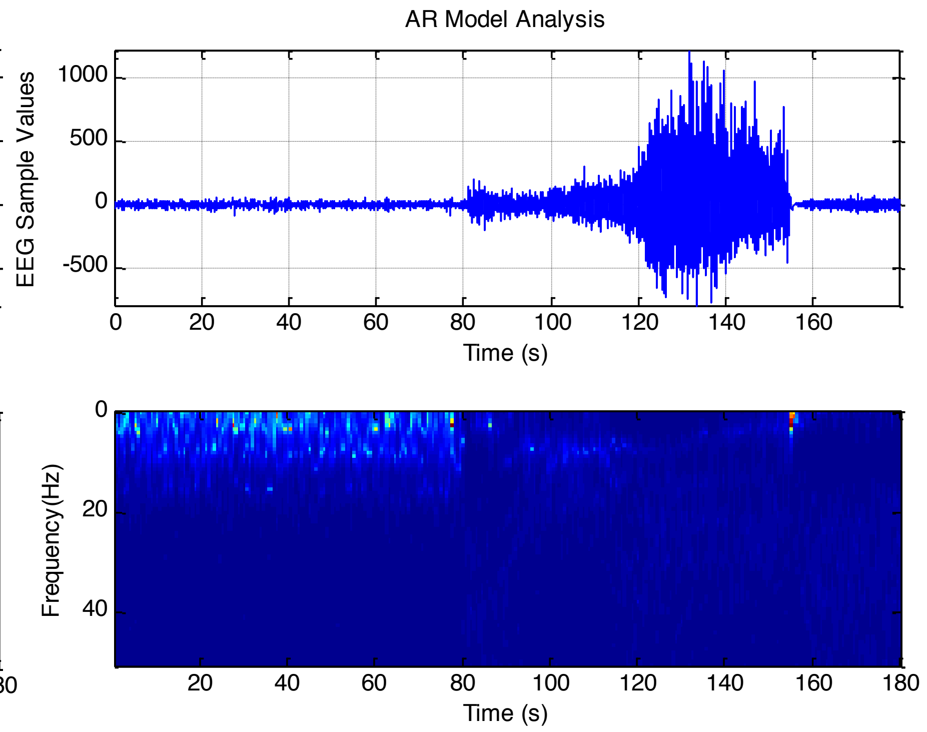




AR(p=12) EEG Model Spectrogram

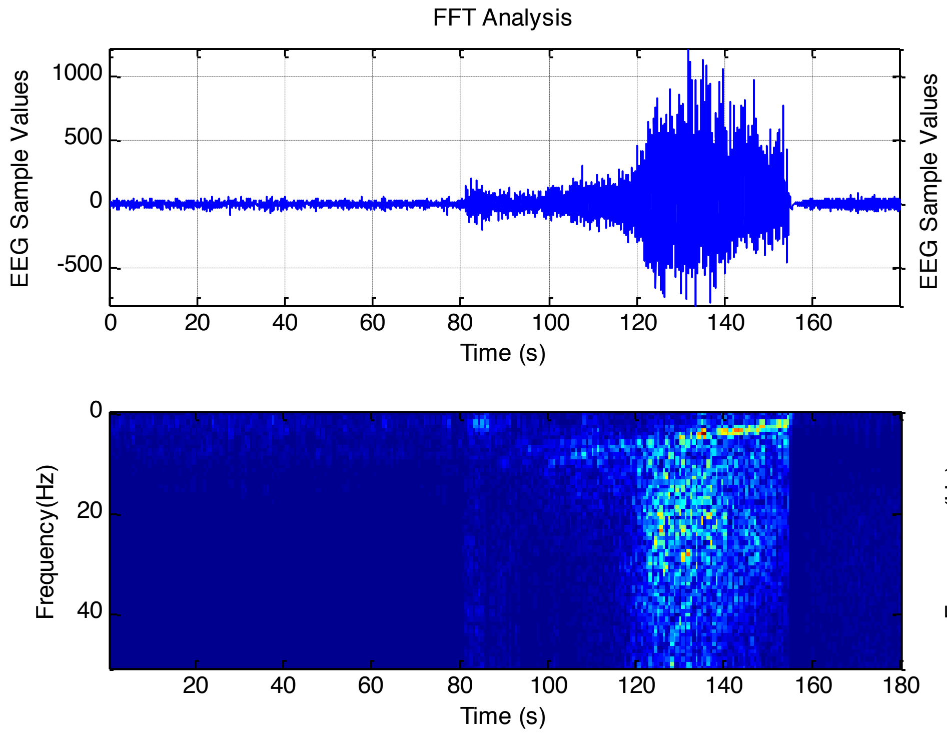


FFT Based Spectrogram

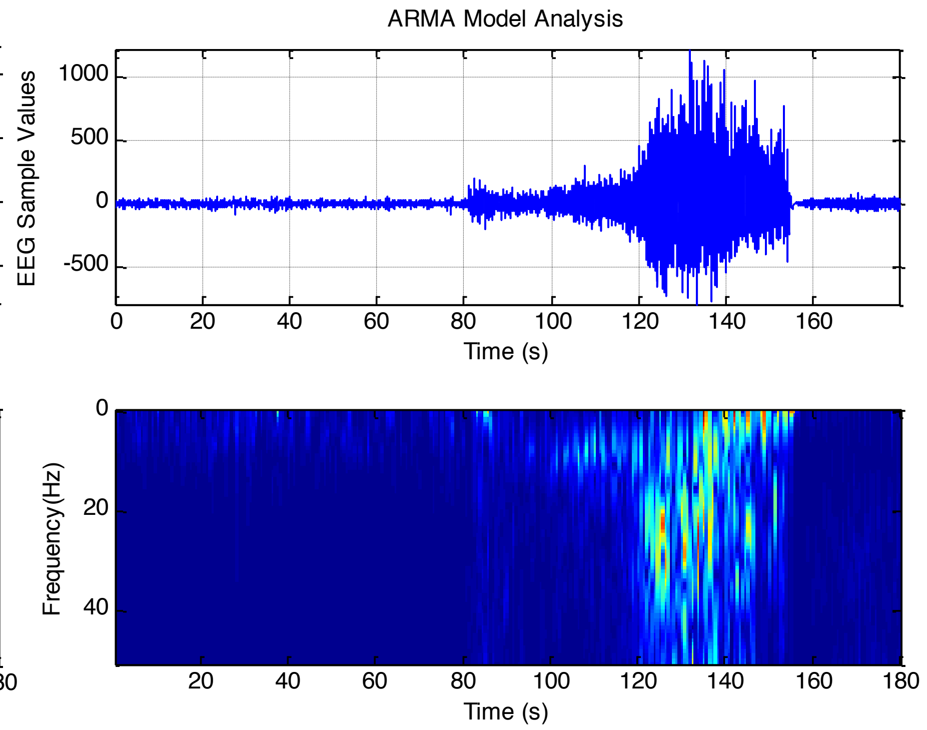


AR(p=12) Model Spectrogram

ARMA EEG Model Performance



FFT Based Spectrogram



ARMA(10,4) Model Spectrogram

SP applications of AR(MA) models

- Like STFT and WT, it offers a valid frequency domain representation of the signal
- Because of the smoother “surface”, it often offers a means of more easily tracking the temporal trajectory of key resonant frequencies or components
- For some applications, the resonant frequencies identified by AR(MA) models can be associated with physical parameters of the system that produced the signal

Synchrony

- Correlation offers a time domain “similarity” measure
- Synchrony measures attempt to quantify the level of synchronisation between certain frequencies (or frequency bands) in two difference signals
- Large number of different measures of linear and non-linear “synchrony”
  - http://www.dauwels.com/Papers/NeuroImage2009.pdf provides an excellent overview of many different measures
  - Commonly used in EEG signal analysis

Cross Coherence

- Cross coherence is one of the simpler measures of synchrony and can be easily calculated from the STFT of the two signals being analysed

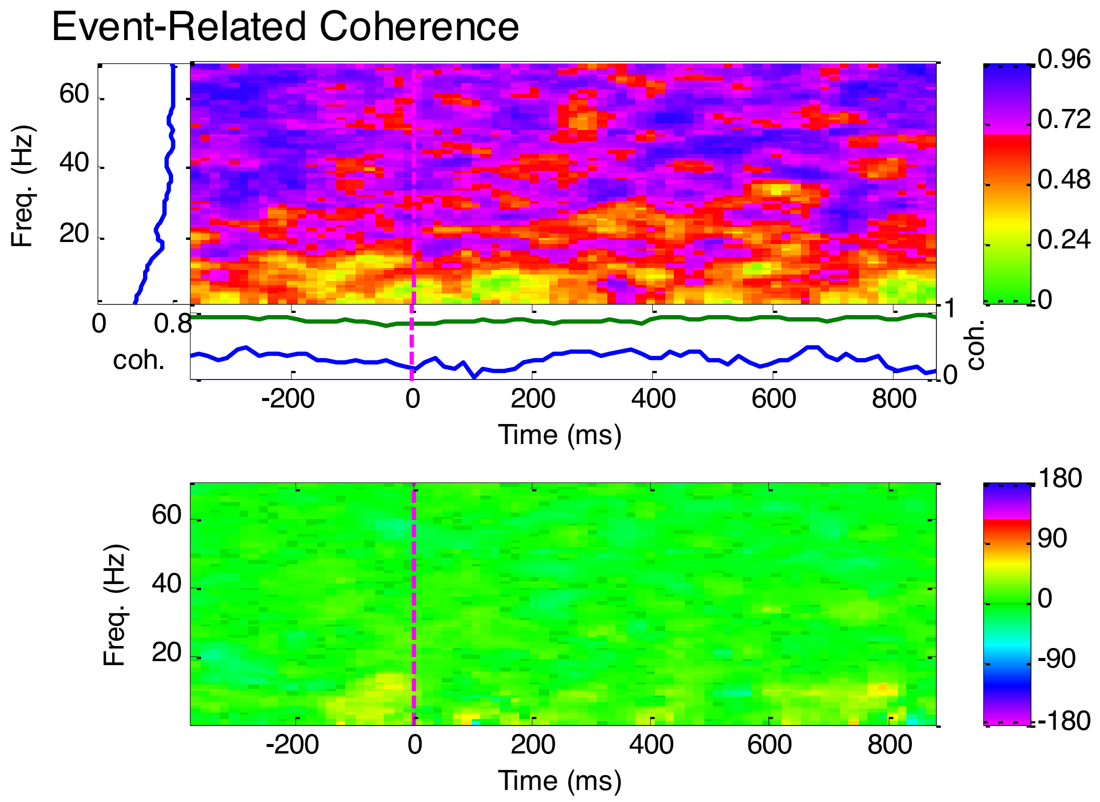
$$Coh_{a,b}(f,t) = \frac{F^a(t,f)F^b(t,f)^*}{|F^a(t,f)F^b(t,f)|}$$

- $F^a(t,f)$ is the STFT of signal “a” at time t and frequency f
- $F^b(t,f)$ is the complex conjugate of the STFT of signal “b” at time t and frequency f

Cross Coherence (2)

- Resultant complex value has a magnitude value between 1 (meaning the two signals are perfectly synchronised at that frequency component) and 0 (meaning there is a complete absence of synchronisation at that frequency component)
- The phase angle of the cross coherence can be easily converted into a corresponding time advance/delay between the two signals at that frequency

Sample EEG Cross Coherence



See Matlab add on toolbox EEGlab: http://sccn.ucsd.edu/eeglab/

Independent Component Analysis (ICA)

- ICA techniques are used in scenarios where it is possible to measure N different signals which are formed by a linear combination of M underlying (but not measurable) signals of interest

$$x = \begin{pmatrix} x_1[n] \\ \\ x_N[n] \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1M} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NM} \end{pmatrix} \begin{pmatrix} s_1[n] \\ \\ S_M[n] \end{pmatrix} = As$$

We can measure $x_i[n]$ but we actually want to know $s_j[n]$ and/or a_{ij}

ICA (2)

- If certain conditions are valid concerning the statistical independence of the source signals\”factors”, then ICA can be used to estimate them
- Large variety of different ICA algorithms depending on particular requirements (under-, over-stated, real time, single channel,…)
- http://cnl.salk.edu/~tewon/Blind/blind_audio.html

Applications of ICA

- Most common application of ICA are in:
  - Wireless communication
  - EEG analysis
  - Separation of biosignals
    - Maternal and foetal ECG separation
  - Investigation of hidden factors in financial data signals
  - Image Noise reduction

ICA Resources

- http://www.ee.columbia.edu/~dpwe/e6820/papers/HyvO00-icatut.pdf (Tutorial paper)
- http://www.bsp.brain.riken.jp/ICALAB/ (Matlab ICA toolbox)

Thank You!