Integral Transforms, the Discrete Fourier Transform and Applications

Gavin Conran

Introduction

This revision paper starts with a description of a set of continuous mathematical constructs, called integral transforms, and their discrete-time cousins, that are used to transform a time domain signal into the frequency domain. Our attention then turns to the details of the Z-Transform and Fourier Transform and continues with a discussion of the algorithms, specifically the Discrete Fourier Transform and the Fast Fourier Transform, used to compute the Fourier Transform. Related applications are based on spectral analysis or filtering so the final sections explore these topics in some detail. The paper finishes with specifics of the world changing applications in Engineering and Science that have been enabled by Integral Transforms.

The Motivation behind the Fourier Series

In mathematics an integral transform maps an equation from its original domain into another domain where it might be manipulated and solved much more easily than in the original domain. The solution is then mapped back to the original domain using the inverse of the integral transform. The precursor of the transforms was the Fourier Series to express functions in finite intervals.

In 1822 Joseph Fourier, a French mathematician, derived the Fourier Series from the Analytic Theory of Heat which was based on Newton's Law of Cooling. The Fourier Series states that any function of a variable, whether continuous or discontinuous, can be expanded in a series of sines of mutiples of the variable. The original motivation behind the Fourier Series was for the Series to be used to solve the Partial Differential Equation for the Conductive Diffusion of Heat:

$$\frac{\partial u}{\partial t} - \alpha \cdot \nabla^2 u = 0 \qquad [1]$$

Later the Fourier Transform was developed to remove the requirement of finite intervals enabling the decompoition of non-periodic functions.

Mathematics & Algorithms of the Fourier Transform

This section describes the Discrete Fourier Series, followed by the Laplace and Fourier Tranforms, both continuous time integral transforms. The section also introduces the corresponding discrete time transforms, namely, the Z-Transform and the Discrete-Time Fourier Transform. This is followed by a description of the two main algorithms associated with computing the Fourier Transform, the Discrete Fourier Transform (DFT) and the Fast Fourier Transform (FFT). After discussing analogue and digital filter design, the section concludes with a summary table mapping the relationship between continuous & discrete time transforms and the associated algorithms.

Discrete Fourier Series

Fourier introduced the concept of representing a given function by a trigonometric series of sines and cosines, i.e. the Fourier Series decomposes any Periodic Function or Periodic Signal into the sum of a (possibly infinite) set of simple oscillating functions, namely sines and cosines or, equivalently, complex exponentials, as follows:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cdot \cos(nx) + b_n \cdot \sin(nx) \right) \quad [2]$$

where the Fourier Coefficients are ortogonal and therefore act as the basis functions of the Fourier Series of a function. The coefficients are given as:

$$a_n = \frac{1}{\pi} \cdot \int_{-\pi}^{\pi} f(x) \cdot \cos(nx) dx \quad [3]$$

$$b_n = \frac{1}{\pi} \cdot \int_{-\pi}^{\pi} f(x) \cdot \sin(nx) dx \quad [4]$$

As an example, take the Square Wave, shown on the left hand side of Illustration 1. As periodic functions can generally be represented by [2] above, the Square Wave can thought of as the sum of a set of simple oscillating functions, known as harmonics', each of which being scaled by a Fourier Coefficient, as given by [3] and [4].

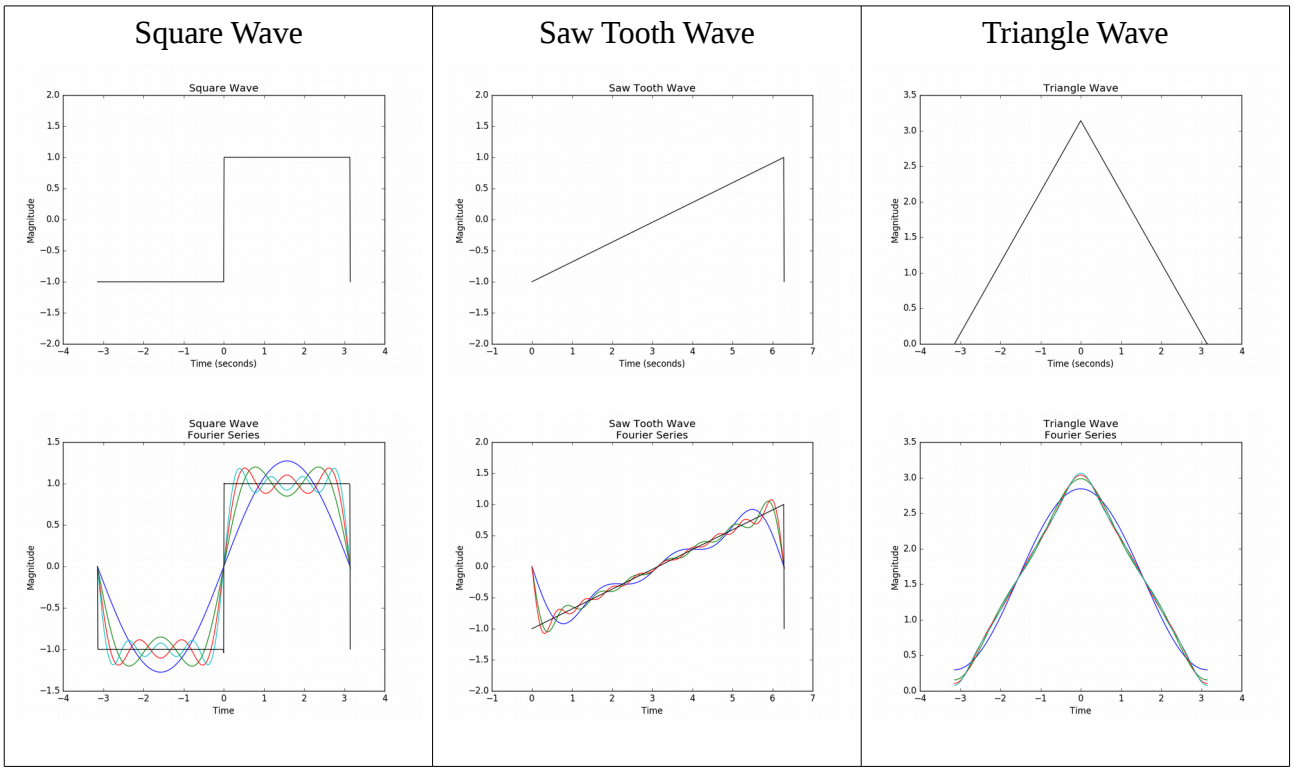


Illustration 1: Waveforms and their corresponding Fourier Series

The idea of the Fourier Series of a function is that it is the closest possible sum of Sines and Cosines that approximate the function on the interval. We use the Root-Mean-Squared distance to minimise the difference between the actual and approximated functions.

Integral Transforms

Generalising the Fourier Series: the Fourier Transform

The Fourier Series states that a function, f, that repeats itself every 2π, can be expressed as a sum of sine and cosine waves:

$$f(x)=\sum_{n=0}^{\infty}\left(A_n\cdot\sin(nx)+B_n\cdot\cos(nx)\right) \quad [5]$$

In the real world, not all functions are periodic and the Fourier Transform is a generalised form of the Fourier Series that enables the decomposition of non-periodic functions of time, i.e. a signal, into the frequencies that make it up. To go from the Fourier Series to the Fourier Transform we go through a number of steps:

Step 1: Taking advantage of Euler's Equation:

$$e^{i\theta}=\cos(\theta)+i\cdot\sin(\theta) \quad [6]$$

it is possible to express the Fourier Series as a single equation:

$$f(x)=\sum_{n=0}^{\infty}C_n\cdot e^{inx} \quad [7]$$

Where

$$C_n=\frac{1}{\pi}\cdot\int_0^{2\cdot\pi}f(x)\cdot e^{inx}\,dx \quad [8]$$

Step 2: Expanding the size of the interval from [0, 2π] to (-∞, ∞) turns the sum along [0, 2π] into an integral along (-∞, ∞) giving:

Synthesis **Analysis**

$$f(x)=\int\hat{f}(n)\cdot e^{i2\pi nx}dx \quad \text{and} \quad \hat{f}(n)=\int f(x)\cdot e^{-i2\pi nx}dx \quad [9]\ \&\ [10]$$

where $\hat{f}$ is the Fourier Transform Operator. Instead of C_n we have $\hat{f}$ and instead of a summation we have an integral.

If x represents time and n, frequency, the Fourier Transform of a function of time is a complex-valued function of frequency whose:

- Absolute value represents the amounts of frequency present in the signal function
- Complex argument is the phase offset of the basic sinusoid in that frequency

The Fourier Transform (FT) refers to both the frequency domain representation and the mathematical operation that associates the Frequency Domain representation to the function of time, also known as Analysis.

The Inverse FT, also known as Synthesis, combines the contributions of all the different frequencies to recover the original function of time.

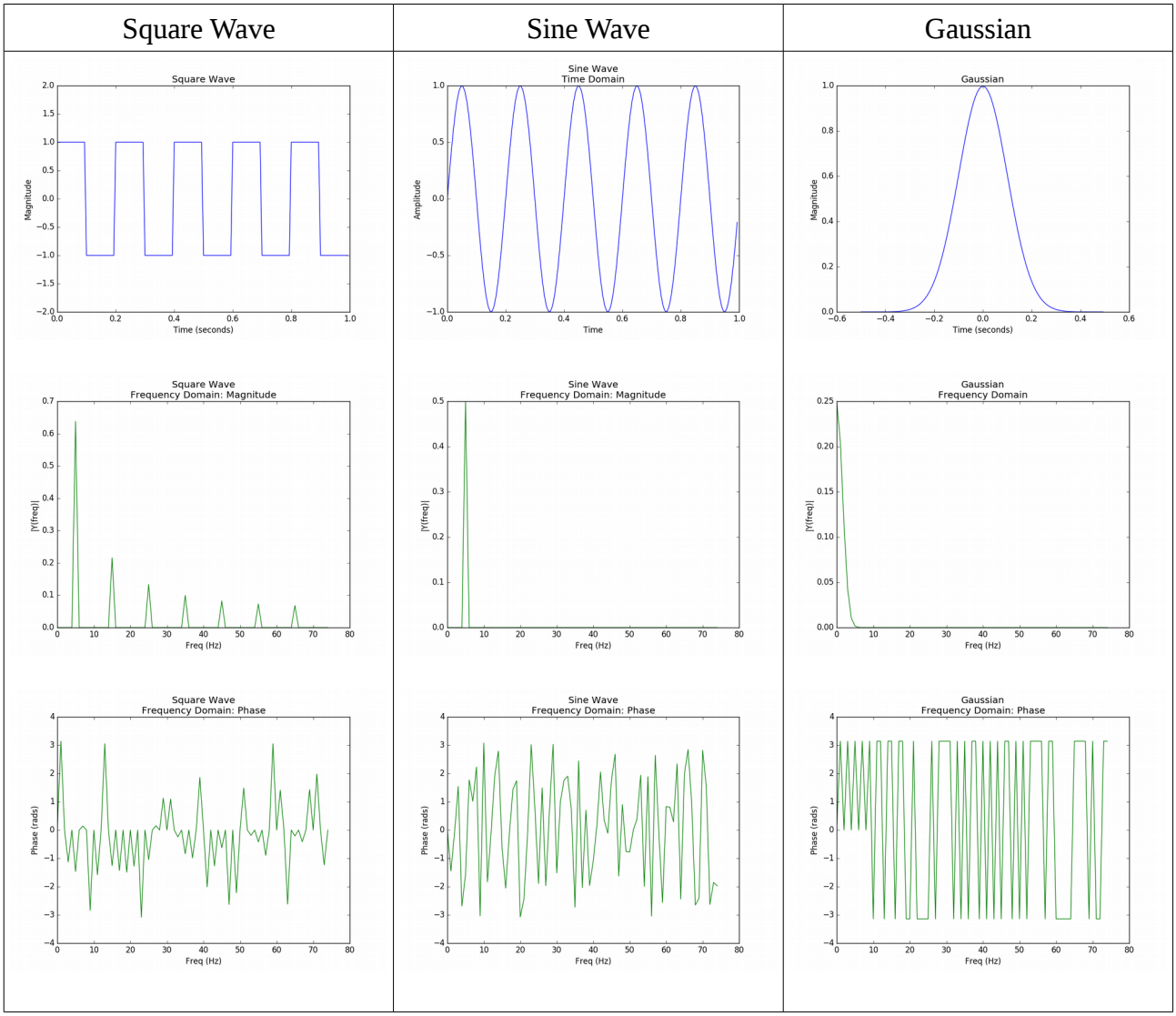


Illustration 2: Waveforms and their corresponding Fourier Transforms (Magnitude & Phase Responses)

Convolution

Convolution in the time domain corresponds to ordinary multiplication in the frequency domain. This means that any Linear Time-Invariant system, such as a filter applied to a signal, can be expressed simply as an operation on frequencies. After performing the desired operation, transformation of the result can be made back to the time domain. The Fourier Transform translates between convolution and multiplication of functions. For example, if f(x) and g(x) are integrable functions with Fourier Transforms $\hat{f}(k)$ and $\hat{g}(k)$, then the Fourier Transform of the convolution is given by the product of the Fourier Transforms $\hat{f}(k)$ and $\hat{g}(k)$.

In other words if

$$h(x)=(f*g)(x)=\int_{-\infty}^{\infty} f(y)\,g(x-y)\,dy \quad \text{Convolution} \qquad [11]$$

Then

$$\hat{h}(k)=\hat{f}(k)\cdot\hat{g}(k) \quad \text{Multiplication} \qquad [12]$$

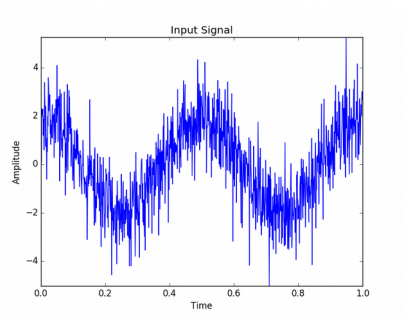
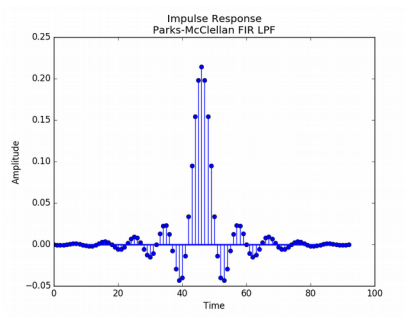
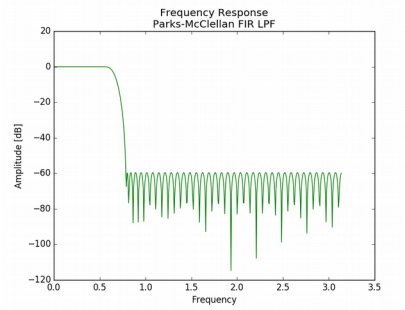
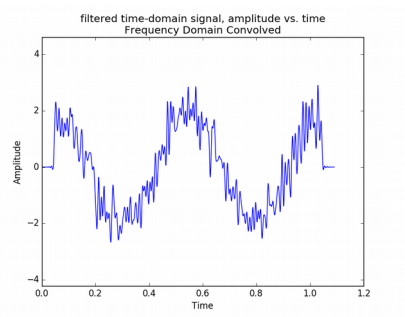
Input Signal	Filter Response	Output Signal
	 	

Illustration 3: Time Domain and Frequency Domain Convolution

Frequency Domain Covolution, based of the Fast Fourier Transform (FFT) algorithm has led to advances in many applications, such as spectral analysis, audio & image processing, digital communication and data compression. Frequency domain convolution is fundamental to many engineering and scientific applications.

Derivative Relationship

Another key property of the Fourier Transform lies squarely around the derivative relationship:

$$\hat{f^{(n)}}(x) = (ik)^n \cdot \hat{f}(x) \quad \text{where } k \text{ is } x \text{ rescaled to a } 2\pi \text{ domain} \qquad [13]$$

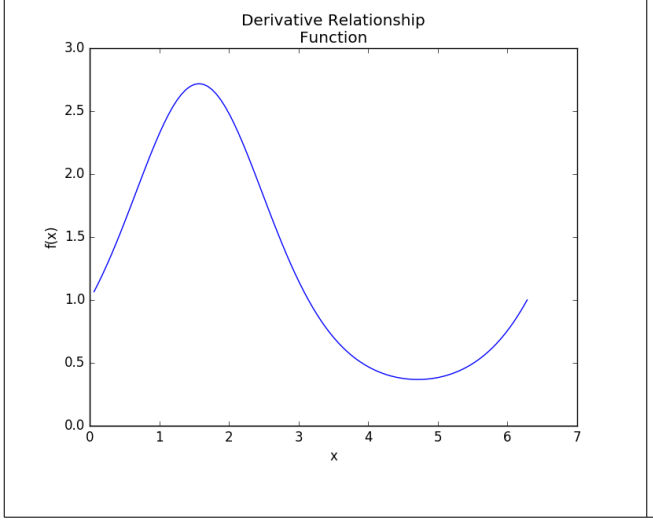
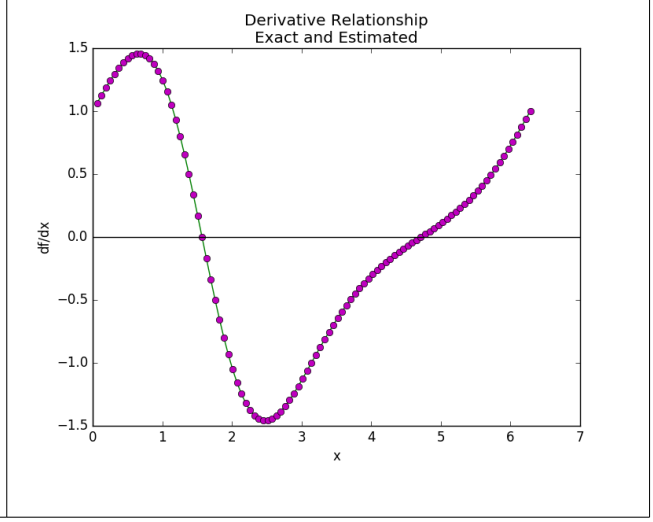
Function $f(x) = e^{\sin(x)}$	Derivative $f'(x) = \cos(x) \cdot e^{\sin(x)}$
	

Illustration 4: Fourier Transform Derivative Relationship

This relationship is what enables the Fourier Transform to be used to find solutions to periodic (Ordinary and Partial) Differential Equations.

Laplace Transform

The Laplace Transform is a more general representation of the Fourier transform. The Laplace Transform is a complex argument, $s=\sigma+i\omega$, of a complex function, e^{-st}, while the Fourier Transform is a real argument, $k=\omega$, of a complex function, e^{-ikx}.

Discrete-Time Fourier Transform (DTFT)

The Discrete-Time Fourier Transform (DTFT) is a form of Fourier analysis that is applicable to the uniformly-spaces samples of a continuous function. The term discrete-time refers to the fact that the transform operates on discrete data, i.e. samples, whose interval often has units of time. From only the samples, it produces a function of frequency that is a periodic summation of the continuous Fourier Transform of the original continuous function.

Given a discrete function, x[n], and a real argument, ω, the DTFT is:

$$X(e^{j\omega})=\sum_{-\infty}^{\infty} x[n]\cdot e^{-j\omega n} \qquad [14]$$

The DTFT is a continuous function of frequency, $X(e^{j\omega})$, but discrete samples of it can be readily calculated via the Discrete Fourier Transform (DFT) which is by far the most common method of modern Fourier analysis.

The Discrete Fourier Transform (DFT)

We have a sequence of length N:

x[n]: Finite length signal of length N

We form a periodic signal of period N:

$$\tilde{x}(n)=x[(n\,modulo\,N)]$$

We compute the Fourier series coefficients:

$$\tilde{X}(k)=x[(k\,modulo\,N)]$$

One period of $\tilde{X}(k)$ *is defined as the Discrete Fourier Transform:*

$$X[k]: DFT\ of\ signal\ x[n]$$

N.B.: There is a big difference between the frequency content of $\tilde{X}(k)$ and $X(k)$.

The DFT is also frequency samples of the DTFT of x[n]
(spacing of samples of the DTFT needs to be sufficiently small to avoid time aliasing)

Table 1: Relationship between the DFS, DTFT & DFT

In reference to Table 1 and Illustration 5, according to Al Oppenheim, mathematically, "the DFT is one period of the discrete Fourier Series Coefficients of the finite length signal, x[n], periodically replicated, i.e $\tilde{x}(n)$". The Fourier series coefficients of $\tilde{x}(n)$ are also defined as the frequency samples of the Discrete-Time Fourier Transform of one period of $\tilde{x}(n)$.

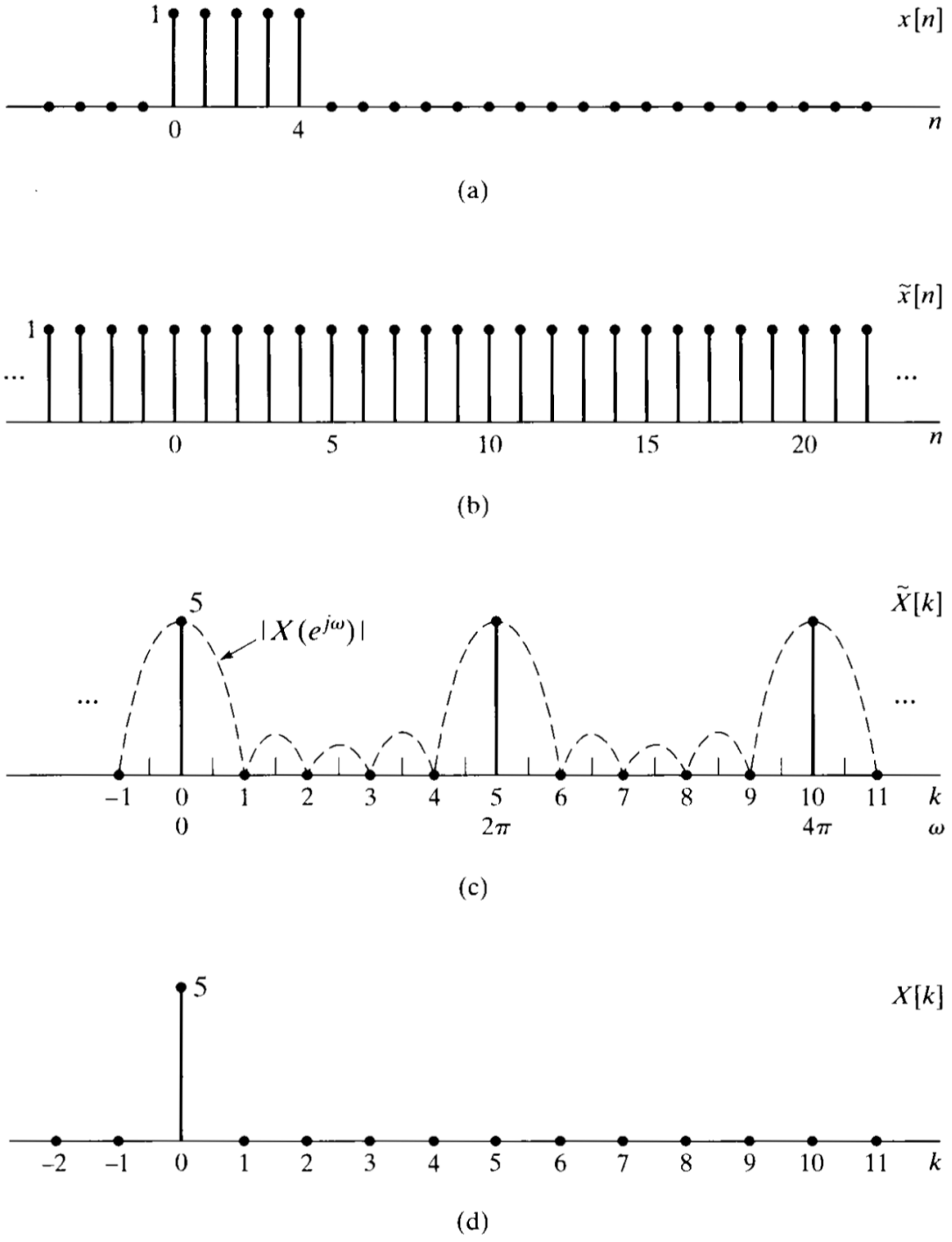


Figure 8.10 Illustration of the DFT. (a) Finite-length sequence $x[n]$. (b) Periodic sequence $\tilde{x}[n]$ formed from $x[n]$ with period $N = 5$. (c) Fourier series coefficients $\tilde{X}[k]$ for $\tilde{x}[n]$. To emphasize that the Fourier series coefficients are samples of the Fourier transform, $|X(e^{j\omega})|$ is also shown. (d) DFT of $x[n]$.

Illustration 5: Illustration of the DFT

Properties of the DFT

Since the DFT is “simply” the DFS of a finite length sequence periodically replicated, it inherits all the properties of the DFS. Consequently multiplication of the DFT by a linear phase term results in a periodic or circular time shift rather than a linear time shift. Similarly, the product of two DFTs is the DFT of the convolution of the two finite length sequences, but the convolution is a ***circular convolution which is equivalent to linear convolution followed by aliasing***. This means that the convolution of two DFTs requires considerable computational ‘book-keeping’. Finally, it is worth noting that the multiplication of the DFT of a sequence by samples of an ideal filter does not result in ideal filtering due to the fact that a filter’s Frequency Response cannot be interpreted as zero in between the frequency samples.

How do we compute the DFT?

The DFT is identical to samples of the Fourier transform at equally spaced frequencies. Consequently, computation of the N-point DFT coresponds to the computation of N samples of the Fourier transform at N equally spaced frequencies, $\omega_k = \frac{2\pi k}{N}$, i.e., at N points on the unit circle in the z-plane. This means that in the most general case, the DFT coefficients, X[k], are complex numbers and they have real and imaginary parts. In order to interpret these coefficients, their magnitude |X[k]| and phase , angle X[k], are wrapped between -π and π and can be plotted on the z-plane.

Therefore, computationally, The DFT is nothing more than a matrix-vector multiplication of $\vec{x}$:

$$\vec{X} = M \cdot \vec{x} \quad \textit{where the matrix,} \quad M_{kn} = e^{-i2\pi\frac{kn}{N}} \qquad [15]$$

Forward DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi\frac{kn}{N}} \qquad [16]$$

Inverse DFT

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{i2\pi\frac{kn}{N}} \qquad [17]$$

Note that when using the Discrete Fourier Transform (or the Fast Fourier Transform), to compute the Fourier Transform the input, x, is a finite domain, i.e. $x \in [-L, L]$ rather than an infinite domain, $x \in [-\infty, \infty]$, as with the continuous and discrete mathematical integral transforms.

The critical draw back of the DFT is that with an algorithmic complexity of $O[N^2]$ its performance is poor for large values of N. In the next section will discuss the Fast Fourier Transform which greatly improves the performance of computing the Fourier Transform with a complexity of $O[N \cdot \log(N)]$.

Goertzel algorithm

If it is only of interest to compute a small number of frequency samples, direct computation via the Goertzel algorithm, $O[N^2]$, is often more efficient than using a highly efficient DFT computation and then only retaining the samples of interest. An example application is the recognition of DTMF tones produced by the buttons pressed on a telephone key pad.

The Fast Fourier Transform (FFT)

In ‘radix-2’ FFT algorithms the FFT re-expresses the DFT of an arbitrary size, $N = N_1 \cdot N_2$, in terms of N_1 smaller DFTs of sizes N_2 , recursively, to reduce the computation time to $O[N \cdot \log(N)]$. It can be shown analytically that if one piece of a problem is simply related to another then it is possible to compute the sub-result only once and save that computational cost. Cooley and Tukey used exactly this approach in deriving the FFT. They showed that it is possible to divide the DFT computation into smaller parts. As explained earlier, the Forward DFT is:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi \frac{kn}{N}} \quad [18]$$

which can be divided into two smaller pieces:

$$\sum_{m=0}^{\frac{N}{2}-1} x_{2m} \cdot e^{-i2\pi k \frac{2m}{N}} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} \cdot e^{-i2\pi k \frac{2m+1}{N}} \quad [19]$$

which in turn is equal to:

$$\sum_{m=0}^{\frac{N}{2}-1} x_{2m} \cdot e^{-i2\pi k \frac{2m}{\frac{N}{2}}} + e^{-i2\pi \frac{k}{N}} \cdot \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} \cdot e^{-i2\pi k \frac{2m+1}{N}} \quad [20]$$

Even Odd

We can see that the range of k is ***0*** <= ***k*** <= ***N*** while the range of n is ***0*** <= ***n***<= ***N/2***. This means that from this symmetry property we only have to perform HALF of the computations for each sub-problem. In addition, as long as the smaller Fourier Transforms have an even-valed m, we can apply the divide-and-conquer approach, halving the computational cost each time, until our arrays are small enough that the strategy is no longer beneficial.

In the asymptotic limit, this recursive approach scales as $O[N \cdot \log(N)]$. The DFT and FFT Python implementations can be seen in Table 2 below.

DFT	FFT

import numpy as np
def DFT(x):
  x = np.array(x, dtype=float)
  N = x.shape[0]
  n = np.arange(N)
  k = n.reshape((N, 1))
  M = np.exp(-2j * np.pi * k * n/N)
  return np.dot(M,x)
	import numpy as np
	def FFT(x):
	  x = np.array(x, dtype=float)
	  N = x.shape[0]
	  if N%2 > 0:
	  print("Error")
	  elif N <= 32:
	  return DFT(x)
	  else:
	  X_even = FFT(x[::2])
	  X_odd = FFT(x[1::2])
	  factor = np.exp(-2j * np.pi * np.arange(N) / N)
	  return np.concatenate([X_even + factor[:N/2] * X_odd, \
	  X_even + factor[N/2:] * X_odd])

Table 2: DFT and FFT functions written in Python

The FFT and inverse FFT are simple function calls within the Python ‘numpy’ module:

X = numpy.fft.fft(x) x = numpy.fft.ifft(X)

For example:

```
If x is ‘array([0, 1, 2])’
Then X is ‘array([ 3. +0.j       , -1.5+0.8660254j, -1.5-0.8660254j])’
```

FFT: input and ouput

Once we have the complex variables from the FFT we can obtain the Magnitude and Phase response with the following code:

```
import numpy
import matplotlib.pyplot as plt

# TIME DOMAIN
Fs = 150.0;                          # sampling rate
Ts = 1.0/Fs;                         # sampling interval
t = numpy.arange(0,1,Ts)             # time vector
ff = 5;                              # frequency of the signal
x = numpy.sin(2*numpy.pi*ff*t)       # signal
plt.plot(t, x); plt.show()           # plot time against signal

# FREQUENCY DOMAIN
n = len(x)                           # length of the signal
k = numpy.arange(n)                  # frequency vector
T = n/Fs
freq = k/T                           # two sides frequency range
freq = freq[range(n/2)]              # one side frequency range
time = t[range(n/2)]                 # one side time range

# FAST FOURIER TRANSFORM
X = numpy.fft.fft(x)/n               # fft computing and normalization
X = X[range(n/2)]                    # one side fft range

X_abs  = abs(X)                      # compute magnitude
plt.plot(freq, X_abs); plt.show()    # plot Frequency Response: Magnitude
X_angle =  numpy.angle(X)            # compute phase
plt.plot(freq, X_angle); plt.show()  # plot Frequency Response: Phase
```

Extract Frequency Response variables from FFT

This code was used to generate the Sine Wave Fourier Transform plots shown in Illustration 2.

Z-Transform, CCDEs & Transfer Functions

In mathematics and signal processing, the Z-Transform converts a ***discrete-time*** signal, which is a sequence of real or complex numbers, into a complex frequency domain representation. It can be considered as a discrete-time equivalent of the Laplace Transform.

Given a discrete function, x[n], and a complex argument, $z = A \cdot e^{j\phi}$, the Z-Transform of x[n] is:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] \cdot z^{-n} \qquad [21]$$

It is worth noting that, if we replace the complex variable, z, with $e^{j\omega}$, then the Z-Transform reduces to the Fourier Transform. The Fourier Transform is simply X(z) when $z = e^{j\omega}$ restricting z to having unity magnitude, i.e. $|z| = 1$.

A Constant Coefficient Difference Equation (CCDE) describes a causal Linear Time-Invariant (LTI) system by a linear combination of past input and output values scaled by constants. CCDEs are difficult to manipulate in the raw because ideal filters cannot be implemented but the Z-Transform greatly reduces the task by creating a new characterisation of a filter in terms of a Rational Transfer Function (ratio of polynomials in variable z).

Consider the generic filter CCDE:

$$y[n] = \sum_{k=0}^{M-1} b_k x[n-k] - \sum_{k=1}^{N-1} a_k y[n-k] \qquad [22]$$

We can get the Rational Transfer Function of the Filter by the following steps:

Step 1: apply the Z-Transform to both sides:

$$Y(z) = \sum_{k=0}^{M-1} b_{kz}^{-k} X(z) - \sum_{k=1}^{N-1} a_k z^{-k} Y(z) \qquad [23]$$

Step 2: Rearrange terms:

$$Y(z) = \frac{\sum_{k=0}^{M-1} b_k z^{-k}}{1 + \sum_{k=1}^{N-1} a_k z^{-k}} \cdot X(k) \qquad [24]$$

'Zeros' and 'Poles' are, respectively, the complex roots of the Numerator and Denominator.

Step 3: The above equation is then given by:

$$Y(z) = H(z) \cdot X(z) \qquad [25]$$

where H(z) is the Transfer Function of the LTI filter described by the CCDE.

By bridging the algorithmic side, i.e. Constant Coefficient Difference Equations, to the analytic side, i.e. spectral properties, the Z-Transform is used to create the Rational Transfer Functions of realisable filters.

	Continuous Mathematics	**Discrete Mathematics**
Laplace: complex argument of a complex function	***Laplace Transform*** Given a continuous function, x(t), and a complex argument, $s=\sigma+i\,\omega$, the Laplace Transform of x(t) is: $X(s)=\int_0^{\infty} x(t)e^{-st}dt$	***Z Transform*** Given a discrete function, x[n], and a complex argument, $z=A\cdot e^{j\phi}$, the Z-Transform of x[n] is: $X(z)=\sum_{n=-\infty}^{\infty} x[n]\cdot z^{-n}$
Fourier: real argument of a complex function	***Fourier Transform*** Given a continuous function, f(x), and a real argument, $k=\omega$, the FT is: $\hat{f}(k)=\int_{-\infty}^{\infty} e^{-ikx}f(x)dx$ and the Inverse FT: $f(x)=\frac{1}{2\pi}\int_{-\infty}^{\infty} e^{ikx}\hat{f}(k)dk$	***Discrete-Time Fourier Transform*** Given a discrete function, x[n], and a real argument, ω , the DTFT is: $X(e^{j\omega})=\sum_{-\infty}^{\infty} x[n]\cdot e^{-j\omega n}$ If we replace the complex variable, z, with $e^{j\omega}$, then the Z-Transform reduces to the DTFT. The DTFT is simply X(z) when $z=e^{j\omega}$ restricting z to having unity magnitude, i.e. $\lvert z\rvert=1$
Algorithm $O[N^2]$	***Discrete Fourier Series (DFS)*** $f(x)=\frac{a_0}{2}+\sum_{n=1}^{\infty}(a_n\cdot\cos(nx)+b_n\cdot\sin(nx))$ *where the Fourier Coefficients are given as:* $a_n=\frac{1}{\pi}\cdot\int_{-\pi}^{\pi} f(x)\cdot\cos(nx)dx$ *and* $b_n=\frac{1}{\pi}\cdot\int_{-\pi}^{\pi} f(x)\cdot\sin(nx)dx$	
Algorithm $O[N^2]$	***Discrete Fourier Transform (DFT)*** *Mathematically*, the DFT is one period of the DFS Coefficients of the finite length signal, x[n], periodically replicated, i.e $\tilde{x}(n)$. The DFT is also frequency samples of the DTFT of x[n]. The DFT inherits the DFS properties. *Computationally*, The DFT is a matrix-vector multiplication of $\vec{x}$: $\overline{X}=M\cdot\vec{x}$ *where the matrix,* $M_{kn}=e^{-i2\pi\frac{kn}{N}}$ <u>*Forward DFT*</u> : $X[k]=\sum_{n=0}^{N-1} x[n]e^{-i2\pi\frac{kn}{N}}$ <u>*Inverse DFT:*</u> $x[n]=\frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{i2\pi\frac{kn}{N}}$	
Algorithm $O[N\cdot\log(N)]$	***Fast Fourier Transform (FFT)*** The FFT re-expresses the DFT of an arbitrary size, $N=N_1\cdot N_2$, in terms of N_1 smaller DFTs of sizes N_2 , recursively, to reduce the computation time to $O[N\cdot\log(N)]$. Applying the DFT or the FFT to the same input will return the same result. The difference between the two algorithms is the computational cost; the FFT is much faster than the DFT for large values of N.	

Table 3: Summary of Integral Transforms and Algorithms

Table 3 above is a summary of the integral transforms, their discrete equivalents and the algorithms used to compute the Fourier Tranform for a discrete and non infinite signal.

Filters: Analogue RLC Circuits

This section consists of a discussion on the time and frequency response of analogue RLC circuits, the use of Operational Amplifiers in analogue filters, and digital filter design.

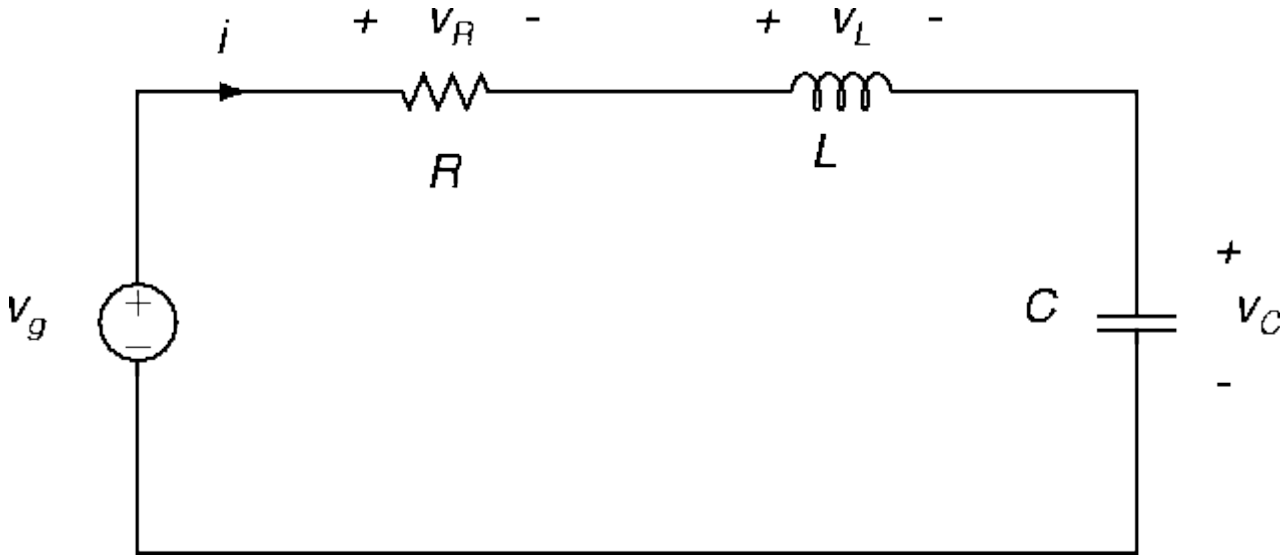


Illustration 6: RLC Series Circuit

Signal Notation

V_A : Upper case symbols, upper case subscripts – D.C. or operating point variables

v_A : Total instantaneous variables

v_a : Incremental or small signal instantaneous variables

V_a : Complex amplitudes (total variables or incremental variables). Recall $v_I = V_i \cdot \cos(\omega t)$, these variables also represent the real amplitudes of sinusoidal sources.

Time Domain Analysis

A discrete-time Difference Equation is the discrete equivalent of a continuous-time Differential Equation . A Second Order Continuous Differential Equation is used to mathematically describe the Time Domain dynamics of an analogue RLC Series Circuit, as shown above in Illustration 6.

With a ***continuous-time*** D.C. input the dynamics of an RLC circuit can be described by a Second Order Differential Equation.

Set up Differential Equation

Step 1: Node Method

Recall Element Rules:

@v: $$\frac{v_A - v}{R} = \frac{C \cdot dv}{dt} \quad [25]$$

$$L: v_L = \frac{L \cdot di_L}{dt} \text{ giving } i_L \int_{-v_o}^{t} dt = i_L \quad [26]$$

@v_A: $$\frac{1}{L} \cdot \int_{-v_o}^{t} (v_I - v_A)\, dt = \frac{v_A - v}{R} \quad [27]$$

$$C: i_C = C \cdot \frac{dv_C}{dt} \text{ giving } \frac{1}{C} \cdot \int_{-v_o}^{t} i_c\, dt = v_C \quad [28]$$

Step 2: Need to get rid of v_A:

From [25]

$$\frac{v_A - v}{R} = \frac{C \cdot dv}{dt} \quad [29]$$

Therefore, from [27]

$$\frac{1}{L} \cdot \int_{-v_o}^{t} (v_I - v_A)\, dt = C \cdot \frac{dv}{dt} \quad [30]$$

After integrating both sides:

$$\frac{1}{L} \cdot (v_I - v_A) = C \cdot \frac{d^2 v}{dt^2} \quad [31]$$

And re-arranging

$$\frac{1}{L \cdot C} \cdot (v_I - v_A) = C \cdot \frac{d^2 v}{dt^2} \quad [32]$$

From [29]

$$v_A = R \cdot C \cdot \frac{dv}{dt} + v \quad [33]$$

Therfore, from [32]

$$\frac{1}{L \cdot C} \cdot \left(v_I - R \cdot C \cdot \frac{dv}{dt} - v\right) = \frac{d^2 v}{dt^2} \quad [34]$$

Giving a Second Order Differential Equation:

$$\frac{d^2 v}{dt^2} + \frac{R}{L} \cdot \frac{dv}{dt} + \frac{v}{LC} = \frac{1}{LC} \cdot v_I \quad [35]$$

Solve Differential Equation

We can now solve the 2nd Order Differential Equation by using the method of homogeneous and particuar solutions:

Step 1: Find the particular solution, v_p, using 'guess & check':

$$v_p = v_I \quad [36]$$

Step 2: Find the homogeneous solution v_h in the following 4 steps:

Step 2a: Assume a solution of the form:

$$v_h = A \cdot \exp^{st} \quad [37]$$

Step 2b: Form the Characteristic Equation, f(s):

$$s^2 + \frac{R}{L} \cdot s + \frac{1}{LC} = 0 \quad \text{where} \quad 2 \cdot \alpha = \frac{R}{L} \quad \text{and} \quad \omega_o = \frac{1}{\sqrt{(LC)}} \quad [38]$$

Step 2c: Find the Roots of the Characteristic Equations s1, s2:

$$s_1 = -\alpha + \sqrt{(\alpha^2 - \omega_o^2)} \quad \text{and} \quad s_2 = -\alpha - \sqrt{(\alpha^2 - \omega_o^2)} \quad [39]$$

Step 2d: General Solution to the Homogeneous Equation is of the form:

$$v_h = A_1 \cdot \exp^{s_1 t} + A_2 \cdot \exp^{s_2 t}$$

Giving

$$v_h = A_1 \cdot \exp^{-\alpha + \sqrt{(\alpha^2 - \omega_o^2)}} + A_2 \cdot \exp^{-\alpha - \sqrt{(\alpha^2 - \omega_o^2)}} \quad [40]$$

Step 3: The Total Solution is the sum of the particular and homogeneous solutions:

$$v_t = v_p + v_h$$

Giving

$$v(t) = v_I + A_1 \cdot \exp^{-\alpha + \sqrt{(\alpha^2 - \omega_o^2)}} + A_2 \cdot \exp^{-\alpha - \sqrt{(\alpha^2 - \omega_o^2)}} \quad [41]$$

Step 4: Use the Initial Conditions to solve for the remaining constants
In this case we will use Zero State Response (ZSR) for our initial conditions:

$$v(0) = 0, \qquad i(0) = 0 \quad [42]$$

Frequency Domain Analysis (Analogue Filters)

With a ***continuous-time*** A.C. input, say $v_I = V_i \cdot \cos(\omega t)$, the dynamics, or Frequency Response, of an RLC circuit can simply be described using Impedences instead of using Differential Equations.

Impedance Method Summary

Step 1:	Replace sinusoidal sources by their complex (or real) amplitudes. $V_i \cdot \cos(\omega t) \text{'} \rightarrow \text{'} V_i$
Step 2:	Circuit elements replaced with boxes (impedances), R → R boxes, L with Ls boxes, C with 1 / sC boxes. Here $s = j\omega$. The circuit topology stays the same but we now have an impedance model of the network.
Step 3:	Determine complex amplitudes of the node voltages and branch currents using any method such as node, superposition, etc.
Step 4:	Not usually needed: Obtain time variables from complex variables:

$$V_a \rightarrow v_A = |V_a| \cdot \cos(\omega t + angle(V_a))$$

Analysis

This time we want the signal across the resistor

$$V_r = \frac{V_i \cdot Z_R}{Z_L + Z_C + Z_R} \qquad [43]$$

We know that:

$$Z_L = s * L \quad , \quad Z_C = \frac{1}{sC} \quad \text{and} \quad Z = R \qquad [44]$$

Giving:

$$V_r = \frac{V_i \cdot Z_R}{sL + \frac{1}{sC} + R} \qquad [45]$$

Therefore, the signal across the Resistor is:

$$\frac{V_r}{V_i} = \frac{\frac{R}{L}s}{s^2 + \frac{R}{L}s + \frac{1}{LC}} \qquad [46]$$

Substituting $s = j\omega$ gives the Frequency Response

In general, the system function can be written

$$\frac{V_o(s)}{V_i(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_{1s} + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_{1s} + a_0} \qquad [47]$$

Which is very similar in form to [22], the Transfer Function derived from Difference Equations and the Z-Transform.

Using a basic theorem of algebra, we can factor each polynomial into a product of terms:

$$\frac{V_o(s)}{V_i(s)} = K \cdot \frac{(s-Z_m)(s-Z_{m-1})+\ldots+(s-Z_1)}{(s-P_n)(s-P_{n-1})+\ldots+(s-P_1)} \qquad [48]$$

Where Z_i and P_i are roots of the numerator and denominator of the polynomials, respectively, and K is a constant. Z_i and P_i are called poles and zeros respectively. In general, these are complex constants of the form $\sigma + j\omega$. Since coefficients of polynomials are real (combinations of R, L, C), these poles and zeros are REAL or COMPLEX CONJUGATE pairs. We can plot the poles and zeros on the s plane.

Damping Scenarios

Three are three general solution scenarios:
1) Over-damped: $\alpha > \omega_0$ (in Time Domain) or $Q > 0.5$ (Frequency Domain)
2) Under-damped: $\alpha < \omega_0$ or $Q < 0.5$
3) Critically-damped: $\alpha = \omega_0$ or $Q = 0.5$

where $Q = \frac{\omega_0}{2 \cdot \alpha}$ and $2 \cdot \alpha = \frac{R}{L}$

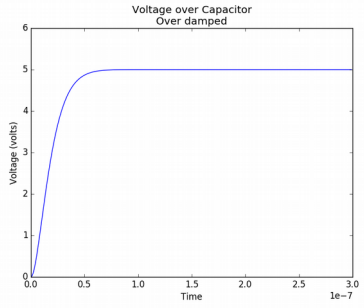
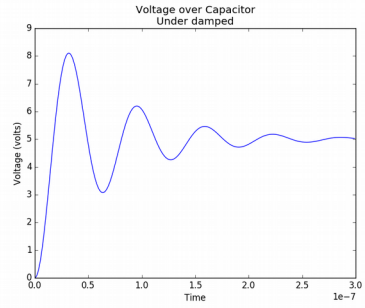
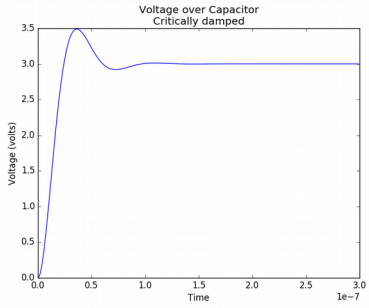
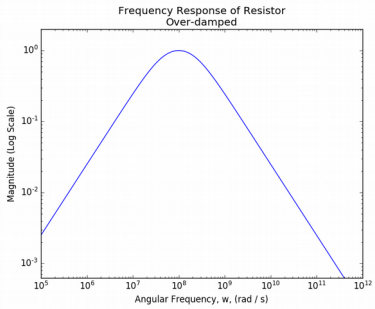
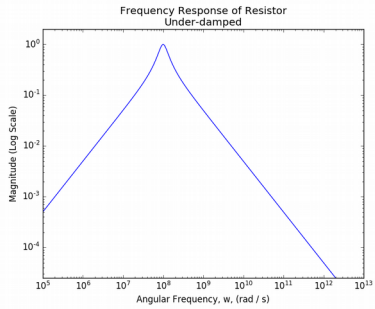
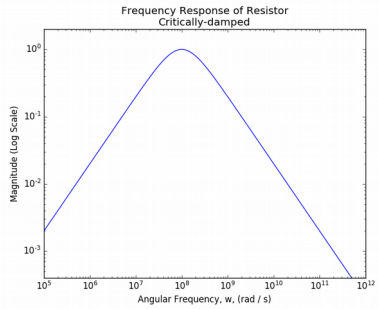
Over-damped	**Under-damped**	**Critically-damped**
Time Domain Response to DC Input (Real)		
	 “Ringing”	
Frequency Domain Response to AC Input (Complex)		
		

Illustration 7: Time Domain and Frequency Domain Analysis of RLC Circuit

Example: Butterworth Low Pass Filter Circuit

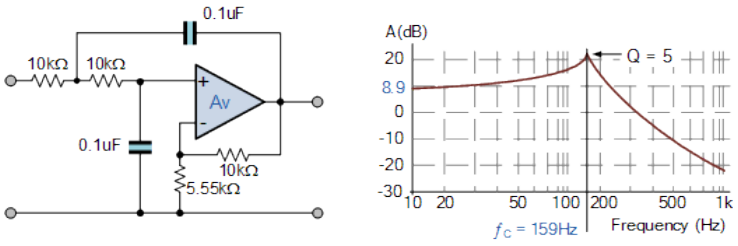


Illustration 8: Butterworth LPF with Freq Response.

In the Butterworth second order low pass filter, as shown in Illustration 8, Inductors have been replaced by Operational Amplifiers. The ‘negative feedback’ method can be clearly seen where the output of the Op Amp is feed back into the negative input port of the Op Amp acting as a ‘stablising techique, similar to antilock brakes.

We can see the peakiness of the frequency response curve is quite sharp at the cut-off frequency due to the high quality factor value, Q = 5. At this point the gain of the filter is given by Q x A = 14, or about +23dB, much higer than the gain of the Op Amp which has a Q value of 2.8 and a gain of +8.9dB.

Digital Filter Design

Infinite Impulse Response (IIR) is a property applying to many linear time-invariant systems. Systems with this property are known as IIR systems or IIR filters, and are distinguished by having an impulse response which does not become exactly zero past a certain point, but continues indefinitely. This is in contrast to a Finite Impulse Response (FIR) in which the impulse response h(t) does become excatly zero at times t > T for some finite T, thus being of finite duration.

In practice, the impulse response, even of IIR systems, usually approaches zero and can be neglected past a certain point. However the physical systems which give rise to IIR or FIR responses are dissimilar, and therein lies the importance of the distinction. For instance, analogue electronic filters composed of resistors, capacitors, and/or inductors (usually replaced by operational amplifiers, op amps) are generally IIR filters. On the other hand, discrete-time filters, usually digital filters, based on tapped delay line employing no feedback are necessarily FIR filters. The capacitors (or inductors) in the analog filter have a “memory” and their internal state never completely relaxes following an impulse. But in the latter case, after an impulse has reached the end of the tapped delay line, the system has no further memory of that impulse has returned to its initial state; its impulse response beyond that point is exactly zero.

Example: IIR & FIR Low Pass Filters

Here I borrow a specific set of specifications used for an interpolation system used to compare vartying system designs. These specifications are taken directly from a commercially available Philips Semiconductors DAC chip. The interpolation system up-samples the input signal by a factor of L = 128. The interpolation filter characateristics from the Philips Semiconductors chip are given in Table 4. All interpolation systems compared in this example are designed to meet these specifications.

Specification	Frequency Band	Value (dB)
Pass-Band Ripple	< 0.45 fs	0.1
Stop-Band Attenuation	> 0.55 fs	50

Table 4: Interpolation filter specifications. f s is the input sampling frequency

It is straight forward to use Python functions to compute the transfer function polynomial coefficients for each filter type and plot them on the complex plane as shown in Table 5.

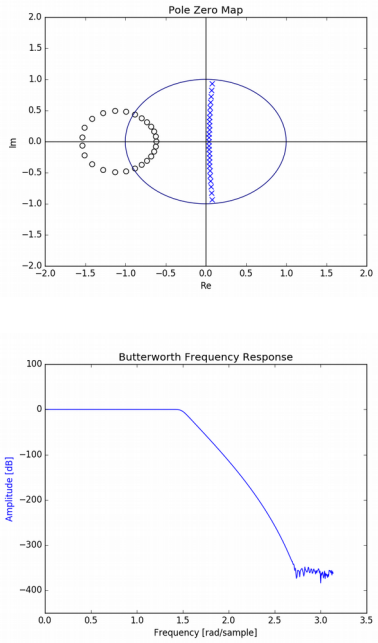
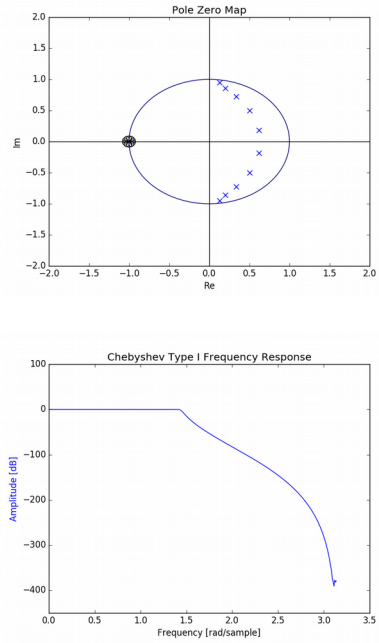
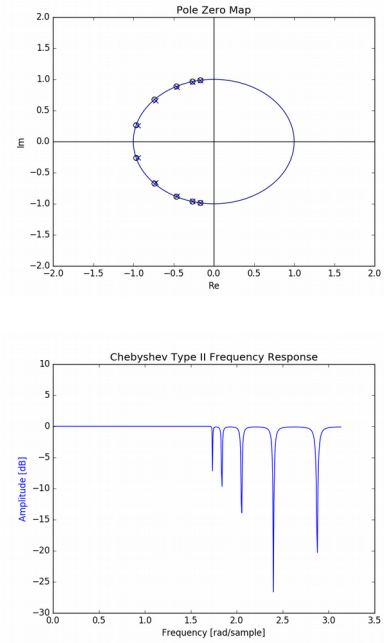
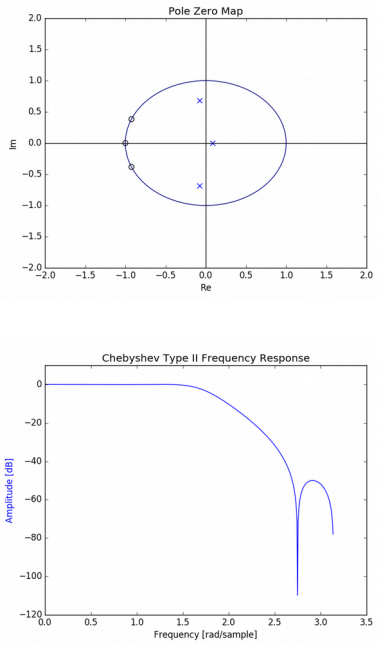
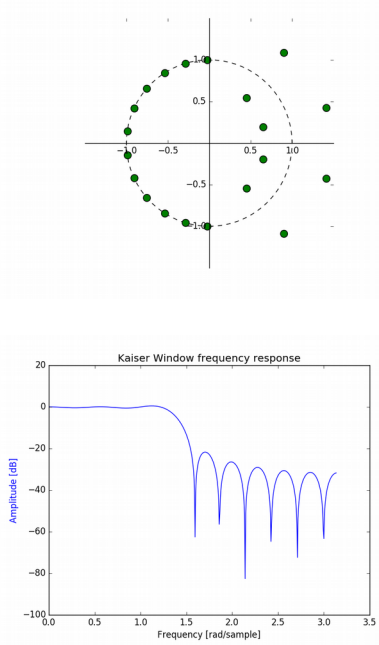
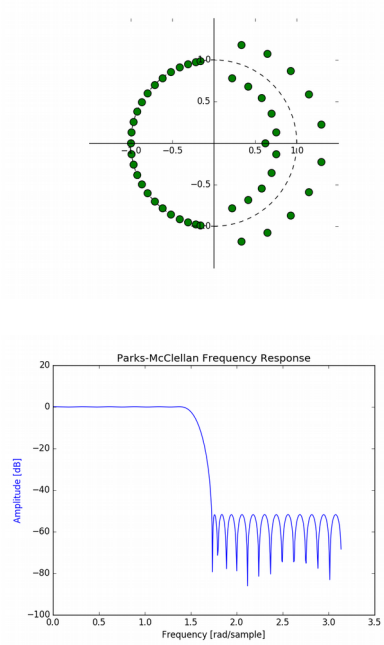
IIR - Butterworth	IIR – Chebyshev Type I	IIR – Chebyshev Type II
		
IIR - Elliptic	**FIR – Kaiser Window**	**FIR – Parks-McClellan**
		

Table 5: Filter Pole Zero Plots & Frequency Response Plots

In addition, we can use the 'Circus Tent' technique to find the Frequency Response. If we consider:

- Zeros pull the tent to the ground
- Poles push it up

Then the Frequency Response (in magnitude) is the tent route around the Unit Circle. To obtain an estimate of the Frequency Response we follow the angle from $-\pi$ to π and plot the corresponding value of the level curve. This can be seen by tracing the route around the unit circle in the Zero Pole plots in Table 5 and comparing it with the associated Frequency Response.

Spectral Analysis

Two primary applications of the DFT are its use in implementing signal processing algorithms as in LTI filtering, and in spectral analysis of deterministic and stochastic signals. In this section we look at how the DFT is used in the spectral analysis of narrowband signals, stochastic wide-band signals using the Periodogram, and then graduates to spectral analysis using the Spectrogram, filter banks and how filter banks are related to the discrete Fourier transform. As well at discussing how the DFT is used in spectral analysis it is also important to understand how not to use the DFT which is captured in the three illusuions which will be discussed in greater detail later:

- If you can't see it, it's not there (the picket fence effect)
- The more zero padding, the better the spectral resolution (resolution vs. sampling, spectral smearing)
- For a random process, as the data record length approaches infinity the magnitude squared of the DTFT, i.e. the Periodogram, converges to the true power spectral density.

A useful cognitive tool to understanding the inherent duality between the time and frequency characteristics is proving the famous Heisenberg uncertainty principle using Fourier analysis which is discussed next.

The Heisenberg Uncertainty Principle

The relationship between the time and frequency domains is mathematically captured in the Heisenberg Uncertainty principle which will become important when understanding Spectral Analysis.

The Fourier Transform of the position of a particle (or anything really) is the momentum. When something has a lot of momentum and energy its wave has a high frequency. Fourier Analysis states that if a wave consists only of a short pulse, such that most of it is located in a small region Δx, then to describe it in terms of sines and cosines will take many different wavelengths. This means that a wave confined to a region Δx in size must contain a range of different spatial frequencies, Δk. The Fourier Math Theorem (or Parseval's Theorem) states that these two ranges have the same relationship:

$$\Delta x \Delta k \geq \frac{1}{2} \qquad [49]$$

Applying this general relatonship to Quantum Mechanics (we replace k, the wave number, by p, the momentum) we arrive at the Heisenberg Uncertainty Principle:

$$\Delta x \Delta p \geq \frac{\hbar}{2} \quad \text{where } \hbar \text{ is Planck's Constant} \qquad [50]$$

As we have seen this equation has nothing to do with quantum behaviour but is a result of calculus and, further more, it applies to the maths of all waves:

- water waves, sound waves, light waves, sesmic waves,
- waves along ropes/piano wires, waves in plasmas & crystals, etc.

Spectral analysis of narrowband signals

The spectral analysis of narrowband signals is often directed at identifying the amplitudes and frequencies of individual sinusoidal components. The DFT is an important tool in this analysis both for deterministic and stochastic signals. The basic procedure is to apply a time-limited window to the data and compute the DFT, as in Illustration 9 with the corresponding signal Fourier transforms in Illustration 10 and a comparison of commonly used windows in Illustration 11.

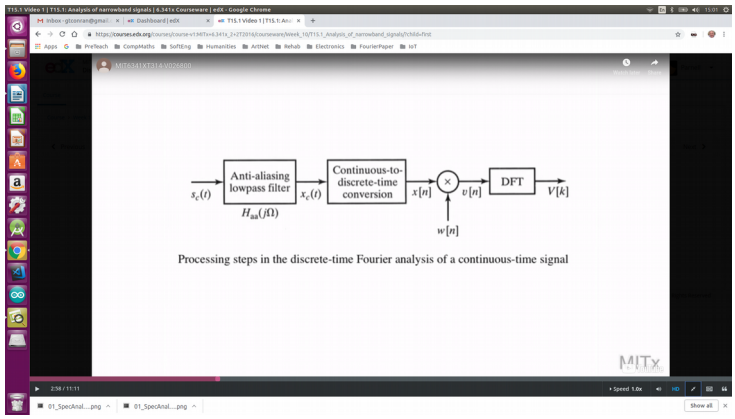


Illustration 9: Processing Steps in the DFT Analysis of a CTS (Deterministic Signal)

If the individual frequencies are to be identified visually from the DFT, then it is important that the window applied to the data be sufficiently long in relation to the minimum spacing of the frequency components so that the spectral "smearing" due to the time-limited window still permits peaks in the result that are visually resolvable. And the length of the DFT needs to be sufficiently large so that simple interpolation either visually or otherwise between the spectral samples provides an accurate representation.

Illusion 1

The first illusion, the picket fence effect, relates to the fact that when the DFT is used for spectral analysis the result is a set of samples of the underlying spectrum, essentially a view of the DTFT through a "picket fence". The underlying DTFT has content at a continum of frequencies and important aspects of the spectrum can be hidden by the sampling process.

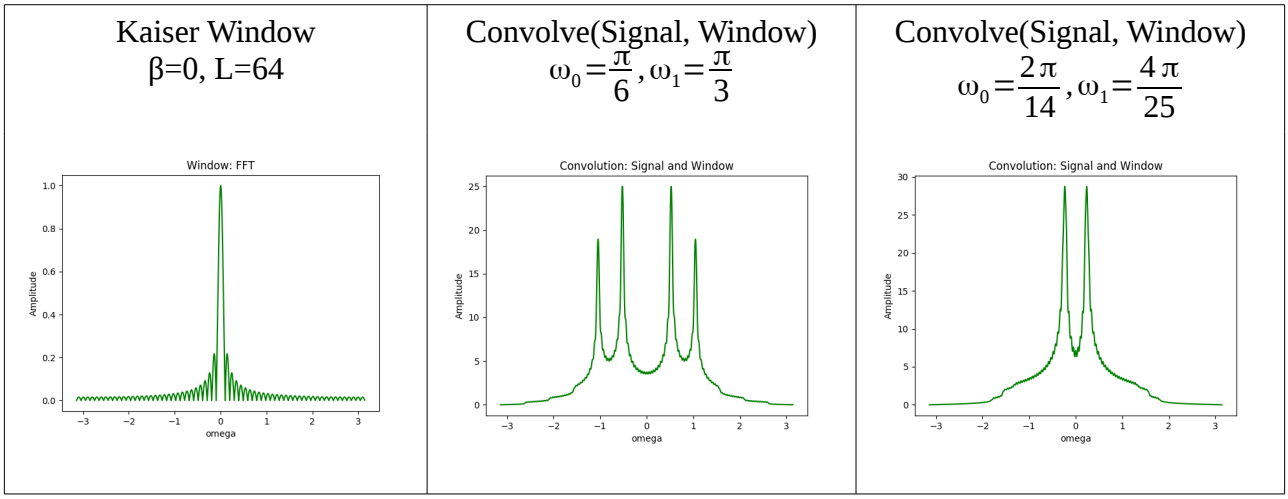


Illustration 10: Illustration of the Fourier Transforms in Illustration 13

Illusion 2

As mentioned earlier, the use of the DFT in spectral analysis typically involves applying a time-limited window of a certain length to the data and then computing the DFT. A second common misconception when the DFT is used to analyze or look for narrowband components is that increasing the size of the DFT but not the length of the window applied to the data results in increased resolution. An essential point is that the visual resolution of individual frequencies is fixed by the length of the window applied to the data. With a fixed window length, increasing the size of the DFT by zero padding will provide finer spectral sampling, which is often helpful in visually interpreting the spectrum, but will have no impact on the resolution.

Data Window

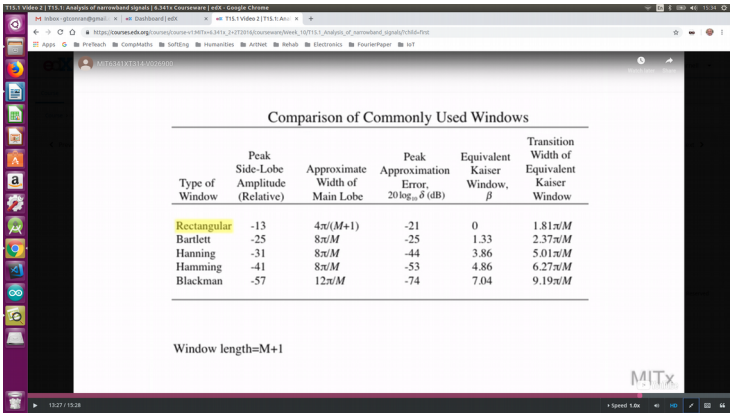


Illustration 11: Comparison of Commonly Used Windows

The ideal frequency characteristics of the data window is impulse-like, i.e. narrow main lobe, very small side lobes. The main lobe of the data window mainly affects ***frequency resolution***. The side lobes impact ***leakage*** between frequencies. ***Leakage*** can be seen in Illustration 10 where the component at one frequency leaks into the vicinity of another component due to the spectral smearing introduced by the window. Also, it becomes apparent that the two frequencies that make up the signal will not be ***resolved*** in the spectrum.

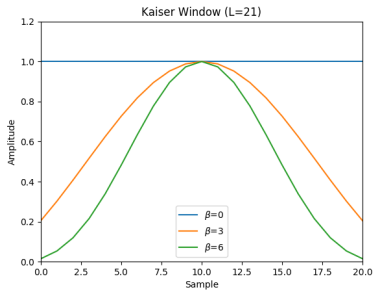
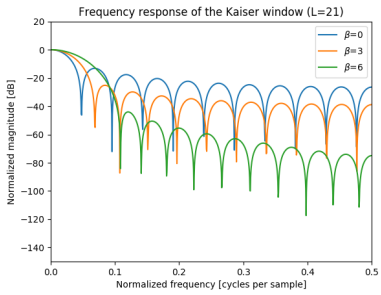
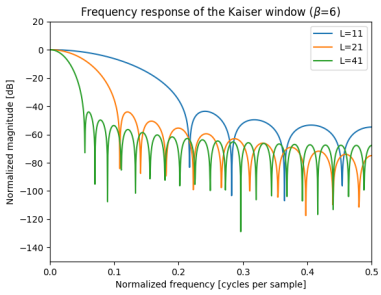
Time Domain Variable β	Frequency Domain Variable β	Frequency Domain Variable Length
		

Illustration 12: Kaiser Window Lobes

The longer the data window the better the resolution in frequency. The shorter the window the better the resolution in time, which becomes important as “frequency content changes”.

As with the use of windows for filter design, window design for spectral analysis has a “you push down here it pops up there” aspect, basically because of the inherent duality between the time and frequency domain characteristics as captured in the uncertainty principle. In filter design, it manifests itself in the trade-off between the ***transition bandwidth*** and ***passband/stopband ripple*** while in spectral analysis it is ***frequency resolution*** and ***frequency leakage***.

Examples of Kaiser Windows in the time and frequency domains are shown in Illustration 12.

Spectral analysis of stochastic wide-band signals

The objective in this case is to analyse the process to estimate the underlying ***power spectral density***. The typical procedure to compute the magnitude-squared of the DFT of a time-windowed segment of data, which is a segment of one realisation of the random process. The result is referred to as the ***periodogram***. The mathematics and process behind the Periodogram can be found in Illustration 14 with an example Periodogram in Illustration 13.

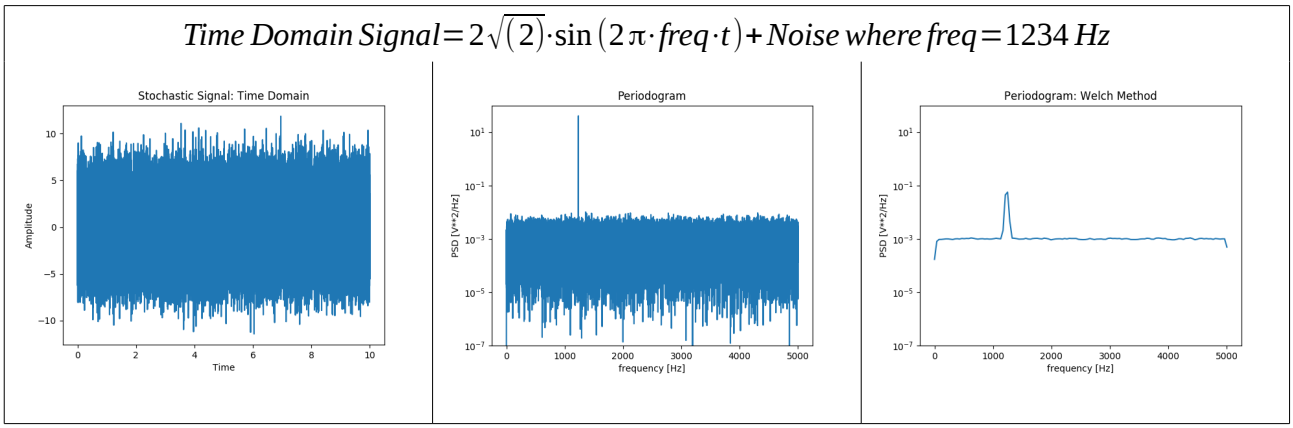


Illustration 13: Periodogram of a Signal in White Noise

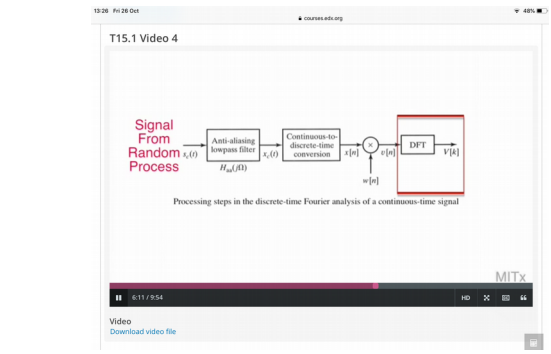
Illusion 3

Here again, of course, with a fixed length time window, increasing the size of the DFT only increases the spectral sampling of the underlying DTFT of that segment. And on average, the periodogram can provide a reasonable estimate of the power spectral density, but with statistical variance at each frequency. More importantly, for a given long data record, increasing the length of the window will increase the resolution of the expected or average value of the periodogram, but the variance at each frequency will not asymptotically go to zero, i.e. the periodogram will not be a consistent estimate of the power spectral density. Another example of the uncertainty principle at play.

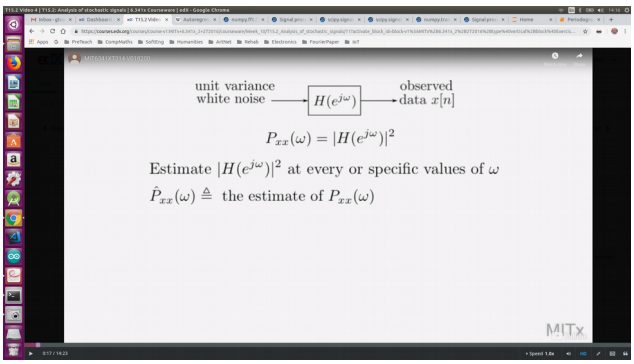
Although not covered here it is worth noting that spectrum analysis can also be conducted through the estimation of the (Auto) correlation function.

Notation:

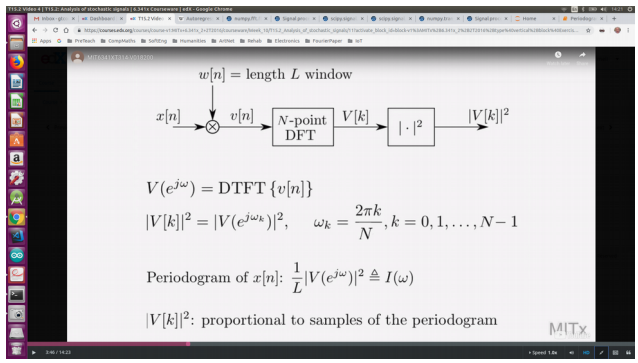
$I(\omega)$:	Periodogram of x[n] at frequency ω.
$P_{xx}(\omega)$:	PSD at frequency ω.
$\phi_{xx}[m]$:	Autocorrelation fun of x[n] at lag m
PSD:	Power Spectral Density



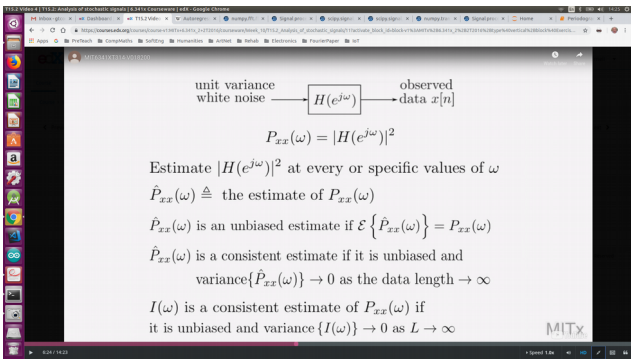
(a)



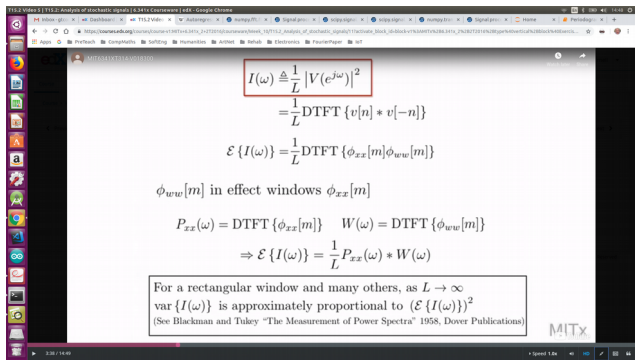
(b)



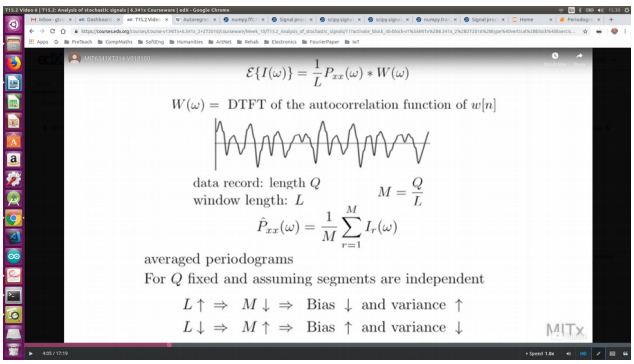
(c)



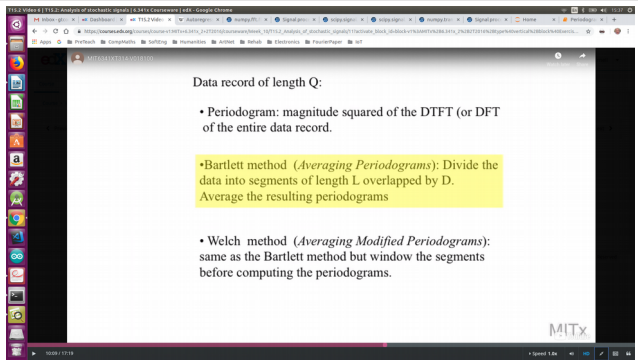
(d)



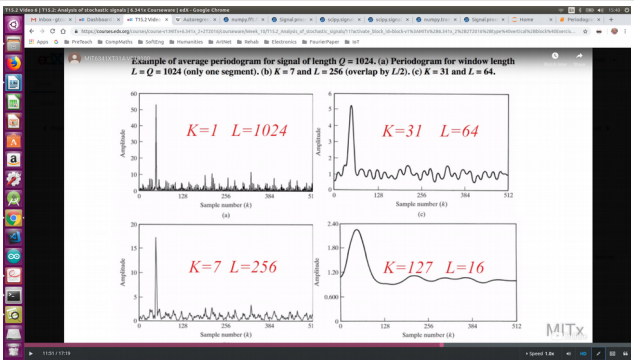
(e)



(f)



(g)



(h)

Illustration 14: (a) – (h): Process and Mathematics of the Periodogram

Fourier Analysis of non-stationary signals

The time-dependent discrete-time Fourier transform (TDDTFT) corresponds to spectral analysis of a signal by examining the Fourier transform of successive segments of the signal where the segments are obtained by moving the signal past a time-limiting window. If, under appropriate conditions, the TDDTFT retains all the information about the signal, then it is an invertible transformation. Many signals, such as, music, speech, and radar/sonar are more appropriately analyzed and understood in the frequency domain through the time-dependent discrete-time Fourier transform (TDDTFT) which reveals how the time-local frequency content is changing with time as shown in Illustration 15 of the 'chirp' signal, i.e. $\cos(\omega\, n^2)$.

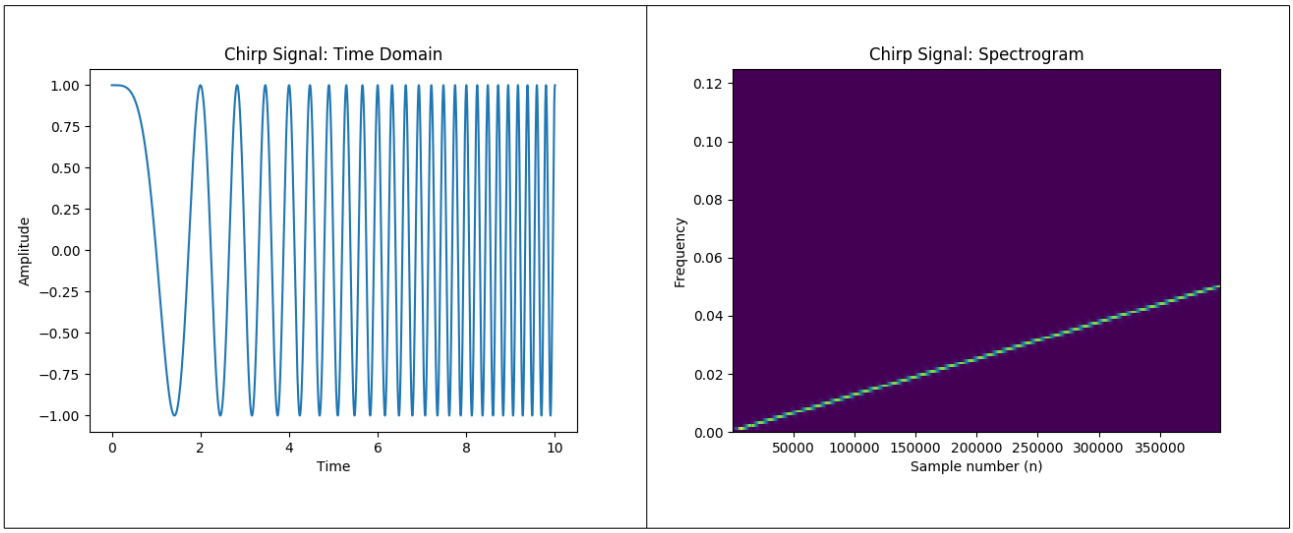


Illustration 15: Spectrogram of 'chirp' signal with its frequency changing over time

Note: There are three common ways of doing time varying spectral analysis: the TDDFT, the use of a modulator and a low-pass filter, and the use of a modulated filter bank (MFB). Mathematically, all three are identical. In comparing the MFB and the TDDFT it is easily seen that the window in the TDDFT is equivalent to the time-reversed low pass filter impulse response in the MFB.

Example: Spectrogram Display of the TDDFT of Speech

Illustration 16 shows a spectrogram display of the time-dependent Fourier transform of a speech signal with the time waveform shown on the same time scale below the spectrogram.

The top figure is a ***wideband spectrogram*** which results from a window (Hamming window of duration 6.7 ms) that is relatively short in time and is characterised by poor resolution in the frequency dimension and good resolution in the time dimension. The broad, dark bars that move horizontally across the spectrogram correspond to the resonance frequencies of the vocal tract, which, as can be seen, change with time. The vertically striated appearance of the spectrogram is due to the quasi-periodic nature of voiced portions of the waveform, as is evident by comparing the variations in the waveform display and the spectrogram.

In the ***narrowband*** time-dependent Fourier analysis of the same speech signal, a longer window (Hamming window of duration 45 ms) is used to provide higher frequency resolution, with a corresponding decrease in time resolution.

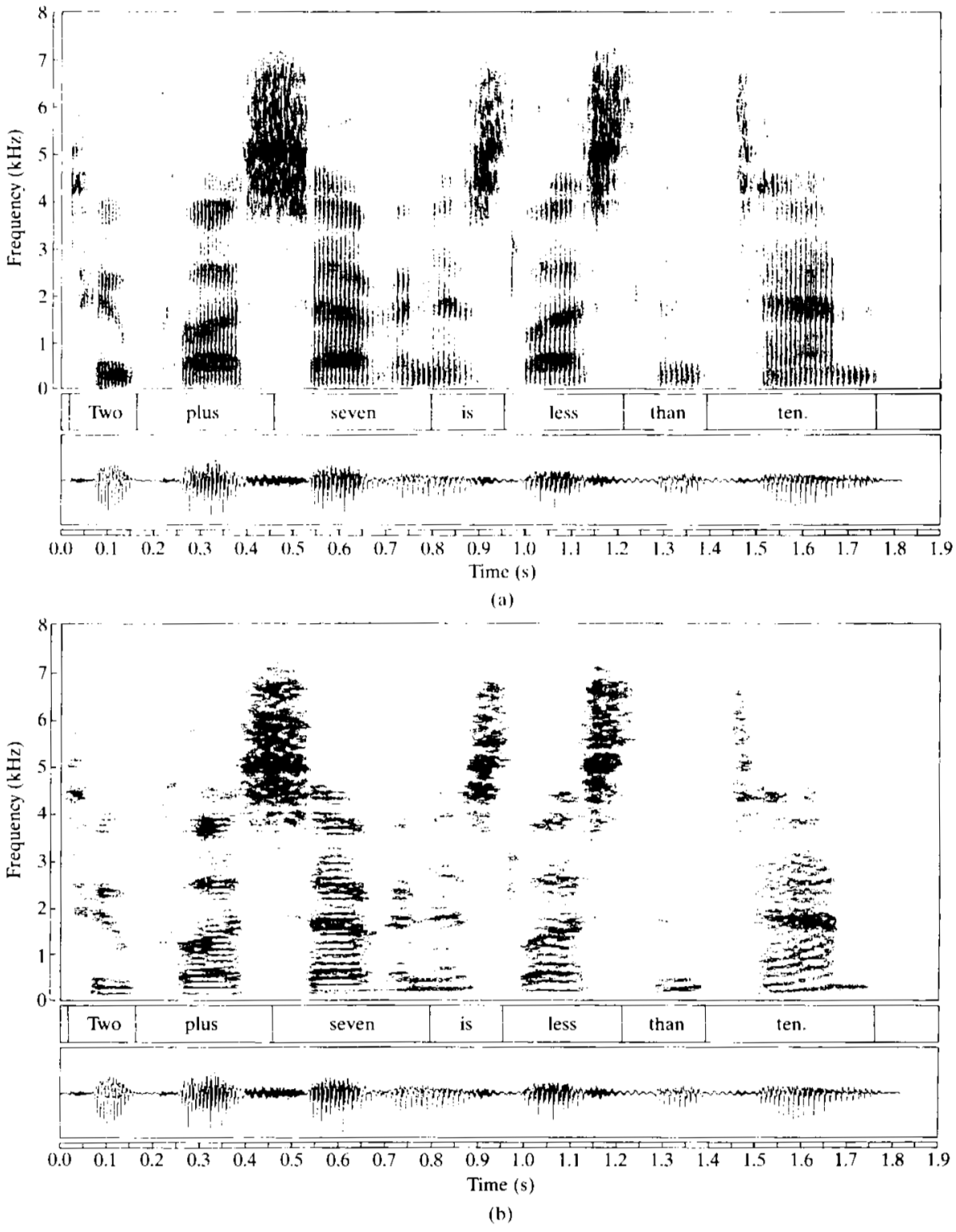


Figure 10.18 (a) Wideband spectrogram of waveform of Figure 10.17. (b) Narrowband spectrogram.

Illustration 16: Wideband and Narrowband Spectrograms of a Speech Signal

Applications

Integral Transforms are widely used in Engineering, Applied Mathematics and Physics and applications can be divided into two basic types, spectral analysis and filtering:

Spectral Analysis

- Geo-Physics
  - Geothermal energy mapping
  - Oil exploration
  - Archeology
- Astro-Physics (Astronomical Spectroscopy)
  - Optical, Radio & X-ray spectroscopy
  - Stars and their Chemical and Physical properties
  - Motion in the universe. e.g. Doppler effect & redshift
- Oceanography (coastal and offshore engineering)
  - Spectrum derived from sequences of radar or video sea surface images
- Bio-Physics
  - Electroencephalograms, electrocardiograms, and electromyograms

Filtering

- Image Processing
  - Computerised photography, e.g. Photoshop
  - Space image processing, e.g. Hubble space telescope images
  - Medical / Biology image processing, e.g. interpretation of X-ray / MRI images
  - Automatic Character Recognition, e.g. zip code
  - Finger print / face / iris recognition
  - Remote sensing / Reconnaissance: aerial and satellite image interpretations
  - Industral applications, e.g. product inspection / sorting
- Audio Processing
  - Broadcasting e.g. radio (and television)
  - Active noise control
  - Audio synthesis
  - Audio effects:
    - echo, flanger, phaser, chorus, equilisation, filtering, overfrive, pitch shift
    - time stretching, resonators, robotic voice effects, modulation, compression
- Digital Communication
  - Multiplexing
  - Compression
  - Echo Control
- Data Compression Standards
  - Image, e.g. JPEG, JPEG 2000
  - Video, e.g. MPEG-1/2/4
  - Audio
    - Music, e.g. AAC, MP-2/3
    - Speech, e.g. Adaptive Multi-Rate (used in GSM and 3GPP)

References

Papers / Books

Although not directly referenced in the paper, the following texts were used extensively:

Kutz, N. (2013). Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data

Oppenheim, A. and Schafer, W. (2013). Discrete-Time Signal Processing

Turek, D. (2004) Design of Efficient Digital Interpolation Filters for Integer Upsampling

Prandoni, P. and Vetterli, M. (2008). Signal Processing for Communications

Agarwa, A. and Lang, J. (2005). Foundations and Digital Electronic Circuits

Krauth, W. (2006). Statistical Mechanics: Algorithms and Computation

Blanchard, P. and Devaney, R. (2011). Differential Equations

Bonfert-Taylor, P. (2015). Complex Analysis

Fowler, J. and Snapp, B. (2014). MOOCulus

Guttag, J. (2013). Introduction to Computation and Programming Using Python

Sedgewick, R. and Wayne, K. (2011). Algorithms

Klein, P. (2013). Coding the Matrix: Linear Algebra through Applications to Computer Science

Wikipedia (multiple links)

Signal Processing (scipy.signal) https://docs.scipy.org/doc/scipy/reference/signal.html

Codes

The following codes were used to generate the various plots in the paper.

FourierSeries_SquareWave.py

FourierSeries_SawToothWave.py

FourierSeries_TriangleWave.py

SquareWave_FourierTransform.py

SineWave_FourierTransform.py

Gaussian_FourierTransform.py

Filters_IIR_Butterworth.py

Filters_IIR_Chebyshev_I.py

Filters_IIR_Chebyshev_II.py

Filters_IIR_Elliptic.py

Filters_FIR_Kaiser.py

Filters_FIR_Parks_McClellan.py

SecondOrder_RLC_SeriesCircuits_TD.py

SecondOrder_RLC_SeriesCircuits_FD.py

LPF_FFT_Convolution.py

DerivativeRelationship.py

KaiserWindow.py

KaiserWindow_ProcessingSteps.py

Periodogram.py

Spectrogram.py

Solving ‘periodic’ Differential Equations

Taking advantage of the ‘Derivative Relationship’ which allows derivatives to become multiplication by a variable when passed through a FT, periodic Differential Equations are easily solved. For example, consider the following Differential Equation:

$$y'' - \omega^2 \cdot y = -f(x) \quad \text{where} \quad x \in [-\infty, \infty] \tag{49}$$

We can solve this by applying the FT to both sides which gives the following reduction:

$$\hat{y''} - \omega^2 \cdot \hat{y} = -\hat{f} \tag{50}$$

By applying the ‘Derivative Relationship’:

$$\hat{f^{(n)}} = (ik)^n \cdot \hat{f} \quad \text{to} \quad \hat{y''} \tag{51}$$

And after rearranging the terms we have

$$(k^2 + \omega^2) \cdot \hat{y} = \hat{f} \tag{52}$$

Which in turn gives us the following polynomial equation

$$\hat{y} = \frac{\hat{f}}{k^2 + \omega^2} \tag{53}$$

To find the solution, y(x), we invert by applying the inverse FT to both sides giving

$$y(x) = \frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} e^{ikx} \cdot \frac{\hat{f}}{k^2 + \omega^2} dk \tag{54}$$

The solution, y(x), is given in terms of an integral which can be solved analytically or numerically.