



1.1 PREAMBLE

We've set out this material as 24 chapters with a target of 10 pages per chapter. That should make each chapter "digestible" at a single sitting. After 24 such sittings, you should know something about DSP. This brief and hopefully enticing recipe comes at a price: it calls for great economy of words, and not much room for elaboration. The chapters may be short, but they demand your full attention. So be warned.

This chapter is introductory, with very little maths. It helps set the scene for what follows. We talk about the different kinds of signals, and about signal attributes such as power and energy. We talk about "frequency", and how signals have alternative descriptions as "spectra". We note the role of signals in instrumentation and control. We say how signals are used for information transfer, and we mention some operations on signals, such as compression, enhancement, and encryption. We finish with some comments on the analogue and digital technologies that make all of these things possible.





1.2 SIGNAL TYPES

In this section we categorise signals as *analogue* or *digital* signals, as *pulses* or *periodic* (*i.e.* repetitive) shapes, as *continuous* or *sampled* data types.

1.2.1 Signals Overview

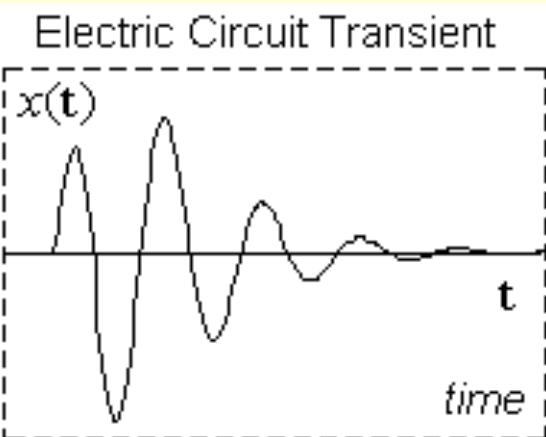
The term *signal* refers to the quantitative measurement of parameters that interest us. Some signals, such as *temperature*, *pressure*, *velocity*, etc, are *physical* parameters, and we often record their variations over time. A velocity signal, for example, might be called $v(\mathbf{t})$ where \mathbf{t} is the time parameter. For any given \mathbf{t} value, the function $v(\mathbf{t})$ gives the velocity at that time. This $v(\mathbf{t})$ is an *analogue* signal quantity that varies *continuously* over time.

Contrast this with a very different signal: the daily variations in currency exchange rates. The value of the Euro against the US dollar changes daily, and the "signal" is a sequence of numbers, one per day, that show the daily exchange rates. This signal too varies with time, but it is *discrete* rather than continuous, just *one* value per day. And, because it is a number, we probably think of it as *digital* rather than analogue. Even analogue signals however, like velocity and temperature, can be converted electronically to digital form and then viewed numerically on a digital display panel.

Not all signals are functions of time. A surveyor will map out a building site as a function $h(x,y)$ that shows the height at each (x,y) co-ordinate position. The signal is continuous, but the surveyor just records an adequate set of sample points and saves them in numeric form for future use.

Some signals call for *complex-number* values. Wind velocity is a good example: it has both magnitude v and direction θ and we can record it as $v\angle\theta$ meaning " v angle θ ". Both v and θ vary with time, so we should call it $v(t)\angle\theta(t)$. In our work, we'll meet lots of complex-valued signals, so we need to get used to the idea.

1.2.2 Analogue Signal Waveforms

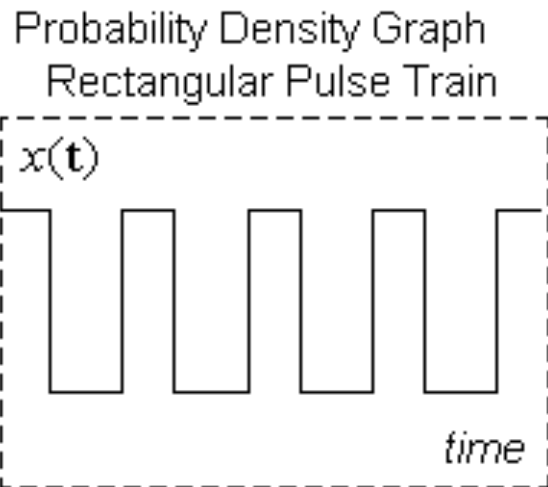


Although many signals vary in irregular manner with no strong distinguishing features, the following distinction is very often applicable:

- 1) signals that describe "one-off" events
- 2) signals that describe events which repeat over and over

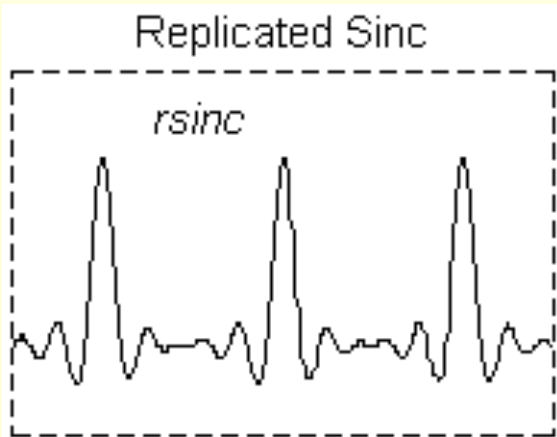
Type 1) signals are *transients* or *pulse* waveforms, like the electric circuit transient shown here (Fig ẽ). They have a shape that happens just once. They frequently begin at a certain time instant (and are zero-valued before that). Some of them terminate and are zero-valued thereafter. Others decay slowly to zero but never quite reach zero. Their common feature is that they have *finite area* and "finite total energy" as we will shortly explain.

Here's a rather different pulse (Fig ç). This probability density function $p(x)$ is a pulse sure enough, but not a "time event" of any kind. It too has finite energy, and for processing purposes, it is similar to many other pulse waveforms.



Type 2) signals are *repetitive* or *periodic* waveforms. Here (Fig ç) we see a portion of a periodic rectangular waveform. It could be a test signal from a function generator, or a clock signal on a digital circuit board. We see just a part of it, through a finite *window* width, but it is presumed to go on forever. That presumption simplifies the maths, since we don't have to specify how it starts or finishes. The same presumption applies to the waves that we call "sin" and "cos".

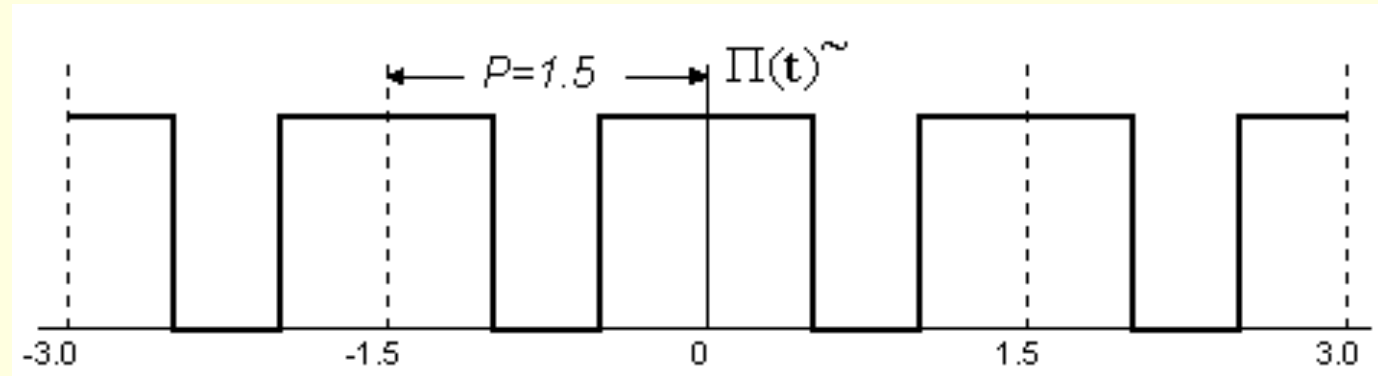
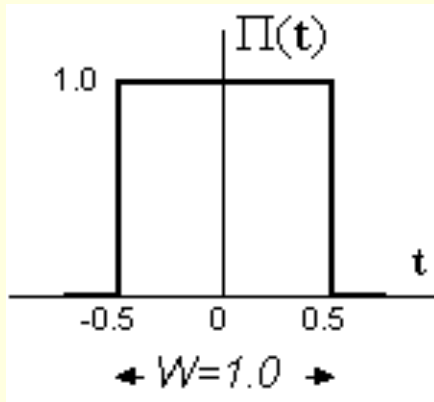
This (Fig í) is another periodic waveform. It has a shape that we will encounter quite often. A single pulse of this kind is called a "sinc" pulse, and the periodic version shown here will be referred to as an "rsinc" waveform, where "rsinc" stands for "replicated sinc". Periodic signals have unlimited energy, but finite mean power. We'll elaborate shortly.



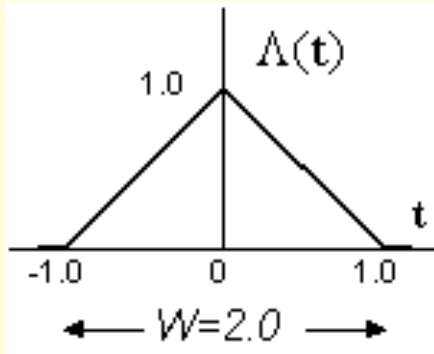
1.2.3 Replication and Periodicity

We can link pulses to periodic waveforms through a process called *replication*. The idea is that we can replicate a pulse and thus create a periodic "version" of the pulse. This is a pulse (Fig 1) that we call $\Pi(t)$, pronounced "rect of t" for its rectangular shape. It's got a height of 1.0 and an area of 1.0, so its width is $W = 1.0$ also (extending from -0.5 to $+0.5$). We'll meet it often, so please note the definition.

We can replicate $\Pi(t)$ at a replication interval P of our choosing. This plot shows the replicated $\Pi(t)$ as $\Pi(t)_{\sim}$ with interval $P = 1.5$, (Fig 2). Notice the \sim symbol on $\Pi(t)_{\sim}$. We use this to mean replication or periodicity.

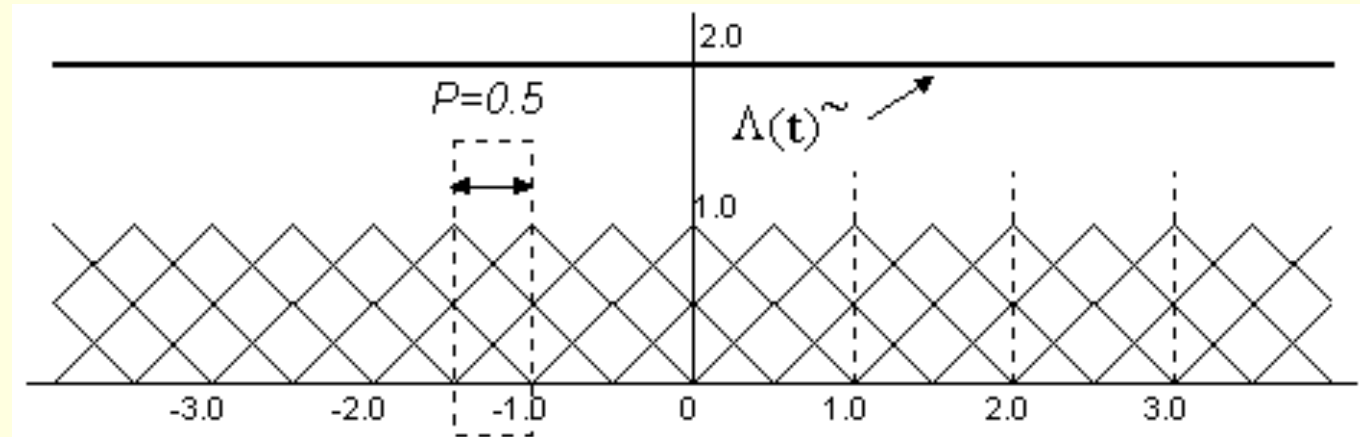


The pulse has been duplicated at regular intervals of $P = 1.5$ all along the axis. The result is the *sum* of such pulses, and it forms a periodic signal that goes on forever in both directions. Because we chose $P > W$, the pulses do not overlap, and we can still see the rectangular shape of the original pulse. Replication is a very important concept in signal processing.

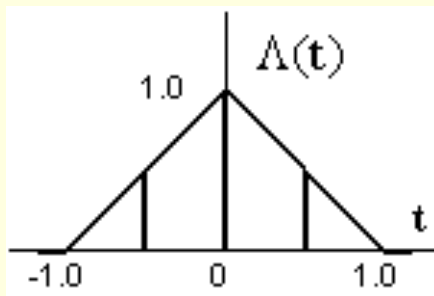


Here's another pulse (Fig ç) called $\Lambda(t)$ or "tri of t " for its triangular shape. It's got a height of 1.0 and an area of 1.0, so its width must be $W = 2.0$ (extending from -1.0 to $+1.0$). We'll meet it often, so please note the definition.

This plot (Fig ê) shows the replicated triangle $\Lambda(t) \sim$ with replication interval $P = 0.5$. The pulse width is $W = 2.0$ and now, because $P < W$, the pulses are overlapping, in fact we see multiple overlaps. The diagram shows us each of the overlapping pulses, but $\Lambda(t) \sim$ is the sum of all these, and it turns out to be the constant level of 2.0 on the top of the diagram.



This tells us a lot about replication. First, if $P < W$, the pulses overlap and *the original pulse shape is lost*. Note carefully, we can always build $\Lambda(t)_{\sim}$ from $\Lambda(t)$ once P is specified, but *we cannot reverse the operation*. The constant level of 2.0 tells us nothing about the triangles that formed it. Other pulse shapes could give the same result, for example, a triangle of height 2 and width 2, replicated at $P = 1.0$. We conclude that, when we replicated $\Lambda(t)$, *we lost information*, and this is true in general for replication with overlap.

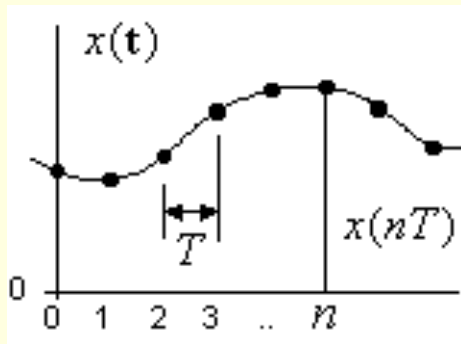


This plot (Fig c) shows $\Lambda(t)$ in slices of width $P = 0.5$. Careful comparison against $\Lambda(t)_{\sim}$ above during a one-period span equalling $P = 0.5$ (the dotted rectangle) shows how $\Lambda(t)_{\sim}$ is a superposition of all four slices. We just move all the slices into the same one-period window and add them together. This has some interesting consequences:

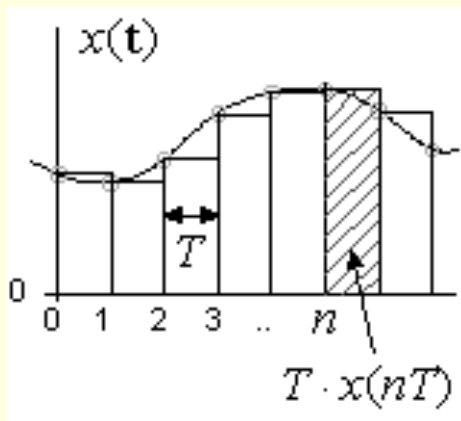
- *the area within a one period window of the replicated pulse waveform equals the area of the original pulse.*
- *any window that is one period (P) in width can be used; sliding it to left or right has no effect.*

The window contains a stack of slices which, if laid out side by side, would re-assemble the original pulse. This is true even for pulses of unlimited width, such as the $p(x)$ that we showed earlier. In such cases, we have an infinity of slices to add together. We can't rebuild $p(x)$ uniquely from $p(x)_{\sim}$, but there remains a connection between them. That connection has surprising importance in DSP generally, as subsequent chapters will reveal.

1.2.4 Sampled Data Signals



We cited the Euro exchange rate as a *discrete* data signal, just a sequence of numbers. In the computer-driven world of today, analogue signals too are routinely converted into number sequences, because computers work exclusively with numbers. This conversion calls for *sampling* of the signal $x(t)$ at regular time intervals of T seconds as the diagram suggests (Fig ç). The signal is sampled at times $t = 0, t = T, t = 2T, t = 3T,$ and so on, and the sample values are converted to numeric form in an Analog-to-Digital converter, usually abbreviated as ADC or as A/D. The sample values are the values of $x(t)$ at times $t = nT$, where n is an integer counter value, $n = 0, 1, 2, 3, 4, ..$ etc. The value $x(nT)$ is the vertical line length on the diagram. It's a numeric equivalent of the analogue signal level at that instant. The A/D converter gives us a stream of these *value-samples* that describe the signal.



This diagram (Fig ç) shows the same information as *area-samples*. Each area-sample gives the *area* of a slice of width T , such as the shaded slice in the diagram. It computes as $T \cdot x(nT)$, as opposed to $x(nT)$ for a value-sample. Therefore area-samples are just value-samples scaled by T , the sample interval. So why use them at all ? Well, first we can observe that:

$$\text{area under } x(t) \approx \sum_n \{T \cdot x(nT)\}$$

New Page 1

The area is approximately the sum of the area-samples. The fact is, area-samples include T and thus retain the link with time, whereas value-samples are just a stream of numbers with no inherent time information. The usefulness of area-samples is at a theory level, where they help us retain the link between analogue signals and their sampled-data counterparts. This is quite important because, when we sample a signal, we *lose* all the signal information between the samples.

We also noted a loss of information when we replicated a signal. Later on, we will discover a connection between replication and sampling !





1.3 POWER AND ENERGY

Its time to say what we mean by power and energy. We'll use the AC mains voltage as an analogue signal example, but we'll extend our ideas to cover sampled-data sequences as well.

1.3.1 Analogue Signal Power and Energy

For an analogue time-signal $x(t)$, we define the signal *power* very simply as:

Instantaneous power :

$$p(t) = x^2(t)$$

The power $p(t)$ is indifferent to the polarity of the signal $x(t)$, and it tends to emphasise the higher values of $x(t)$. If we integrate $p(t)$ over the life of the signal, we get:

Total Energy:

$$E = \int_{-\infty}^{\infty} p(\mathbf{t}) \cdot d\mathbf{t}$$

This is a "global" measure of signal activity. With *pulse* waveforms, we can expect the integration to give a finite result, that is, a finite value of total energy E . Not so with periodic waveforms. Because they go on forever, their total energy is infinite, and that's not a useful measure. The solution is to measure the energy over one cycle only:

Energy over one cycle:

$$E_1 = \int_P p(\mathbf{t}) \cdot d\mathbf{t}$$

where the upper-case P is the period of the waveform (or the duration of one cycle) in seconds. Then we can also specify the *mean power* over a cycle as:

Mean power:

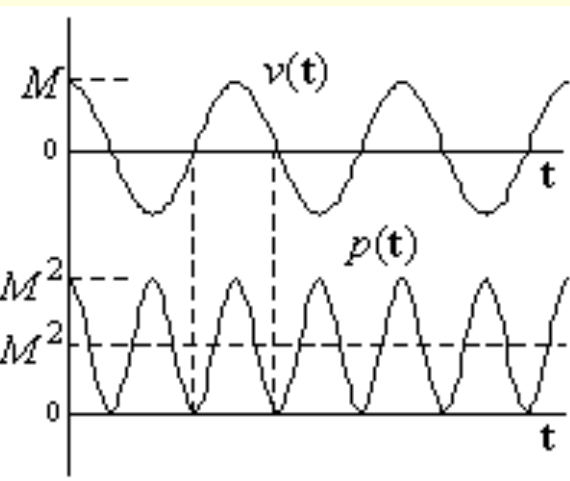
$$\bar{p} = \frac{E_1}{P} = \frac{1}{P} \int_P p(\mathbf{t}) \cdot d\mathbf{t}$$

These concepts of power and energy are not unlike the *true* power and energy from which this terminology is borrowed. If $x(\mathbf{t})$ were the voltage across a resistor R , then the instantaneous power in the resistor would be $x^2(\mathbf{t})/R$, or $p(\mathbf{t})/R$ as we have defined it. The only difference is in the factor R , but *true* power defines a very real heating effect as a *rate of energy consumption*, measured in Watts. The time integral of that activity is E , the energy used, which is also the total heat generated, in Joules.

The AC mains voltage provides a useful illustration. We can specify it as:

$$v(t) = M \cdot \cos(2\pi ft + \alpha), \quad M = 220\sqrt{2} = 311 \text{ volts}, \quad f = 50 \text{ Hertz}$$

for a typical European distribution system. The angle α is the signal angle at the time we choose to call $t = 0$. Any angle will do. This is known as a "220 volt" mains, although its peak value is 311 volts. There's a reason for this, as follows.



This plot (Fig ç) shows a few cycles of the actual mains voltage, $v(t)$, and underneath (Fig í) is a plot of the "instantaneous power" as we define it in signal theory, that is:

$$p(t) = v(t)^2 \quad \text{in watts}$$

The power waveform goes from a minimum of zero to a maximum of M^2 , the peak value squared. It is a "raised cosine" shape, with a frequency of 100 Hz. Every half-cycle of the voltage equates to one full power cycle. It is obvious from the shape of the power wave that the average power is one-half of the peak power:

$$\bar{p} = M^2 / 2 \quad \text{watts into 1 Ohm}$$

We like to specify the sinusoidal mains in such a way that this average power (into $R = 1\Omega$) equals the voltage-squared, just as it is for DC voltages. Thus, the voltage is the *root* of the mean power, and we say:

$$V_{\text{rms}} = \sqrt{\bar{p}} = M / \sqrt{2} = 220 \text{ volts}$$

This voltage is "the root of the mean of the squared waveform", hence the abbreviation "rms" that describes it. In AC *true*-power calculations, when the resistor R is included, it enables us to make statements such as:

$$\bar{p} = (V_{\text{rms}})^2 / R \quad \text{or} \quad R = (V_{\text{rms}})^2 / \bar{p}$$

– just as we would do for a DC circuit. Thus, the resistance of a 1 kilo-Watt heater element is just $R = 220^2 / 1000 = 48.4 \Omega$. The rms value of 220 volts is the significant quantity as far as power usage is concerned. Then the rms current is calculated as $220 / 48.4 = 4.5$ Amps. In the signals context, and for future reference, we should remember that the mean power of a sinusoid is $M^2/2$, or one-half of the peak-value squared.

Compare this with the mean value of mains voltage $v(t)$: taken over one or more cycles, the mean value is zero, which renders it useless as a "global" measure of signal activity. The power, by contrast, is $v(t)^2$. It is always positive, its integral is always increasing, and it works well as the global measure that we require.

The idea of power is very useful for a *noise* signal. We don't know the noise *value* at any given time, but we can still quantify it in terms of its mean power, and thus decide whether it is large enough to be objectionable.

1.3 2 Sampled Data Power and Energy

When a signal $x(t)$ is sampled, with T seconds between samples, the sample values are $x(nT)$ for $n = 0, 1, 2, \dots$ etc. We simplify our notation by calling them $x[n]$, the same sequence of numbers, but with no mention of T .

Signal power $x^2(t)$ is an instantaneous thing, and so the sample power becomes $x[n]^2$, with no time dependency. Energy, however, is (power \times time), so we should think of the energy per sample as $T \cdot x[n]^2$. In practice, we often drop the T , thus ignoring the time aspect, but still obtaining a useful global measure. The total energy of a sampled signal is therefore $T \cdot \sum x[n]^2$ (summed over all samples), or just $\sum x[n]^2$ if we ignore the time factor. We would then find the mean power over N samples as either $T \cdot \sum x[n]^2 / (NT)$, or as $\sum x[n]^2 / N$ when T is ignored. Either way, the mean power is independent of the sample interval T .





1.4 TIME AND FREQUENCY DOMAINS

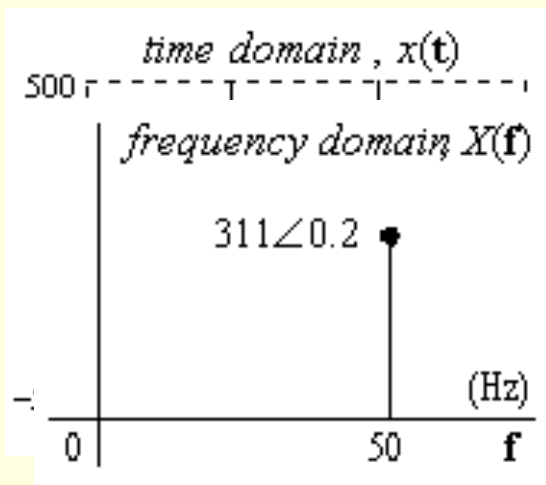
A great deal of DSP work is about signals that change over time, but in our efforts to describe them, we also speak of *frequencies*, such as the 50 Hz frequency of the AC mains. Such references are hard to avoid. They are part of an alternative viewpoint that we can bring to bear on a signal. We can describe a signal fully in the *time domain*, or equally well in the *frequency domain*. Each description has its good and its bad features, but the frequency approach is of such importance that it runs all through this work, and through most books on signal processing. We will try to explain the differences in these approaches.

1.4.1 Time versus Frequency

The time description of our mains voltage might have been:

$$x(t) = 311 \cdot \cos(2\pi 50t + 0.2)$$

(We tend to use $x(t)$ for analogue time signals generally). This *time-domain* description is the collection of all possible signal values. We've drawn it alongside over a 60 milli-second time window (Fig 4). Even a rough plot of $x(t)$ requires several data points to convey the general shape.



There's a simple alternative, as this diagram illustrates (Fig 5). We can just say:

$$X(f) = 311\angle 0.2 \quad @ \quad f = 50 \text{ Hz}$$

This is the frequency-domain description. It says just as much as the time-domain description, but it says it more compactly. The $X(f)$ plot needs only a single complex-number value placed at $f = 50$ Hz. Its magnitude is 311, the cosine peak value, and its angle is the initial angle of 0.2 radians (when $t = 0$).

But what of other signal shapes? Actually, this method would not be much good were it not for the fact that *we can construct any shape we please from a sum of sine waves*. We'll see the evidence of this as we continue. We can thus have a *spectral diagram* $X(f)$ that shows one complex number for each sine wave that makes up the signal.

In the future, we can describe our signals as $x(\mathbf{t})$, the time–domain view, or alternatively and equivalently as $X(\mathbf{f})$, the frequency–domain view. If we know $x(\mathbf{t})$, we can find $X(\mathbf{f})$ and vice versa. Indeed, the two are linked by a transform (formula) known as the Fourier Transform (FT). There are different FT's to suit different signal types, as follows:

- The Continuous Fourier Transform (CFT) is for analogue time pulses.
- The Discrete–time Fourier Transform (DtFT) is for *sampled* time signals, or for number sequences.
- The Discrete–frequency Fourier Transform (DfFT) is for periodic time signals, like the sine wave that we mentioned, and these signals are represented by a number sequence in frequency.
- The Discrete Fourier Transform (DFT) connects data *samples* in time to data *samples* in frequency. It is a fully digital transform, and is widely used to do calculations by computer.

These transforms are developed over the next two chapters. They will allow us to work interchangeably in the two domains. The DFT works only with numbers, making it ideal for computer use, and it has a *fast* version that we call the Fast Fourier Transform (the FFT). Not only does it switch quickly between the domains, but it can speed up various number–crunching jobs, doing them much more rapidly than would otherwise be possible. The FFT is the workhorse of signal processing, and is very widely used.





1.5 TECHNOLOGY REVIEW

As recently as 1970, digital signal processing was a little known area, just beginning to emerge. Thirty years later, in the year 2000, its effects were everywhere to be seen, in music, in computers, in communication, and about to conquer television as well. And the world wide web had arrived, persuasive evidence of an emerging "Information Age". We will take a brief glimpse backward over this era of extraordinary development.

1.5.1 The Early Days

The technology of world war two was high technology, for its time, with impressive achievements in electronics, in spite of a very limited technology base. Radio communication had arrived, and signal processing had commenced, with signals ranging from morse code through analogue radio signals to the early radar signals and, of course, *everything* was done using *analogue* techniques. The bipolar transistor had not yet arrived, and thermionic valves provided the link into the future. These were soon made obsolete by the transistor, which for many years prospered as a stand-alone three-terminal device. With the transistor, analogue design methods flourished, and quite a lot could be done with just a small number of these devices. Rapid progress in radio and in television provided ample evidence of all this.

The first experimental computers used thermionic valves, and the transistor soon transformed them into serious computing machines, but in a price range that was affordable to big business, and to military users, but not to many others. All that began

to change with the arrival of the integrated circuit.

1.5.2 Development Milestones

The planar integrated circuit (IC) was introduced by Robert Noyce in 1959, making it possible to build and to interconnect many devices simultaneously on a single "chip", and to reproduce these circuits easily and in quantity. The IC set the scene for the accelerated growth which followed.

Nine years later, Noyce joined with Gordon Moore to start the Intel Corporation in Santa Clara, California. The first microprocessor was built by Intel in 1971, and it heralded the beginning of the personal computer era. The early highlights of this era included the appearance of the Apple II in 1977 and the IBM PC in 1981.

The microprocessor is a general-purpose binary machine, with logic and arithmetic capability. Typical microprocessors can add and subtract in a single machine cycle, but a multiplication involves many such steps, and must be expressed as a software routine that takes several machine cycles to execute.

This approach is too slow for a majority of DSP tasks that must operate in real-time. Fast multipliers are a priority, and this is a primary distinction between DSP processors and other processors. DSP processors have their own internal multiplier "engine", implemented in silicon, and can generally complete a multiplication in a single machine cycle.

Multipliers have come a long way, from the multipliers of the seventies which filled complete circuit boards, to the single-chip multipliers of the eighties, and to the vastly reduced geometries of the nineties. The time to perform a multiplication has fallen dramatically as well, from several hundred nanoseconds for the board-level products to less than 20 nanoseconds in 1997. This latter figure is a mere 20×10^{-9} seconds, and its reciprocal is 50×10^6 . This multiplier could execute 50 million multiplications per second !

1.5.3 Here and Now

Looking back a quarter century or so, the DSP of that time had few engineering roles, and very little impact that we could see. In the interim, everything has changed. That change is largely due to the advent of fast inexpensive digital technologies. The most visible manifestation of change is in the proliferation of computing equipment. But, alongside these advances, a great many analogue engineering functions have been replaced by new digital counterparts. Most information flow is now in digital form. This includes audio information, which is now mostly digital, and digital video signals, which are not far behind.

The migration to digital is limited by two main factors: cost and speed. As the cost of digital functions continues to fall, most of the remaining analogue systems will become obsolete. Speed is a more significant limitation, and certain specialised high frequency processes will continue in analogue form for a long time to come.

This does not alter the fact that we already live in a mostly digital world, and the current boundaries between radio, television, telephones, computers, etc are being eroded. We face a new era in which all these systems, and more, are just different manifestations of *information transfer and processing*.





1.6 DSP TODAY

In this section, we will take a brief glimpse at the signal processing operations of today, the parameters that interest us, and the underlying motivations.

1.6.1 Signal Monitoring Tasks

Peak Value
Average Value
Zero Crossings
Slope
Error
Mean Squared Value
Power
Energy

The information that we want from a signal can be in any of several forms, some of which are listed alongside (Fig ç). Here are some examples.

- *Peak value* and *average value* are important in weather monitoring (hottest day, average rainfall), and in a host of other areas.
- *Zero crossing* detectors count the magnetic flux reversals which denote "0" and "1" on the hard disk of a computer.
- The *slope* of a velocity signal $v(t)$ is the acceleration that generates dangerous "g forces" in traffic accidents and otherwise.
- *Error* signals are deviations from desired values (of room temperature, of product dimensions, etc). Positive and negative errors (too high or too low) are often equally unacceptable, and the *mean squared value* of error over an agreed time period is a useful parameter for the quality control people. We could call it *mean error power*, and then do our best to minimise it !

1.6.2 Signal Processing Tasks

The list is a very long one, but we will try to give some small impression of the tasks to be done and the reasons that motivate them.

Virtual Instrumentation. Instead of using dedicated oscilloscopes, spectrum analyzers, and other instruments, we can use an ADC to digitise a signal, then do all the processing on a computer, while using the computer screen as the instrument display panel.

Systems Control. A primary reason for monitoring a signal is so that we can make corrections, as when we measure room temperature and use the information to adjust the supply of heat. This is as simple form of *feedback control system*. Control systems abound in industrial applications.

Filtering. A filter modifies a signal by blocking some of its frequency components and allowing the rest to go through unaffected.

Noise Removal. Some filters are for the removal of unwanted *noise* signals. Narrow-band noise is often easy to remove. Broadband noise is more challenging.

Information Transfer. This covers a multitude. Even "local transfers", such as the transfer of data from a Compact Disk (CD) or Mini-disk (MD) to the music output circuitry can involve complex *line codes* and *de-compression* algorithms. Long distance transfers are made over the air waves, or through co-ax cables, or through fiber-optic cables. These involve *multiplexing* methods (in time or in frequency) so that several signals can share the same path, *modulation* methods which place the signal in a frequency band that suits the transmission medium, and *de-modulation* methods that restore the original signal at the receiver. All this applies to telephone, radio and television links, both analogue and digital, and to the world-wide

web links, which are exclusively digital.

Signal Enhancement. This includes making noisy voice recordings more intelligible, and making blurred video images more recognisable. There are many techniques, and they use digital processing for the most part.

Signal Compression. This reduces the data length with little or no damage to the content. With *lossless* compression methods, we can restore the original data exactly, but compression factors of 2 to 4 may be all we can achieve. With *lossy* compression methods, full restoration is impossible, but that is often unimportant, as with audio or video data, provided the final sound or image seems satisfactory. Lossy compression can achieve far higher compression ratios, as much as 10 to 50 times, but with some loss of quality as the ratio increases. There are two main benefits from data compression: less *space* is needed to store the data, and less *time* (and cost !) is needed to send it over a data link. Compression is sometimes a *necessity*: without it, digital television transmissions would not be viable.

Message Encryption. The internet (and private intranets) make communication easy, but they also create a need for *data security*. The answer is to *encrypt* a message before transmission, which makes it unintelligible, and to reverse the process later. The methods are mostly digital, and highly sophisticated.

1.6.3 Digital Versus Analogue.

Some of the listed operations have used *analogue* methods in the past, particularly systems-control, filtering and information transfer. Now, more and more, they use digital methods instead. We will briefly set out the reasons for the changeover.

- Analogue methods are subject to large component variations over process and over temperature, making it very difficult to achieve high precision. With digital methods, the precision is limited only by the word-length employed.
- Analogue design can be difficult and time-consuming. Digital solutions are getting less and less expensive, smaller in size, and easier to automate.
- Digital solutions have great immunity to noise. In a digital phone link, a signal will pick up noise along the way, but the binary digital data (0's and 1's) can be fully re-constituted and rendered "noise-free" again (except in extreme

high-noise conditions).

- Analogue solutions serve one purpose only, but digital solutions are highly re-configurable. The ultimate example is the computer. It re-configures itself for a new role every time we open a new application.

That leaves only a few categories in which analogue methods win out. At very *low price* levels, an analogue solution may be cheaper, and still suffice (cheap radios, etc). For very *high-speed* systems, the digital option may be too slow, and an analogue solution must be used. And finally, at the Analogue/Digital interface (in A/D and D/A converters), some analogue circuitry cannot be avoided.

1.6.4 Algorithms and Implementations.

There are two distinct parts to a problem: the *algorithm* and the *implementation*. The algorithm is the *mathematical strategy*, and it applies to both analogue and digital solutions, although the word algorithm most often refers to software.

The *implementation* refers (mainly) to the hardware employed. The resistors, capacitors, transistors, etc, of the analogue implementation give way to logic circuits in the digital implementation. The algorithm defines the result that we want, but the way that we get that result will depend on the implementation.

The algorithm is the *concept* side, and is relatively unchanging. The implementation is decided by technology, and can change quite rapidly. This book is mostly about algorithms. Our references to implementations are mostly for illustration, and to give a more practical bias to the work.

The future of electronics is mostly about information, and with less reason to separate the various audio, video, and other information systems. We will see some merging of traditional roles, and a growing reliance on computers, while commercial firms compete to define the shape of things to come.





2.1 PREAMBLE

First we show how to describe signals, both as maths expressions, and as pictures of waveforms. We explain signal symmetry. We introduce phasors, their properties, and we show how two phasors make a sine wave. We discuss sums of sine waves, and how periodic signals are built from sine waves. Then we switch to a spectral viewpoint, and we show how to find the phasors in a periodic signal. We arrive at the Discrete-time Fourier Transform, better known as Fourier Series.

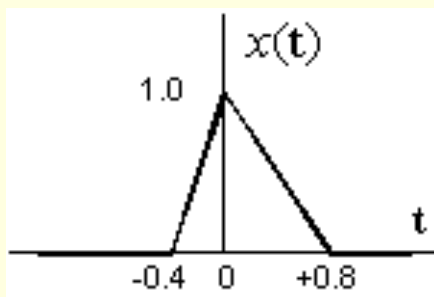




2.2 SIGNALS IN TIME DOMAIN

Now we revisit the pulses, the replication, and the periodic signals of Chapter 1, but from a more mathematical viewpoint, that allows precise descriptions.

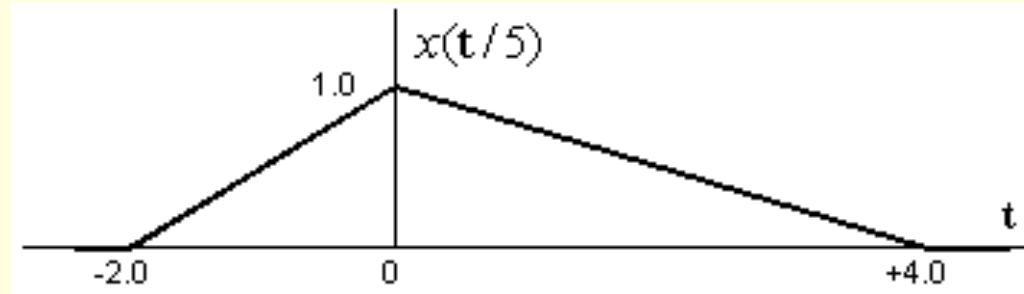
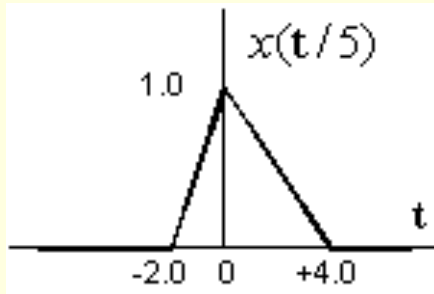
2.2.1 Sketching Signal Waveforms



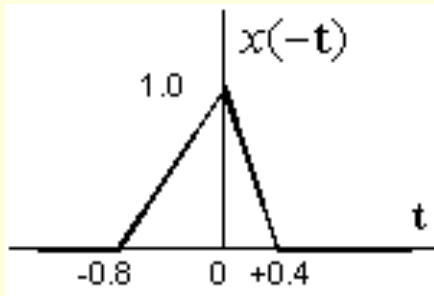
This $x(t)$ is a pulse (Fig ç) with no special symmetry. We'll show some simple ways to move it and to stretch it, both vertically and horizontally, and even to flip it around an axis. Vertical stretching is easy: the signal $5x(t)$ is 5 times taller than $x(t)$, with a peak value of 5.0, and no change in shape.

Now we'll stretch it horizontally by 5. There are two ways to show the result. Here the stretch is visible (Fig ê), but space-consuming. Here (Fig í) we've used less space, and changed only the numbers. Both methods are equally valid, but

we'll usually opt for the space-saving method.

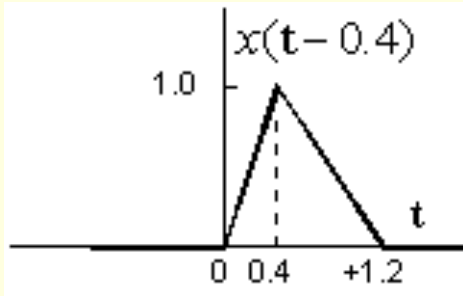


Watch how the maths label has changed. It now reads $x(t/5)$. To explain, $x(t)$ comprises a *function* $x(\quad)$ with a *parameter*, normally t , inside the brackets. We give a parameter value to $x(\quad)$, and $x(\quad)$ gives us back a function value. By changing the parameter to $t/5$, we now need a t which is 5 times larger to get the same parameter value. This stretches the x -shape by 5 on the time axis. We could compress it by 5 in a similar way by using $x(5t)$ as our new function.

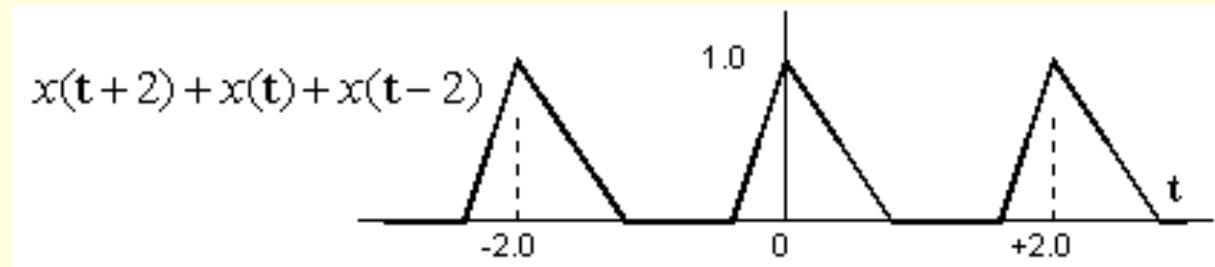


To flip $x(t)$ around its *horizontal* axis, we just take $-x(t)$. This reverses the sign but not the magnitude for all values of t . To flip $x(t)$ around its *vertical* axis, we take $x(-t)$ instead, as shown here (Fig 4). The positive- t axis becomes the negative- t axis and vice versa. This is *time-reversal*, and we will meet it occasionally.

To slide a signal along the t -axis by τ seconds, we use $x(t-\tau)$. Here we see a plot of $x(t-0.4)$, representing a 0.4-sec *time delay* (Fig 1). In $x(t-0.4)$, t has to be *greater* by 0.4 to give the same parameter value as $x(t)$, and this explains the right-shift. The function $x(t+0.4)$ performs a 0.4sec left-shift, or *time advance*.



This (Fig 1) is a plot of $x(t)$, and two copies of $x(t)$, shifted right and left by a distance of $P = 2$. It looks periodic, but there are 3 periods only.



To make it truly periodic, we must add more pulses to left and right ad infinitum. Here are a few of those pulses:

$$\dots x(t+6) + x(t+4) + x(t+2) + x(t) + x(t-2) + x(t-4) + x(t-6) \dots$$

The fully periodic signal would be:

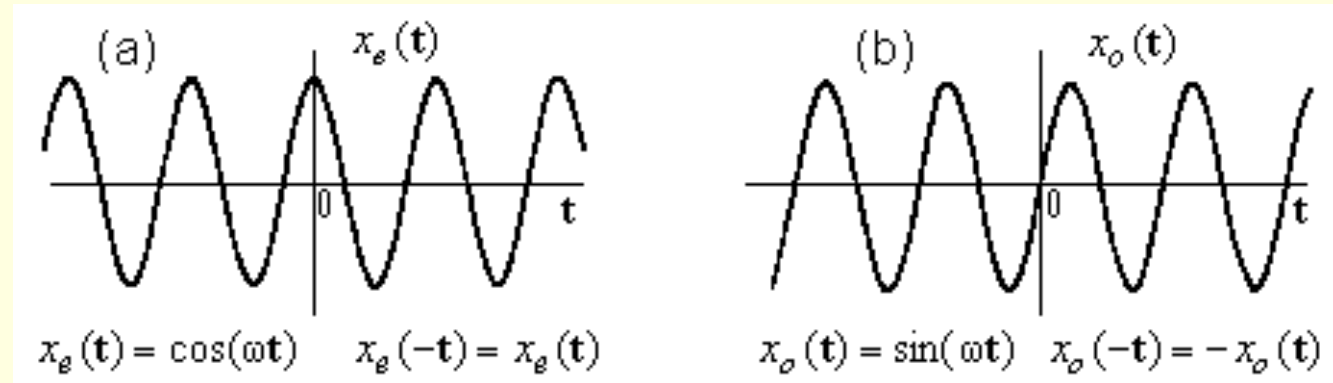
$$x(\mathbf{t})^{\sim} = \sum_{n=-\infty}^{\infty} x(\mathbf{t} - nP), \quad \text{with } P = 2$$

We called the result $x(\mathbf{t})^{\sim}$ because this is the *replicated* version of $x(\mathbf{t})$, with replication interval $P = 2$. Because P exceeds the pulse width, the replication is without overlap, and the pulse identity is not lost. By this expression, we've defined what we mean by replication, and the replicated $x(\mathbf{t})^{\sim}$ has some notable features, as we described earlier (i [Ch 1.2.3](#)).

The symbol \sim in $x(\mathbf{t})^{\sim}$ is a mark of periodicity, and most signals to follow will be periodic but, as no confusion is likely, we will revert to the simpler form $x(\mathbf{t})$.

2.2.2. Signals and Symmetry

Many signals have important symmetry properties. The familiar cosine and sine are good examples. This plot (Fig \hat{e}) shows both. Note that both describe the same sinusoidal waveform, except that "sin" lags "cos" by a quarter cycle (which is $\pi/2 = 1.57$ radians). This means that $\sin(\omega\mathbf{t}) = \cos(\omega\mathbf{t} - \pi/2)$.

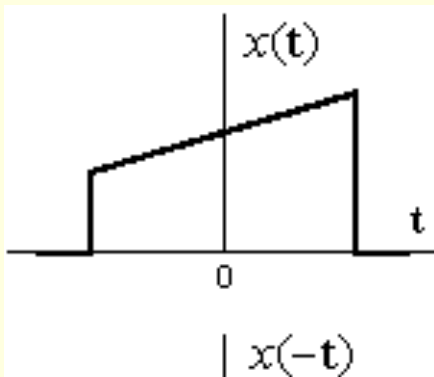


We've called the cosine wave $x_e(t)$ because of its *even* symmetry about the vertical zero-axis. We've called the sine wave $x_o(t)$ because of its *odd* symmetry about the same axis. These symmetries are defined by saying:

$$x_e(-t) = x_e(t) \quad \text{and} \quad x_o(-t) = -x_o(t)$$

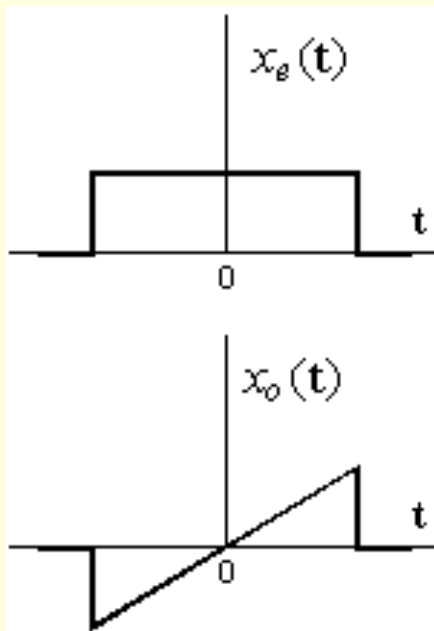
A given (real-valued) $x(t)$, such as this one (Fig 4), might have neither symmetry, but we can routinely decompose it into even and odd parts. Its even part is:

$$x_e(t) = \frac{1}{2}[x(t) + x(-t)]$$



Notice, the right-hand-side (rhs) of the equation is unchanged if we use $-t$ in place of $+t$. This proves that $x_e(-t) = x_e(t)$ as per the definition. The odd part of $x(t)$ is found as:

$$x_o(t) = \frac{1}{2}[x(t) - x(-t)]$$



Notice, the rhs of this equation is *sign-reversed* if we use $-t$ in place of $+t$. This proves that $x_o(-t) = -x_o(t)$ as per the definition.

To illustrate the method for the $x(t)$ given above (Fig 1.1), we've also shown $x(-t)$, and we've sketched the $x_e(t)$ and the $x_o(t)$ that result from our equations (Fig 1.2). All plots use the same scale factor. Take a moment to check that they make sense.

Periodic signals likewise can have even or odd symmetry. The rectangular waveform (Fig 1.1) has even symmetry if we place the $t = 0$ axis on the middle of the pulse (or midway between pulses), but not otherwise.

2.2.3 Phasors, Sines and Cosines

A vector can be described as $L\angle\theta$ meaning "length L at angle θ ". This is shorthand for the maths expression $L \cdot e^{j\theta}$, because $e^{j\theta}$ is a vector of length 1 at angle θ . A vector that rotates at some uniform rate, call it f_1 (in units of cycles per second or Hertz), is a *phasor* of frequency f_1 .

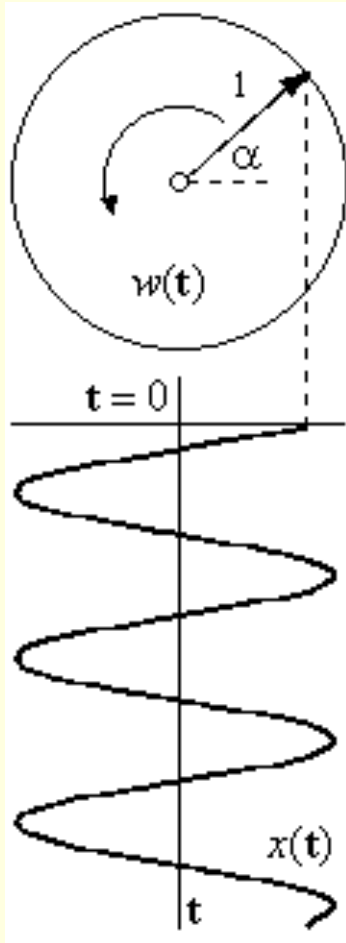
A sinusoid of frequency f_1 can be traced out as the projection of this rotating vector, or phasor, (which also explains why a rotating turbine–generator yields a sine–shaped mains voltage). It resembles a *wheel*, rotating as:

$$w(t) = e^{j(2\pi f_1 t + \alpha)}$$

where α is the initial angle at $t = 0$. The *horizontal projection* of $w(t)$, as $w(t)$ rotates, is seen here (Fig 5). This sine–shaped $x(t)$ is the *real part* of $w(t)$.

$$x(t) = \operatorname{Re}\{w(t)\} \quad \text{the real part of } w(t)$$

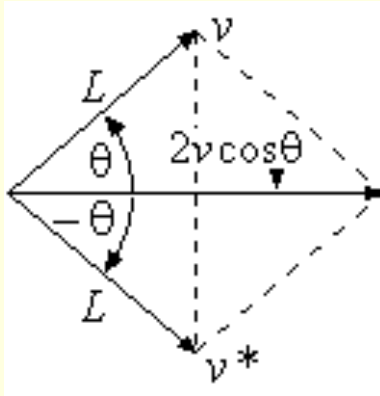
Note that the *vertical projection* is the *imaginary part* of $w(t)$, or $\operatorname{Im}\{w(t)\}$, and both parts are featured in this well–known relationship:



$$w(t) = e^{j(2\pi f_1 t + \alpha)} = \cos(2\pi f_1 t + \alpha) + j \sin(2\pi f_1 t + \alpha)$$

The cos is the real part; and the sin is the so-called imaginary part. We'll use these ideas when we talk about band-pass signals. Another way to generate a sine wave is based on the sum of a vector v and its *complex conjugate* v^* .

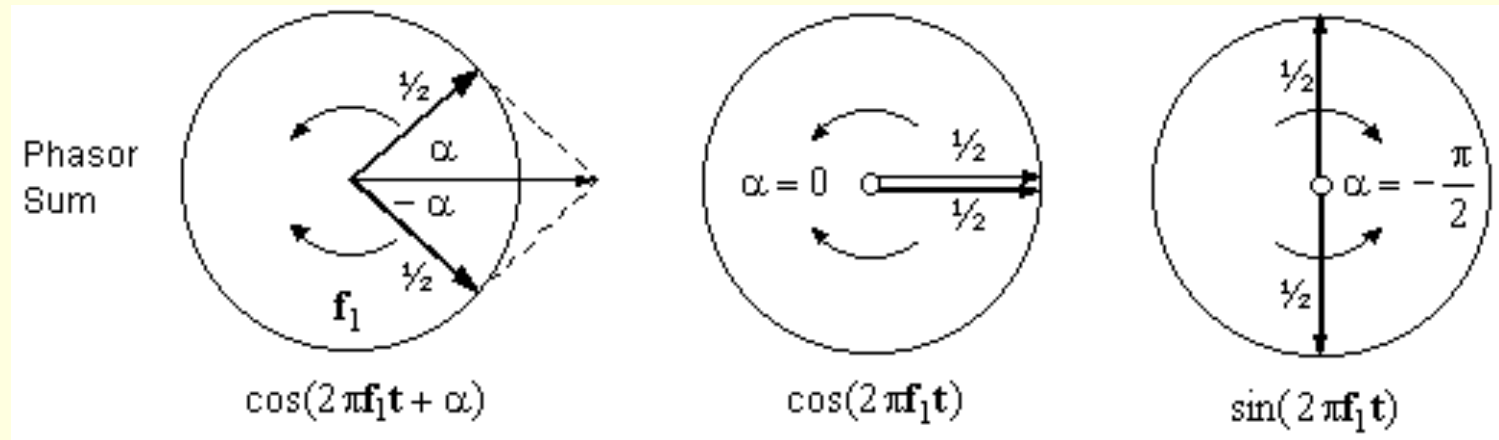
$$\text{If } v = L\angle\theta \text{ then } v^* = L\angle-\theta \text{ and } v + v^* = 2L\cos\theta$$

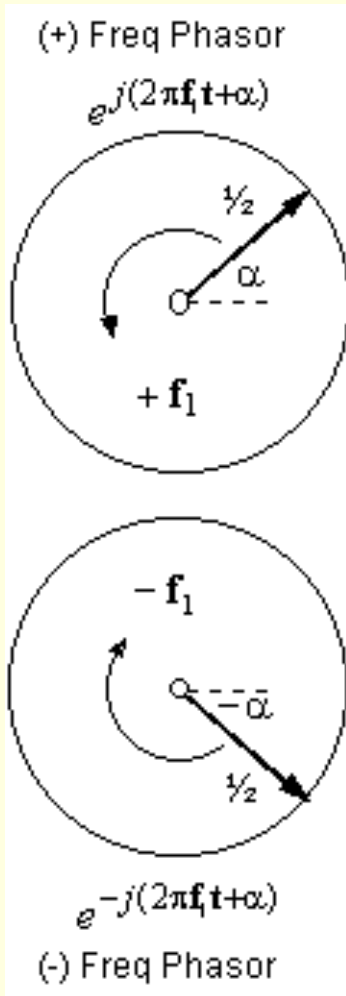


This diagram (Fig ζ) explains how it works. For a phasor, θ becomes $(2\pi f_1 t + \alpha)$, and it increases uniformly over time. We can then generate $x(t)$ as:

$$x(t) = \frac{1}{2} \cdot e^{j(2\pi f_1 t + \alpha)} + \frac{1}{2} \cdot e^{-j(2\pi f_1 t + \alpha)} = \cos(2\pi f_1 t + \alpha)$$

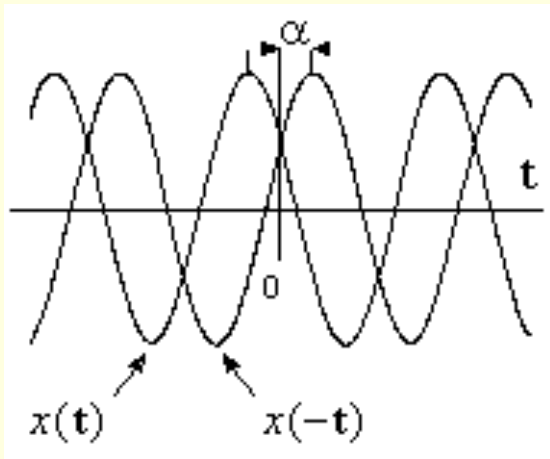
This combines two phasors, both of length $\frac{1}{2}$, to form a unit-amplitude sinusoid. The first phasor rotates (positive, anti-clockwise) at frequency $+f_1$ with initial angle of $+\alpha$. The second rotates (negative, clockwise) at frequency $-f_1$ with initial angle of $-\alpha$. This diagram (Fig ζ) shows the individual phasors. Below (Fig ι) we see how the phasor sum is real-valued (horizontal), and its *length* is the *height* of the sinusoid at that same instant.





This phasor sum (Fig é) is the general sinusoid, and can have any value of initial angle α . The signals "cos" and "sin" (Fig 2.2.2...1) are special cases. Setting $\alpha = 0$ gives us $\cos(2\pi f_1 t)$. Its vectors are horizontal at $t = 0$ (Fig é), and they add to 1.0, the cosine peak at that same instant. Setting $\alpha = -\pi/2$ gives us $\sin(2\pi f_1 t)$. Its vectors are vertical at $t = 0$ (Fig é), and they add to 0.0, which marks the *positive-going* zero-crossing of the sine at that instant. We prefer to write $\sin(2\pi f_1 t)$ as $\cos(2\pi f_1 t - \pi/2)$, since this identifies the angle correctly. That makes the "sin" somewhat redundant.

We've shown how signals can be advanced or delayed, or even flipped about the time axis. Its of some interest to see how a sine wave would respond. We'll take time reversal first. Suppose $x(t) = \cos(2\pi f_1 t + \alpha)$. Then $x(-t) = \cos(2\pi f_1 (-t) + \alpha)$ which is the same as saying $x(-t) = \cos(2\pi f_1 t - \alpha)$ because the cos is *even* ($\cos(-\theta) = \cos\theta$). *The effect of time reversal is to change the sign of α .* This plot offers graphical confirmation of that idea (Fig ë).

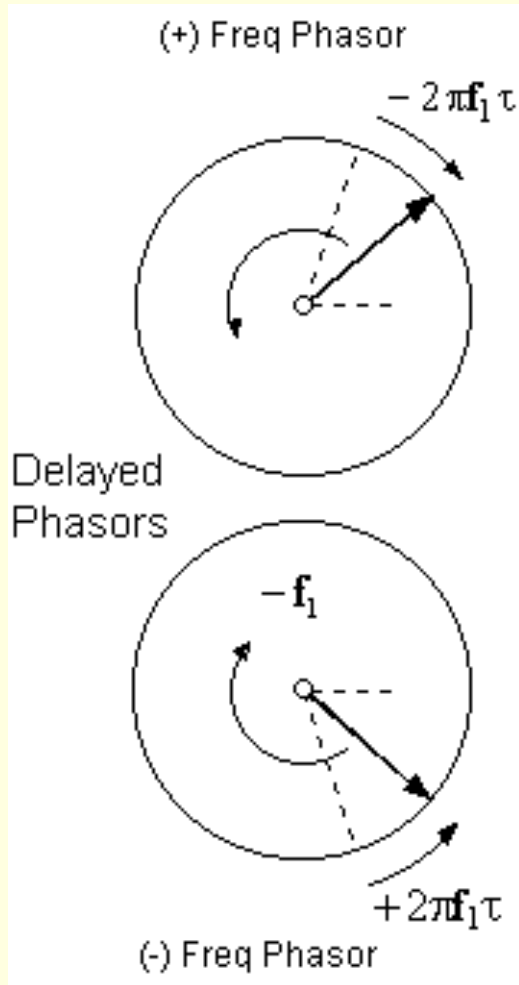


To delay a sine wave $x(t)$ by τ seconds, we must wind *back* both phasors by a corresponding angle, going *against* their direction of rotation (Fig c). At frequency f_1 , the delay angle is $f_1\tau$ cycles, which is $2\pi f_1\tau$ radians. The result is $x(t - \tau)$, expressed as:

$$x(t - \tau) = \frac{1}{2} \cdot e^{j(2\pi f_1 t + \alpha - 2\pi f_1 \tau)} + \frac{1}{2} \cdot e^{-j(2\pi f_1 t + \alpha - 2\pi f_1 \tau)}$$

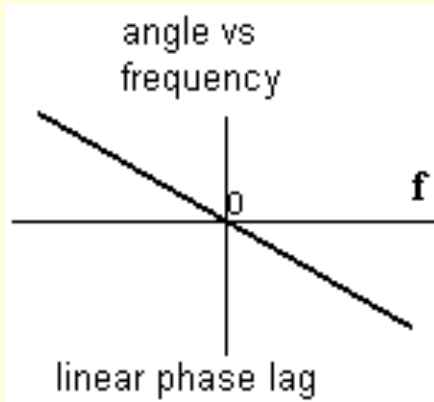
The angular change is $-2\pi f_1 \tau$ for the $+f_1$ phasor and is $+2\pi f_1 \tau$ for the $-f_1$ phasor. If we delayed several sines of different frequencies by the same τ secs, the delay angles would be different for each, and would be *linearly proportional* to the frequency of each. A plot of delay angle versus frequency would then look like this (Fig í), a *linear phase lag*. This is very important, because it happens when we delay *any* signal $x(t)$, whatever its shape. The $x(t)$ may be a sum of many sine waves, and all are delayed by different angles, but, if we have *linear phase lag*, the shape of $x(t)$ will not be distorted — which is often our major concern.

To illustrate, suppose $x(t)$ has 50Hz and 100Hz components:



$$x(t) = \cos(2\pi 50t) + 0.9 \cos(2\pi 100t)$$

A 5-msec delay is $\frac{1}{4}$ cycle at 50Hz or $\frac{1}{2}$ cycle at 100Hz, and the delayed signal becomes:

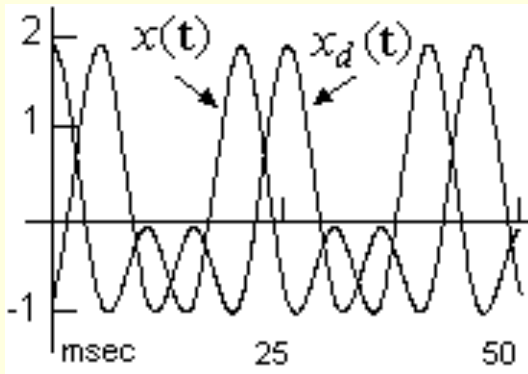


$$x_d(t) = \cos(2\pi 50t - \pi/2) + 0.9 \cos(2\pi 100t - \pi)$$

We can leave it like this, or re-write it as:

$$x_d(t) = \sin(2\pi 50t) - 0.9 \cos(2\pi 100t)$$

Either way, $x(t)$ and its delayed version $x_d(t)$ look like this (Fig c). There is no change in shape. Signal integrity has been preserved.



2.2.4 Sums of Sine Waves

We can build arbitrary signals from sine waves. The following is a sum of N sine waves, all of different amplitudes M_k , frequencies f_k , and angles α_k :

$$x(t) = \sum_{k=1}^N M_k \cdot \cos(2\pi f_k t + \alpha_k)$$

Almost any result is possible, but if we limit our options, it becomes more interesting. One way is to use the *same* frequency f_1 for all.

$$x(t) = \sum_{k=1}^N M_k \cdot \cos(2\pi f_1 t + \alpha_k)$$

The magnitudes and phase angles are still arbitrary, but the sum $x(t)$ is far from arbitrary. It is still a sine wave, and of frequency f_1 . We can find its peak value M_x and its angle α_x from the relationship:

$$M_x \angle \alpha_x = \sum_{k=1}^N M_k \angle \alpha_k$$

To understand why, recall that the phasors which make up the sine waves are all rotating together at the same frequency. Relative to one another, they are stationary. That's why we can add them all as vectors to get this result. In strict maths notation, we must replace $\angle\theta$ by $e^{j\theta}$ everywhere. This is the method that we use in AC circuit theory. We can treat all voltages and currents as vectors, because they all have the same frequency, and when they add they retain their sinusoidal shape. You can't say that about other waveforms (square, triangle, etc).

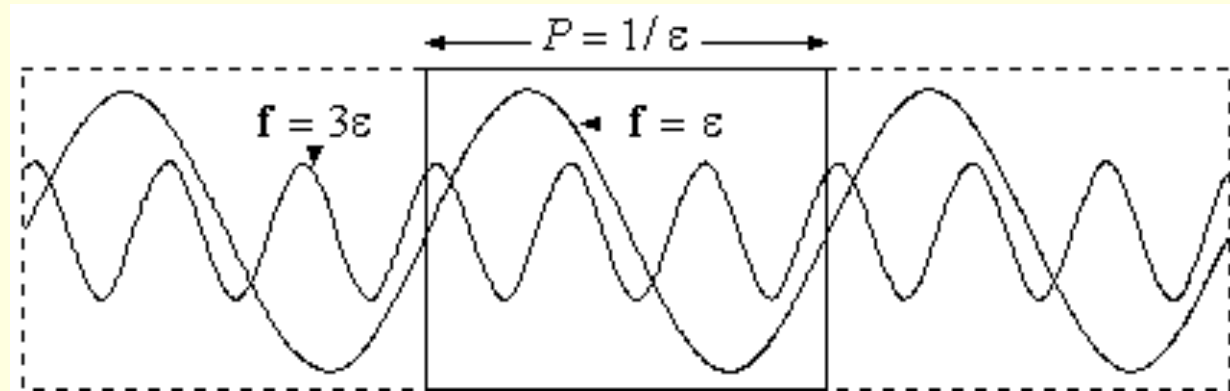
Moving on from just one frequency, we will allow different frequencies \mathbf{f}_k , but with this constraint:

$$\mathbf{f}_k = k \cdot \varepsilon, \quad k = 0, 1, 2, 3, \dots$$

The permitted frequencies are integer multiples of ε , where ε is the *fundamental* frequency. The result is a *periodic* $x(\mathbf{t})$ of period P such that:

$$P = 1/\varepsilon$$

The sinusoid of frequency $\mathbf{f}_k = k \cdot \varepsilon$ is called the k -th *harmonic* component. The shape of $x(\mathbf{t})$ is decided by the M_k and the αk of the various harmonics, noting that rapid changes in $x(\mathbf{t})$ call for high harmonic frequencies. It turns out that *any* periodic $x(\mathbf{t})$ of period P can contain the harmonic frequencies $\mathbf{f}_k = k \cdot \varepsilon$, where $\varepsilon = 1/P$, and *only those frequencies*. This diagram (Fig 6) explains why. It shows the first and third harmonic components of an $x(\mathbf{t})$ that repeats itself in windows of width P . The third harmonic has a period of $P/3$, but more importantly, *it is also periodic in P* , that is, it repeats identically in successive windows. This is essential for periodicity, and only harmonic frequencies have this property.



Notice, we allowed for a $k = 0$ term. This is a zero-frequency or DC component. It's action is to raise or lower the waveform by a fixed amount, and that does not affect the periodicity.





2.3 SIGNALS IN FREQUENCY DOMAIN

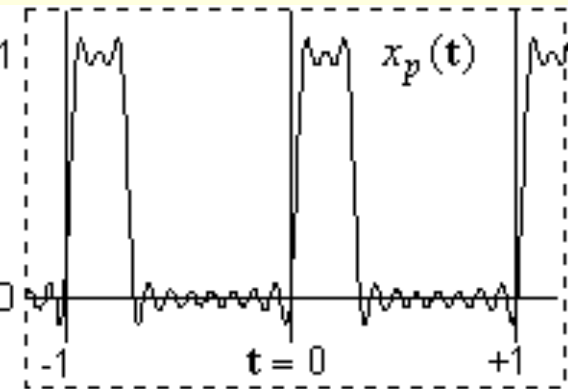
We can assemble a periodic $x(t)$ of arbitrary shape by specifying its various harmonic components. The shape of $x(t)$ over a period is the *time-domain* view. The list of harmonics (M_k, α_k) , is an alternative view, a *frequency-domain* view, or a *spectral* view, which we will now present.

2.3.1 Periodic Signal Spectra

This table (Fig 2.3.1) is a list of (M_k, α_k) values up to the tenth harmonic. We can build the waveform that it describes as:

k	M_k	α_k
0	0.250	0
1	0.450	-45
2	0.318	-90
3	0.150	-135
4	0.000	-180
5	0.090	-45
6	0.106	-90
7	0.064	-135
8	0.000	-180

9	0.050	-45
10	0.064	-90

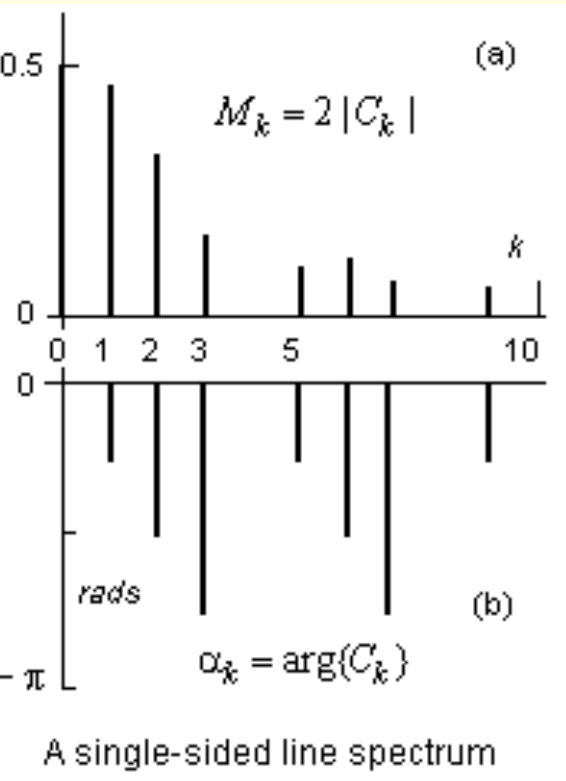


$$x(t) = \sum_{k=0}^{10} M_k \cdot \cos(2\pi(k \cdot \epsilon)t + \alpha_k)$$

When we do this, we get the waveform shown here (Fig ). It resembles a rectangular waveform of amplitude 1.0 and period $P = 1.0$, with pulse-width 0.25. The waveform repetition frequency is $\epsilon = 1$ Hz. The highest frequency present is 10 Hz, the 10-th harmonic, and this limits the available steepness at pulse edges.

The table of (M_k, α_k) values is more often seen pictorially as a *line spectrum* such as the *single-sided* line spectrum shown here (Fig ). We have a set of line lengths describing M_k values, and a separate set for α_k values (now in radians), with frequencies identified by k on the horizontal axis. Each (M_k, α_k) pair describes one of the sine waves that make up the signal.

But the sine is not the most basic signal element. Each sinusoid is a sum of two conjugate phasors. The phasor of frequency $+f_k = k \cdot \epsilon$ can be represented as:



$$C_k = \frac{1}{2}M_k \angle \alpha_k = \frac{1}{2}M_k e^{j\alpha_k}$$

The phasor of frequency $-\mathbf{f}_k = -k \cdot \epsilon$ can be represented as:

$$C_{-k} = \frac{1}{2}M_k \angle -\alpha_k = \frac{1}{2}M_k e^{-j\alpha_k}$$

Notice how $|C_{-k}| = |C_k| = \frac{1}{2}M_k$, but $\arg\{C_{-k}\} = -\arg\{C_k\} = -\alpha_k$, where "arg $\{C_k\}$ " just means "the angle of C_k ". The k -th harmonic becomes:

$$x_k(t) = M_k \cos(2\pi k \epsilon t + \alpha_k) = C_k \cdot e^{j2\pi k \epsilon t} + C_{-k} \cdot e^{-j2\pi k \epsilon t}$$

There may be more here than meets the eye. M_k and α_k are real numbers, but C_k and C_{-k} are complex numbers whose angles give the phasors their initial phase. We could also write this k -th harmonic as:

$$x_k(t) = |C_k| \cdot e^{j2\pi k \epsilon t} \cdot e^{j \arg\{C_k\}} + |C_{-k}| \cdot e^{-j2\pi k \epsilon t} \cdot e^{j \arg\{C_{-k}\}}$$

The angles combine by addition:

$$x_k(t) = |C_k| \cdot e^{j(2\pi k \epsilon t + \arg\{C_k\})} + |C_{-k}| \cdot e^{-j(2\pi k \epsilon t - \arg\{C_{-k}\})}$$

We can eliminate C_{-k} by using $|C_{-k}| = |C_k|$ and $\arg\{C_{-k}\} = -\arg\{C_k\}$:

$$x_k(t) = |C_k| \cdot e^{j(2\pi k \epsilon t + \arg\{C_k\})} + |C_k| \cdot e^{-j(2\pi k \epsilon t + \arg\{C_k\})}$$

The $|C_k|$ is now common to both, and since $(e^{j\theta} + e^{-j\theta}) = 2\cos\theta$, we get:

$$x_k(\mathbf{t}) = 2|C_k| \cdot \cos(2\pi k \varepsilon \mathbf{t} + \arg\{C_k\})$$

This is the k -th harmonic. The complete $x(\mathbf{t})$ is the sum of the DC term and all of the harmonics, resulting in:

$$x(\mathbf{t}) = C_0 + \sum_{k=1}^{\infty} 2|C_k| \cdot \cos(2\pi k \varepsilon \mathbf{t} + \arg\{C_k\})$$

Or, equivalently, we could just add all the phasors to give:

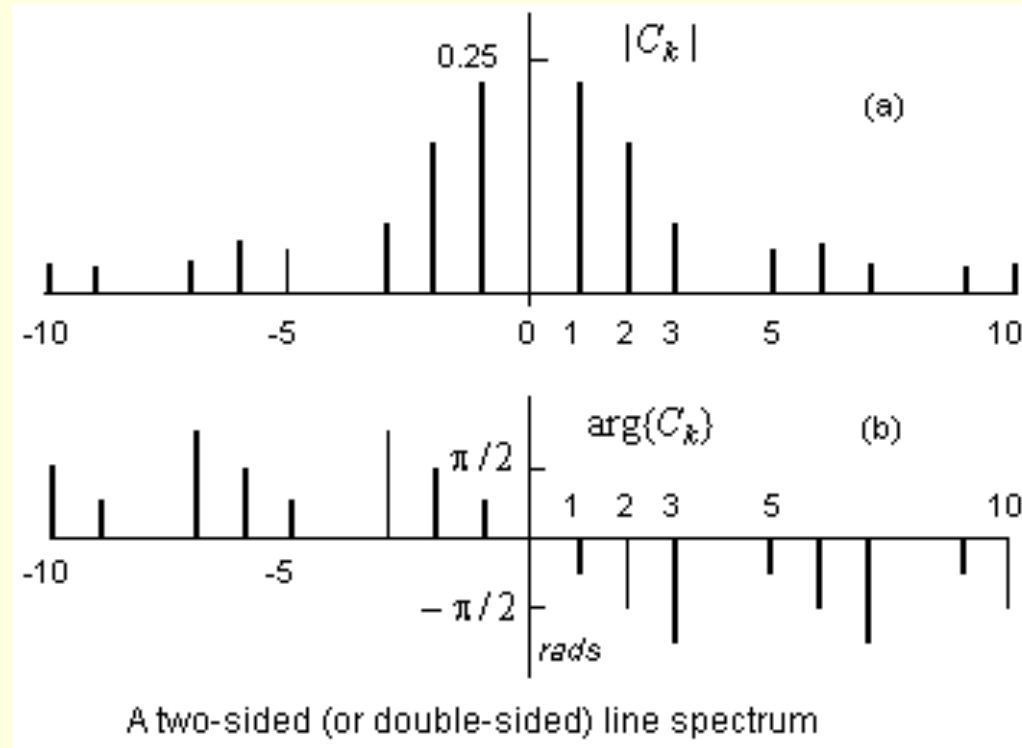
$$x(\mathbf{t}) = \sum_{k=-\infty}^{\infty} C_k \cdot e^{j(2\pi k \varepsilon \mathbf{t})}$$

This very compact form includes the DC term C_0 at $k = 0$, while the phasors combine in pairs at $k = \pm 1$, $k = \pm 2$, $k = \pm 3$, etc to build the sine-wave components. Note that $C_0 = M_0$ is the (real) DC level, but all other C_k are complex, with only half the length of M_k . We can have any number of harmonics, so we've extended the sums to infinity.

If we think in terms of phasors, then we will display the phasors on a *double-sided spectrum*, such as that shown below (Fig 6), for the same rectangular waveform as before. This spectrum shows the negative frequencies too, and some symmetry rules apply. Because $|C_k| = |C_{-k}|$, the *magnitude* plot shows *even* symmetry about the vertical zero-axis. Because $\arg\{C_{-k}\} = -\arg\{C_k\}$, the *phase* plot shows *odd* symmetry instead. The spectra of *real-valued* signals will always have

these symmetries. That makes it frequently unnecessary to use a two-sided spectrum, since one side tells us all that we need to know. But there are advantages in doing so, and this will be more apparent as we progress.

k	M_k	α_k
0	0.250	0
1	0.450	-45
2	0.318	-90
3	0.150	-135
4	0.000	-180
5	0.090	-45
6	0.106	-90
7	0.064	-135
8	0.000	-180
9	0.050	-45
10	0.064	-90



Returning to our table of spectral data (Fig 8), we note that the $|C_k|$ line lengths are just half the M_k values. We must also wonder how the M_k and α_k , or the corresponding C_k values, were chosen. This is the problem of finding one phasor in a signal build from many phasors, which we will now address.

2.3.2 Testing For Phasors

We'll start with phasor integration, by observing that:

$$\int_{t_0}^{t_0+P} e^{j2\pi k\epsilon t} dt = 0, \quad \text{when } P = 1/\epsilon$$

We've integrated a phasor over k phasor cycles, for integer k , and the result is zero. This is always true when the integration covers an exact number of cycles, regardless of the starting point, t_0 . This is intuitively correct, but we may also note that:

$$e^{j\theta} = \cos\theta + j\sin\theta$$

This reminds us that k cycles of the phasor equates to k cosine periods for its real part and k sine periods for its imaginary part. Both parts separately integrate to zero over any k -cycle interval — for integer values of k .

Suppose we want to find a phasor of frequency $+m\cdot\epsilon$ in a periodic $x(t)$ of period $P = 1/\epsilon$. Our supposition is that $x(t)$ may have a phasor of this frequency, with phasor length L_m and initial angle α_m , but this $C_m = L_m\angle\alpha_m$ is as yet unknown to us, and we know that $x(t)$ contains many other phasors as well. Therefore:

$$x(t) = L_m \cdot e^{j\alpha_m} \cdot e^{+j2\pi m\epsilon t} + \text{other phasors}$$

We propose the following test to find C_m :

$$C_m = \frac{1}{P} \int_{t_0}^{t_0+P} x(t) \cdot e^{-j2\pi m\epsilon t} dt$$

To find a phasor of frequency $+m \cdot \epsilon$ in $x(t)$, the strategy is to multiply it by a phasor of frequency $-m \cdot \epsilon$, and then integrate over $P = 1/\epsilon$. We get:

$$C_m = \frac{1}{P} \int_{t_0}^{t_0+P} \left[L_m \cdot e^{j\alpha_m} \cdot e^{j2\pi m\epsilon t} + \text{other} \right] \cdot e^{-j2\pi m\epsilon t} dt$$

In place of "other", we'll use a phasor of some other frequency $k \cdot \epsilon$.

$$C_m = \frac{1}{P} \int_{t_0}^{t_0+P} \left[L_m \cdot e^{j\alpha_m} \cdot e^{j2\pi m\epsilon t} + L_k \cdot e^{j\alpha_k} \cdot e^{j2\pi k\epsilon t} \right] \cdot e^{-j2\pi m\epsilon t} dt$$

yielding:

$$C_m = \frac{1}{P} \int_{t_0}^{t_0+P} \left[L_m \cdot e^{j\alpha_m} \cdot e^{j2\pi(m-m)\varepsilon t} + L_k \cdot e^{j\alpha_k} \cdot e^{j2\pi(k-m)\varepsilon t} \right] dt$$

The phasors rotating at $+m \cdot \varepsilon$ and at $-m \cdot \varepsilon$ (Hz) are now "frozen" by multiplication into a stationary vector, and we get:

$$C_m = \frac{1}{P} \int_{t_0}^{t_0+P} \left[L_m \cdot e^{j\alpha_m} + L_k \cdot e^{j\alpha_k} \cdot e^{j2\pi(k-m)\varepsilon t} \right] dt$$

The first term returns $L_m \cdot e^{j\alpha_m}$, which is precisely what we want, but it will work only if the "other" term is zero. If $x(t)$ were just any signal, it could not work, and the test would fail. But $x(t)$ is periodic with period P , and the only frequencies that *can* occur are integer multiples of $\varepsilon = 1/P$. This has the effect that k and m must be integers, so that $(k - m)$ is an integer too. Therefore, the "other" term is assured to cover an integer number of phasor cycles, and the result will be zero every time. We can conclude that:

$$C_m = \frac{1}{P} \int_{t_0}^{t_0+P} x(t) \cdot e^{-j2\pi m \varepsilon t} dt = L_m \cdot e^{j\alpha_m}$$

It identifies this C_m , and we can similarly find all other C_k values which define the spectrum of a periodic $x(t)$ of period $P = 1/\varepsilon$. We now have a pair of "Transform Relationships" which link $x(t)$ with C_k , as follows:

$$C_k = \frac{1}{P} \int_{t_0}^{t_0+P} x(t) \cdot e^{-j2\pi k \epsilon t} dt \quad \text{and} \quad x(t) = \sum_{k=-\infty}^{\infty} C_k \cdot e^{j2\pi k \epsilon t}$$

Forward Transform

Inverse Transform

Through the "Forward Transform" we get the spectral data (a set of complex numbers C_k) that describe a known $x(t)$. Through the "Inverse Transform", we can build the continuous periodic $x(t)$ if we know its spectral numbers C_k .

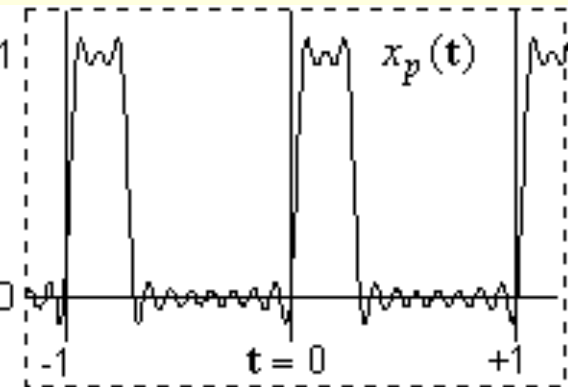
2.3.3 Rectangular Waveform Spectrum

As a first test of the method, we'll return to our sample rectangular waveform and we'll try to find its C_k spectrum:

$$C_k = \frac{1}{P} \int_{t_0}^{t_0+P} x(t) \cdot e^{-j2\pi k \epsilon t} dt = \int_0^1 x(t) \cdot e^{-j2\pi k \epsilon t} dt$$

The $x(t)$ period is $P = 1$, which gives $\epsilon = 1$ also. It has value 1 for $(0 < t < 1/4)$ and is zero for the rest of the period. Therefore:

$$C_k = \int_0^{1/4} 1.0 \cdot e^{-j2\pi kt} dt = \frac{1}{-j2\pi k} \left[e^{-j2\pi kt} \right]_{t=0}^{t=1/4} = \frac{1}{j2\pi k} \left[1 - (-j)^k \right]$$



We wrote $e^{-j\pi/2}$ as $(-j)$ in this result. (They both indicate $1.0\angle-\pi/2$). It is now a simple matter to check that these C_k agree with the numbers in our $(M_k, \alpha k)$ table. Remember, the M_k are twice the $|C_k|$, but the angles are the same for both.

We've also seen how these C_k build a good approximation to the desired waveform (Fig 5), so we can start to gain confidence in this new way of working with signals. But, before we leave it, we want to formally define the "Transform" method that we've uncovered. We also want to "stand back" from all the details, so that we can appreciate the full significance of this result.

2.3.3 The DfFT (or Fourier Series)



We've just described a transform mechanism that is commonly known as Fourier Series. It builds a periodic $x(t) \sim$ as a sum of sinusoids:

$$x(\mathbf{t})^{\sim} = C_0 + \sum_{k=1}^{\infty} 2|C_k| \cos(2\pi k\epsilon \mathbf{t} + \arg\{C_k\})$$

and it shows us how to identify the C_k values, commonly known as *Fourier Series coefficients*. The above summation is convenient for building $x(\mathbf{t})^{\sim}$, but as a general statement of the transform and its inverse, we will use our earlier description (i Eqn xx, but including \sim to show periodicity) :

$$C_k = \frac{1}{P} \int_{t_0}^{t_0+P} x(\mathbf{t})^{\sim} \cdot e^{-j2\pi k\epsilon \mathbf{t}} \cdot d\mathbf{t} \quad \text{and} \quad x(\mathbf{t})^{\sim} = \sum_{k=-\infty}^{\infty} C_k \cdot e^{j2\pi k\epsilon \mathbf{t}}$$

Forward DfFT

Inverse DfFT

This implies that we can use $x(\mathbf{t})^{\sim}$ to find C_k and then use C_k to rebuild the same $x(\mathbf{t})^{\sim}$. There are a few subtleties, but this is indeed the case, and therefore we have a "Transform pair", a way of working interchangeably with a signal in two domains. We like to think of $x(\mathbf{t})^{\sim}$ as a *real-valued* signal. A complex $x(\mathbf{t})^{\sim}$ seems more abstract, but it is allowable. When $x(\mathbf{t})^{\sim}$ is complex, the C_k coefficients lose their symmetry. Note that a complex $x(\mathbf{t})^{\sim}$ is just a sum of two real signals which have been "packaged" as one complex signal: $x(\mathbf{t})^{\sim} = x_r(\mathbf{t})^{\sim} + j x_i(\mathbf{t})^{\sim}$.

The *frequency-domain* view (the *spectrum*) gives important insights that are not apparent in the time domain — but the converse is also true, so we should normally draw our information from both domains.

This is a transform that links a *continuous* periodic signal in one domain with a *discrete* set of numbers in the other domain. The two descriptions are interchangeable. This is a Discrete-frequency Fourier Transform, or DfFT for short. More generally, our data can be continuous (analogue) or discrete (numeric) in both domains, and the data "shape" can be either pulse-like or periodic. To cover all of these possibilities, we'll need some more Fourier Transforms, and that will be our work for the next chapter.





3.1 PREAMBLE

This is the keynote chapter. It develops the transforms that we need for continuous (analogue) data and for discrete (sampled) data, and for waveforms that are either pulse-shaped or repetitive (periodic). It is condensed, and needs careful reading, and it lays the groundwork for virtually everything that follows.





3.2 THE CONTINUOUS FOURIER TRANSFORM (CFT)

At present, we have a DfFT (discrete in frequency), also called Fourier Series. It links a periodic time signal $x(t) \sim$, of period P , to a set of spectral samples C_k (the Fourier Series coefficients), at sample interval $\epsilon = 1/P$, in Hertz. We will use it to derive the CFT (continuous in both domains), that links a pulse in time to a pulse-shaped spectrum.

3.2.1 From DfFT to CFT

The DfFT (using $\epsilon = 1/P$) took the form:

$$C_k = \frac{1}{P} \int_{t_0}^{t_0+P} x(t) \sim \cdot e^{-j2\pi k\epsilon t} \cdot dt \quad \text{and} \quad x(t) \sim = \sum_{k=-\infty}^{\infty} C_k \cdot e^{j2\pi k\epsilon t}$$

Forward DfFT

Inverse DfFT

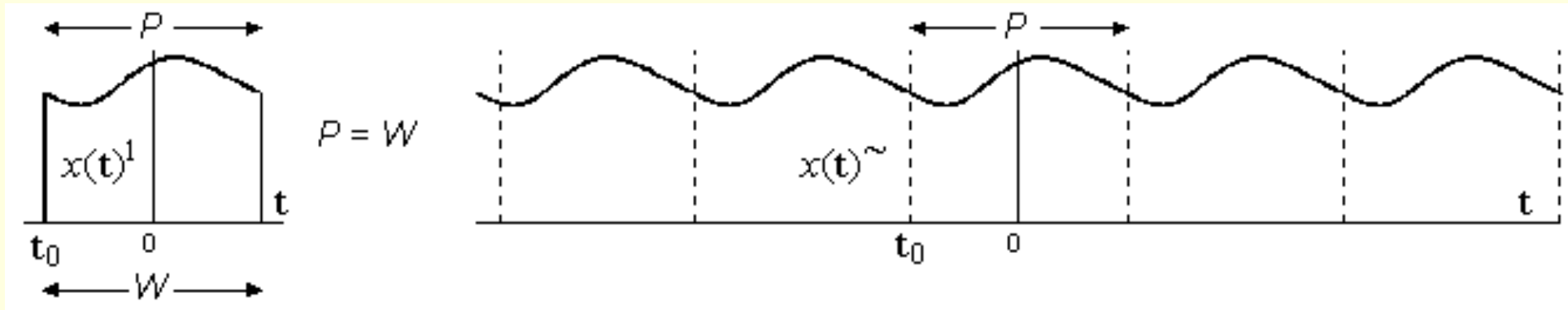
We've included \sim in $x(\mathbf{t})\sim$ to mark its *periodicity*. In this chapter, the simpler $x(\mathbf{t})$ will mean a *pulse-shaped* signal. The forward DfFT integral above refers to the harmonic frequencies $\mathbf{f} = k\varepsilon$, and we can think of C_k as being samples from a spectral function $X(\mathbf{f})$ taken at the frequencies $\mathbf{f} = k\varepsilon$, where:

$$X(\mathbf{f}) = \int_{\mathbf{t}_0}^{\mathbf{t}_0+P} x(\mathbf{t})\sim \cdot e^{-j2\pi\mathbf{f}\mathbf{t}} \cdot d\mathbf{t}$$

and then:

$$C_k = \varepsilon \cdot X(k \cdot \varepsilon) \quad \text{with } \varepsilon = 1/P$$

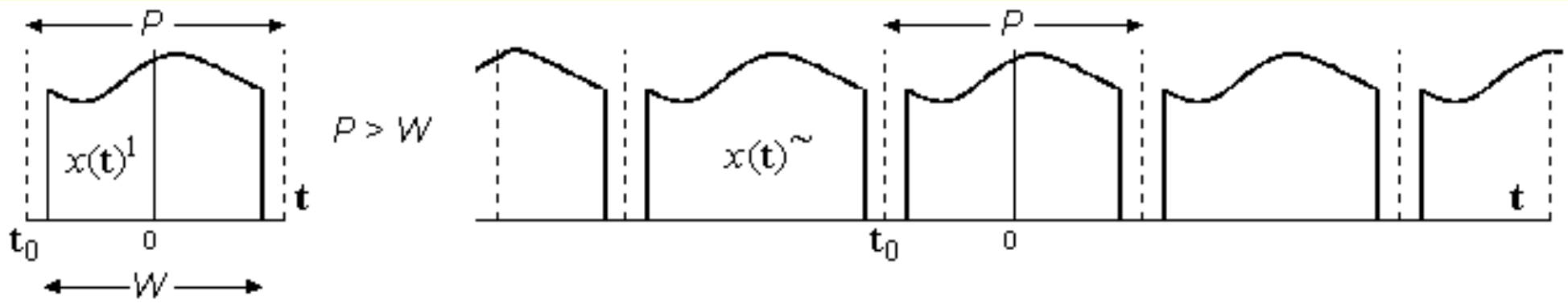
These C_k are *area-samples* of $X(\mathbf{f})$ (value-samples $X(k\varepsilon)$ \times sample-spacing ε). $X(\mathbf{f})$ is the result of integration over a one-period segment of $x(\mathbf{t})\sim$. This segment is a *pulse* which we will name $x(\mathbf{t})^1$, as illustrated here (Fig 1). It is clear that $x(\mathbf{t})\sim$ is just a replication of these $x(\mathbf{t})^1$ pulses with spacing P . We'll refer to the replicated pulse set as $x(\mathbf{t})^1\sim$.



It follows that:

$$X(\mathbf{f}) = \int_{t_0}^{t_0+P} x(t)^\sim \cdot e^{-j2\pi\mathbf{f}t} \cdot dt = \int_{t_0}^{t_0+P} x(t)^1 \cdot e^{-j2\pi\mathbf{f}t} \cdot dt$$

At present, the pulse spacing P (or the *replication interval*) is also the width W of the $x(t)^1$ pulse in the diagram. We could increase the interval P between the $x(t)^1$ pulses, and our $x(t)^\sim$ will change accordingly as shown (Fig 1).



The C_k will, of course, change also. But, *we will still have the same $X(\mathbf{f})$* . That's because the empty space beyond the pulse adds nothing to the $X(\mathbf{f})$ integral. The new C_k samples are different only because of their reduced spacing ($\epsilon = 1/P$) on the same $X(\mathbf{f})$ waveform. We can carry this to the extreme by letting $P \rightarrow \infty$, then watching what happens to $C_k = \epsilon \cdot X(k\epsilon)$:

$$X(k\epsilon) = \int_{t_0}^{t_0+P} x(t)^{1\sim} \cdot e^{-j2\pi k\epsilon t} \cdot dt$$

The pulses move further apart and the integration range expands, while the spectral sample-spacing $\epsilon \rightarrow 0$, and the samples $X(k\epsilon)$ crowd together into a continuum over $X(\mathbf{f})$, causing \mathbf{f} to replace $k\epsilon$ in the limit. The central $x(t)^1$ pulse on the $t = 0$ axis remains, but all of its replicas are removed to infinity, and then $X(\mathbf{f})$ becomes this pulse's spectrum:

$$X(\mathbf{f}) = \int_{-\infty}^{\infty} x(t)^1 \cdot e^{-j2\pi \mathbf{f} t} \cdot dt$$

Meanwhile, the harmonics crowd ever more closely together:

$$x(\mathbf{t})^{\sim} = \sum_{k=-\infty}^{\infty} C_k \cdot e^{j2\pi k\varepsilon t} = \sum_{k=-\infty}^{\infty} X(k \cdot \varepsilon) \cdot e^{j2\pi k\varepsilon t} \cdot \varepsilon$$

As $k \cdot \varepsilon \rightarrow \mathbf{f}$, this summation becomes an integral, and all that remains of $x(\mathbf{t})^{\sim}$ is the central $x(\mathbf{t})^1$ pulse:

$$x(\mathbf{t})^1 = \int_{-\infty}^{\infty} X(\mathbf{f}) \cdot e^{j2\pi \mathbf{f} \mathbf{t}} \cdot d\mathbf{f}$$

Combining our two results:

$$X(\mathbf{f}) = \int_{-\infty}^{\infty} x(\mathbf{t})^1 \cdot e^{-j2\pi \mathbf{f} \mathbf{t}} \cdot d\mathbf{t} \quad \text{and} \quad x(\mathbf{t})^1 = \int_{-\infty}^{\infty} X(\mathbf{f}) \cdot e^{j2\pi \mathbf{f} \mathbf{t}} \cdot d\mathbf{f}$$

Forward CFT

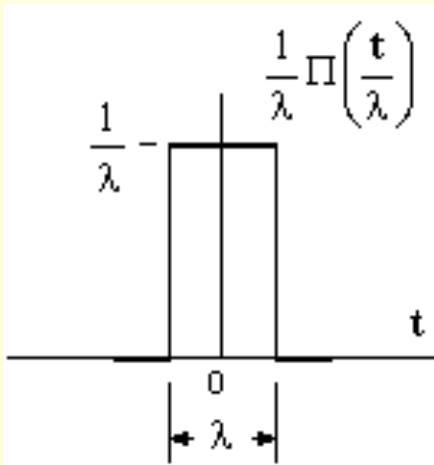
Inverse CFT

This is a new transform, a Continuous Fourier Transform (CFT), and it links a pulse-shaped time signal $x(\mathbf{t})^1$ to a pulse-shaped spectrum $X(\mathbf{f})$. The CFT is notable for its symmetry: the two integrals are almost identical, with only $(-j)$ versus $(+j)$ to separate the Forward from the Inverse. It might seem that the DfFT doesn't share the CFT's symmetry, but that can change with the way we write it:

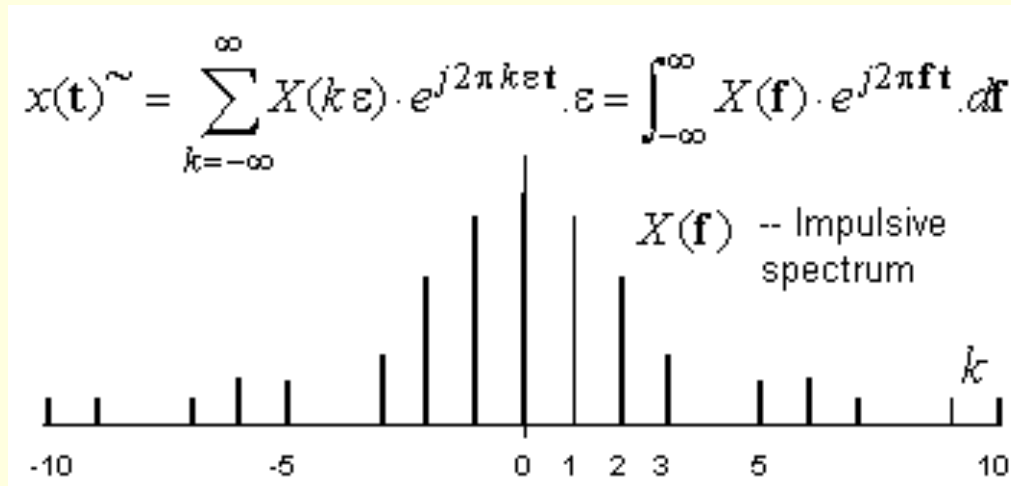
$$X(k \cdot \varepsilon) = \int_{t_0}^{t_0+P} x(t) \cdot e^{-j2\pi k \varepsilon t} dt \quad \text{and} \quad x(t) = \sum_{k=-\infty}^{\infty} X(k \cdot \varepsilon) \cdot e^{j2\pi k \varepsilon t} \cdot \varepsilon$$

Forward DfFT

Inverse DfFT



This has better symmetry, although the inverse DfFT is still numeric, it *sums* over a set of *numbers*, very different from *integration* over a *function*. But we can make even this sum look like an integration, if we choose to think of a number as a special type of pulse, which we will call an *impulse*. We can arrive at the impulse as an extension of this rectangular pulse (Fig 5). As $\lambda \rightarrow 0$, this pulse grows taller and narrower, but its area remains fixed at 1.0. In the limit, when $\lambda = 0$, the pulse becomes an impulse. If we integrate across the impulse, it adds 1.0 to the area. The impulse is at once a numeric (digital) value and an (analogue) function shape. It is our bridge between the analogue and the digital. Until now, we used a double-sided line spectrum to describe a set of numbers. Each line represents a number, but we can also view each line as an impulse. The line spectrum becomes an *impulsive* $X(f)$ spectrum, such that an Inverse CFT integral over this $X(f)$ becomes a sum over impulse areas, or an ordinary numeric summation.

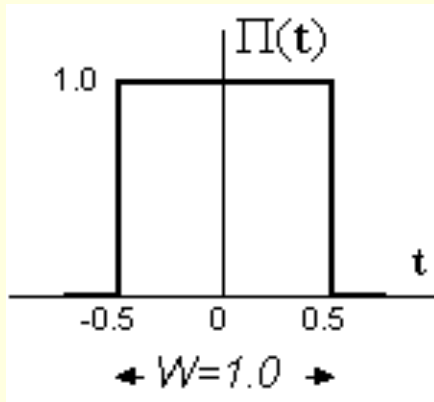


This diagram (Fig 6) shows how the Inverse DtFT sum over a line spectrum is the same as an Inverse CFT integral over an impulsive $X(f)$. The impulse areas are the C_k values. They are the area-samples $\{\varepsilon \cdot X(k\varepsilon)\}$ of $X(f)$. In this way, the DtFT is just a special (impulsive) version of the CFT.

3.2.2 CFT Pairs: rectangle and sinc

A pulse in time has a pulse-shaped CFT spectrum. As an example of this, we will find the CFT spectrum of the rectangle pulse $x(t) = \Pi(t)$ shown here (Fig 7). The integration is easy:

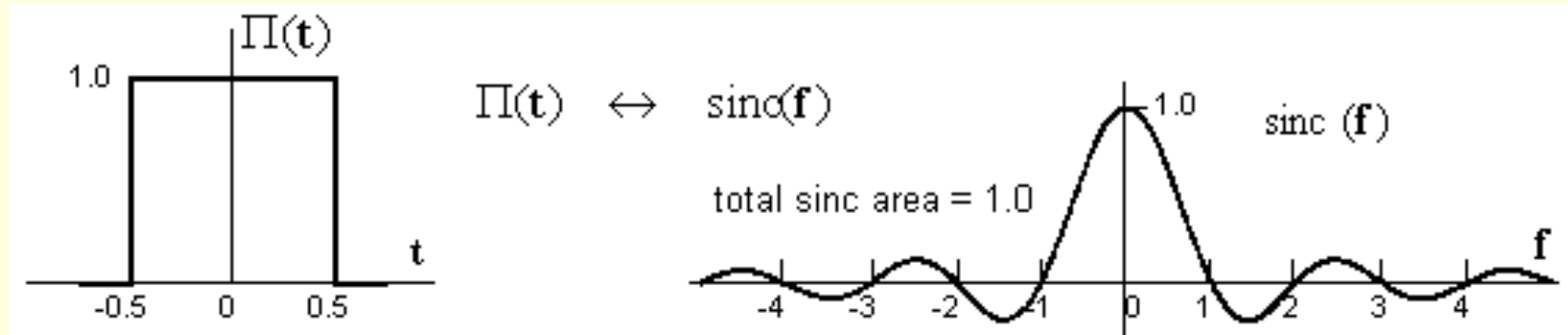
$$X(f) = \int_{-\infty}^{\infty} \Pi(t) \cdot e^{-j2\pi ft} \cdot dt = \int_{-1/2}^{1/2} 1.0 \cdot e^{-j2\pi ft} \cdot dt$$



The integral of e^{at} is e^{at}/a (whether a is real or complex), and so:

$$X(\mathbf{f}) = \left[\frac{e^{-j2\pi\mathbf{f}t}}{-j2\pi\mathbf{f}} \right]_{t=-1/2}^{t=1/2} = \frac{e^{j\pi\mathbf{f}} - e^{-j\pi\mathbf{f}}}{j2\pi\mathbf{f}} = \frac{\sin(\pi\mathbf{f})}{\pi\mathbf{f}} \equiv \text{sinc}(\mathbf{f})$$

The result is $\sin(\pi\mathbf{f})/(\pi\mathbf{f})$, a well-known shape (Fig 6), which we call a "sinc" pulse. This pulse is infinitely wide and very smooth. Its value is zero for all integer values of \mathbf{f} , except at $\mathbf{f} = 0$, where it peaks at a value of 1.0. The total area under $\text{sinc}(\mathbf{f})$ is 1.0, the same as the area under $\Pi(t)$.



The Inverse CFT integral (or ICFT) over this sinc gives a finite result (in spite of its infinite pulse width). Quite remarkably, the result will be either 0 or 1, the only possible values of $\Pi(t)$. Notice how we use \leftrightarrow to symbolise a CFT pair.

From the symmetry of these waveforms, and the symmetry of the CFT/ICFT integrals, it also emerges that the CFT of a sinc in time is a rectangle-shaped spectrum, that is:

$$\text{sinc}(t) \leftrightarrow \Pi(f)$$

To interpret this result: because the sinc-shaped time-pulse is so *smooth*, its spectrum is *band-limited*. It has no frequencies higher than $\frac{1}{2}$. These CFT pairs have great significance, and we will use them shortly.





3.3 THE DfFT AND THE DtFT

We've seen the DfFT (discrete in frequency), and we will soon find a corresponding DtFT (discrete in time). We'll develop them in a very general way, using a CFT pair $\{ x(\mathbf{t}) \leftrightarrow X(\mathbf{f}) \}$ as our starting point, so that the DfFT and the DtFT can both be viewed as impulsive extensions of the CFT.

3.3.1 Sampling and Replication

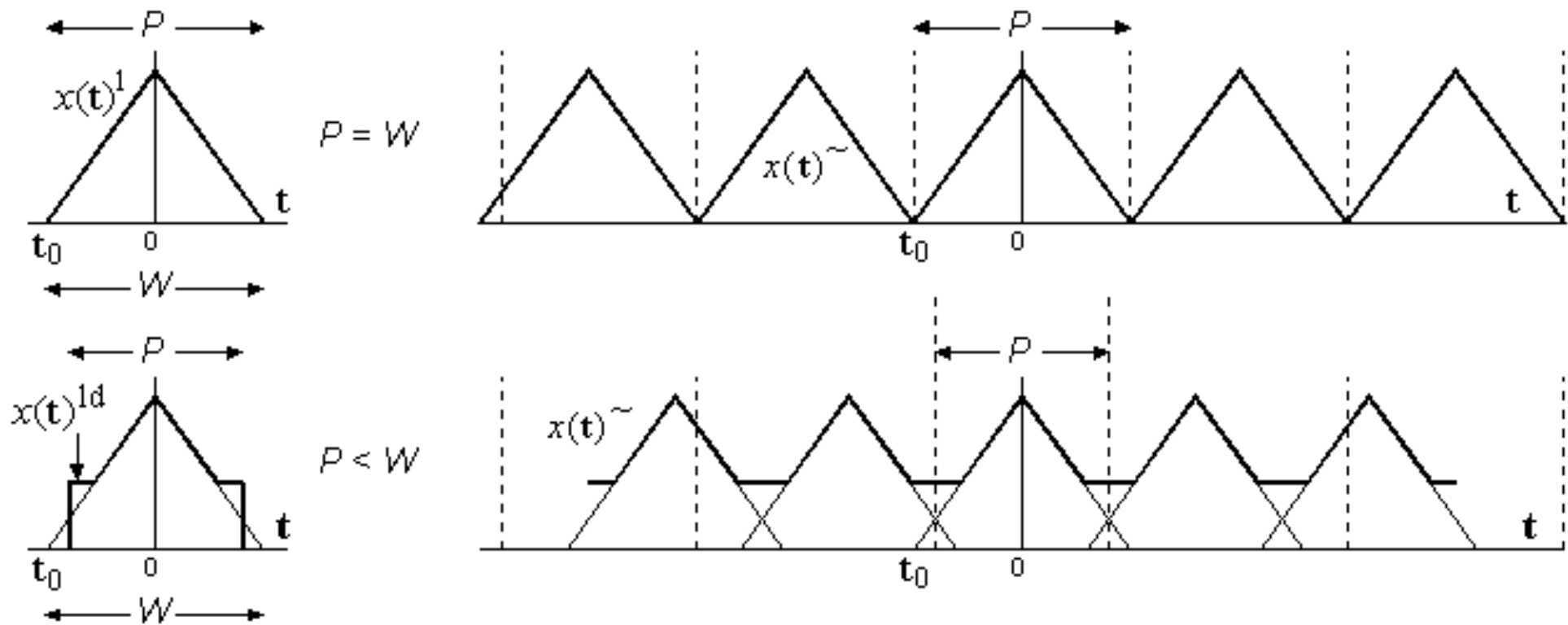
We've seen how a periodic $x(\mathbf{t})_{\sim}$ is fully described by its one-period pulse $x(\mathbf{t})^1$, and is in fact a replication of such pulses, that is, $x(\mathbf{t})_{\sim} = x(\mathbf{t})^1_{\sim}$. The replication interval P is also the pulse width W of $x(\mathbf{t})^1$. We also saw that $x(\mathbf{t})^1$ has a CFT spectrum $X(\mathbf{f})$.

Then we moved the $x(\mathbf{t})^1$ pulses apart. The new $x(\mathbf{t})_{\sim}$ is still a replication, except that $P > W$ now. We found that the new impulsive spectrum is still a set of area-samples C_k from the same $X(\mathbf{f})$ pulse. As P increases further, the C_k samples are more closely spaced on $X(\mathbf{f})$. Eventually, they fill all of $X(\mathbf{f})$, which is now the CFT of $x(\mathbf{t})^1$ (with its replicas removed to infinity).

We can understand the role of $X(\mathbf{f})$ when $P > W$, because its pulse shape is defined by $x(\mathbf{t})^1$, and that shape is preserved as P increases. But, if we replicate $x(\mathbf{t})^1$, using a spacing $P < W$, the pulses that form $x(\mathbf{t})^1 \sim$ will overlap, yielding a new periodic $x(\mathbf{t}) \sim$ in which the shape of $x(\mathbf{t})^1$ will be lost. We've illustrated this below (Fig 6) using a triangular waveform, in which $x(\mathbf{t})^1$ is a triangular pulse, and it has a CFT spectral pulse $X(\mathbf{f})$. When $P = W$ we can say that:

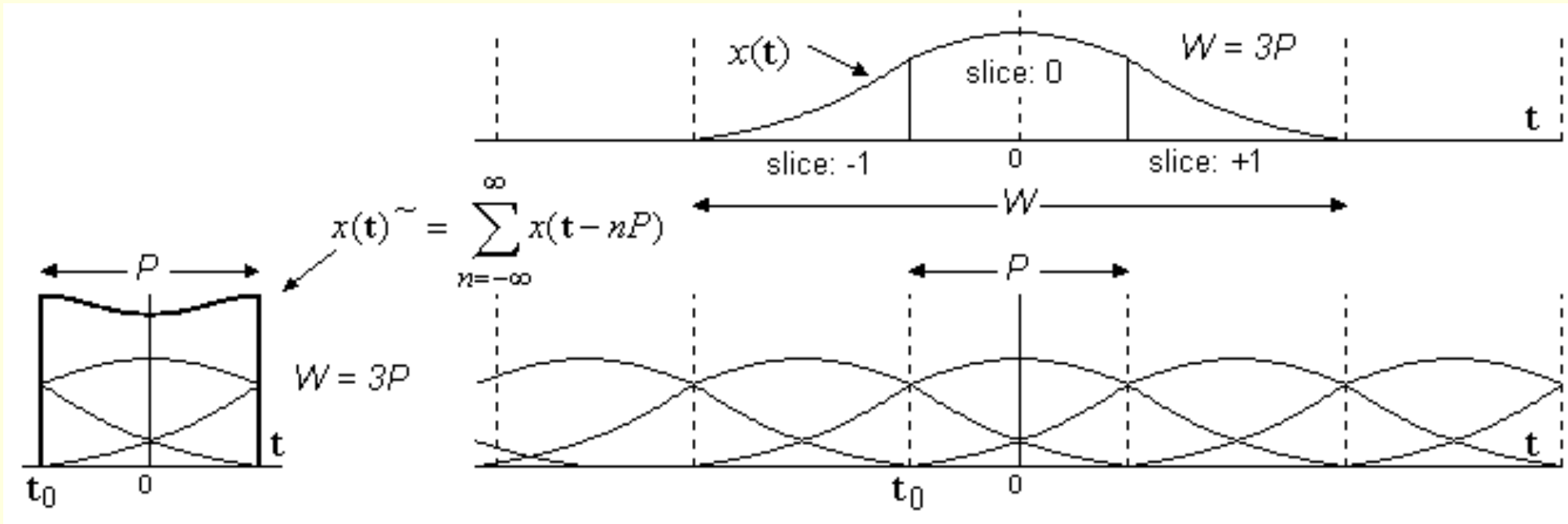
$$X(\mathbf{f}) = \int_{\mathbf{t}_0}^{\mathbf{t}_0 + P} x(\mathbf{t})^1 \sim \cdot e^{-j2\pi\mathbf{f}\mathbf{t}} \cdot d\mathbf{t}$$

When $P < W$, the overlapped pulse–shape is no longer triangular (Fig 6). Over one period, this pulse is narrower than $x(\mathbf{t})^1$ and its shape is *different*. We've shown it as $x(\mathbf{t})^{1d}$ (Fig 7), and it must have a *different* CFT spectrum $X(\mathbf{f})^d$.



It now appears that the above integral should return $X(\mathbf{f})^d$ instead of $X(\mathbf{f})$, and this is true in general. But we will show that $X(\mathbf{f})$ is still relevant, in spite of the overlap, but *only* at the sample frequencies $\mathbf{f} = k\epsilon$.

To do this, we'll use a pulse $x(t)$, and its replicated version $x(t)^\sim$, as given in this diagram (Fig 6). The $x(t)$ pulse is 3 times wider than the replication interval P , and we've sectioned $x(t)$ as 3 slices of width P . Then, one period P of the replicated $x(t)^\sim$ is the sum of the three slices of $x(t)$, as shown.



It follows that, if we integrate over one period of $x(t)^\sim$, we integrate over a *sum of all the slices* of $x(t)$. We cover all of its area. Therefore:

$$\int_P x(t)^\sim .dt = \int_{-\infty}^{\infty} x(t) .dt$$

We used $W = 3P$ for illustration, but even an infinite W , with unlimited overlap, is allowable. We noted this integral property earlier (i [Ch 1.2.3](#)) when we said:

- *the area within a one period window of a replicated pulse waveform equals the area of the original pulse.*

But the Fourier integral that we've encountered has an additional term, an exponent term, and when we include this term:

$$\int_P x(t) \sim \cdot e^{-j2\pi ft} dt \neq \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} dt$$

Note the inequality. They are not the same. To make them the same, the exponent term *should look identical in each and every slice* of $x(t)$. That calls for an exponent that is periodic in P , and this condition *will be met* when $f = k\varepsilon$. We can now say that:

$$\int_P x(t) \sim \cdot e^{-j2\pi k\varepsilon t} dt = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi k\varepsilon t} dt, \quad \text{for } \varepsilon = 1/P$$

We have an equality again, and we can use it on our overlapped triangular waveform ($P < W$) to find that:

$$X(k\varepsilon)^d = \int_{t_0}^{t_0+P} x(t)^{1\sim} \cdot e^{-j2\pi k\varepsilon t} dt = \int_{-\infty}^{\infty} x(t)^1 \cdot e^{-j2\pi k\varepsilon t} dt = X(k\varepsilon)$$

$X(k\varepsilon)^d$ and $X(f)$ give the same result at sample points, $f = k\varepsilon$. The picture now emerging is that if a pulse $x(t)$ with a CFT spectrum $X(f)$ is replicated to form a periodic $x(t) \sim$, the FS coefficients describing $x(t) \sim$ are $C_k = \varepsilon \cdot X(k\varepsilon)$, for any replication interval $P = 1/\varepsilon$, *even* when it causes the original $x(t)$ pulses to overlap one another. There is another way to say this:

- If a time pulse $x(t)$ has a CFT spectrum $X(f)$, then the DtFT spectrum of a periodic $x(t)$

\tilde{x} , formed by replication of $x(\mathbf{t})$ at intervals $P = 1/\epsilon$, is a set of area-samples $\{ C_k = \epsilon \cdot X(k\epsilon) \}$ from $X(\mathbf{f})$.

Even more concisely, we could say that:

- *area-sampling in frequency corresponds to replication in the time domain*

We will find that the converse is also true, that we can interchange the two domains, and then we will have a DtFT as well. Now we need a more formal language to describe them both.

3.3.2 DfFT and DtFT Definitions

We will need a notation to describe number sets as impulse functions. For a solitary impulse in time, we adapt the symbol $\delta(\mathbf{t})$. It represents a pulse of infinite height, zero width, and unit area, occurring at time $\mathbf{t} = 0$. As with any pulse, we can scale it and shift it: for example, the impulsive function $5\delta(\mathbf{t} - 4)$ is an impulse of area 5 occurring at time $\mathbf{t} = 4$. The impulse area is also called its *strength*. If we *multiply* some continuous function $x(\mathbf{t})$ by $\delta(\mathbf{t} - 4)$ we get:

$$x(\mathbf{t}) \cdot \delta(\mathbf{t} - 4) = x(4) \cdot \delta(\mathbf{t} - 4)$$

All that remains after multiplication is an impulse of strength $x(4)$ at time $\mathbf{t} = 4$. By this action we *sampled* $x(\mathbf{t})$ at time $\mathbf{t} = 4$. To sample $x(\mathbf{t})$ at regular intervals of T seconds, we can define:

$$x(\mathbf{t})\rangle = x(\mathbf{t}) \cdot \sum_{n=-\infty}^{\infty} \delta(\mathbf{t} - nT) \quad \text{value-sampling}$$

We use $x(\mathbf{t})\rangle$ to mean the *value*-sampled $x(\mathbf{t})$. It is a string of impulses (or a set of numbers) describing value-samples $x(nT)$ taken from $x(\mathbf{t})$. We can also define:

$$x(\mathbf{t})\gg = x(\mathbf{t}) \cdot \sum_{n=-\infty}^{\infty} T \cdot \delta(\mathbf{t} - nT) \quad \text{area-sampling}$$

We use $x(\mathbf{t})\gg$ to mean the *area*-sampled $x(\mathbf{t})$. It is a string of impulses (or a set of numbers) describing area-samples $T \cdot x(nT)$ taken from $x(\mathbf{t})$. Note, the numbers are the impulse *strengths* (or impulse *areas*), while the impulse *heights* are infinite.

We can define identical operations in frequency. For example, to perform area-sampling of $X(\mathbf{f})$, we write:

$$X(\mathbf{f})\gg = X(\mathbf{f}) \cdot \sum_{k=-\infty}^{\infty} \varepsilon \cdot \delta(\mathbf{f} - k\varepsilon) \quad \text{area-sampling}$$

After a function is sampled, only the sample values remain. Thus:

$$X(\mathbf{f})^{\gg} = X(\mathbf{f}) \cdot \sum_{k=-\infty}^{\infty} \varepsilon \cdot \delta(\mathbf{f} - k\varepsilon) = \sum_{k=-\infty}^{\infty} \{\varepsilon \cdot X(k\varepsilon)\} \cdot \delta(\mathbf{f} - k\varepsilon)$$

The result on the right has impulses of strength $\varepsilon \cdot X(k\varepsilon)$. Values of $X(\mathbf{f})$ at frequencies other than $\mathbf{f} = k\varepsilon$ have been lost. In similar manner:

$$x(\mathbf{t})^{\gg} = x(\mathbf{t}) \cdot \sum_{n=-\infty}^{\infty} T \cdot \delta(\mathbf{t} - nT) = \sum_{n=-\infty}^{\infty} \{T \cdot x(nT)\} \cdot \delta(\mathbf{t} - nT)$$

We also need to recall how we defined *replication* of $x(\mathbf{t})$:

$$x(\mathbf{t})^{\sim} = \sum_{n=-\infty}^{\infty} x(\mathbf{t} - nP)$$

The notation that we need is now in place, and it allows us to present the CFT and the DtFT in the tabular form shown below (Fig 6).

Row (a) of the table is for a CFT pair. The $x(\mathbf{t})$ and $X(\mathbf{f})$ are shown as similar bell-shaped pulses, and some pairs of this kind do exist. They are also convenient for our illustrations. The column on the left shows the Forward and Inverse integrals (the CFT and the ICFT) that connect $x(\mathbf{t})$ with $X(\mathbf{f})$.

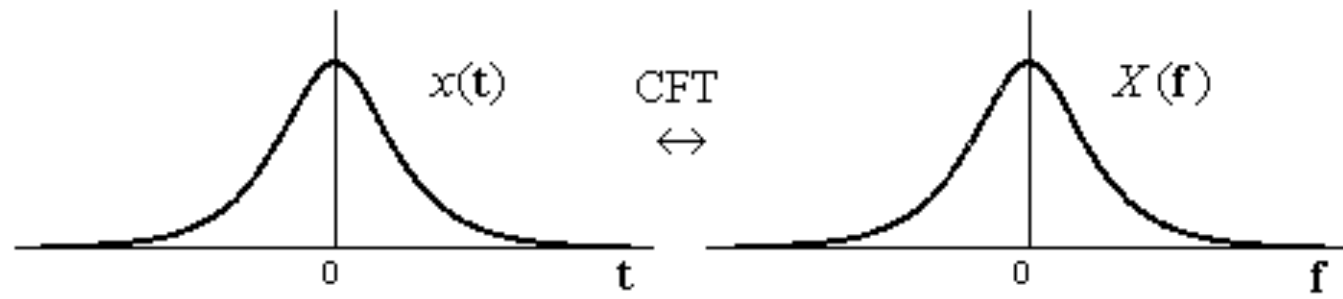
Row (b) of the table is for a DfFT pair. The diagram shows the replication of $x(\mathbf{t})$ to form $x(\mathbf{t})\sim$, and the corresponding area-sampling of $X(\mathbf{f})$ to give us a set of impulses. We've used a special symbol \lceil to mean an area-impulse. Its drawn height is the value $X(k\varepsilon)$, but its strength, or impulse area, is understood to be $\varepsilon \cdot X(k\varepsilon)$. In this way, there is no change of scale. The sums overhead the diagrams are just our definitions of $x(\mathbf{t})\sim$ and of $X(\mathbf{f})\gg$. Notice, the spectral sample interval is ε , and the time replication interval is $1/\varepsilon$. We don't need an additional symbol P . The column on the left shows the Forward DfFT integral and the Inverse DfFT (or IDfFT) summation that link $x(\mathbf{t})\sim$ to $X(\mathbf{f})\gg$. We could say all this very compactly as:

$$x(\mathbf{t})\sim \leftrightarrow X(\mathbf{f})\gg$$

a DfFT pair

$$= \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} dt$$

$$= \int_{-\infty}^{\infty} X(f) \cdot e^{j2\pi ft} df$$

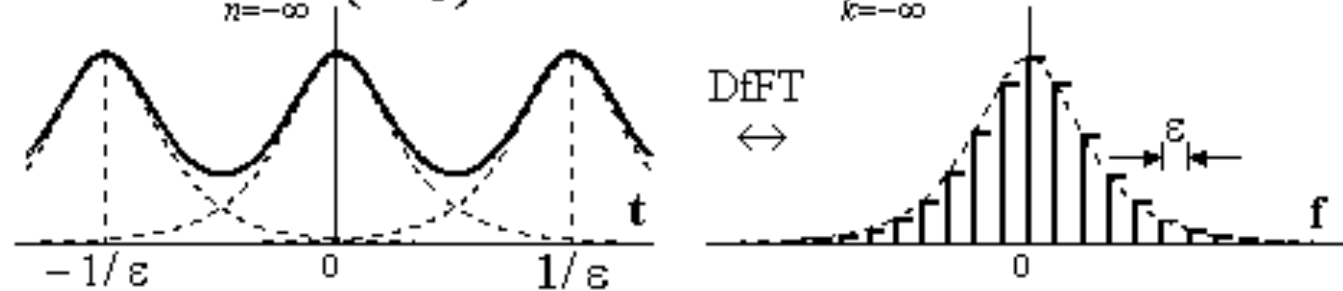


$$\int_{-1/\varepsilon}^{1/\varepsilon} x(t) \sim \cdot e^{-j2\pi k\varepsilon t} dt$$

$$\sum_{k=-\infty}^{\infty} \{\varepsilon \cdot X(k\varepsilon)\} \cdot e^{j2\pi k\varepsilon t}$$

$$x(t) \sim = \sum_{n=-\infty}^{\infty} x\left(t - \frac{n}{\varepsilon}\right)$$

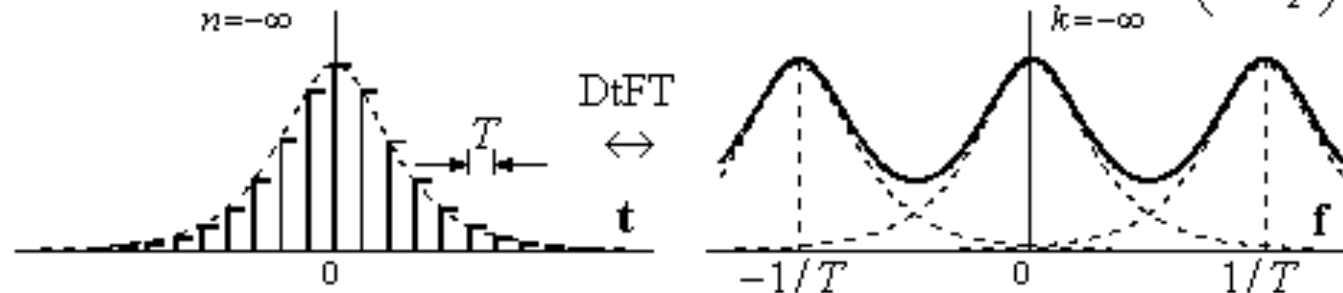
$$X(f) \gg = \sum_{k=-\infty}^{\infty} \{\varepsilon \cdot X(k\varepsilon)\} \cdot \delta(f - k\varepsilon)$$



$$\sum_{n=-\infty}^{\infty} \{T \cdot x(nT)\} \cdot e^{-j2\pi fnT}$$

$$x(t) \gg = \sum_{n=-\infty}^{\infty} \{T \cdot x(nT)\} \cdot \delta(t - nT)$$

$$X(f) \sim = \sum_{k=-\infty}^{\infty} X\left(f - \frac{k}{T}\right)$$



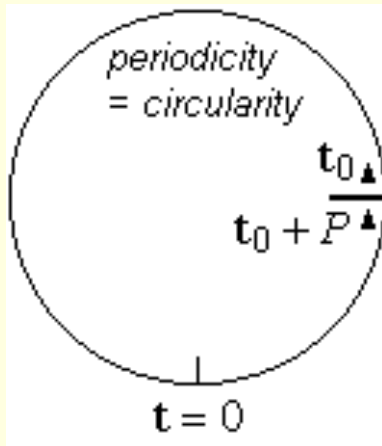
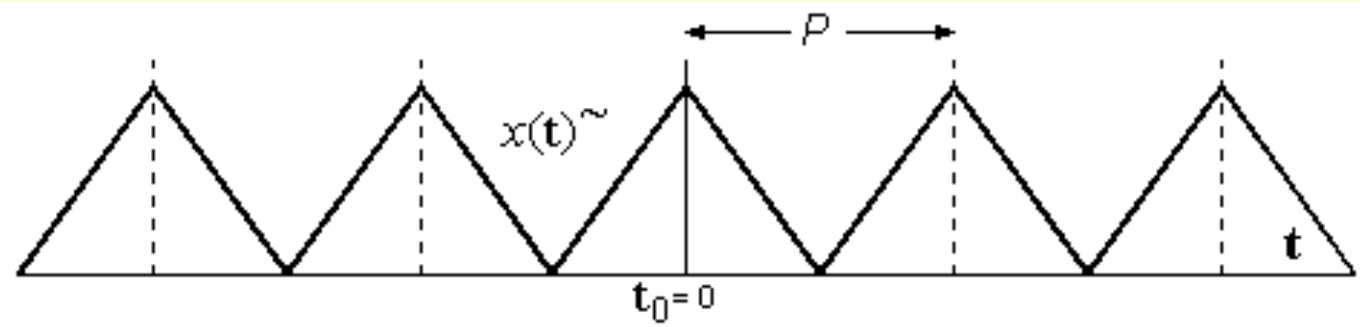
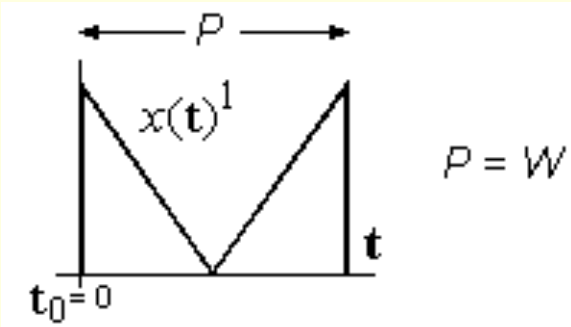
If we look at the symmetry of the CFT and ICFT integrals, we realise that the DfFT must have a counterpart, a DtFT, with very close similarity. We only have to swap $(-j)$ with $(+j)$, to interchange \mathbf{f} and \mathbf{t} , and to use a time-domain sample interval T where previously we had ϵ , the spectral sample spacing. Whereas the DfFT linked $x(\mathbf{t}) \sim$ to $X(\mathbf{f}) \gg$, the DtFT will link $x(\mathbf{t}) \gg$ to $X(\mathbf{f}) \sim$. We've presented the DtFT in row (c) of the table, and careful inspection should convince us of its validity. It would be easy to repeat all of the arguments that gave us the DfFT, and to arrive at the DtFT instead. Now we can generalise our earlier statement:

- *area-sampling in one domain corresponds to replication in the other domain*

The switch between $(-j)$ and $(+j)$ does not affect this conclusion. Notice in the left column of (b) and (c) above, the transform summations run over all frequency (from $k = -\infty$ to $k = +\infty$), or over all time (from $n = -\infty$ to $n = +\infty$). As such, they are just CFT/ICFT integrals applied to impulsive functions. But, the transform integrals in these columns run over one period only (over $1/\epsilon$ or over $1/T$). In fact, there is more to be said about these, and some interesting conclusions as well.

3.3.3 Observations on Periodicity

Our triangular waveform is re-drawn here (Fig 6), slightly modified. We've moved the one-period starting point t_0 from $t = -P/2$ to $t = 0$. That gives us a very different $x(t)$ pulse (Fig 7), but when it is replicated, it gives the same $x(t) \sim$ as before. This new $x(t)^1$ has a different $X(\mathbf{f})$ spectrum as well, but its spectral samples $X(k\epsilon)$ have *not* changed, because they describe the same $x(t) \sim$.



The same is true for *any* other value of t_0 . Every different t_0 gives a different $X(f)$ but they all share the same samples $X(k\varepsilon)$. This lends a *circular* property to the periodic $x(t)^\sim$ as depicted here (Fig c). We can place our t_0 at any point and, after P seconds, we re-visit the same values again, as if traversing a circular time path. In fact we can say this more generally:

- any signal which is discrete in one domain has a circular behaviour in the other domain

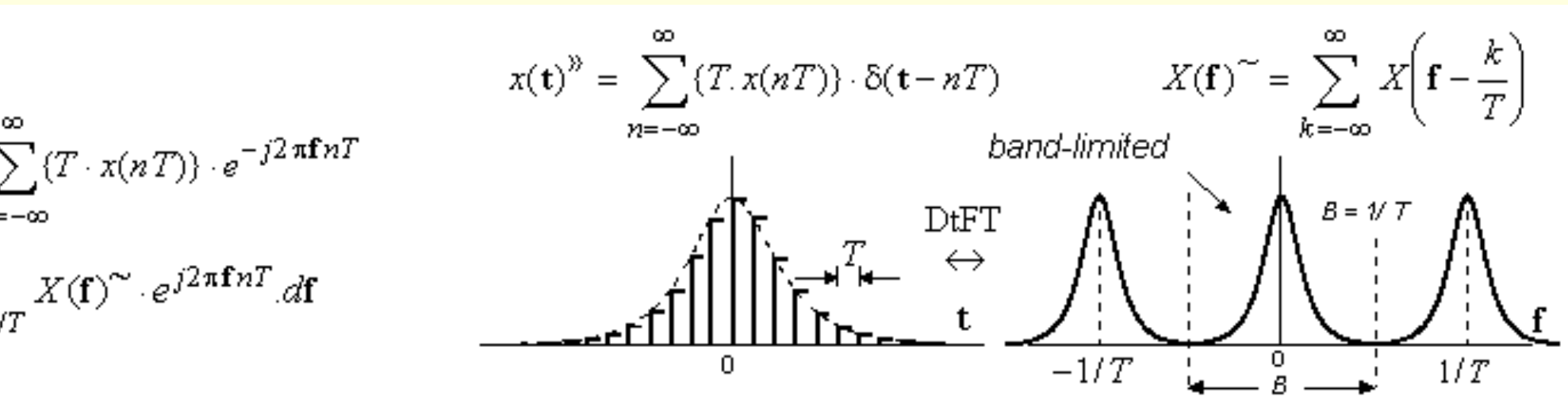
Each new t_0 yields a new $x(t)^1$ with its own unique $X(f)$, but these are not the only $X(f)$ pulses that share the same $X(k\varepsilon)$ samples. We can also include the $X(f)$ spectra of all those *wider* $x(t)$ pulses which overlap when replicated, but still yield the

same $x(t) \sim$ as do the $x(t)^1$ pulses. In general, therefore, *all* those time pulses which replicate to form the same $x(t) \sim$ will also share the same $X(k\epsilon)$ values.

We can interchange the two domains, and make identical observations. The maths are much the same, but we tend to view the two domains differently. With this in mind, we will switch our attention to the DtFT of row (c) in the Table, and then continue our discussion from there.

3.3.4 The Sampling Theorem

The DtFT diagram is repeated here (Fig 6), slightly altered. We've shown the $X(f)$ pulse as *band-limited*, that is, it has a finite width which we will call its *bandwidth*, B .



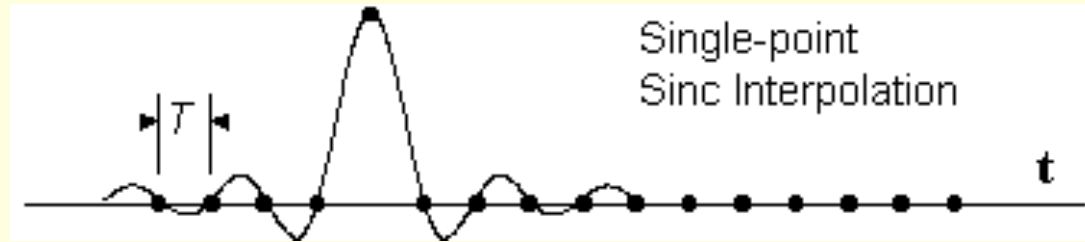
$X(\mathbf{f})$ is the spectrum of a time pulse $x(\mathbf{t})$, and the replicated $X(\mathbf{f})\sim$ is the spectrum of $x(\mathbf{t})\gg$, the *area-sampled* time sequence. We chose a sample spacing of $T = 1/B$ sec, thus ensuring that the replicated $X(\mathbf{f})$ pulses would just touch one another, but without overlapping. This means that the identity of $X(\mathbf{f})$ is *not lost* on replication. That would seem to imply that *just the time samples* from the $x(\mathbf{t})$ pulse may be sufficient to define $x(\mathbf{t})$ completely, that is, even at points *between* the known sample points. We get further evidence of this if we increase the rate of sampling by making T even smaller. Then the replicas of $X(\mathbf{f})$ in $X(\mathbf{f})\sim$ move further apart, but the $X(\mathbf{f})$ pulse at $\mathbf{f} = 0$ does not change. Meanwhile, the samples of $x(\mathbf{t})$ grow denser and eventually merge into a continuous $x(\mathbf{t})$ which has the $X(\mathbf{f})$ at $\mathbf{t} = 0$ as its CFT spectrum. We can find $X(\mathbf{f})$ by DtFT from a sample set $x(\mathbf{t})\gg$ at *any spacing which does not exceed* $T = 1/B$. (If that limit is exceeded, spectral overlap occurs, and then $X(\mathbf{f})$ is not recoverable from $X(\mathbf{f})\sim$). Having found $X(\mathbf{f})$ from samples at the *maximim* spacing of $T = 1/B$, the Inverse CFT of $X(\mathbf{f})$ can tell us all other values of $x(\mathbf{t})$. It follows that:

- a signal $x(\mathbf{t})$ whose double-sided spectrum is band-limited to B Hertz is fully recoverable from its samples taken at a rate exceeding B samples/second.

This Sampling Theorem has great importance for sampled time signals, but it is equally true that a time-pulse $x(\mathbf{t})$ of finite width P sec has a spectrum $X(\mathbf{f})$ that is fully specified by its samples taken at $\epsilon = 1/P$ Hertz, or less.

3.3.5 Sinc Interpolation

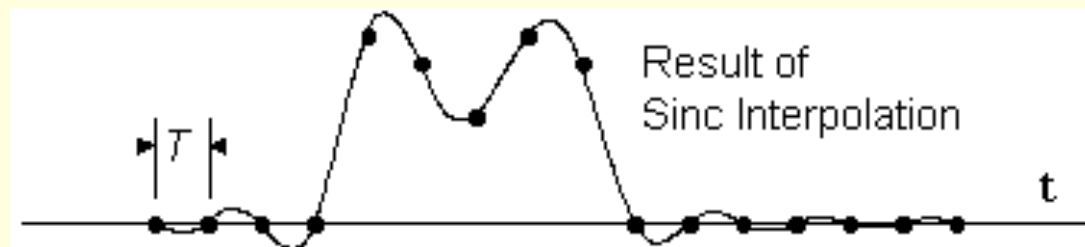
Given a sample set $x(\mathbf{t})\gg$ from some $x(\mathbf{t})$, the process of finding intermediate $x(\mathbf{t})$ values is called *interpolation*. We'll first consider a very simple $x(\mathbf{t})\gg$ in which *only one* of its sample-values is non-zero. Even this has a valid interpolation, and we already know one pulse shape that can pass through all the sample values and also fill the spaces in between. This is the time-domain sinc, or sinc-t pulse, as shown here (Fig 6).



An arbitrary sequence can be considered to be a sum of one-point sequences like that shown above. This sequence (Fig \hat{e}) has five non-zero values, each of them interpolated by a sinc-t pulse of corresponding height.



The final interpolation is the sum of all the sinc pulses. It is always a *smooth* interpolation (Fig \hat{i}), although it can be quite oscillatory at times.



This sum of sincs meets the time-domain constraints in that it passes through *all* of the given sample points. It must also meet the frequency constraint of being band-limited to $B = 1/T$ Hertz. To check this, we recall the CFT pair $\text{sinc}(t) \leftrightarrow \Pi(f)$ in which the zeros of $\text{sinc}(t)$ have a spacing of 1.0, and its spectrum $\Pi(f)$ has a width of 1.0. But the sinc pulses in our diagram have a zero-spacing of T , and this changes the spectral width to $1/T$, a result that matches our bandwidth constraint exactly. This argues strongly for the sinc as the true interpolator for band-limited signals. More formal methods would confirm that this is correct. The sinc- t interpolation yields a *pulse* that we call $x(t)^\wedge$:

$$x(t)^\wedge = \sum_{n=-\infty}^{\infty} x(nT) \cdot \text{sinc}\left(\frac{t-nT}{T}\right)$$

where the symbol \wedge denotes interpolation. We can conclude that $x(t)^\wedge$ and $X(f)$ are a CFT pair:

$$X(f) = \int_{-\infty}^{\infty} x(t)^\wedge \cdot e^{-j2\pi ft} \cdot dt \quad \text{and} \quad x(t)^\wedge = \int_{-1/2B}^{1/2B} X(f) \cdot e^{j2\pi ft} \cdot df$$

Integration limits of $(-\infty, \infty)$ could be used for both integrals, because $X(f)$ is band-limited to B .

We've spoken only of a band-limited $X(f)$, but the one-period $X(f)^\wedge$ pulse of a *non*-band-limited $X(f)$ is by definition band-limited, and the same result must apply to this. This gives us the result in row (d) below (Fig 6).

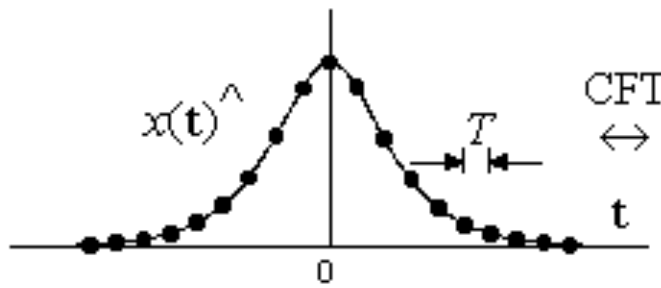
Interpolation

$$\hat{x}(t) = \int_{-\infty}^{\infty} \hat{x}(t) \cdot e^{-j2\pi ft} dt$$

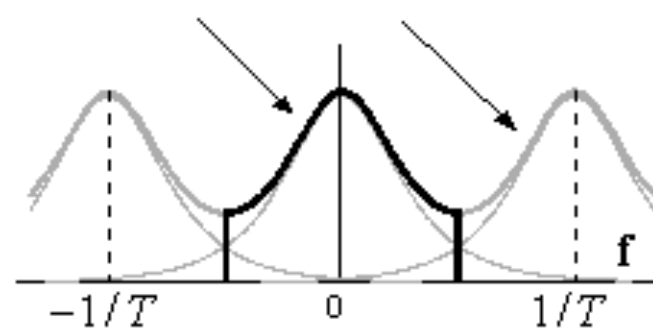
$$= \int_{-1/2T}^{1/2T} X(f) \cdot e^{j2\pi ft} df$$

$$= \int_{-1/T}^{1/T} X(f) \cdot e^{j2\pi f n T} df$$

$$\hat{x}(t) = \sum_{n=-\infty}^{\infty} x(nT) \cdot \text{sinc}\left(\frac{t-nT}{T}\right)$$



$$X(f) \hat{=} X(f) \cdot \Pi(fT)$$



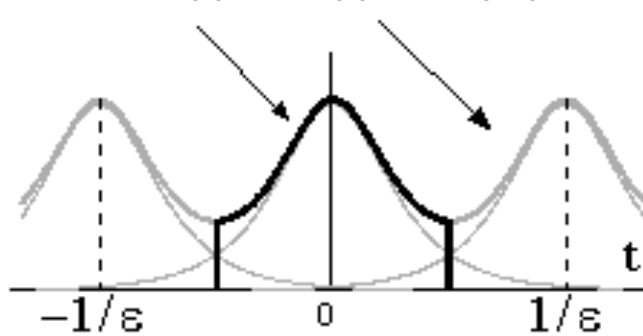
Interpolation

$$\hat{x}(t) = \int_{-1/2\varepsilon}^{1/2\varepsilon} \hat{x}(t) \cdot e^{-j2\pi ft} dt$$

$$= \int_{-\infty}^{\infty} X(f) \cdot e^{j2\pi ft} df$$

$$= \int_{-1/\varepsilon}^{1/\varepsilon} X(f) \cdot e^{-j2\pi k\varepsilon t} dt$$

$$\hat{x}(t) \hat{=} x(t) \cdot \Pi(t\varepsilon)$$



$$X(f) \hat{=} \sum_{k=-\infty}^{\infty} X(k\varepsilon) \cdot \text{sinc}\left(\frac{f-k\varepsilon}{\varepsilon}\right)$$



Eqns (d) 1 and 2 are the CFT/ICFT integrals. We could write both with $(-\infty, \infty)$ limits because $X(f)^1$ is band-limited to $1/T$. Eqn (d) 3 refers to sample points only. In this case, *any* one-period width of $X(f)_{\sim}$ will do, regardless of where it starts.

Some of the detail from the original CFT pair $\{x(t) \leftrightarrow X(f)\}$ is no longer available in the pair $\{x(t)^\wedge \leftrightarrow X(f)^1\}$. $x(t)^\wedge$ has only the *samples* from $x(t)$, and $X(f)$ cannot be fully recovered from $X(f)^1$ because of spectral overlap. Both $x(t)^\wedge$ and $X(f)^1$ express the same loss of information between time-samples, but they do it in different ways.

We extracted $X(f)^1$ from $X(f)_{\sim}$ above through multiplication by the *window function* $\Pi(fT)$. This $\Pi(fT)$ equals 1.0 for $(-1/2T < f < 1/2T)$ and is zero everywhere else. Windowing is a widely-used concept in DSP.

When we interchange the domain roles, row (e) gives us the results for DfFT interpolation. Eqns (e) 1 and 2 are the CFT/ICFT integrals. We could write both with $(-\infty, \infty)$ limits because $x(t)^1$ is band-limited to $1/\epsilon$. Eqn (e) 3 refers to sample points only. In this case, *any* one-period width of $x(t)_{\sim}$ will do, regardless of where it starts.

In the next section, we will sample the band-limited pulses $x(t)^1$ and $X(f)^1$ and this will replicate their interpolated spectra. This process will bring us to the Discrete Fourier Transform (DFT), a transform which is discrete in *both* domains.





3.4 THE DFT

The DFT has importance as the all-numeric transform for computer use. We will derive it now, but we will defer all use of the DFT until later.

3.4.1 From Interpolated DfFT and DtFT to the DFT

Starting from the CFT pairs:

$$\{ x(t) \} \leftrightarrow \{ X(f) \} \text{ and } \{ x(t) \} \leftrightarrow \{ X(f) \}$$

we will area-sample the band-limited $x(t)$ and $X(f)$ pulses at a spacing which equals the band-width divided by N , where N is any positive integer. This has the effect of replicating $X(f)$ and $x(t)$ at a spacing which is $1/(\text{band-width})$. These operations give us the signals shown here (Fig 6).

FT

T

$$x(nT) \sim \delta(t - nT)$$

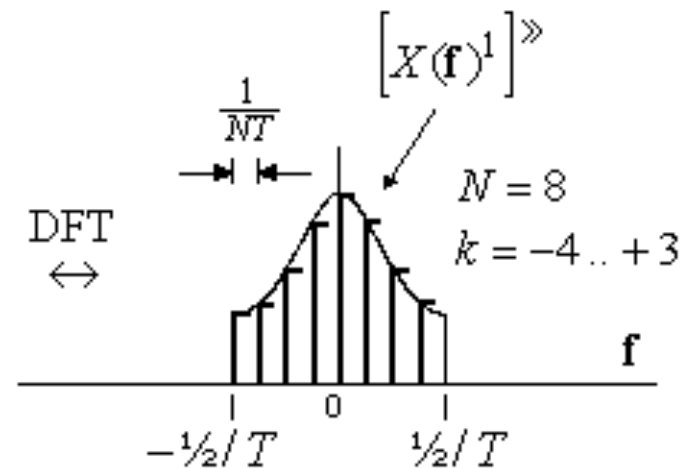
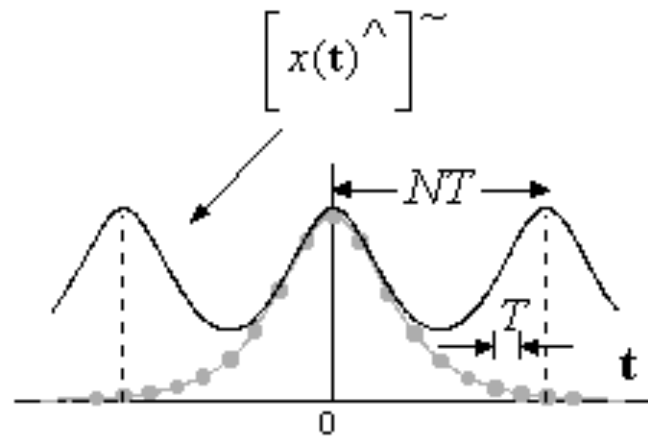
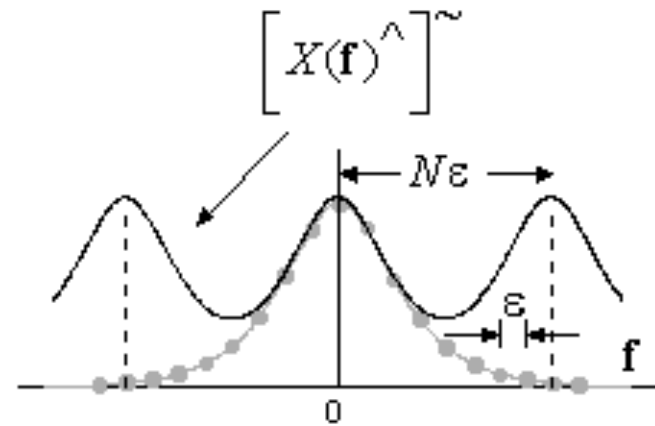
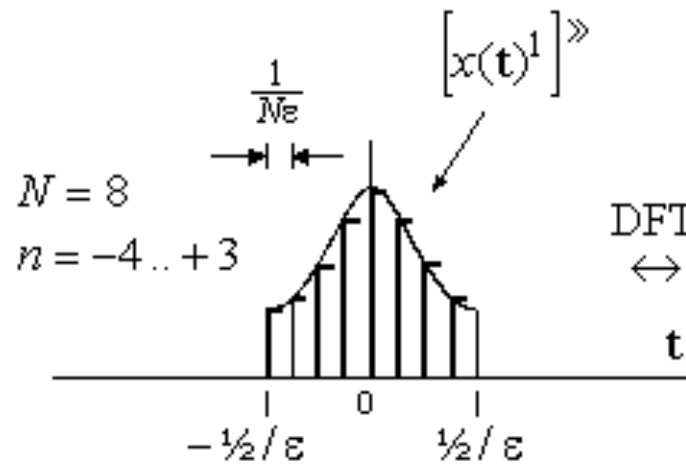
$$X(k\epsilon) \sim \delta(f - k\epsilon)$$

FT

ϵ

$$X(k\epsilon) \sim \delta(f - k\epsilon)$$

$$x(nT) \sim \delta(t - nT)$$



In row (f), we had sampled $X(\mathbf{f})$ with sample spacing ϵ . This gave a bandwidth of $1/\epsilon$ for $x(\mathbf{t})^1$ and the sample-spacing in time becomes $1/N\epsilon$ as shown. This time-sampling causes $X(\mathbf{f})^\wedge$ to be replicated at intervals of $N\epsilon$ to yield a new periodic signal $[X(\mathbf{f})^\wedge]^\sim$. When the replicas are added, the sample points of all the replicas in $[X(\mathbf{f})^\wedge]^\sim$ will be aligned, but only because we chose N as an integer. Thanks to this alignment, and because the samples of $X(\mathbf{f})^\wedge$ are also samples of $X(\mathbf{f})$, the samples of $[X(\mathbf{f})^\wedge]^\sim$ will now become samples of $X(\mathbf{f})^\sim$. This allows us to write a forward CFT over the area-samples of $x(\mathbf{t})^1$, but we will evaluate it only at the points $\mathbf{f} = k\epsilon$.

$$[X(k\epsilon)^\wedge]^\sim = X(k\epsilon)^\sim = \int [x(\mathbf{t})^1]^\gg \cdot e^{-j2\pi k\epsilon t} \cdot dt$$

The signal $[x(\mathbf{t})^1]^\gg$ is a set of area-impulses, and the integration reduces to a summation over time-samples at spacing of $1/N\epsilon$. It becomes:

$$X(k\epsilon)^\sim = \sum_n \left\{ \frac{1}{N\epsilon} \cdot x\left(\frac{n}{N\epsilon}\right)^\sim \right\} \cdot e^{-j2\pi k\epsilon \frac{n}{N\epsilon}} = \sum_n \left\{ \frac{1}{N\epsilon} \cdot x\left(\frac{n}{N\epsilon}\right)^\sim \right\} \cdot e^{-j2\pi \frac{k}{N} n}$$

We can make a corresponding statement about the signals in row (g):

$$x(nT) \sim = \sum_k \left\{ \frac{1}{NT} \cdot X\left(\frac{k}{NT}\right) \sim \right\} \cdot e^{j2\pi \frac{k}{NT} nT} = \sum_k \left\{ \frac{1}{NT} \cdot X\left(\frac{k}{NT}\right) \sim \right\} \cdot e^{j2\pi \frac{k}{N} n}$$

The signals in row (f) have two spacing parameters, N and ϵ , while the signals in row (g) have spacing parameters N and T . There is no necessary relationship between ϵ and T , but, if we choose T and ϵ such that:

$$1/N\epsilon = T \text{ or, equivalently, } 1/NT = \epsilon$$

we will have the same sample positions (in time and in frequency) for row (e) that we have for row (f). Then both rows will refer to the same numeric data and we can re-write our summations as:

$$X(k\epsilon) \sim = \sum_n \{T \cdot x(nT) \sim \} \cdot e^{-j2\pi \frac{k}{N} n}$$

Forward DFT

and

$$x(nT) \sim = \sum_k \{ \varepsilon \cdot X(k\varepsilon) \sim \} \cdot e^{j2\pi \frac{k}{N} n}$$

Inverse DFT

The diagram shows summing ranges for n and k to cover one period with $N = 8$. But the range $-4 \dots +3$ can be replaced by the range $0 \dots 7$ without penalty. In fact, *any* set of N consecutive samples will do, and the range $0 \dots N-1$ is widely used as a matter of convenience.

These equations apply to the replicated signals $\{x(\mathbf{t}) \sim, X(\mathbf{f}) \sim\}$, but not to the original CFT pair $\{x(\mathbf{t}), X(\mathbf{f})\}$. They connect N time-samples that span one period of $x(\mathbf{t}) \sim$ to N spectral samples that span one period of $X(\mathbf{f}) \sim$. They use a sum over *area-samples* in one domain to determine *value-samples* in the other domain. On a purely numeric level, the DFT uses an N -point time vector $x[n]$ to find an N -point spectral vector $X[k]$, and the Inverse DFT (or IDFT) uses $X[k]$ to restore the original $x[n]$.

The use of replicated signals $\{x(\mathbf{t}) \sim, X(\mathbf{f}) \sim\}$ rather than $\{x(\mathbf{t}), X(\mathbf{f})\}$ is a necessary consequence of sampling, because replication in one domain is an expression of what was lost between samples in the other domain. For a fully digital transform, replication in both domains is inevitable.

But overlap is not inevitable. For a given CFT pair $\{x(\mathbf{t}), X(\mathbf{f})\}$, it is possible for $x(\mathbf{t})$ or $X(\mathbf{f})$, but not both, to be band-limited. Thus, in one domain only, we can have replication without overlap. This can mean that $x(\mathbf{t})^1 = x(\mathbf{t})$, or that $X(\mathbf{f})^1 = X(\mathbf{f})$, but not both of these simultaneously.

This concludes our introduction to Fourier Transform theory. We now have the transform base that we need for the DFT application work of later chapters.





4.1 PREAMBLE

The DfFT, also known as Fourier Series, provides discrete spectral descriptions of periodic time signals. Periodic signals are important in several areas of engineering, and their spectra are frequently of interest. This chapter takes a closer look at Fourier Series.



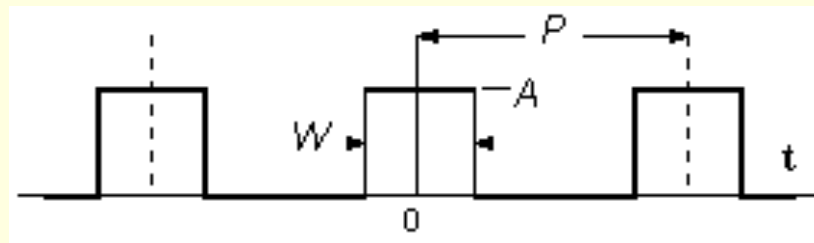


4.2 PERIODIC WAVEFORMS AND THEIR PROPERTIES

We will examine a number of periodic waveforms, with observations about their symmetry, and the spectral implications. We will see the spectral impact of time-shifting a signal. We will show the Energy and Power expressions for these signals. We will look at periodic impulse trains with their periodic impulsive spectra. Finally, we will talk about Harmonic Distortion, which is a useful measure of non-linear effects in amplifiers and related equipment.

4.2.1 The Rectangular Waveform

The rectangular waveform (Fig 4.2) is a replication of rectangular pulses, each of width W , and with pulse spacing $P > W$.



We can start by taking the CFT of the pulse, which we will call $x(t)$:

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} dt = \int_{-1/2W}^{1/2W} A \cdot e^{-j2\pi ft} dt$$

$$= \frac{-A}{j2\pi f} \left[e^{-j2\pi ft} \right]_{-1/2W}^{1/2W} = \frac{A}{j2\pi f} \left[e^{j\pi W f} - e^{-j\pi W f} \right]$$

$$= \frac{AW}{\pi W f} \left[\frac{e^{j\pi W f} - e^{-j\pi W f}}{2j} \right] = AW \cdot \frac{\sin(\pi W f)}{\pi W f}$$

Finally: $X(f) = AW \cdot \text{sinc}(Wf)$

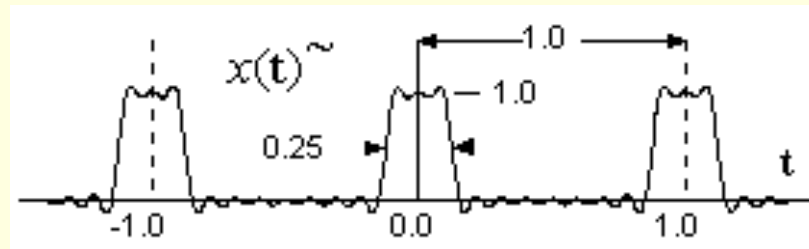
We can then specify the FS coefficients as:

$$C_k = \varepsilon \cdot X(k\varepsilon) = \frac{1}{P} X\left(\frac{k}{P}\right) = \frac{AW}{P} \text{sinc}\left(\frac{W}{P}k\right)$$

From these, we can re-construct the periodic waveform as:

$$x(t) \sim \sum_{k=-\infty}^{\infty} C_k \cdot e^{j2\pi k \frac{t}{P}} = C_0 + \sum_{k=1}^{\infty} 2|C_k| \cdot \cos\left(2\pi \frac{k}{P} t + \arg\{C_k\}\right)$$

A direct summation that used $A = 1$, $P = 1$ and $W = 0.25$, up as far as $k = 10$ (the tenth harmonic) gave this result (Fig 2.3).



We can see an oscillation (often called "Gibb's oscillation") that is widely observed in Fourier synthesis. It occurs when we attempt sharp transitions with a limited bandwidth (frequencies no higher than $10/P$ or 10 Hz in this case). If we include more harmonics, the *frequency* of the oscillation will increase but its *amplitude* will *not* diminish. Later on, we'll describe ways to curb the oscillation where needed. If this waveform looks familiar, it's because we saw it before (in [Ch 2.3](#)). On that occasion the C_k expression seemed quite different, and this brings us to our next topic.

4.2.2 Time-shifted Waveforms

The present $x(t) \sim$ has its $t = 0$ axis on the centre of a pulse, and this imparts *even symmetry* to the waveform. The C_k that we found for it are *real-valued* numbers. This means that $\arg\{C_k\} = 0$, causing $x(t) \sim$ to be built from a sum of *pure* cosines. That makes sense, because the cosines themselves have even symmetry also.

Our earlier $x(t) \sim$ (i [Ch 2.3](#)) had its $t = 0$ axis on a pulse edge, equivalent to a right-shift (a delay) of the present waveform by $\tau = W/2$ sec. We also noted (i [Ch 2.2.3](#)) that a τ sec delay was equivalent to a phase change of $-2\pi f_1 \tau$ radians at some frequency f_1 . To apply this phase change to our C_k coefficients, where the harmonic frequencies are $(k\varepsilon)$ Hertz, we require:

$$C_k \rightarrow C_k \angle(-2\pi k\varepsilon\tau) = C_k \cdot e^{-j2\pi k\varepsilon\tau}$$

We will apply this to the waveform that we just plotted:

$$C_k = \frac{AW}{P} \operatorname{sinc}\left(\frac{W}{P}k\right) \cdot e^{-j2\pi \frac{kW}{P} \frac{W}{2}} = 0.25 \cdot \operatorname{sinc}\left(\frac{k}{4}\right) \cdot e^{-jk\frac{\pi}{4}}$$

Compare this with our earlier result:

$$C_k = \frac{1}{j2\pi k} \left[1 - (-j)^k \right]$$

They may not look the same, but they are the same. That's easily checked numerically, but can you manipulate the equations to prove it ?

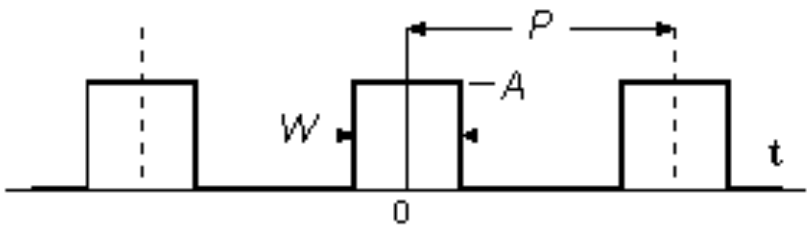
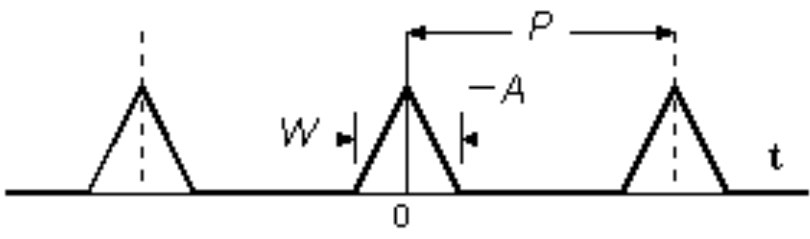
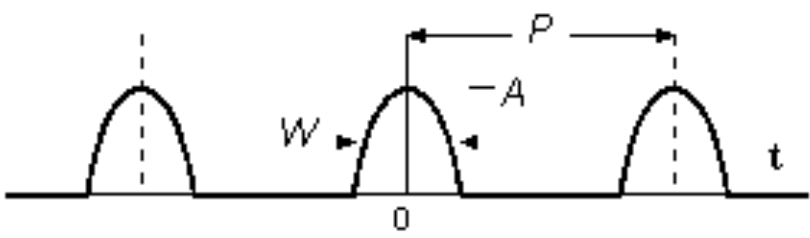
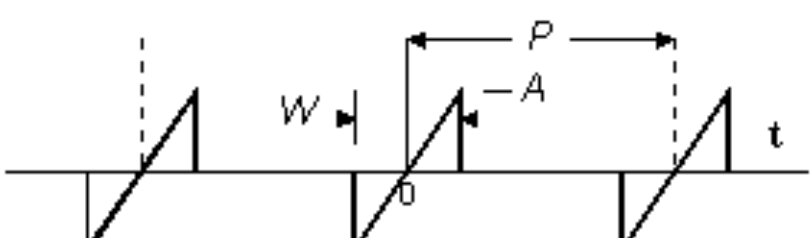

We've just shown how a signal's spectrum changes when the signal is shifted in time. A shift of τ seconds multiplies the spectrum $X(\mathbf{f})$ by $e^{-j2\pi\mathbf{f}\tau}$. This multiplier has a magnitude of 1.0. We conclude that:

- *time shifting of a signal does not alter the spectral magnitude, but it applies a linear-phase adjustment to the phase.*

This result holds for all periodic waveforms, and equally for other waveforms, so we will meet it quite a lot in the future.

4.2.3 Periodic Waveform Listing

We've tabulated several waveforms here (Fig 4.2.3).

Shape Function $X(\mathbf{f})$	Shape Description $C_k = \varepsilon \cdot X(k\varepsilon), \quad \varepsilon = 1/P$
Rectangle $AW \cdot \text{sinc}(W\mathbf{f})$	
Triangle $\frac{1}{2}AW \cdot \text{sinc}^2(\frac{1}{2}W\mathbf{f})$	
Half-Cosine $\frac{AW}{2} \left\{ \begin{array}{l} \text{sinc}(W\mathbf{f} - \frac{1}{2}) \\ + \text{sinc}(W\mathbf{f} + \frac{1}{2}) \end{array} \right\}$	
Ramp $\frac{-jA}{\pi^2 W \mathbf{f}^2} \left\{ \begin{array}{l} \sin(\pi W \mathbf{f}) - \\ (\pi W \mathbf{f}) \cdot \cos(\pi W \mathbf{f}) \end{array} \right\}$	
Sign	

For the Rectangular waveform, we used the Forward CFT integral of the one–period pulse to arrive at the $X(\mathbf{f})$ formula. The same approach, with a little more algebra, gives us the others $X(\mathbf{f})$ expressions in the Table. If the Triangle waveform were to be constructed using $W < P$, we would see the change of shape, due to overlap, that we predicted earlier (i [Ch 3.3.1](#)).

The Rectangle, Triangle and Half–Cosine all have even symmetry about $\mathbf{t} = 0$. They also have real–valued $X(\mathbf{f})$ expressions, and the C_k values, which are samples from $X(\mathbf{f})$, will be real–valued too. This is as it should be, because waveforms with even symmetry are constructed from cosines, and cosines have zero phase.

A waveform which has *odd symmetry* about $\mathbf{t} = 0$ is built from sines rather than cosines, because the sines themselves have odd symmetry. In this case, the C_k values will have phase–angles of $\pm\pi/2$. As such, they are *imaginary* numbers. The Ramp and Sign waveforms have odd symmetry, and their $X(\mathbf{f})$ expressions are imaginary as anticipated.

It makes sense to avail of waveform symmetry wherever possible. If we shift a symmetric waveform horizontally, we destroy that symmetry, and the coefficients C_k become complex (rather than just real *or* imaginary). However, they do so in a very predictable way, by application of a *linear phase lag*, as was demonstrated earlier. To recap, if we modify any of these $X(\mathbf{f})$ according to:

$$X(\mathbf{f}) \rightarrow X(\mathbf{f}) \cdot e^{-j2\pi\mathbf{f}\tau} \quad \text{applying a } \tau\text{-second delay}$$

the $x(\mathbf{t})\sim$ reconstruction is moved horizontally by τ seconds, to the right (a time *delay*) when τ is positive, and to the left (a time *advance*) when τ is negative.

4.2.4 Odd Half–wave Symmetry

The Fourier Series coefficients of a periodic $x(t) \sim$ can be found from:

$$C_k = \frac{1}{P} \int_0^P x(t) \sim \cdot e^{-j2\pi \frac{k}{P} t} \cdot dt$$

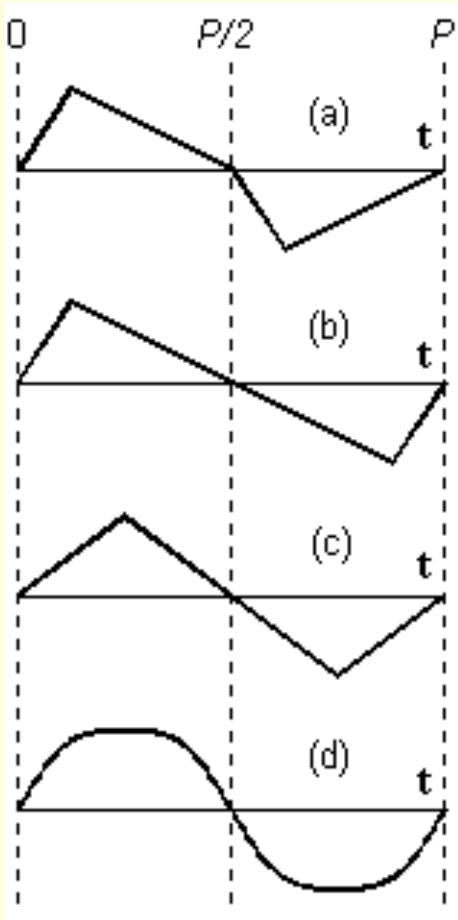
A waveform having *odd half-wave symmetry* is one for which:

$$x(t \pm \frac{1}{2}P) \sim = -x(t) \sim$$

Here (Fig ç) we see one period from different waveforms, and waveform (a) has odd half-wave symmetry. For even harmonics (when k is a multiple of 2), the exponent term in the range $(0 < t < \frac{1}{2}P)$ is repeated exactly in the range $(\frac{1}{2}P < t < P)$, while the $x(t) \sim$ shape just changes sign. The two halves of the C_k integral will have equal and opposite areas. It follows that $C_k = 0$ when k is even, and that odd half-wave symmetry causes the even harmonics to disappear altogether.

Waveform (b) might seem more symmetric, but it does *not* have odd half-wave symmetry. Waveform (c) is a special case of waveform (a), so it too has no even harmonics. The same can be said of waveform (d). Waveforms like this can occur as the *distorted* responses to sine-wave inputs in *balanced* amplifier circuits, and the absence of even harmonics is well known in this context.

Some of the waveforms that we tabulated have no even harmonics under certain conditions. The Sign waveform has no DC level, and it has no even harmonics when $W = P$. The Triangle waveform has no even harmonics when $W = P$, but it does have a DC level of $A/2$.



4.2.5 Power and Energy

As a matter of definition, the mean power \overline{p}_x of a periodic $x(t) \sim$ is the Energy over a period P , divided by P , that is:

$$\bar{p}_x = \frac{1}{P} \int_P |x(t)\tilde{}|^2 dt$$

In general, $x(t)\tilde{}$ is built from sine waves of amplitude $M_k = 2|C_k|$, each with a mean power of $\frac{1}{2}M_k^2$, which is $2|C_k|^2$. Each of the two phasors that make up the sine wave contributes *half* of this power, that's a mean power of $|C_k|^2$ for each phasor. It follows that:

$$\bar{p}_x = \frac{1}{P} \int_P |x(t)\tilde{}|^2 dt = \sum_{k=-\infty}^{\infty} |C_k|^2$$

We've used a modulus symbol $||$ on both expressions to mean the *magnitude*, or the vector length. We *must* do that for complex numbers (such as C_k), but its use is optional for real signals (such as a real-valued $x(t)\tilde{}$).

The integral over one period of $x(t)\tilde{}$ is an integral over the $x(t)$ pulse which has a finite width P . The Energy of this pulse (noting that $C_k = \epsilon X(k\epsilon)$) is:

$$E_x = \int_P |x(t)|^2 dt = \epsilon \cdot \sum_{k=-\infty}^{\infty} |X(k\epsilon)|^2$$

This refers to the CFT pair $\{x(t) \leftrightarrow X(f)\}$, and the $X(k\epsilon)$ are samples from $X(f)$. For an arbitrary pair $\{x(t) \leftrightarrow X(f)\}$, we could not express the energy in terms of samples from $X(f)$, but we can do so here because $x(t)$ has finite width P and can be represented by its spectral samples, taken at a spacing of $1/P$ or less.

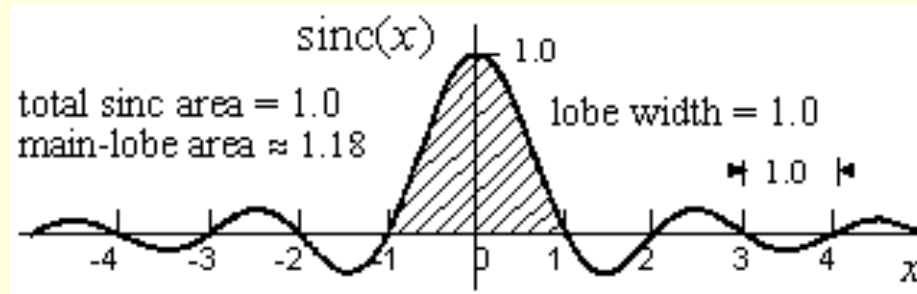
Notice, the summation describing E_x is the *approximate area* under $[X(f)]^2$. In this case, because $x(t)$ has finite width P , it is also the *true area* under $[X(f)]^2$. This is a consequence of sinc interpolation, using some properties of the sinc pulse which we will now briefly introduce.

4.2.6 Properties of the Sinc Pulse

We defined the sinc pulse to be:

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

and its main characteristics are shown here (Fig 6). It is a smooth pulse which, for integer n , satisfies $\text{sinc}(n) = 0$, but with the exception that $\text{sinc}(0) = 1$.



The distance between zero-values is the *lobe width*, which is 1.0. The shaded area above is called the *main lobe*, but it is two lobes wide and its area is about 1.18. The sinc has some notable properties under integration. Firstly, the total sinc area is exactly 1.0, that is:

$$\int_{-\infty}^{\infty} \text{sinc}(x) \cdot dx = 1.0 \quad \text{sinc area is 1.0}$$

Secondly, for integer values of m and n :

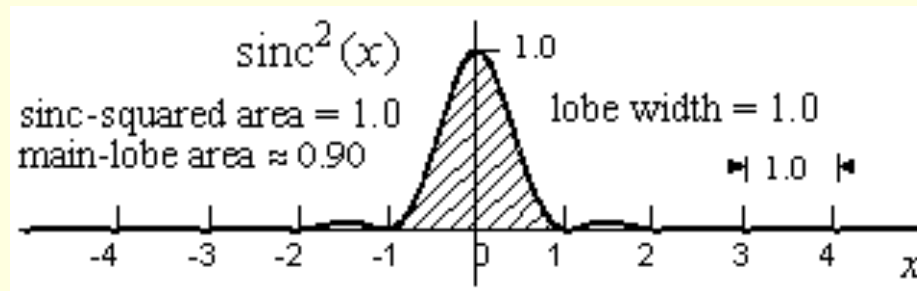
$$\int_{-\infty}^{\infty} \text{sinc}(x-m) \cdot \text{sinc}(x-n) \cdot dx = 1 \text{ for } m=n, \quad = 0 \text{ otherwise}$$

This is the *orthogonality* property of overlapping sinc pulses. Except when they coincide, they interact destructively, and their product area is zero. The sincs coincide when $m=n$, and if we set $m=n=0$, we obtain:

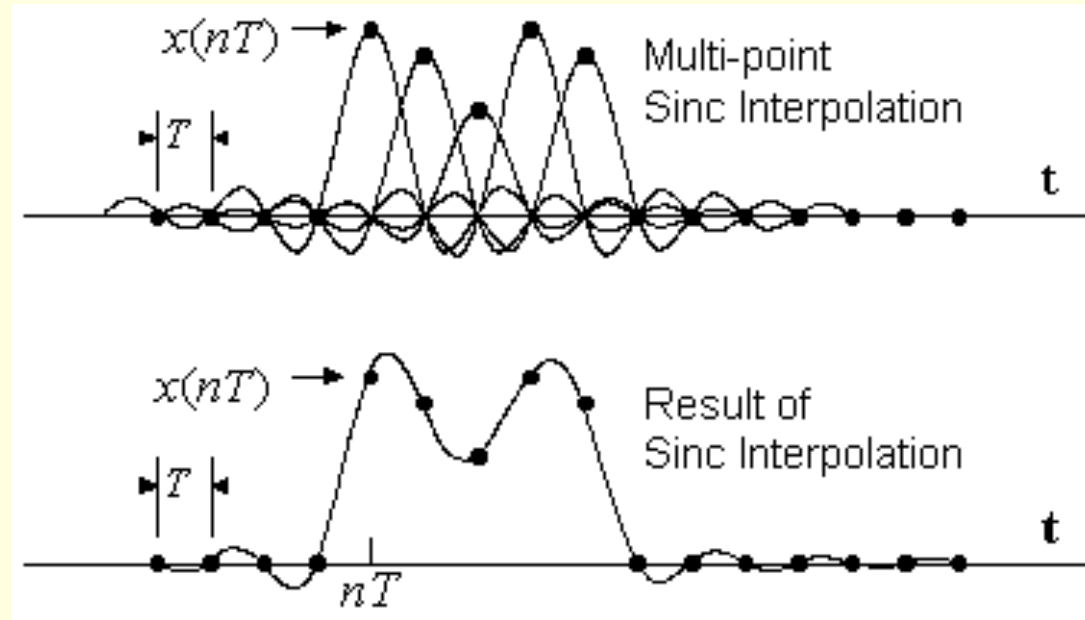
$$\int_{-\infty}^{\infty} \text{sinc}^2(x) dx = 1.0$$

sinc area is 1.0

This sinc-squared pulse is always positive, as shown here (Fig 6).



With these results, we can comment on the area under sinc-interpolated data sets.



The sincs that we see here (Fig 6) have a *lobe* width of T seconds, and a *height* which equals the sample value, $x(nT)$. The nominal sinc area of 1.0 is scaled by T horizontally and by $x(nT)$ vertically. This sinc's area is $T \cdot x(nT)$, the value of an area-sample. The sum of area-samples is our usual approximation to the area under the curve but, for sinc-interpolated samples, *it is also the exact area* under the curve. The same applies to energy calculations, where the estimate would be:

$$E_x = T \cdot \sum_n x(nT)^2$$

Here again, the estimate yields an *exact* result when the samples are replaced by their sinc pulses, but for more than one reason, as follows. The square of the sum of sinc pulses results in sinc^2 terms *and also in cross-product terms* from the various pulse pairs. The sinc^2 terms are exact because $\text{sinc}^2(x)$ has unit area, and the cross-product areas go to zero because

$\text{sinc}(x - n) \cdot \text{sinc}(x - m)$ integrates to zero when $n \neq m$. A little earlier (in [Ch 4.2.5](#)), we wrote the energy of a band-limited time pulse as a sum over spectral samples:

$$E_x = \int_P |x(t)|^2 dt = \varepsilon \cdot \sum_{k=-\infty}^{\infty} |X(k\varepsilon)|^2$$



Here too, the sum gave an exact result, identical to the area under $|X(f)|^2$, because the $X(f)$ of a band-limited time signal is related to its samples by sinc interpolation.



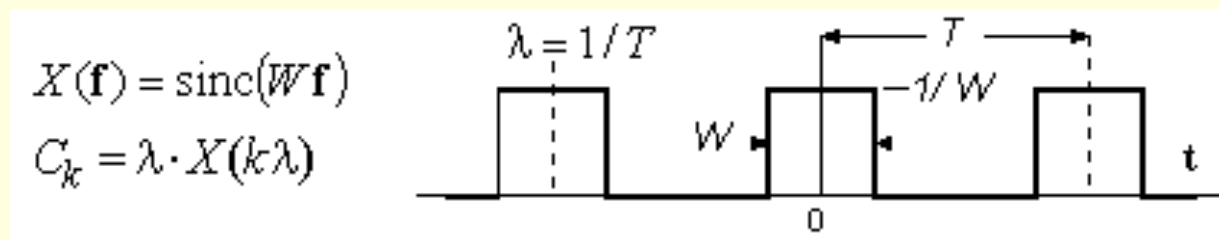


4.3 IMPULSIVE WAVEFORMS

Here we derive some impulsive periodic transform pairs. They are comprised of impulse trains in both domains. Their main significance is as the theoretical sampling functions that convert analogue signals into sampled-data sequences.

4.3.1 Railings Transform Pairs

We now return to the rectangular pulse train, but with a few minor changes included (Fig 6). We've set the pulse height A to be $1/W$, and we've re-defined the pulse spacing with symbol T rather than P .



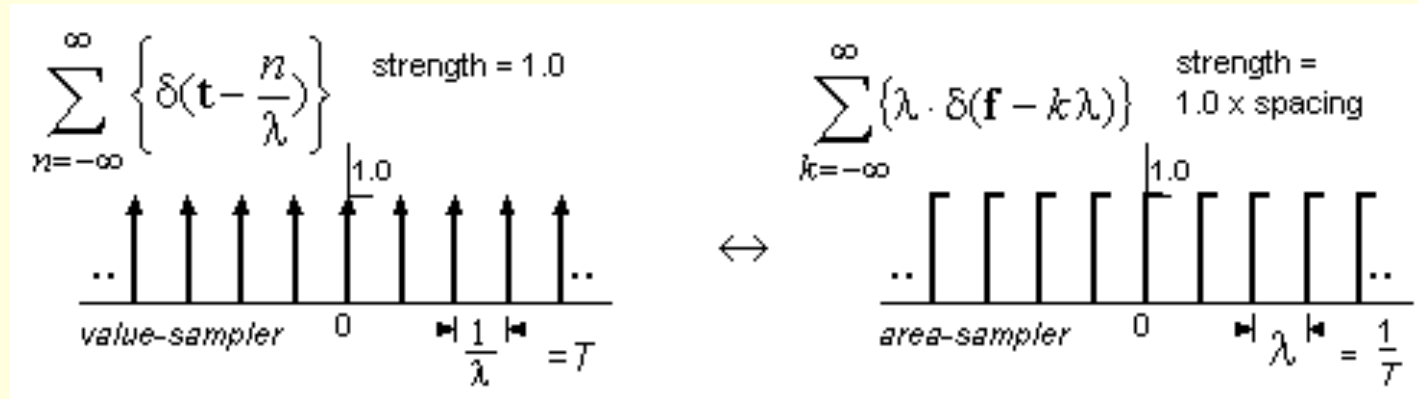
With these changes, $AW = 1$, and then $X(\mathbf{f}) = \text{sinc}(W\mathbf{f})$. The C_k values are its area-samples, and if we define $\lambda = 1/T$ we get:

$$C_k = \lambda \cdot X(k\lambda) = \frac{1}{T} \text{sinc}\left(k \frac{W}{T}\right)$$

The result becomes more interesting if we let $W \rightarrow 0$. Then each pulse becomes tall and narrow, and eventually becomes a *unit-area impulse* when $W = 0$. We then have a string of impulses of strength 1.0 with spacing T , and their spectrum is:

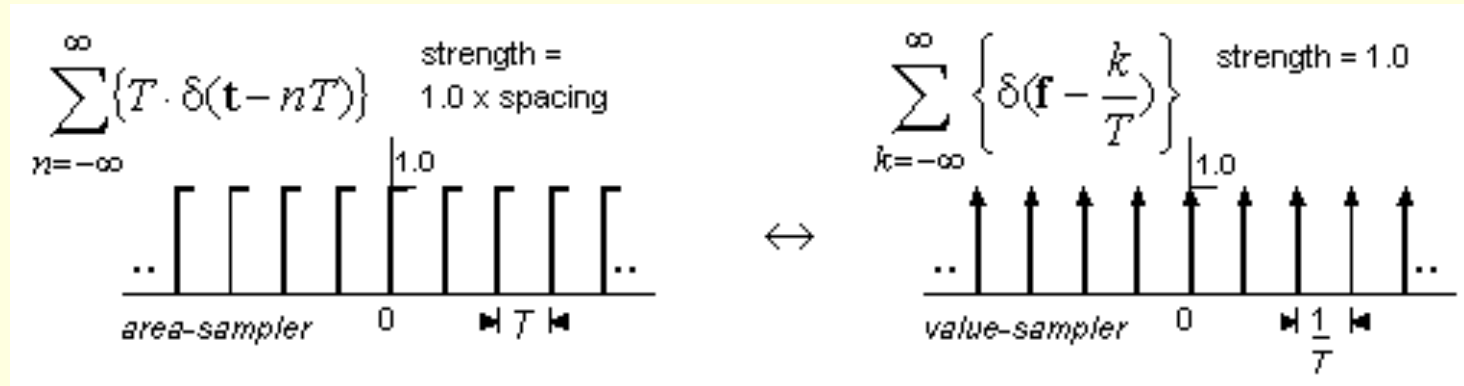
$$C_k = \frac{1}{T} \text{sinc}(0) = \frac{1}{T} = \lambda$$

All coefficients now have the same value, λ . This C_k spectrum is a sequence of numbers, but we can treat each number as an impulse, with impulse strength (or *area*) as given by the numeric value. The result is a highly unusual transform pair that looks like this (Fig 6) :



Both sides are impulse trains, with reciprocal spacing, T and $1/T$, (or $1/\lambda$ and λ if we prefer). By adapting two different impulse symbols, we can conveniently have the same drawn height of 1.0 on both sides. On the left, we have *value-impulses*, of strength equal to drawn height. On the right, we have *area-impulses*, of strength equal to (drawn height \times impulse-spacing). The left side is called a *value-sampler*. The right side is called an *area-sampler*. They are sometimes called "Railings" because of their resemblance to a railing-type fence.

We obtained this from a pulse train with a pulse height $A = 1/W$, which gave a pulse area of 1.0, and this became the impulse strength on the left-hand side. If we had set $A = T/W$ instead, the left-side impulse strength would be T , and the right-side C_k coefficients would be larger by T , with a value of 1.0. This gives us an alternative rendering of the same transform pair, as shown here (Fig 6).



Notice how the value-sampler and the area-sampler have changed places. If a continuous signal $x(t)$ were to be multiplied by the area-sampler on the left, the result would be a set of impulses with spacing T and with impulse strengths $\{T \cdot x(nT)\}$. In numeric form, this is just a set of area-samples from $x(t)$, which is the reason that we called this an area-sampler. By the same reasoning, we could multiply $x(t)$ by a value-sampler to get a set of value samples $x(nT)$. We can summarise our findings by saying:

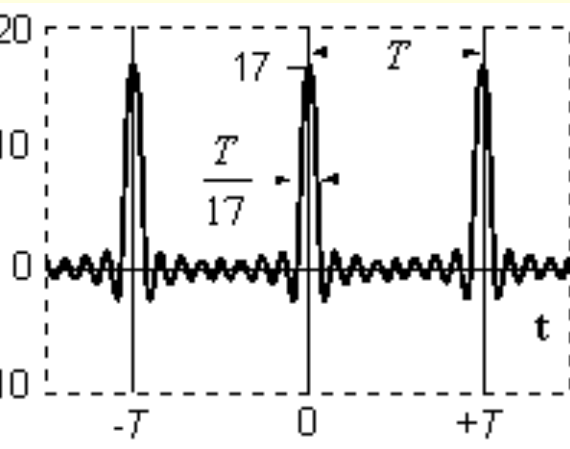
- an area-sampler transforms to a value sampler
- a value-sampler transforms to an area-sampler

These are rather extreme transform pairs. To give some appreciation of how they work, we will sum the harmonics of the most recent pair, in an attempt to build the area-sampler on the left. The right side says that $C_k = 1$ for all k , and so:

$$x(t) \sim \sum_{k=-\infty}^{\infty} C_k \cdot e^{j2\pi k \varepsilon t} = \sum_{k=-\infty}^{\infty} 1.0 \cdot e^{j2\pi \frac{k}{T} t}$$

This reduces to a DC level of 1.0 plus a sum of cosines:

$$x(t) \sim 1.0 + \sum_{k=1}^{\infty} \left(e^{j2\pi \frac{k}{T} t} + e^{-j2\pi \frac{k}{T} t} \right) = 1.0 + \sum_{k=1}^{\infty} 2 \cos \left(2\pi \frac{k}{T} t \right)$$



To see how this $x(t) \sim$ develops, we summed as far as $k = 8$, and got the result shown here (Fig ç). In the limit, we expect a set of impulses, each of area T . This result is tending that way, with a pulse area of about $(17 \times T/17)$ on inspection. At multiples of T , the cosines are in phase and they add up. At all other times they are out of phase, and ultimately sum to zero. In this way, we can account for these Railings transform pairs.

Later on, we may find these pairs of some assistance, in that they can help us to visualise the consequences of sampling.





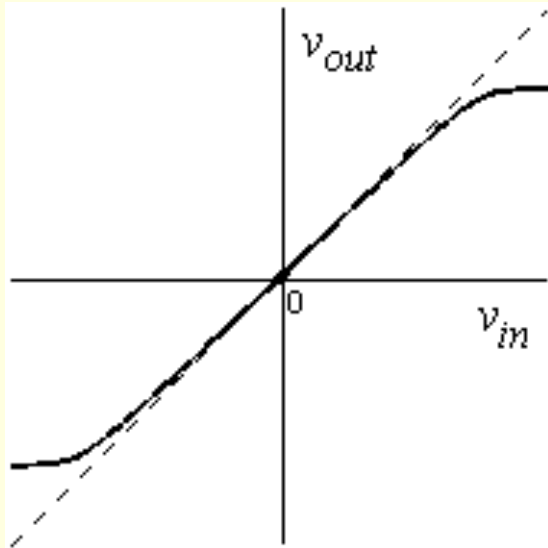
4.4 HARMONIC ANALYSIS

One of the practical uses for Fourier Series is in the measurement of how a sine wave is distorted by its transmission through an amplifier, or through some other analogue system. Because of system non-linearities, there is some distortion of the signal shape at the output. It is no longer a pure sine wave, but its frequency has not changed. The distortions will be seen as small harmonic components of the sine-wave frequency. We use the amplitude of these harmonics as a measure of the distortion introduced by the system.

4.4.1 Measuring Harmonic Distortion

On this plot (Fig ζ), the dotted line is a straight line at 45° describing the input-output relationship of an ideal unity-gain amplifier. The solid line is closer to reality. It saturates at the highest output levels, and shows some non-linearity within the signal range. A sinusoidal v_{in} will emerge somewhat distorted as v_{out} . A spectral analysis of v_{out} will show up some harmonic components. The amplitude of these harmonics is a useful measure of the distortion.

The signal v_{out} is the sum of its harmonic components :



$$v_{out} = v_1(\mathbf{t}) + v_2(\mathbf{t}) + v_3(\mathbf{t}) + \dots$$

where $v_1(\mathbf{t})$ is the fundamental, $v_2(\mathbf{t})$ is the 2nd harmonic, etc. The distortion is from the sum of the harmonics, with instantaneous distortion power:

$$p_d(\mathbf{t}) = (v_2(\mathbf{t}) + v_3(\mathbf{t}) + \dots)^2$$

$$= v_2(\mathbf{t})^2 + v_3(\mathbf{t})^2 + 2v_2(\mathbf{t})v_3(\mathbf{t}) + \dots$$

We want only the mean power over a cycle, and we will find that the mean of the cross-product terms like $2v_2(\mathbf{t})v_3(\mathbf{t})$ is zero. (That's because the different harmonics are *uncorrelated* with one another). Consequently

$$\bar{p}_d(\mathbf{t}) = \frac{1}{P} \int_P (v_2(\mathbf{t})^2 + v_3(\mathbf{t})^2 + \dots) .d\mathbf{t} = \overline{v_2(\mathbf{t})^2} + \overline{v_3(\mathbf{t})^2} + \dots$$

where the bar overhead indicates mean value. We typically define the harmonic voltages to be the square-roots of these mean-power terms:

$$V_2 = \left(\overline{v_2(t)^2} \right)^{1/2}, \quad V_3 = \left(\overline{v_3(t)^2} \right)^{1/2}, \quad \dots$$

and the total harmonic distortion, or THD, in per-cent, becomes:

$$\text{THD} = \sqrt{\frac{\overline{p_d(t)}}{\overline{p_1(t)}}} = \frac{\sqrt{V_2^2 + V_3^2 + \dots}}{V_1} \cdot 100\%$$

where V_1 is the rms value of the fundamental. Notice however that we can use peak sine-wave values to get the same result, because all of these waveforms share the same sinusoidal shape.

The distortion due to *one* harmonic component may be of special interest. For example, we can write the distortion due to the second harmonic component as:

$$\text{HD}_2 = \frac{V_2}{V_1} \cdot 100\%$$

EXAMPLE. A spectrum analyser shows the distortion components of a periodic waveform. Relative to the fundamental, the second, third and fourth harmonics are at -24dB , -36dB , and -30dB respectively, while higher harmonics are negligible. Find the 2nd harmonic distortion, and the THD.

Solution: The harmonic voltages, relative to the fundamental, are:

$$\frac{V_2}{V_1} = 10^{-\frac{24}{20}} = 0.063, \quad \frac{V_3}{V_1} = 10^{-\frac{36}{20}} = 0.016, \quad \frac{V_4}{V_1} = 10^{-\frac{30}{20}} = 0.032$$

The 2nd harmonic distortion is at a level of 6.3%. The total harmonic distortion is calculated as:

$$\text{THD} = \sqrt{\left(\frac{V_2}{V_1}\right)^2 + \left(\frac{V_3}{V_1}\right)^2 + \left(\frac{V_4}{V_1}\right)^2} \cdot 100\% = 7.2\%$$

Total Harmonic Distortion, expressed as a percentage, is the popular measure of non-linear distortion for amplifiers, recorders, and other analogue instruments.





5.1 PREAMBLE

A signal, after sampling, becomes a data sequence, and we will look at the effects of sampling of a periodic signal. The rate of sampling must be high enough to avoid *aliasing*, a phenomenon that we will investigate. From the sampling of periodic signals, we will see how spectral replication brings us once again to the DFT, but by a different route from before, and with some helpful new insights to be added. Although we've seen the DFT before now, this chapter brings it much closer from an applications perspective.



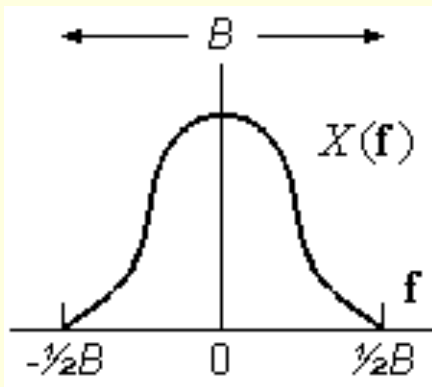


5.2 SINE-WAVE SAMPLING AND IMAGING RULES

If we sample a signal frequently, such that it changes only slightly between samples, then the samples will give a good impression of the underlying shape. Less frequent sampling (larger T) gives a poorer result, with substantial signal variations between samples. It would appear that the signal values between samples are lost forever, but the sampling theorem which we discussed (in [Ch 3.3.4](#)) assures us that this is not necessarily so :

- a signal x

(t) whose double-sided spectrum is band-limited to B Hertz is fully recoverable from its samples taken at a rate exceeding B samples/second.



To meet the *band-limited* condition, the spectrum $X(f)$ that describes $x(t)$ must go to zero for $|f| > \frac{1}{2}B$, (Fig c). We can then use a sampling rate f_s as low as $f_s = B = 2f_{\max}$, where f_{\max} is the highest signal frequency in $x(t)$. This rate of sampling gives us only *two* samples per cycle at f_{\max} . We're assured that this very sparse sampling is sufficient for accurate reconstruction. We'll need some time to appreciate this fully, but we can start the process now by looking at some sampled sinusoidal waveforms.

5.2.1 Sampled Phasors and Sampling Ambiguity

Suppose we take a sine wave of frequency f and initial angle α .

$$x(t) = \cos(2\pi f t + \alpha)$$

Then we sample it at a rate of $f_s = 1/T$ samples/sec :

$$x[n] = x(nT) = \cos(2\pi f nT + \alpha) = \cos(2\pi \frac{f}{f_s} n + \alpha)$$

This result depends only on the ratio (f/f_s), a *normalised* measure of frequency, to which we will assign the symbol f :

$$f = \frac{f}{f_s} \quad \text{cycles per sample-interval (normalised freq)}$$

Then: $x[n] = \cos(2\pi f n + \alpha)$

With this step, we've moved from \mathbf{f} and \mathbf{t} to a normalised f and to an integer time parameter n , which is *time expressed as a sample count*. This is a routine transition when we perform Analog to Digital (A/D) conversion on a signal:

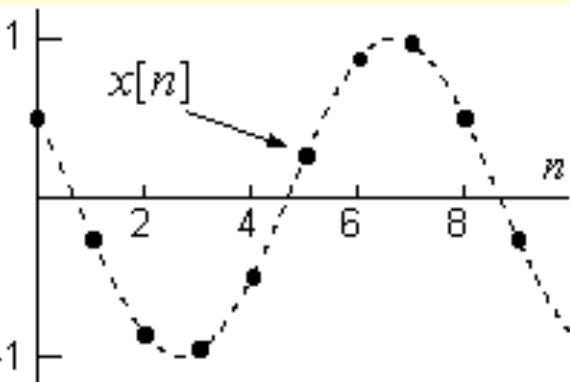
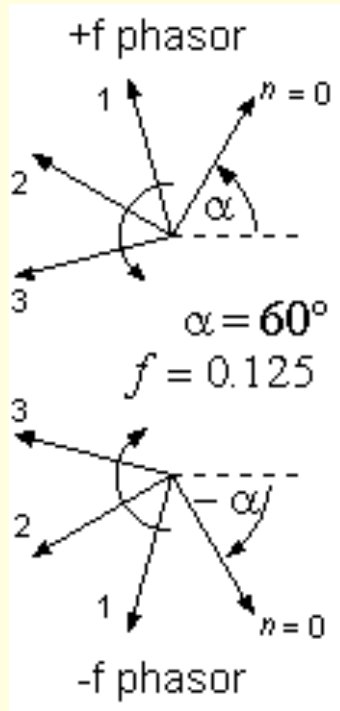
$$\text{(cycles/sec)} \times \text{secs} \quad \mathbf{ft} \xrightarrow{\text{A/D}} \mathbf{fn} \quad \text{(cycles/sample)} \times \text{samples}$$

It establishes a new time scale, in which the sample-interval replaces the second as "one unit of time". Once a signal is sampled, its best to use f rather than \mathbf{f} because it allows us to process the data *independently of the actual rate of sampling*. We now think of the signal frequency f as *cycles per sample interval* and, according to the sampling theorem, it can be as high as 0.5, (equivalent to $0.5 \times \mathbf{f}_s$ samples/sec) but not any higher. The range of signal frequencies becomes ($0 < f < 0.5$) for a single-sided spectral view, or ($-0.5 < f < 0.5$) for a two-sided spectral view.

We can show rather simply why these limits must apply. We begin by expressing $x[n]$ as a sum of two phasor terms, just as we did earlier (i [Ch 2.2.3](#)) for a continuous sinusoidal $x(\mathbf{t})$:

$$x[n] = \cos(2\pi f n + \alpha) = \frac{1}{2} e^{j(2\pi f n + \alpha)} + \frac{1}{2} e^{-j(2\pi f n + \alpha)}$$

But these are *sampled* phasors, and we can view the anti-clockwise ($+f$) phasor and the clockwise ($-f$) phasor at successive sample positions as they rotate (Fig ç). We should recognise f as the *angle between successive samples*, expressed as a fraction of a cycle. The drawn angle here (Fig ç) is 45° , and it describes a frequency of $f = 1/8$ cycles/sample = 0.125. Also shown is the initial angle α when $n = 0$, drawn as 60° on the diagram. The ($+f$) phasor starts at $+\alpha$, while the ($-f$) phasor starts at $-\alpha$. The sum of the two phasors at any instant is a *real* number, and it traces out the sampled sinusoid, $x[n]$. The $x[n]$ that these phasors describe are shown here (Fig í).



We now consider what happens if we take the signal frequency f above, and far beyond, the limit of $0.5 f_s$. This will take f far beyond 0.5 . At $f = 0.5$, there is 180° between samples. At higher frequencies, a confusion sets in, as follows.

Our drawing showed the $(+f)$ phasor at $f = 0.125$, (Fig 6), or $1/8$ cycle between samples. We could not distinguish this result from $f = 1.125$, which includes an extra (unseen) cycle between samples. The same is true for $f = 2.125$, and for $f = 3.125$, and so forth.

All this presumes that the $(+f)$ and $(-f)$ rotations are as indicated. But the samples alone do *not* convey the direction of rotation. We still get real-valued $x[n]$ samples if *both* of the drawn directions are reversed. We could add 1.0 (cycles/sample) to the $(-f)$ frequency of $f = -0.125$ causing this phasor to change direction to a new $(+f)$ frequency of 0.875. This phasor is *presumed* to rotate at $f = -1/8 = -0.125$, but it cannot be distinguished from phasors at $f = +7/8 = +0.875$, or at $f = 1.875$, or at $f = 2.875$, and so forth. As part of this reversal, the $(+f)$ phasor at $f = +1/8 = +0.125$ becomes the $(-f)$ phasor with frequencies of $f = -7/8 = -0.875$, $f = -1.875$, $f = -2.875$, and so forth. A further consequence of reversal is that the new $(+f)$ phasor has an initial angle of $-\alpha$ rather than $+\alpha$.

It turns out that the presumed f value ($f = 0.125$ in this case) is only the lowest of many possible *sine-wave* frequencies that would give us the same set of samples. Sine-wave frequencies, unlike phasor frequencies, are inherently positive. For directions of rotation as drawn, the possibilities are:

$$f = f_0, 1 + f_0, 2 + f_0, 3 + f_0, \dots \quad (\text{with } f_0 = 0.125 \text{ and initial angle } \alpha)$$

These are the result of adding 1, 2, 3 etc, to the phasor at $+f_0 = +0.125$. If the directions of rotation are interchanged, we get more possibilities:

$$f = 1 - f_0, 2 - f_0, 3 - f_0, \dots \quad (\text{with } f_0 = 0.125 \text{ and initial angle } -\alpha)$$

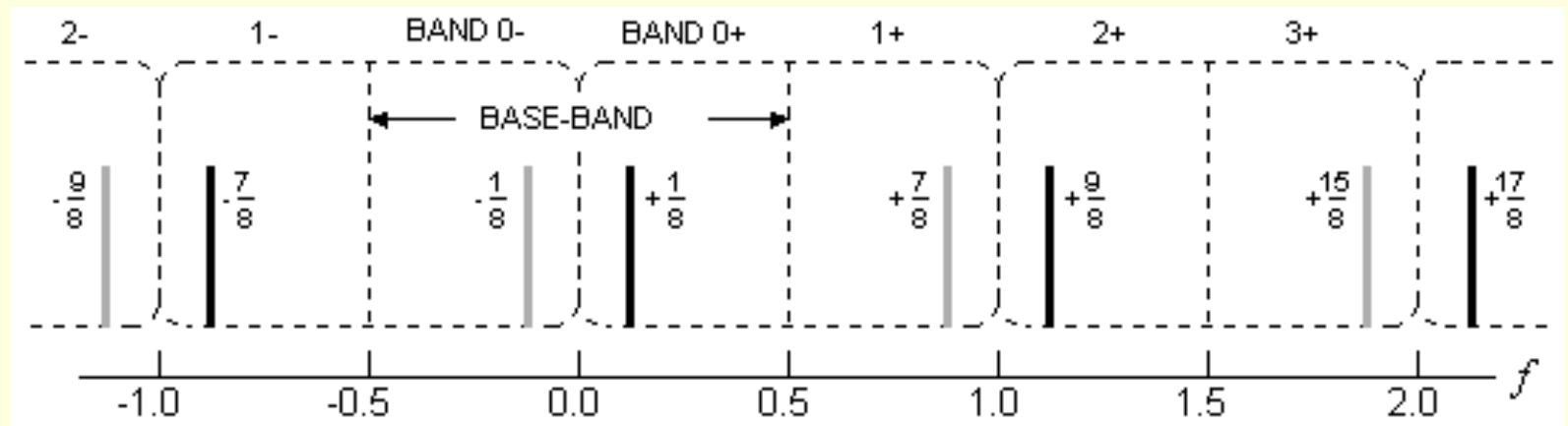
These are the result of adding 1, 2, 3 etc, to the phasor at $-f_0 = -0.125$. The set of all possible sine-wave frequencies becomes:

$$f = f_0, 1 \pm f_0, 2 \pm f_0, 3 \pm f_0, \dots \quad (\text{for } 0 < f_0 < 0.5)$$

The eligible frequencies above f_0 are called the *images* of f_0 . The actual frequencies in samples/sec (or Hertz) are just these f values multiplied by f_s . If a signal $x(t)$ could contain more than one of these frequencies, then its samples $x(nT)$ cannot distinguish one from another. From the samples alone, there is no way to say from which of the images they originated. The samples describe the *sum* of image components, and there is no way to separate them. This describes the *ambiguity* that surrounds a signal after sampling. The only way to avoid it is to guarantee, in advance, that $x(t)$ can contain only one (usually the lowest) of a whole set of image frequencies. In other words, $x(t)$ must be band-limited, and the limits are the same as was predicted by the Sampling Theorem. This brings us to the subject of image bands.

5.2.2 Normalised Frequency and Image Bands

This spectral diagram (Fig 5.2) shows the various images that we described. Each line describes a phasor with a (fractional) frequency label attached. The phasors at $(\pm 1/8)$ describe the f_0 sinusoid. The phasors at $(\pm 7/8)$ are for the $(1 - f_0)$ sinusoid. The phasors at $(\pm 9/8)$ are for the $(1 + f_0)$ sinusoid. This pattern is repeated at higher frequencies. Notice, the black $(+1/8)$ phasor is replicated up and down the spectrum with a spacing of 1.0 on the f scale. The gray $(-1/8)$ phasor is replicated in an identical manner. Taken together, the (black + gray) phasor pair at $(\pm 1/8)$ is *replicated* at intervals of 1.0 to form a periodic spectrum. This concurs with our earlier finding that sampling in time equates to spectral replication (see [Ch 3.3.2](#)).



The dotted frames on the diagram are a set of one-period windows, each of width 1.0, and each divided in two by a vertical line at the centre. This divides the spectrum into numbered *bands*, with the band numbers overhead, such that each band contains one of the possible frequencies. The range $(-0.5 < f < +0.5)$ is band-0, usually called the *base-band*, with half-bands labels (0-, 0+) overhead. The ranges $(-1.0 < f < -0.5)$ and $(+0.5 < f < +1.0)$ define band-1 with half-band labels (1-, 1+) overhead. Similarly for band-2, with labels (2-, 2+) overhead. When a base-band sinusoid $x_0(t)$ of frequency $f = f_s/8$ is sampled at rate f_s , the samples can be interpreted as base-band phasors at $f = \pm 0.125$, or as band-1 phasors at $f = \pm 0.875$, or as band-2 phasors at $f = \pm 1.125$, and so forth. Thus, any of several analogue signals could have given us the same set of samples :

$$x_0(t) = \cos(2\pi \frac{1}{8} f_s t + 60^\circ) \quad \text{the band-0 interpretation}$$

$$x_1(t) = \cos(2\pi \frac{7}{8} f_s t - 60^\circ) \quad \text{the band-1 interpretation}$$

$$x_2(t) = \cos(2\pi \frac{9}{8} f_s t + 60^\circ) \quad \text{the band-2 interpretation}$$

Notice how the initial angle *changes sign* in the odd-numbered bands. If we have only the samples to work with, they could have come from *any* of these analogue signals. These signals, with one signal per band, are just some of the possible "interpretations" of a single set of samples. We can easily confirm this by substituting $t = nT$ in $x_0(t)$, $x_1(t)$ and $x_2(t)$ to find that $x_0(nT)$, $x_1(nT)$ and $x_2(nT)$ are identical sets of data values.

From a set of image frequencies $\{f_0, 1 \pm f_0, 2 \pm f_0, 3 \pm f_0, \text{ etc } \}$, it is usually the base-band frequency f_0 that is the *intended* frequency. To protect this signal from the corrupting effect of images, we must guarantee that the higher image frequencies do not exist in the signal, or are reduced to an acceptable level. To do this, the analogue $x(t)$ *must be filtered in advance by an analogue low-pass filter* so as to remove those parts of $x(t)$ at frequencies higher than $\frac{1}{2}f_s$. In this way, $x(t)$ becomes band-limited to the base-band region. Because image frequencies are sometimes called "aliases", the analogue filter that does this is called an *anti-alias* filter. We'll have more to say about these in due course.





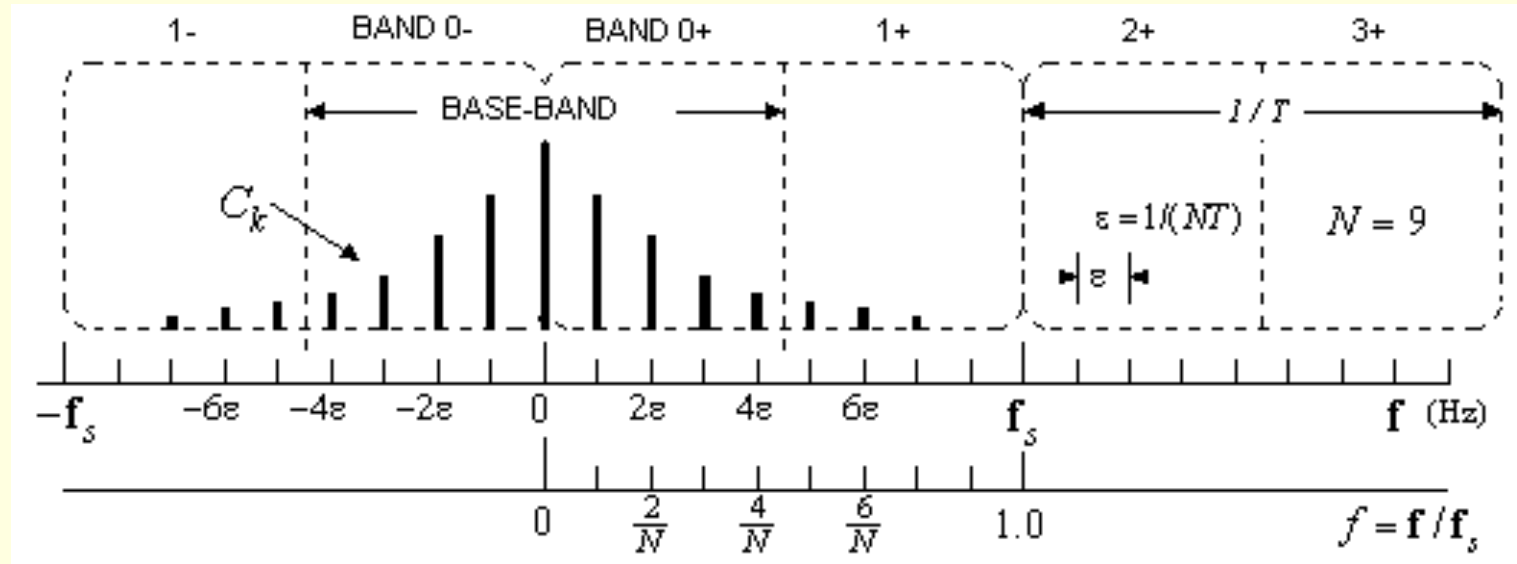
5.3 PERIODIC SIGNAL SAMPLING AND THE DFT

Now that we know the imaging rules, we will apply them to periodic signals, so as to determine the consequences of sampling. We will see how these efforts lead us directly to the Discrete Fourier Transform (DFT).

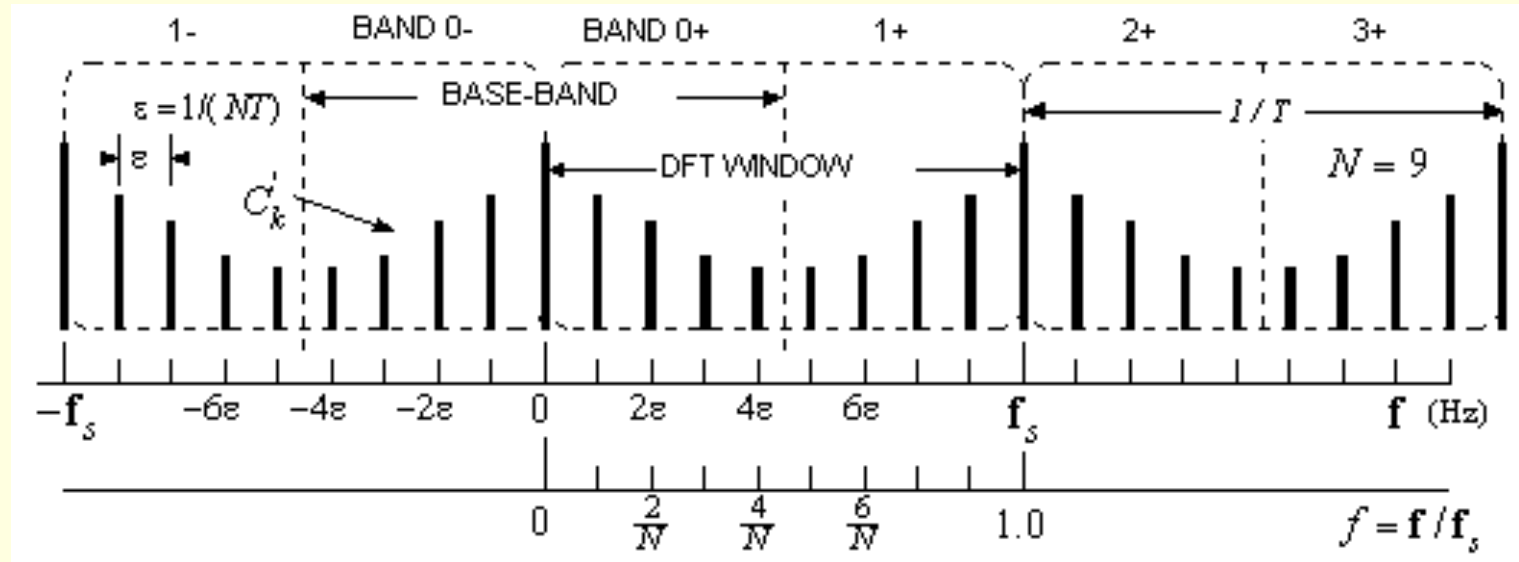
5.3.1 Aliased FS Coefficients

This is the spectral plot (Fig 6.1) of a periodic signal $x(t) \sim$. The heavy black lines are the FS coefficients C_k , with line-spacing $\epsilon = 1/P$ the reciprocal of the waveform period.

The harmonics imply periodicity, but do not account for the band-structure that we've superimposed. These bands describe a *proposed* sampling of this signal at a rate $f_s = N\epsilon$, with $N = 9$ on the diagram. The window width is f_s , which is also $1/T$, where T is the sample-interval. We've also shown the normalised frequency scale. In terms of f , the harmonics now occur at $1/N, 2/N, 3/N$, etc, reaching $N/N = 1.0$ at the sampling frequency.



The spectral lines describe coefficients C_k before $x(t)$ is sampled. When we sample $x(t)$, the spectrum is replicated. The entire set of lines is duplicated at intervals of f_s , or at intervals of 1.0 on the f scale. The result is shown here (Fig 6). The final set of lines is the sum of all the replicated sets.



This final set is periodic as shown. Notice, all of the lines fall into harmonic positions because $f_s = N\epsilon$, with *integer* N . If N were *not* an integer, a much more difficult and complex picture would emerge, and we don't want to deal with that just yet. The result is a new set of coefficients C_k' such that:

$$C_k' = \{ \dots C_{-2N+k} + C_{-N+k} + C_k + C_{N+k} + C_{2N+k} + \dots \}$$

We've shown just a few terms of an infinite set. A more compact description would be:

$$C'_k = \sum_{m=-\infty}^{\infty} C_{mN+k}$$

Because there are N harmonics in a spectral window, the new set C'_k must be periodic, with a period of $N = 9$. It means that C'_k has only N unique values, occupying one spectral window, and the same values are repeated in each and every window.

The action of sampling with $N = 9$ generates N samples over precisely one period of $x(\mathbf{t}) \sim$. We then have N unique sample values and, because N is an integer, the same set of N samples occur for all other periods of $x(\mathbf{t}) \sim$. If N were not an integer, the sample set from $x(\mathbf{t}) \sim$ would not be periodic in N . It might be periodic over some higher number of samples, or, it might not be periodic at all !

With $N = 9$, we have 9 unique time samples described by 9 unique C'_k values. It makes sense that a sampling action which reduces the data to a mere 9 samples in time should have a corresponding effect in frequency, giving us 9 spectral coefficients as well. More generally, this is a transform mechanism that links a set of N time samples to a set of N spectral coefficients. It is a Discrete Fourier Transform, or a DFT.

5.3.2 The Discrete Fourier Transform (DFT) -- Again

We've already arrived at the DFT, but by a different route (i [Ch 3.4](#)), and with some difference in notation, as follows:

$$X(k\varepsilon)\tilde{\sim} = \sum_n \{T \cdot x(nT)\tilde{\sim}\} \cdot e^{-j2\pi\frac{k}{N}n}$$

Forward DFT

and

$$x(nT)\tilde{\sim} = \sum_k \{\varepsilon \cdot X(k\varepsilon)\tilde{\sim}\} \cdot e^{j2\pi\frac{k}{N}n}$$

Inverse DFT

with: $1/N\varepsilon = T$ or, equivalently, $1/NT = \varepsilon$.

The indexes n and k normally use the range $0 \dots N-1$, but a range such as $m \dots m+N-1$ (where m is any integer), will do just as well. Both of these DFT expressions perform a *sum over area-samples* to deliver a set of *value-samples*, and both refer to periodic signals $x(\mathbf{t})\tilde{\sim}$ and $X(\mathbf{f})\tilde{\sim}$ which can (optionally) be considered to be the replicated versions of pulses $x(\mathbf{t})$ and $X(\mathbf{f})$. In our Fourier Series discussion, the C_k were the *area-samples* $\{\varepsilon \cdot X(k\varepsilon)\}$ from $X(\mathbf{f})$. By observing that $N \cdot C_k' = N \cdot \varepsilon \cdot X(k\varepsilon)\tilde{\sim} = X(k\varepsilon)\tilde{\sim}/T$, we can re-write the DFT and the IDFT in terms that use C_k' as the spectral quantity:

$$C_k' = \frac{1}{N} \sum_n x(nT)\tilde{\sim} \cdot e^{-j2\pi\frac{k}{N}n}$$

Mathcad CFFT

$$x(nT)^{\sim} = \sum_k C'_k \cdot e^{j2\pi \frac{k}{N}n}$$

Mathcad ICFFT

CFFT and ICFFT are the function names in Mathcad that perform these DFT operations, but they do so using a "fast" computational algorithm. The label CFFT means "Complex Fast Fourier Transform", and ICFFT is its Inverse. The MATLAB definitions, however, are in terms of NC'_k rather than C'_k :

$$NC'_k = \sum_n x(nT)^{\sim} \cdot e^{-j2\pi \frac{k}{N}n}$$

Mathcad N -CFFT, MATLAB fft

$$x(nT)^{\sim} = \frac{1}{N} \sum_k NC'_k \cdot e^{j2\pi \frac{k}{N}n}$$

Mathcad ICFFT/ N , MATLAB ifft

The MATLAB functions are called fft and ifft, again using a fast algorithm. The C'_k notation is fine for Fourier Series work, but a popular alternative uses:

$$x[n] = x(nT)^{\sim}, \quad X[k] = NC'_k$$

and then:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi \frac{k}{N}n}, \quad k = 0..N-1$$

Mathcad N -CFFT, MATLAB fft

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \cdot e^{j2\pi \frac{k}{N}n}, \quad n = 0..N-1$$

Mathcad ICFFT/ N , MATLAB ifft

This notation is a purely numeric one, in which $x[n]$ and $X[k]$ are simply data vectors of length N . We'll use these as our standard DFT/IDFT definitions, and we'll use the $x[n]$ and $X[k]$ symbols throughout most of our DFT work. When dealing directly with sine waves however, we may prefer CFFT/ICFFT with the C_k notation. For now, we'll return to the C_k notation, and to the role of the DFT in a Fourier series setting.

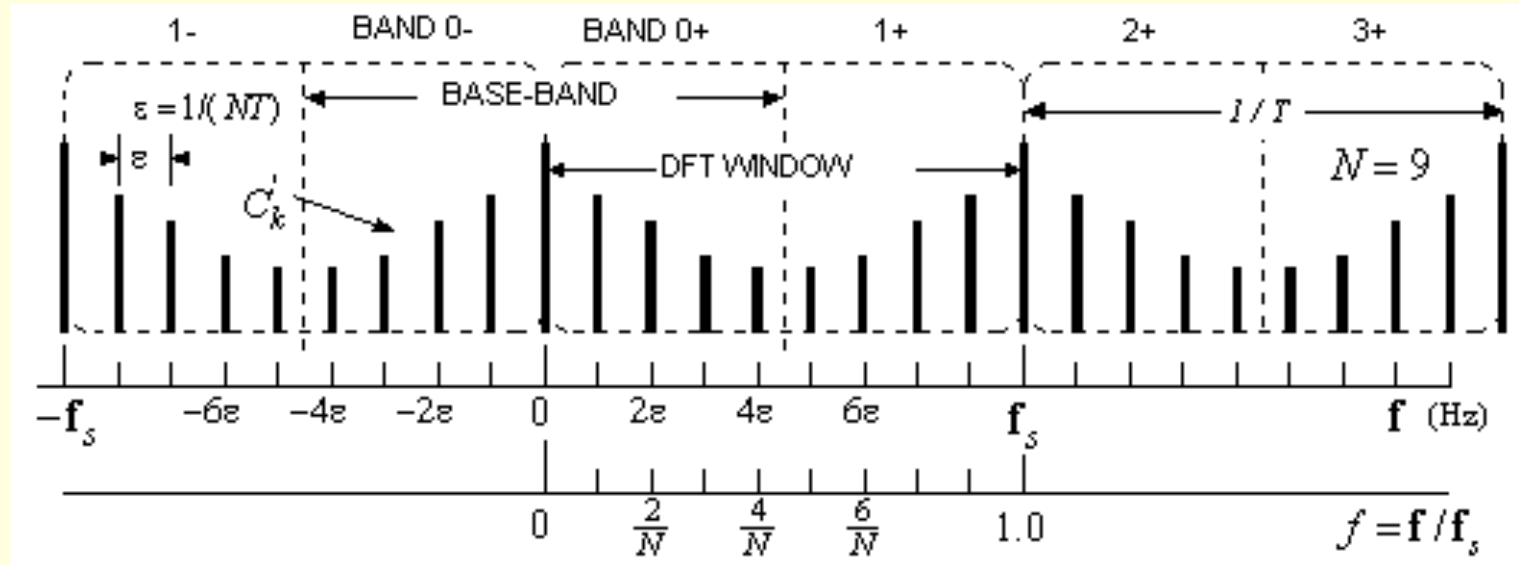
5.3.3 Spectral Symmetry and the DFT

We can use the DFT for our Fourier Series work but, because we use samples $x(nT)~$ rather than the continuous $x(t)~$, the spectrum is inherently replicated, forcing us to work with C_k' rather than C_k . If the replication causes C_k lines to overlap, we have an *aliasing* problem, and the C_k' will be different from C_k . Very often, however, we can *minimise* aliasing effects by using a long DFT. That means using large N so that $T = P/N$ is made smaller. This increases the sampling frequency $f_s = 1/T$, and then the spectral replicas will be further apart and the overlap is thereby reduced.

Most often, the $x(nT)~$ are samples from a *real-valued* time signal, and this leads to spectral symmetries which we have noted before now. On a two-sided spectral diagram, where the harmonics have peak value M_k with angle αk :

$$\frac{1}{2}M_k = |C_k| = |C_{-k}| \quad \text{and} \quad \alpha_k = \arg\{C_k\} = -\arg\{C_{-k}\}$$

This is because the two phasors that build the sine-wave have equal lengths of $\frac{1}{2}M_k$ but their angles are αk for the $+f$ phasor and $-\alpha k$ for the $-f$ phasor. These symmetries still apply for the aliased coefficients that we call C_k' .



Within the baseband therefore (Fig 6), we can say that:

$$C'_{-k} = C'_k^* \quad \text{a conjugate pair}$$

The symbol * denotes "complex conjugate of ..", meaning equal length but the angle changes sign. It follows that if we know the BAND 0+ lines, then the BAND 0- lines are known too, provided $x(nT)$ are real numbers. (On the diagram, only C_0 and C_k for $k = 1..4$ are unique, but the four C_{-k} are complex, with two numeric values for each, so we still need nine spectral numbers to correspond with nine sample values in time).

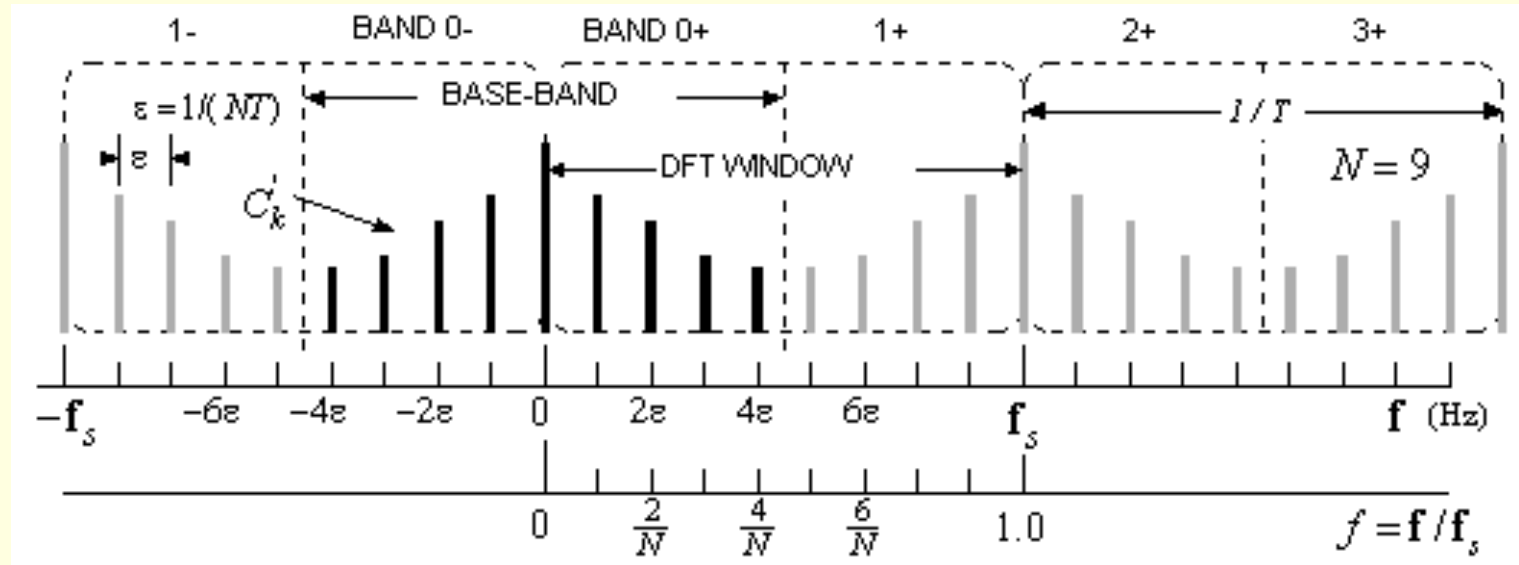
The DFT window (Fig 6), by convention, uses the index range $0..N-1$, so that it extends over BAND 0+ and BAND 1+. We must then remember that BAND 1+ is both the top half of the DFT window and an exact image of BAND 0-. So, for a

normal two-sided spectral view, we must mentally transpose the top-half of the DFT window into the BAND 0- position. This peculiarity stems from our use of $0 \dots N-1$ as the *convenient* DFT summing range. Referring to our diagram with $N = 9$, while it's OK to sum the DFT over $0 \dots 9$, but we'll soon see that the *true* range of $-4 \dots +4$ must be used in any attempt to rebuild $x(t)_{\sim}$ from its samples.

5.3.4 Continuous Signal Reconstructions

It took a possibly unlimited set of C_k coefficients to describe the analogue $x(t)_{\sim}$, but the sampled $x(nT)$ has only N unique sample values and, correspondingly, only N coefficients C_k' . Clearly, much detail is lost when we sample a signal. We cannot re-build a detailed picture of $x(t)_{\sim}$ from C_k' alone. What we can do is to build an *interpretation* of $x(t)$, one that agrees with $x(t)_{\sim}$ at the sample points. In fact, we can build *numerous* interpretations, a different one for every band ! We will demonstrate how this re-construction process works.

BAND-0 Reconstruction. For a BAND 0 reconstruction, we *assume* that $x(t)_{\sim}$ has no components at higher than base-band frequencies. That means using the set of lines which we see here in black (Fig 6).



The resulting reconstruction is:

$$x_0(t) \sim = \sum_{\substack{k < \frac{N}{2} \\ k > \frac{-N}{2}}} C'_k \cdot e^{j2\pi k\epsilon t} \quad (\text{for } N \text{ odd})$$

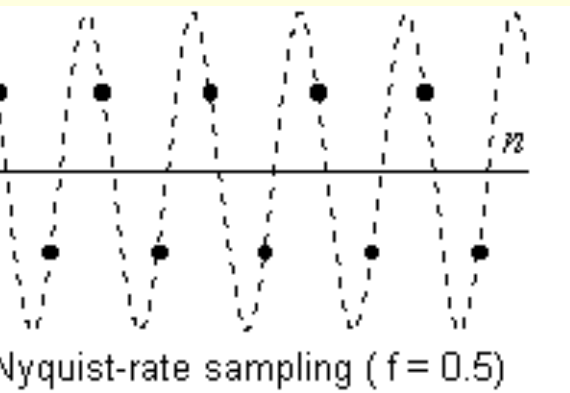
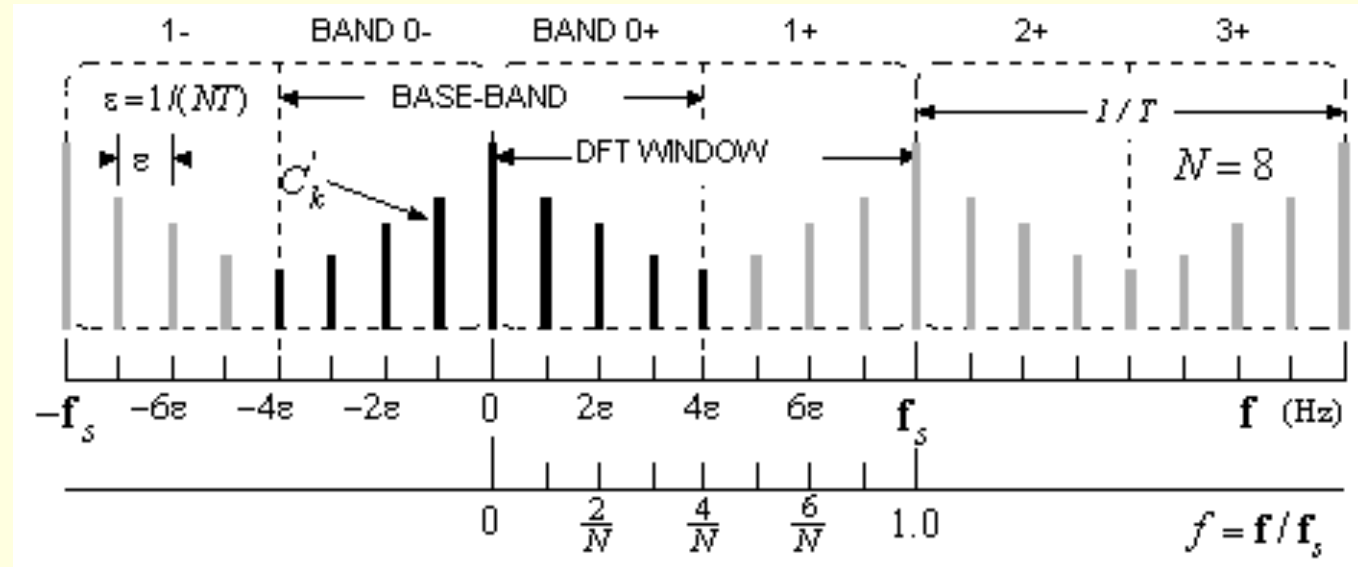
In our example, the sum is over $\{k = -4 \dots +4\}$. This $x_0(t) \sim$ agrees with $x(t) \sim$ at sample points. If $x(t) \sim$ had *no* spectral lines beyond the base-band, then $x_0(t) \sim$ is identical to $x(t) \sim$ everywhere. If $x(t) \sim$ *did* have spectral lines beyond the base-band, then their *base-band images* are included in $x_0(t) \sim$. They will still agree at sample points but, between samples, $x_0(t) \sim$ will vary less rapidly than $x(t) \sim$ because the higher frequencies in $x(t) \sim$ have been given a lower-frequency interpretation.

BAND-0 Reconstruction when N is Even. There are some notable differences when N is even, as for the $N = 8$ case shown here (Fig 6).

We now have a spectral line at the Nyquist frequency, at $f = 4\epsilon$ in this example. When the original C_k lines at $\pm 4\epsilon$ are replicated at intervals of $f_s = 8\epsilon$, they replicate with *self-overlap*, and a consequent "doubling-up" effect. As a result, the re-construction must use only *one-half* of the C_k' values at $\pm 4\epsilon$. We get the following re-construction formula:

$$x_0(t) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} R_k' \cdot e^{j2\pi k\epsilon t} \quad \text{with} \quad \begin{aligned} R_k' &= C_k', & |k| < \frac{N}{2} \\ R_k' &= \frac{1}{2}C_k', & |k| = \frac{N}{2} \end{aligned}$$

The summing range is $\{-4 \dots 4\}$ in this example. Again, $x_0(t)$ agrees with $x(t)$ at sample points, provided that any non-zero component at the Nyquist frequency has been handled correctly !



The difficulty at the Nyquist frequency is illustrated here (Fig ζ). At only 2 samples per cycle, the samples could fall on signal peaks, or on zero-crossings, or somewhere in between as we have illustrated. Therefore the sample values do not tell us the true phase of the signal. Suppose, in the example, that the original signal data included the following:

$$C_4 = 1 \angle \alpha, \quad C_{-4} = 1 \angle -\alpha$$

This describes a sinusoid of peak value 2 and angle α at frequency 4ϵ . Replication at intervals 8ϵ then causes C_{-4} to be superimposed on C_4 and vice versa. The resulting aliased coefficients are:

$$C'_4 = C'_{-4} = (1\angle\alpha + 1\angle-\alpha) = 2\cos\alpha$$

The reconstruction then yields:

$$x_{4\epsilon}(\mathbf{t}) = \frac{1}{2}C'_{-4} \cdot e^{-j2\pi 4\epsilon\mathbf{t}} + \frac{1}{2}C'_4 \cdot e^{j2\pi 4\epsilon\mathbf{t}} = (2\cos\alpha) \cdot \cos(2\pi 4\epsilon\mathbf{t})$$

This does *not* describe the true sinusoid with an amplitude of 2 at angle α . This reconstruction describes an amplitude of $2\cos\alpha$, with zero phase. It assumes that the sample points are at the *peak* of the signal, whether that be true or not. It does this of necessity, because no better information is available when sampling at this frequency.

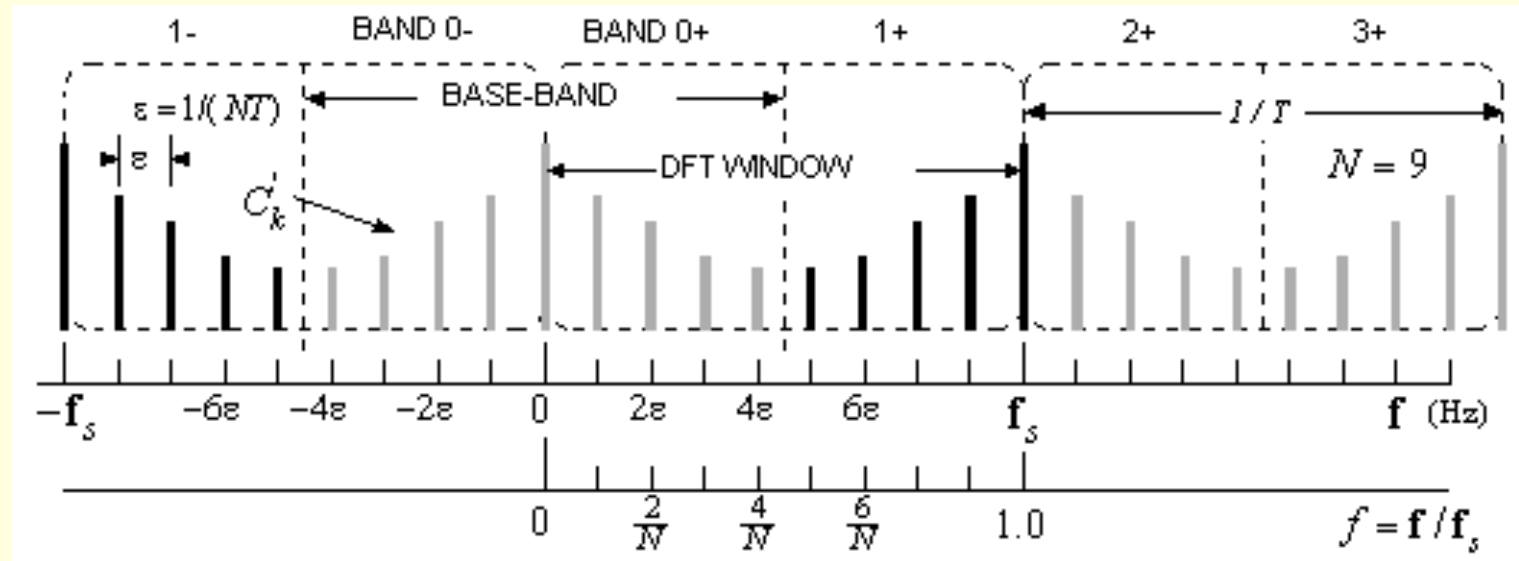
This constitutes a failure to recognise a signal correctly at the Nyquist frequency, and it is a fundamental sampling limitation. Notice that, because of this "phase-blindness" at the Nyquist frequency, the coefficient C_k' , at $k = N/2$, will always be real-valued, so long as $x(\mathbf{t})$ is a real-valued time signal.

BAND-1 Reconstruction. For a BAND 1 reconstruction, we use the black spectral lines as shown here (Fig 6) for $N = 9$. These are the BAND 1 lines. The lines at $\pm f_s$ will self-overlap on replication, so they must be used at one-half of their face value:

$$x_1(t) = \frac{1}{2}C'_{-9} \cdot e^{-j2\pi 9\epsilon t} + \frac{1}{2}C'_9 \cdot e^{j2\pi 9\epsilon t} + \sum_k C'_k \cdot e^{j2\pi k\epsilon t}$$

$$k = \{-8 \dots -5, 5 \dots 8\}$$

We still have agreement at sample points, meaning that $x_1(nT) = x_0(nT) = x(nT)$, but $x_1(t)$ varies much more rapidly between sample points. At the frequency $f_s = 9\epsilon$, we have only one sample per cycle. The same value recurs over and over, and is indistinguishable from a DC component at $f = 0$.



The same reconstruction process applies in higher bands also. This is significant for band-pass signal processing where, under proper conditions, a high-frequency band-pass signal can be sampled and, with no further processing, these samples can serve as a base-band image of the signal. The intermediate "demodulation" step is thereby rendered unnecessary !



This chapter has helped to broaden our understanding of the DFT, and we are ready to proceed to some practical DFT application topics. That will be our focus in the chapter just ahead.





6.1 PREAMBLE

We'll begin our work with the DFT in this chapter. We'll use the fast algorithm, the FFT, to speed the computation. We'll show how the DFT can display the spectra of sine waves and of periodic data, and how it obeys the imaging rules. We'll write the DtFT in terms of normalised frequency, and we'll use it to find the spectra of number sequences. Because filters are described by number sequences, the DtFT can compute a filter spectrum, and we'll see how the FFT can do this work more quickly, and more systematically.





6.2 DFT WINDOWS AND THE FFT ALGORITHM

This section describes the scaling of DFT windows, and the fast DFT algorithms that we have available to us.

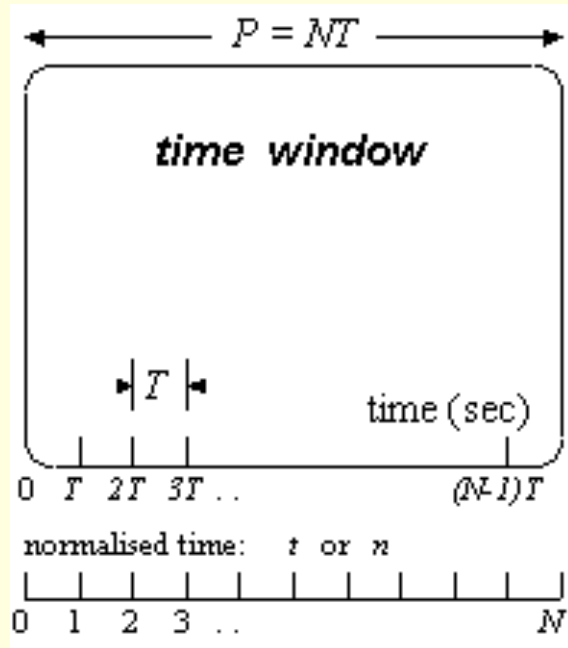
6.2.1 DFT Time and Frequency Windows

It is vital that we understand the *scaling* of DFT time and frequency windows. The scaling of *both* these windows is *fully defined* by just two parameters, T and N . We use other symbols as well, but they are not essential.

The time-window scaling is shown here (Fig 6.2.1). Its main parameters are sample interval T , and window width $P = NT$, both in seconds. The sample set runs from 0 to $(N - 1)T$, a total of N points. Because of periodicity, the sample at time NT is identical to the sample at time 0.

A normalised time scale t (or n) is drawn underneath. On the normalised scale, we consider that $T = 1$ sample interval, and that $\{t = 0, 1, 2 \dots N-1\}$ counts the *sample intervals* over the window. When $t = N$, the cycle starts to repeat. The corresponding (normalised) frequency variable is f , with units of "cycles per sample interval", or just "cycles per sample". Although we use f quite widely, we prefer the integer symbol n , rather than t , as our normal time counter. We do this to

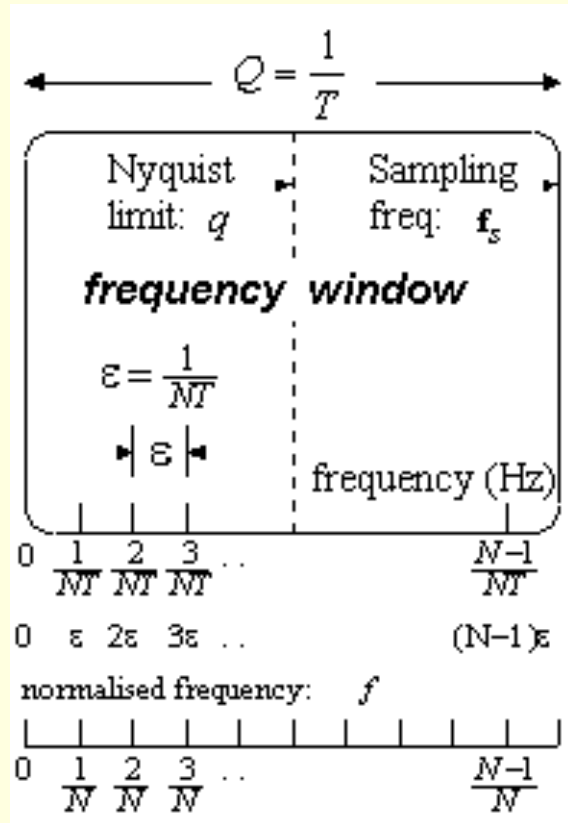
conform with popular usage, but we risk losing sight of n as the normalised time variable that it is.



The frequency-window is shown here (Fig 1). Its main parameters are the sample interval $\epsilon = 1/NT$, and the window width $f_s = 1/T$. Clearly, $f_s = N\epsilon$, and both ϵ and f_s are measured in Hertz (cycles per second). Occasionally, we use the symbol Q , noting that $Q = f_s = N\epsilon$.

The frequencies at which spectral lines are calculated are the frequencies $0, \epsilon, 2\epsilon, \dots, (N-1)\epsilon$, where $\epsilon = 1/(NT)$. These are often referred to as "frequency bins", in the sense that the total signal power is distributed over a set of N "bins". We will often use the term "bin" in this connection.

The normalised frequency scale f is also shown (Fig 2). It has a range of 0 to 1, with a normalised bin spacing of $(1/N)$. The normalised bin frequencies are the set of values $f = (k/N)$, with $k = 0, 1, 2, \dots, N-1$. We can also treat the bins as *numbered* bins, with k as the bin number, ranging from $k = 0$ up to $k = N-1$, for a total of N frequency-bins.



The scaling of the time and frequency windows is such that the *window-width* in one domain is the *reciprocal* of the *sample-interval* in the other domain :

$$P = \frac{1}{\varepsilon} \quad \text{and} \quad Q = \frac{1}{T}$$

The DFT time and frequency scales, as presented above, must be clearly understood. These scales are the first, and perhaps the most important, requirement for a good understanding of DFT practice.

By choosing a DFT summing range of $0 \dots N - 1$, we choose a frequency window that covers BAND 0+ and BAND 1+ . The lines in BAND 1+ are a copy of the BAND 0- lines. For example, with $N = 8$, a bin-3 sinusoid has phasors at $(-3\epsilon, +3\epsilon)$, but the phasor at -3ϵ is not visible. Instead, we see its BAND 1+ image at $-3\epsilon + 8\epsilon = 5\epsilon$.

The spectra of real signals have magnitude-even and phase-odd symmetries. It follows that the upper half of the DFT window (the BAND 1+ area) offers no new information. It contains the BAND 0- data, and it obeys the symmetry rules.

6.2.2 Using FFT Algorithms

The "Fast Fourier Transform", or FFT, is a fast algorithm for the DFT, nothing more. It computes the Forward and Inverse DFT summations which we presented, but it can do so far more quickly than the direct method. Actually, there are several variations on this algorithm, but that need not concern us. The FFT algorithm works best when N is a power of 2, such as $N = 32$ or $N = 1024$. Normal DFT evaluation computes N values of $X[k]$, each of which uses N complex multiplications for its summation over N terms. That's a total of N^2 multiplications, and it can take some time to compute. In comparison, the FFT needs only $\frac{1}{2}N \cdot \log_2 N$ multiplications. To illustrate the saving, when $N = 1024$, the number of multiplies is reduced from $N^2 = 1048576$ to $\frac{1}{2}N \cdot \log_2 N = 5120$, a reduction of more than 200 times. Not surprisingly then, the various software packages use a fast algorithm, but they do not restrict N to powers of 2. They can handle any value of N (and pay some time penalty), and we will avail of this flexibility in our examples, using DFT lengths such as $N = 20$, which is not a power of 2, but it helps keep the numbers simple.





6.3 DFT SPECTRA OF PERIODIC SIGNALS

When working with sine waves, we prefer to see a spectral view that shows phasor values (C_k or C_k') directly. We will therefore use:

$$C_k' = \frac{1}{N} \sum_n x(nT) \cdot e^{-j2\pi \frac{k}{N}n}$$

Mathcad CFFT

$$x(nT) \sim = \sum_k C_k' \cdot e^{j2\pi \frac{k}{N}n}$$

Mathcad ICFFT

We'll use CFFT to find the spectra of signals that are sums of sinusoids. We'll also see how to re-build a signal, using ICFFT, when the spectrum is known. We'll see the evidence of imaging, first for sine-wave signals, and later from the spectral aliasing of a sampled periodic waveform.

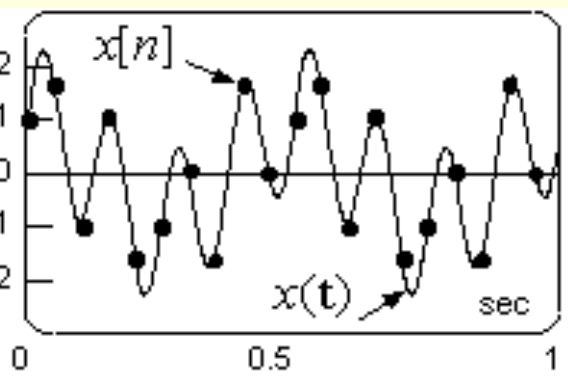
6.3.1 Sinusoids at bin frequencies

We'll start with some signals which we will sample at intervals of $T = 0.05$ sec, for a sample rate of $f_s = 1/T = 20$ Hz. Then, by choosing a DFT length of $N = 20$, we get 20 bins from DC up to f_s , so the spectral bin spacing (conveniently) becomes 1 Hz. The time window width is $NT = 1$ sec. For our first trial, we choose:

$$x(t) = \cos(2\pi 2t) + 1.4 \sin(2\pi 8t)$$

After sampling:

$$x[n] = x(nT) \quad \text{for } n = 0 \dots N-1$$

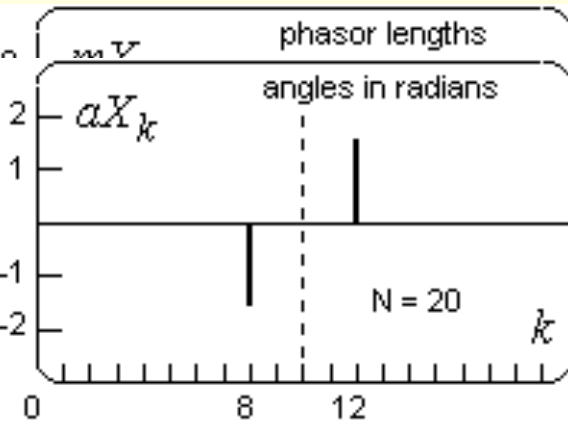


The signal $x(t)$ and the samples $x[n]$ are plotted here (Fig 5). We have a 2 Hz tone and an 8 Hz tone. Although the sampling is evidently *sparse*, the signal is inside the baseband, which extends as far as $\frac{1}{2}f_s = 10$ Hz. It follows that there will be no spectral overlaps on replication, so we can be sure that $C_{k'} = C_k$.

The $x[n]$ values become our time vector x of length N . We obtain the spectral vector X as $X = \text{CFFT}(x)$, also of length N , and this contains the C_k values.

Because C_k are complex, we must use separate magnitude and angle plots:

$$mX_k = |X_k|, \quad aX_k = \arg\{X_k\} \quad \text{for } k = 0 .. N-1 \dots$$

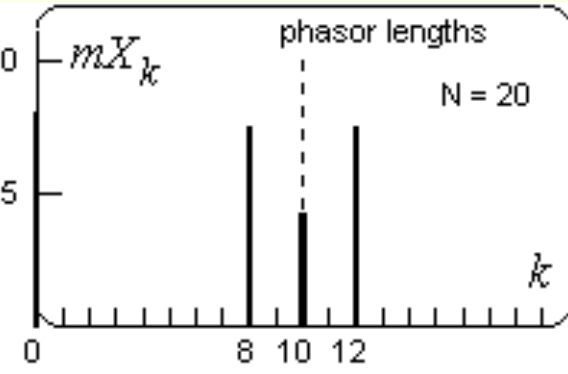


The results are shown alongside (Fig ζ). They show phasors in bins 2 and 8, for the 2 Hz and 8 Hz components. The -2 Hz phasor is in bin $-2+N = 18$. The -8 Hz phasor is in bin $-8+N = 12$. We can also view these as BAND 1+ phasors, as the *images* located at 12 Hz and 18 Hz. The phasor lengths are 0.5 and 0.7, or one-half of the sine-wave amplitudes, as expected. The angle plot (Fig ζ) shows zero phase, $\alpha = 0$, for the 2 Hz cosine and an angle of $\alpha = -\pi/2 = -1.57$ rads for the 8 Hz sine. But the -8 Hz phasor angle is $-\alpha = +\pi/2$, as seen in bin 12. All of this agrees with our earlier discussion on sine waves (i [Ch 5.2](#)).

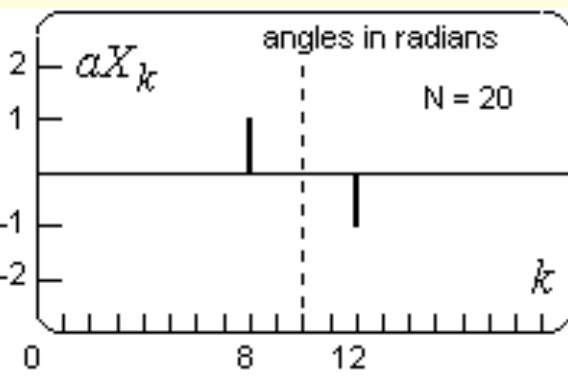
We'll follow this with a further example:

$$x(t) = 0.8 + 1.5 \cos(2\pi 8t + 1.0) + 0.6 \cos(2\pi 10t + 0.8)$$

We have a DC level of 0.8, an 8 Hz tone and a 10 Hz tone, but the 10 Hz tone is at the Nyquist frequency. Following the same procedure as before, we get these results (Fig ζ). The DC level is seen at its *true* value of 0.8 in bin 0, and its angle is



inherently zero. The 8 Hz tone shows up in bins 8 and 12, with phasor lengths of $1.5/2 = 0.75$, and with phasor angles of ± 1 radian, all as expected. The 10 Hz tone is mis-interpreted. It comes across as a pure cosine of peak value $0.6\cos(0.8) = 0.418$. The length of X_{10} is 0.418, the apparent peak value, twice as long as normal phasor-lengths, which are only half the peak value. This is because both phasors at the Nyquist boundaries of ± 10 Hz appear in *the same* bin located at $N/2 = 10$, where they overlap and add, with a loss of phase information. The reported angle is zero, even though the true signal angle was 0.8 radians. This is the "phase blindness" that we expect to see at the Nyquist frequency.



To summarise our findings, the pure cosine is the "zero-angle" sinusoid, and the general sinusoid in bin number k is:

$$x_k(\mathbf{t}) = M_k \cos\left(2\pi \left[\frac{k}{N} \mathbf{f}_s\right] \mathbf{t} + \alpha_k\right)$$

The brackets [] contain the frequency in Hz. To sample, we set $\mathbf{t} = nT = n/f_s$ s and the sample set becomes:

$$x_k[n] = M_k \cos\left(2\pi \frac{k}{N}n + \alpha_k\right)$$

Time is counted by n (in samples) and the normalised frequency is k/N . This sinusoid is described by phasors in bin k , and in bin $-k+N$, at phasor lengths of $\frac{1}{2}M_k$ and with phasor angles of $\pm\alpha_k$. A pure sine is an $x_k(t)$ with initial angle $= -\pi/2$. The DC component, and any component at $\frac{1}{2}f_s$, are treated differently, as noted above.

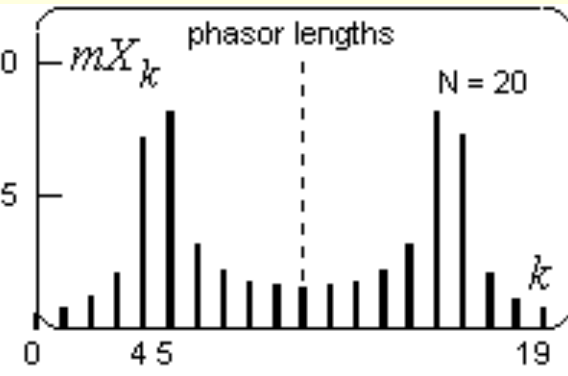
The story so far is of a very well-behaved transform which accurately identifies the sine-wave tones that we give it. The reality is more complex, as the next example demonstrates.

6.3.2 Sinusoids at non-bin frequencies

Everything worked well for sinusoids at bin-frequencies. Using the same DFT length and sample interval ($N = 20$, $T = 0.05$), we'll now try the signal:

$$x(t) = 2.0 \cdot \cos(2\pi 4.5t + 1.0)$$

The frequency is mid-way between bin 4 and bin 5. The spectral magnitude plot that emerges looks like this (Fig 9). We might hope to see two lines, each of length 1.0, but this is what we get instead. Even the longest lines are around 20% too short, and more seriously, we have spectral lines at all the other bin frequencies. This sine wave is "making itself felt" at frequencies far removed from the true frequency. It's a problem known as *spectral leakage*. It puts an end to our hopes of a well-behaved DFT (unless we can guarantee that only bin frequencies will occur). It's too soon to deal with this problem now. Later on, we'll find ways to reduce the harmful effects that we see here.



6.3.3 Checking out the Imaging Rules

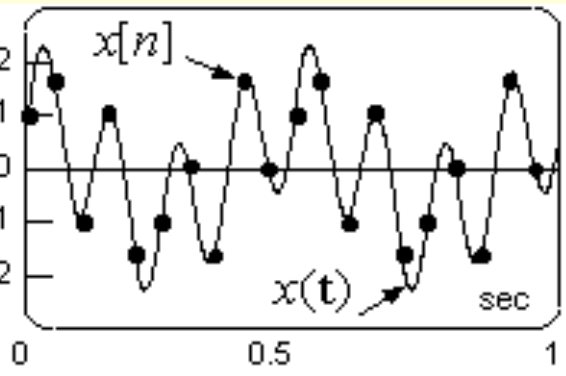
Returning to our first example:

$$x(t) = \cos(2\pi 2t) + 1.4 \sin(2\pi 8t)$$

we can re-write it in standard cosine form as:

$$x(t) = \cos(2\pi 2t + 0) + 1.4 \cos(2\pi 8t - \pi/2)$$

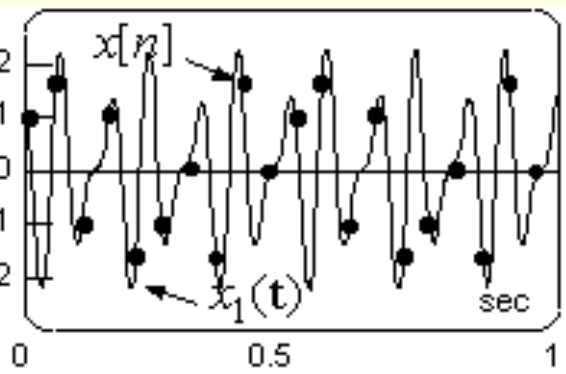
To get the BAND-1 image signal $x_1(t)$, we change the sign of the angles, and we add $f_s = 20$ Hz to the *negative* phasor frequencies:



$$x_1(t) = \cos(2\pi 18t) + 1.4 \cos(2\pi 12t + \pi/2)$$

We could have written the 12 Hz term as $-1.4\sin(2\pi 12t)$. To get the BAND-2 image signal $x_2(t)$, we leave the angles unchanged, and we add $f_S = 20$ Hz to the *positive* phasor frequencies:

$$x_2(t) = \cos(2\pi 22t) + 1.4 \cos(2\pi 28t - \pi/2)$$

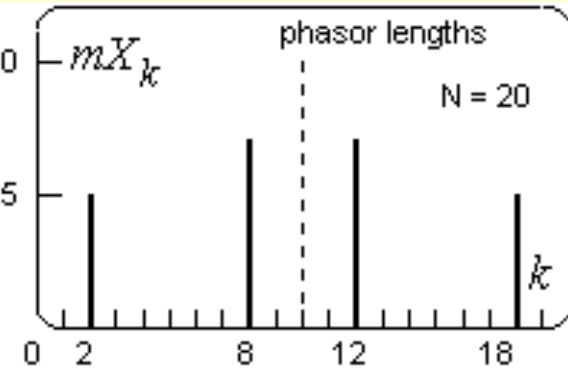


We could have written the 28 Hz term as $+1.4\sin(2\pi 28t)$.

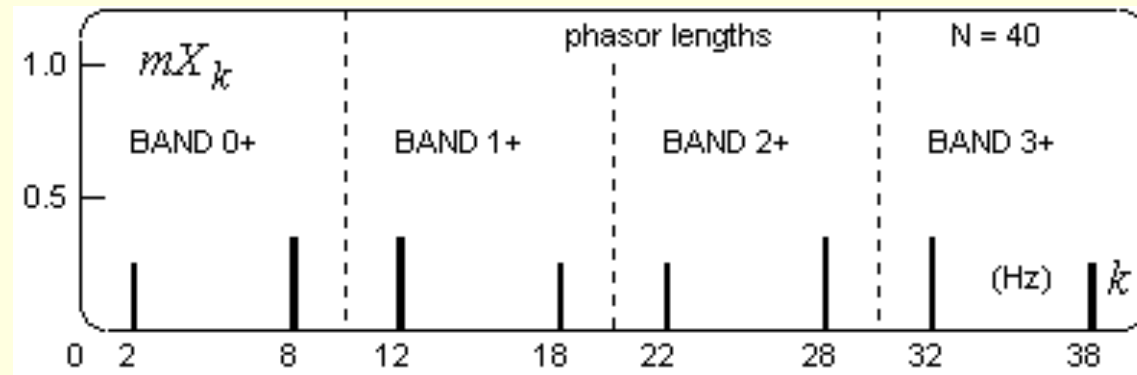
The image signals $x_1(t)$, $x_2(t)$ etc have progressively higher frequencies than $x(t)$, but they share the same sample set. That will be evident in this comparison (Fig 5) between $x(t)$ and $x_1(t)$ and their samples. We conclude that $x[n] = x(nT) = x_1(nT) = x_2(nT)$, etc. Because $x(t)$ and its image signals share the same sample set $x[n]$, they also have identical DFT spectra. By showing their spectra to be identical, we also verify their imaging property.

We can also obtain a DFT spectrum that *shows us* the higher image bands. To do this, we double the DFT length to $N = 40$, and we enter our time samples as:

$$x[n] = \{x(0) \ 0 \ x(T) \ 0 \ x(2T) \ 0 \ x(3T) \ 0 \dots \dots x(19T) \ 0\}$$



We use the same 20 sample values as before, but we double the sample count by placing zeros in between samples. The DFT now sees a sequence with only $T/2$ sec between its samples (including the zeros), and so the spectrum $X[k]$ goes from DC up to $2/T = 2f_s$. This spectrum covers bands 0+, 1+, 2+ and 3+. For the signal of our first example, which previously gave us this plot (Fig 5), we now obtain this new result (Fig 6).



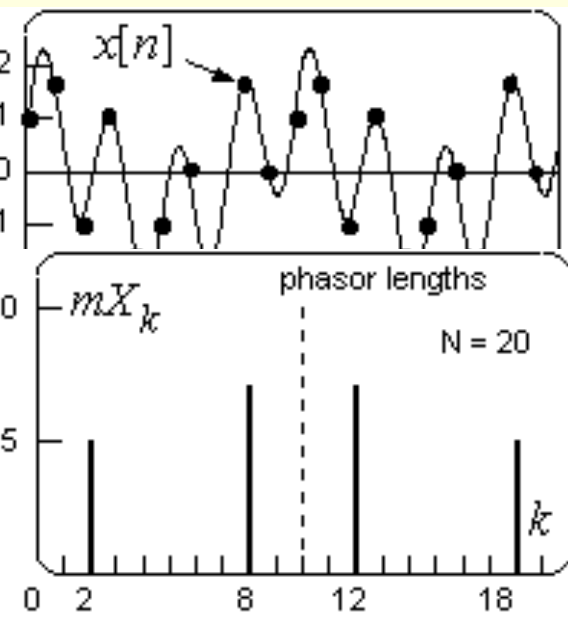
The time axis shows the bin numbers, which are also the frequency in Hertz. This window is just two adjacent copies of the original window (Fig 5), but with one difference. All of the magnitudes are halved. Other than this, it shows the images in all four bands just as they should be. Had we drawn the phase plot, we would again see two identical windows side by side, but without the scaling by $1/2$ that we see on the magnitude plot.

The zero-insertion that we used just now gave an effective *doubling* of the sample rate, or an "up-sampling" in the ratio

$U = 2$. This up-sampling by zero-insertion is called an *expansion* of the data sequence. Except for the magnitude scaling, it doesn't change the spectrum that we see, but it lets us see *more cycles* of the same periodic spectrum. If we were to expand by $U = 4$, we would insert 3 zeros between adjacent samples, thus increasing the DFT length, and the sample rate, by $U = 4$ times. We would then see image bands from BAND 0+ up to BAND 7+, and the spectral magnitudes would fall by $1/U = 1/4$. This change in spectral magnitudes is a general feature of expanded sequences. We will again refer to expanded sequences when we come to talk about *multi-rate* systems (ø Ch 22).

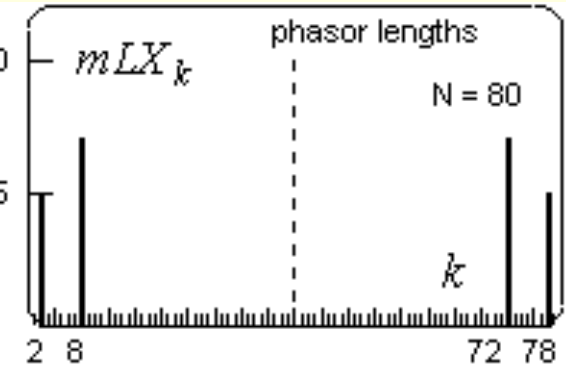
6.3.4 Signal Reconstruction by zero-padding

On our first example, we noted that the sampling was sparse (Fig ç), but we added that the signal frequencies were inside the baseband (below $1/2f_s$). This inferred that the samples were sufficient to *allow full reconstruction* of the signal, which was:



$$x(t) = \cos(2\pi 2t) + 1.4 \sin(2\pi 8t)$$

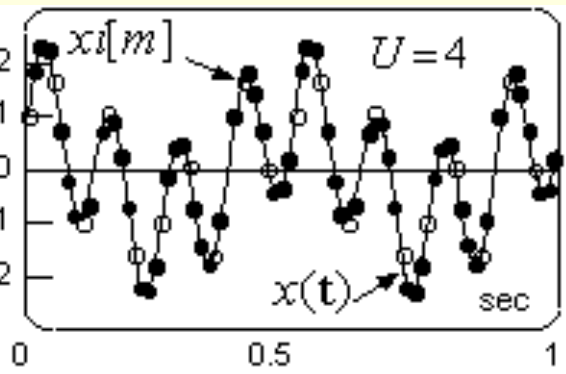
We'll now show one way to reconstruct it. The sample sequence $x[n] = x(nT)$ gave us a spectral vector $X = \text{CFFT}(x)$, and its magnitude mX is redrawn here (Fig ç).



But we *could have* used a longer DFT, perhaps of length $M = 80$, four times longer that we did use. We could have used the extra length to obtain 4 times as many samples in the same 1-second time window. That would quadruple the sampling rate, and also the number of frequency bins, but the bin spacing in Hz would be unchanged. The new longer spectrum, which we've named LX , would then look like this (Fig 5).

The same BAND 0+ lines would occur, and in bins 2 and 8 as before. The BAND 1+ images would be found in bin $-2+M = 78$, and in bin $-8+M = 72$, (Fig 5). There would be no difference in these line values, and all the other lines would be zero. In fact, our spectrum X for $N = 20$ has all the information that we need to build this new LX spectrum. We simply set:

$$LX_2 = X_2 \quad LX_8 = X_8 \quad LX_{-2+M} = X_{-2+N} \quad LX_{-8+M} = X_{-8+N}$$



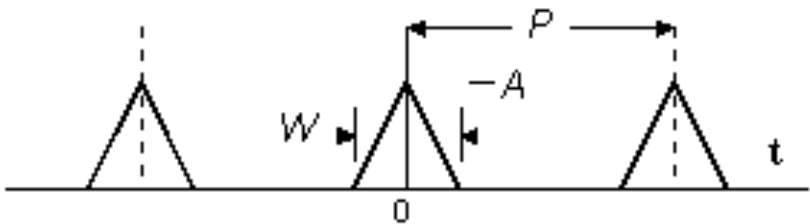
and we set all other LX_m values to zero. We then find the time signal from LX via the operation $x_i = \text{ICFFT}(LX)$. This new x_i (where i denotes *interpolation*) gives us 80 sample points in the same 1-second interval, whereas previously we had only 20, (Fig 6). We can still recognise the original 20 points (with symbol \circ) as $x[n] = x_i[4n]$ for $n = 0 \dots 19$. But we'll have 3 new points between any pair of these points, and they will "fill the gaps" between the sparse sample set $x[n]$. They will also match $x(t)$ exactly, such that $x_i[m] = x(mT/4)$ for $m = 0 \dots 79$.

This is an example of *interpolation* by $U = 4$. It is different from expansion in that the new samples are *authentic* samples from $x(t)$, provided that $x(t)$ was band-limited. We could have used a *higher* U value to get an even more detailed reconstruction. There is no upper limit on U . Therefore, we can reconstruct $x(t)$ to any degree of detail, using only the sparse 20-point sample set $x[n]$. This is what we were promised by the sampling theorem.

We obtained LX from X by a process of *spectral zero-padding*. We used the lines that we had available, and then filled out the middle of the spectrum with zeros. This did not alter the amplitude or the shape of the time signal, but it gave us a much more detailed set of time samples. Later on, we'll perform zero-padding on a *time* signal, in order to obtain a more detailed set of spectral samples. Zero padding (in whatever domain) adds no new information, but it correctly fills the spaces between samples in the other domain.

6.3.5 DFT Spectra of Periodic Signals

We tabulated this triangle waveform (Fig 6.3.5) much earlier (in [Ch 4.2.3](#)), and the diagram also shows how we can find its C_k coefficients exactly. For example, if we choose $A = 1$ and $W/P = 0.2$, we obtain:

<p>Shape Function $X(\mathbf{f})$</p>	<p>Shape Description $C_k = \varepsilon \cdot X(k\varepsilon), \quad \varepsilon = 1/P$</p>
<p>Triangle $\frac{1}{2}AW \cdot \text{sinc}^2(\frac{1}{2}W\mathbf{f})$</p>	

$$C_k = \frac{1}{2} A \frac{W}{P} \operatorname{sinc}^2 \left(\frac{W}{2} \frac{k}{P} \right) = \frac{1}{10} \operatorname{sinc}^2 \left(\frac{k}{10} \right)$$

This method relies on a derivation that gave us a formula for the C_k values. Derivations get more difficult as the waveform shape gets more complex, but the DFT can do the same job without a derivation. If the periodic signal is $x(t) \sim$ with period P , we take a DFT of length N , and we choose a sample interval of $T = P/N$. Then we form:

$$x[n] = x(nT) \sim, \quad n = 0..N-1$$

This time window covers exactly one period, from $t = 0$ to $t = P$. Then we find the spectrum as $X = \text{CFFT}(x)$, and we know that $C_k' = X_k$. We did this for the triangle waveform using $N = 32$ (a power of 2), and we tabulated the C_k from theory and the C_k' from the DFT as far as $k = 10$, with this result:

k	0	1	2	3	4	5	6	7	8	9	10
C_k	0.100	0.097	0.088	0.074	0.057	0.041	0.025	0.014	0.005	0.001	0.000
C_k'	0.102	0.098	0.089	0.075	0.059	0.042	0.028	0.016	0.008	0.003	0.002

We see that the C_k' are *too high* in all cases. That's because of the additive effect of the aliases from higher frequencies, which are always real and positive, because sinc^2 is real and positive. To make C_k' come far closer to the true C_k , we must reduce the overlap due to aliasing. The way to do this is to use far higher N , such as $N = 256$, and then set $T = P/N$ as before. We then get many more time samples over the same one-period time window. The sampling frequency $f_s = 1/T$ is greatly increased. This f_s is the spacing between spectral replicas and, by increasing f_s , we've reduced the amount of overlap. If we

repeat our tabulation using $N = 2^8 = 256$, we will find $C_{k'}$ very close to the true C_k values. We will have harmonics up as far as $\frac{1}{2}N = 128$, not because the higher harmonics are of any interest (usually), but because large N makes the lower $C_{k'}$ more nearly equal to C_k .



The big advantage of the DFT method is that it works for any shape of periodic waveform. By progressively increasing the DFT length N , and monitoring the resulting $C_{k'}$, we can make a reasoned judgement as to when aliasing becomes insignificant. This confirms our ability to view periodic-signal spectra with the help of the DFT, so long as we take care to minimise spectral aliasing.





6.4 THE NORMALISED DtFT AND THE FFT

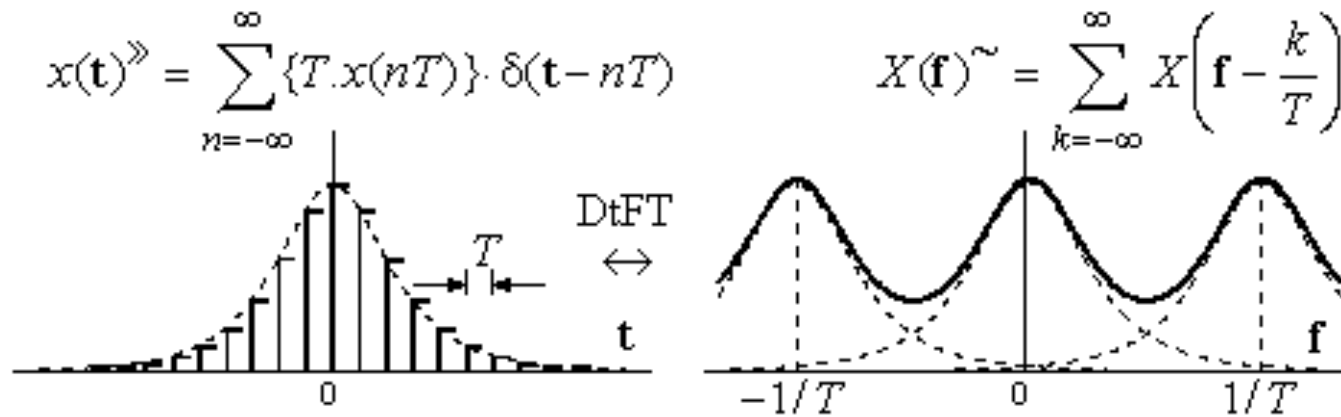
The DtFT (Discrete-time Fourier Transform) links a discrete-time signal to its continuous periodic spectrum. The time signal is a number sequence, and it may be the sample set from some analogue pulse waveform. As such, the sequence has many possible analogue interpretations, a different one for every band of the spectrum. This is why the spectra of data sequences are inherently periodic.

We will first introduce the DtFT in terms of normalised frequency f . Afterwards, we'll see how the the DFT gives us a fast and convenient way to evaluate it.

6.4.1 The DtFT in Normalised Form

We introduced the DtFT much earlier (i [Ch 3.3.2](#)) in the form that we see here (Fig \hat{e}).

$$\text{DFT} \\ \sum_{n=-\infty}^{\infty} \{T \cdot x(nT)\} \cdot e^{-j2\pi f n T} \\ \int_{-1/T}^{1/T} X(\mathbf{f}) \sim \cdot e^{j2\pi f n T} \cdot d\mathbf{f}$$



The number sequence is depicted here as the set of area-samples $x(\mathbf{t})\gg$ from a pulse $x(\mathbf{t})$ whose spectrum is $X(\mathbf{f})$. The spectrum of the $x(\mathbf{t})\gg$ sequence is just the replicated $X(\mathbf{f})\sim$. The forward DtFT is a sum over sample values. The inverse DtFT is an integral over one period of $X(\mathbf{f})\sim$. Both (Fig 6) are shown alongside.

When dealing with sampled data, the normalised frequency f is more convenient, and we generally use the DtFT in its normalised form. We'll also find out very soon that digital filters are fully specified by a number sequence, which we usually call $h[n]$. The (normalised) DtFT of this $h[n]$ becomes the filter spectrum $H(f)$. We will switch over to the filter notation, as follows:

$$2\pi f n T \rightarrow 2\pi f n \frac{1}{f_s} = 2\pi f n, \quad X(\mathbf{f}) \sim \rightarrow H(f), \quad T \cdot x(nT) \rightarrow h[n]$$

Then the Forward DtFT summation becomes:

$$H(f) = \sum_{n=-\infty}^{\infty} h[n] \cdot e^{-j2\pi f n} \quad \text{normalised DtFT}$$

and the inverse integral becomes:

$$h[n] = \int_{-0.5}^{0.5} H(f) \cdot e^{j2\pi f n} \cdot df \quad \text{normalised IdtFT}$$

This integration is over the base-band ($-0.5 < f < 0.5$), or over an equivalent one-period width of 1.0. The period width is different by a factor T from the width before normalisation, which was $1/T$. That is why we see $h[n] = T \cdot x(nT)$ on the left side, instead of the $x(nT)$ of the original form. (Also, because $H(f) = X(f) \sim$, the heights under both integral forms are identical).

We'll make extensive use of these normalised DtFT forms. For any numeric sequence $h[n]$ of length L , the DtFT sum gives us its periodic spectrum $H(f)$. Computation of $H(f)$ can be time-consuming, because it requires L complex multiplications for every value of f in $H(f)$. This is where the FFT can produce a time saving.

6.4.2 DtFT Evaluation by FFT

For this application, we prefer to use the DFT in the $x[n]/X[k]$ notation (i [Ch 5.3.2](#)). We defined $x[n] = x(nT) \sim$ and $X[k] = NC^{-k}$, and the DFT/IDFT became:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi \frac{k}{N}n}, \quad k = 0..N-1$$

Mathcad N -CFFT, MATLAB fft

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \cdot e^{j2\pi \frac{k}{N}n}, \quad n = 0..N-1$$

Mathcad ICFFT/ N , MATLAB ifft

The only difference from the earlier C_k form is that the spectral heights $X[k]$ are larger by N , the DFT length. We'll re-write the Forward DFT using filter symbols $h[n]$ and $H[k]$ instead of $x[n]$ and $X[k]$:

$$H[k] = \sum_{n=0}^{N-1} h[n] \cdot e^{-j2\pi \frac{k}{N}n}, \quad k = 0..N-1$$

Mathcad N -CFFT, MATLAB fft

Now we'll write the Forward DtFT for comparison:

$$H(f) = \sum_{n=-\infty}^{\infty} h[n] \cdot e^{-j2\pi f n} \quad \text{normalised DtFT}$$

Direct comparison of $H[k]$ with $H(f)$ reveals that, provided $h[n]$ is zero outside the range $0..N-1$, then $H[k]$ is a set of N values from $H(f)$, evaluated at the DFT bin frequencies k/N . We have to choose a DFT of sufficient length to contain all the elements of $h[n]$. After that, the FFT will find the filter spectrum $H(f)$ very rapidly, at the DFT bin frequencies.

By way of an example, we'll use the following $h[n]$ sequence:

$$h[n] = \{ \underline{0.2} \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.2 \}$$

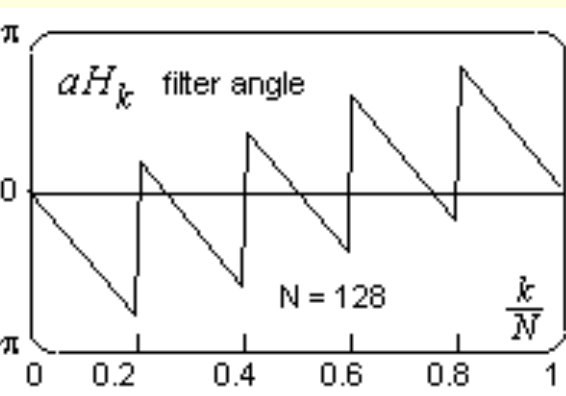
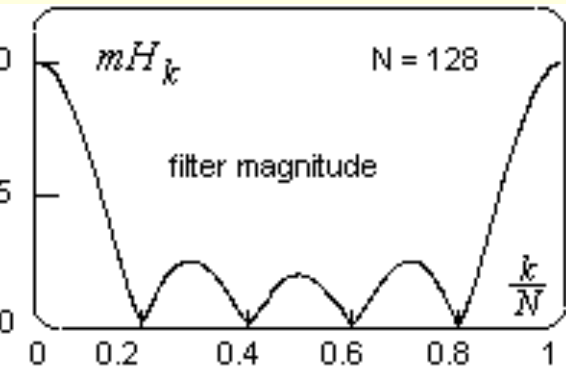
This is a filter of length $L = 5$, and all of its "tap values" are 0.2. We use a DFT of length $N = 128$ (a power of 2), far longer than the filter length requires. We specify $h[n]$ for the DFT as:

$$h[n] = 0.2, \quad n = 0..4 \quad \quad h[n] = 0, \quad n = 5..N-1$$

This $h[n]$ vector is of length N , as required. The filter values come first, then it is *padded out with zeros* to the full DFT length N . We compute the spectrum as

$$H = N \cdot \text{CFFT}(h)$$

Notice the multiplication by N in this version of the DFT. We now have a spectrum with elements H_k in spectral bins $k = 0 \dots 127$. Alongside (Fig 5), we show the spectral magnitude plot mH_k and the angle plot aH_k . On the frequency axis, we've used normalised frequency k/N instead of the bin numbers k . This gives a frequency range of $(0 < f < 1.0)$, the { BAND 0+, BAND 1+ } range.



These are 128-point plots (since $N = 128$), but they are drawn as continuous line plots. We chose large N simply to get large spectral point density. The *time-domain zero padding* achieved this. It added nothing new in information terms, but it filled in the detail of the spectrum. If we now doubled up N to $N = 256$, it would again halve the spectral point spacing, but the plots would look nearly identical to what we see here.

These plots offer a full description of the filter $h[n]$, as we will come to see in the next chapter. As such, they are very important, we will use them extensively, and the FFT offers a fast and convenient framework for obtaining them.





6.5 DFT POWER AND ENERGY

When the DFT describes a periodic time signal, a DFT *power spectrum* is sometimes more appropriate than the normal signal spectrum. When the DFT describes a pulse in time, then the DFT *energy spectrum* becomes appropriate.

6.5.1 DFT Power Spectra

We demonstrated earlier (in [Ch 4.2.5](#)) that the power associated with a phasor C_k is $|C_k|^2$. From this, we saw that the total signal power became:

$$\bar{P}_x = \frac{1}{P} \int_P |x(t)^\sim|^2 dt = \sum_{k=-\infty}^{\infty} |C_k|^2$$

The DFT gives us the aliased coefficients as C_k' (or as $X[k]/N$) and allows us to calculate mean signal power as:

$$\bar{P}_x = \sum_{k=0}^{N-1} |C'_k|^2 \quad \text{total (mean) signal power}$$

This is a summation over one DFT window, equivalent to a summation over the baseband. It gives total mean power because any signal power outside the baseband is aliased into the baseband when the signal is sampled. Because of this result, we can also think in terms of *bin-power*, defined as:

$$\text{PBIN}_k = |C'_k|^2 = |X[k]/N|^2, \quad k = 0..N-1 \quad \text{mean power in bin-}k$$

If we plot this, rather than plotting mX_k and aX_k , we obtain a DFT *power spectrum*, which we could also call a *bin-power* plot. The total power is spread out over the base-band, often in a non-uniform way. We sometimes like to speak of power spectral density (PSD), which is *power per unit of bandwidth*. The PSD level at bin number k is the bin power divided by the bin-width, where bin-width is expressed in cycles per sample:

$$\text{PSD}_k = \frac{\text{bin}_k \text{ power}}{\text{bin width } (\Delta f)} = \frac{|X[k]/N|^2}{(1/N)} = \frac{|X[k]|^2}{N} \quad \text{power spectral density}$$

Notice, the normalised bin width is just $1/N$. PSD and bin-power plots give essentially the same information, but we may favour one over the other, depending on the application.

The power spectrum gives us only the spectral amplitudes, it contains *no phase information*, and hence *no timing information*. For signals whose timing is highly variable or unknown, we can often obtain a power spectrum, even though a full $X[k]$ spectrum might not be obtainable. This helps explain the importance of the power spectrum in many practical situations. It just tells us at what frequencies the signal power will tend to concentrate. This may be vital information. For example, if we want to filter a transmission signal, so that it fits in a limited bandwidth, the power spectrum of the signal tells us what frequencies we should retain, so that very little of the power will be lost.

6.5.2 DFT Energy Spectra

If the DFT time sequence $x[n]$ describes a "one-off" pulse shape, then the signal is confined to one DFT period, which is NT in seconds, or simply N using normalised time units. Because mean power is Energy per time period, we can identify the pulse Energy as (mean power) $\times NT$, or just as (mean power) $\times N$ using the normalised units. To convert from power quantities to normalised Energy quantities, we just multiply by N . In this way, we get:

$$\text{ESD}_k = |X[k]|^2 \quad \text{energy spectral density}$$

and

$$\text{EBIN}_k = |X[k]|^2 / N \quad \text{energy in bin } k$$

We can view EBIN_k as just $\text{ESD}_k \times \Delta f$, where $\Delta f = 1/N$.

To convert ESD and EBIN figures into time-related Energy quantities, we just multiply by T .

This concludes our first application-oriented session with the DFT. We've seen it do some jobs very well, and with the speed advantage of the FFT to make it more attractive. But we've also hinted at some difficulties. We will tackle those difficulties when we next return to the DFT.





7.1 PREAMBLE

Our theoretical framework is not yet complete, and we will deal with it further in Chapter 8. Meanwhile, we want to make a start on some practical DSP work, by way of a simple introduction to digital filters. We'll begin with the sampling of some frequently-occurring signals. Then we introduce filters of the FIR and IIR types, using the DtFT to find the filter spectra, and we explain the importance of these spectra. We show how filters can be connected, in series and in parallel, and we introduce correlation as a matched-filtering procedure.





7.2 SAMPLING AND SEQUENCES

The signals that we introduce here have importance in digital filtering. We will look at the sampled versions of some signal shapes $x(\mathbf{t})$, and will introduce new parameters to describe these sampled versions of $x(\mathbf{t})$.

7.2.1 Sampled Step and Impulse Waveforms

We will start with some elementary test signals. Prominent among these is the unit step function, defined as:

$$\begin{aligned} u(\mathbf{t}) &= 1, & \mathbf{t} &\geq 0 \\ u(\mathbf{t}) &= 0, & \mathbf{t} &< 0 \end{aligned}$$

If we sample this step at sample points $\mathbf{t} = nT$, the resulting sequence is $u[n]$:

$$u[n] = \{ \dots 0 \underline{0} 1 1 1 \dots \}$$

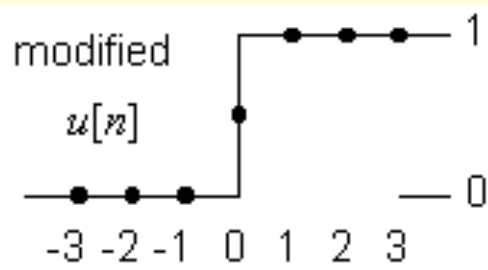
The underline marks the zero-th element, $u[0]$. The double-dots denote continuation of the trend. The brackets $\{ \}$ suggest that we treat all elements together as a single entity, as a data-sequence. The sequence is presumed to have infinite length in both directions ($-\infty < n < \infty$). A different but equivalent definition is shown alongside (Fig ẽ).

The Unit Step

$$u[n] = 1, \quad n \geq 0$$

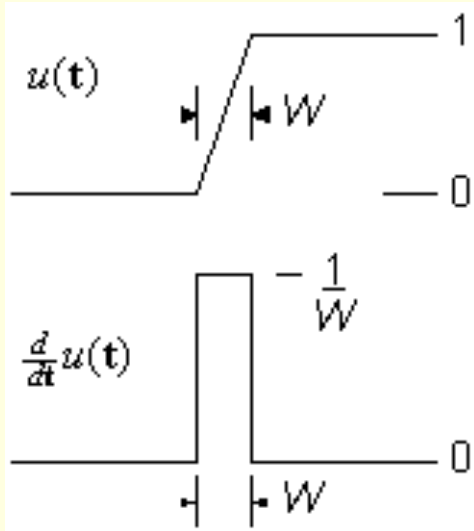
$$u[n] = 0 \quad \text{otherwise}$$

A slightly modified definition (Fig ç) would show $u[0] = 1/2$. This is certainly correct for Fourier synthesis, when we try to build a step from sine waves. But the first definition is favoured for its simplicity, and is more widely used.



The derivative of the step $u(t)$ is the impulse $\delta(t)$, already defined (i [Ch 3.3.2](#)). This is easy to see if we think of the step as the limiting case of this ramp waveform (Fig í) when the ramp width W is reduced to zero. The derivative is a rectangle of area $(W)(1/W) = 1.0$. As $W \rightarrow 0$, it becomes a unit-area impulse, $\delta(t)$.

We have no close digital equivalent for the impulse. The smallest meaningful W value is the sample interval T . If we sample the rectangular-shaped derivative using $W = T$, we get a single sample of value $1/T$, and all other sample values are zero. Thus, we can reasonably define:



$$\delta_T[n] = \{ \dots 0 \ 0 \ \underline{\frac{1}{T}} \ 0 \ 0 \dots \} \quad \text{digital-}T \text{ impulse}$$

as our nearest digital equivalent to an analogue impulse $\delta(t)$. We will use this definition in the future, but it is also common practice to define a time-independent digital impulse as :

The Unit Impulse
 $\delta[n] = 1, \ n = 0$
 $\delta[n] = 0 \text{ otherwise}$

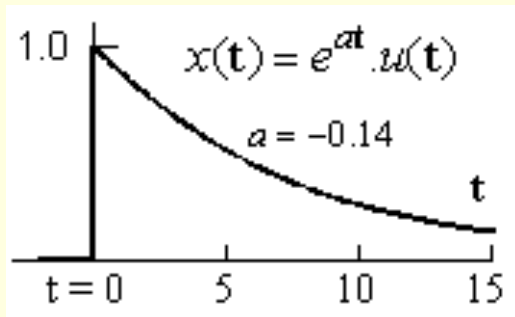
$$\delta[n] = \{ \dots 0 \ 0 \ \underline{1} \ 0 \ 0 \dots \} \quad \text{digital impulse}$$

or as the "Unit Impulse" shown here (Fig c). This "time-free" definition is more widely-used, although it is more remote from the analogue impulse that gave it its name. We will use both both $\delta[n]$ and $\delta_T[n]$ as the situation demands.

7.2.2 Sampled Exponentials

The right-sided analogue exponential is defined by:

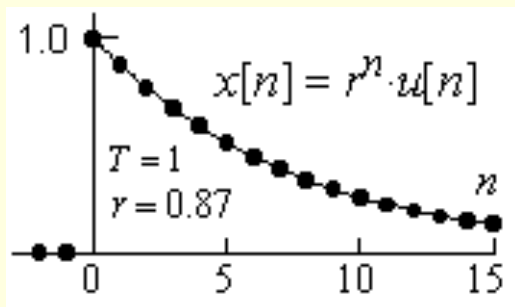
$$x(t) = e^{at} \cdot u(t)$$



where $a < 0$ for a *decaying* exponential as shown (Fig ç). Taking value-samples of $x(t)$ with sample interval T , we get:

$$x[n] = x(nT) = e^{anT} \cdot u(nT) = \left(e^{aT} \right)^n \cdot u[n]$$

Even more simply:

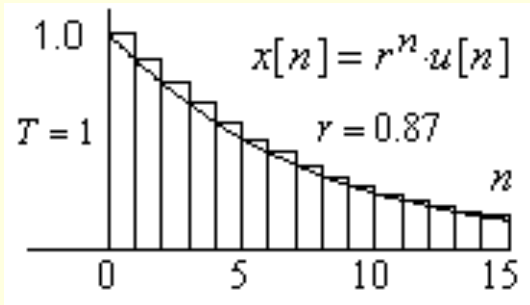


$$x[n] = r^n \cdot u[n], \quad r = e^{aT}$$

This sequence is shown here (Fig ç) for $a = -0.14$ with sample interval $T = 1$. The rate of decay from sample to sample depends both on a and on T , such that the two combine to form a single decay constant, $r = 0.87$. This r becomes the *ratio* of any sample value to the sample value just before it, that is, $r = x[n]/x[n-1]$.

The area under the $x(t)$ exponential (for $a < 0$) has a finite value, found as:

$$\int_{-\infty}^{\infty} x(t) dt = \int_0^{\infty} e^{at} dt = \frac{1}{a} \left[e^{at} \right]_0^{\infty} = \frac{-1}{a}$$



The value-samples $x[n]$ above give no sense of this area, but the *area-samples* do (Fig c). Clearly, the sum of area-samples $\{T \cdot x[n]\}$ is an approximation to the area under $x(t)$, that is:

$$\int_0^{NT} x(t) dt \approx \sum_{n=0}^{N-1} \{T \cdot x(nT)\} \approx \sum_{n=0}^{N-1} \{\text{Area - samples}\}$$

This is true for other waveforms also. To see how it works for our exponential $x(t)$, we must first sum the $x[n]$ samples above. If we use S_N to mean the sum over the first N samples of $x[n]$, we have:

$$S_N = \sum_{n=0}^{N-1} r^n = 1 + r + r^2 + r^3 + \dots + r^{N-1}$$

Now we multiply both sides by r to obtain:

$$r \cdot S_N = r + r^2 + r^3 + \dots + r^{N-1} + r^N$$

Subtracting $r \cdot S_N$ from S_N , most terms cancel and we find:

$$S_N - r \cdot S_N = 1 - r^N$$

Finally,

$$S_N = \sum_{n=0}^{N-1} r^n = \frac{1 - r^N}{1 - r}$$

This is a very useful closed-form expression for S_N . It continues to hold as $N \rightarrow \infty$, provided the sequence decays (requiring that $r < 1$). Thus:

$$S_\infty = \sum_{n=0}^{\infty} r^n = \frac{1}{1 - r}, \quad (r < 1)$$

Geometric Sums

$$\sum_{n=0}^{N-1} x^n = \frac{1-x^N}{1-x}$$

and ..

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$$

provided $|x| < 1$

These *geometric sums* are equally valid when r is a *complex* number. They play a big part in DSP work, and are summarised alongside (Fig 7.2.3) in terms of some complex parameter x . Returning to our exponential, the sum of value-samples is S_{∞} and the sum of area-samples is $T \cdot S_{\infty}$. Thus:

$$T \cdot S_{\infty} = \frac{T}{1-r} = \frac{T}{1-e^{aT}} = \frac{T}{1-\{1+aT+\dots\}} \approx \frac{-1}{a}, \quad (aT \ll 1)$$

We've used the approximation $e^x \approx 1 + x + \frac{1}{2}x^2 + \dots$ with $x = aT$, valid for small aT , to show how the sum of area-samples converges to the area under $x(t)$ when T is small (frequent sampling). A familiar sum occurs when $r = 0.5$. Then:

$$S_{\infty} = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots = \frac{1}{1-\frac{1}{2}} = 2.0$$

We could have written this sum on inspection. And, since we can find the sum for *any* value of $r < 1$, we can *always* integrate over a decaying exponential to get a finite result. It even holds for *oscillatory* decay, which is our next topic.

7.2.3 Decaying Oscillatory Sequences

The *decaying sinusoid* is a signal of the form :

$$y(t) = e^{at} \cdot \sin(\omega_0 t) \cdot u(t)$$

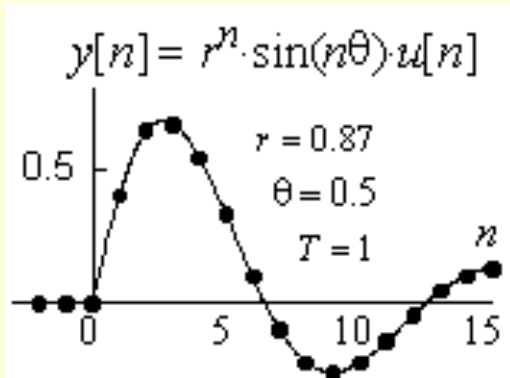
where $\omega_0 = 2\pi f_0$ is just the radian frequency (rads/sec). After sampling:

$$y[n] = y(nT) = e^{anT} \cdot \sin(\omega_0 nT) \cdot u(nT)$$

We now have *two* constant parameters:

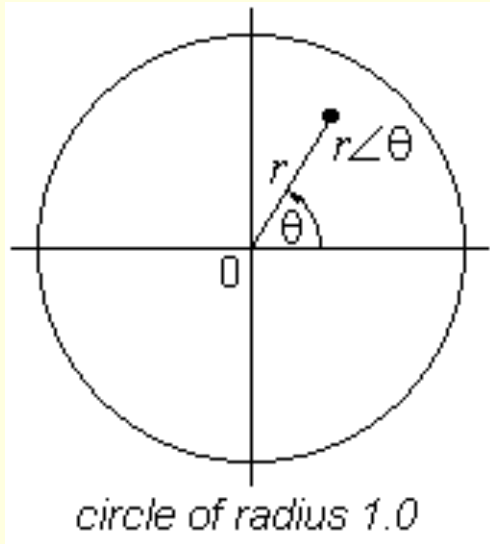
$$r = e^{aT}, \quad \theta = \omega_0 T$$

such that:



$$y[n] = r^n \cdot \sin(n\theta) \cdot u[n]$$

The sequence is shown here (Fig 5) for $r = 0.87$ and $\theta = 0.5$. Recall that a sine wave is a sum of two rotating phasors, and the angle θ is the amount by which the phasors rotate from one sample to the next, in units of radians/sample, a description which is *independent* of the sampling frequency. The decaying sinusoid can always be represented by a single complex number $r\angle\theta$ as shown here (Fig 1). The radius r gives the *decay* between samples, while θ is the *rotation* between samples. As long as $r\angle\theta$ lies inside the circle of radius 1.0, the signal will decay, and its area under integration will be finite, because it lies within the envelope of a decaying exponential. Signals of this type, and diagrams like this, will recur again and



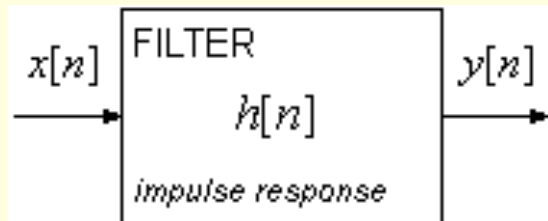
again in our later work.





7.3 DIGITAL FILTERS AND CONVOLUTION

Digital filters are used to modify a signal by removing (or re-shaping) some of its frequency components. The input signal is represented by its samples $x[n]$, and the "filter" is just a set of calculations that are repeated over and over on the input samples $x[n]$, to generate a set of output samples, $y[n]$. The filter action repeats under control of a "clock" signal, where $n = 0, 1, 2 \dots$ etc counts the clock cycles such that, on each clock cycle, the filter takes in a new input sample $x[n]$, and outputs a new output sample $y[n]$. Most often, the action of the filter can be specified by another set of samples called $h[n]$, which is known as the "Impulse Response" of the filter (Fig 7.3.1).



7.3.1 LTI Filters

LTI is short for "Linear and Time-Invariant". Most of our filters will be LTI filters, and their impulse response $h[n]$ defines them completely. It is so named because, if the input signal to the filter is the impulse sequence $\delta[n]$, (a solitary "1" at $n = 0$ and nothing thereafter), the resulting output sequence $y[n]$ is the sequence of numbers that we call $h[n]$, the "response to an impulse". The filter's response to some different input sequence is determined by $h[n]$ and by the LTI nature of the filter, and we will illustrate this with an example.

We'll choose a filter whose impulse response is $h[n] = r^{-n}u[n]$ with $r = 0.5$. Thus:

$$h[n] = \{ \underline{1} \quad 0.5 \quad 0.25 \quad 0.125 \quad \dots \}$$

This sequence continues indefinitely. For the input $x[n]$, our choice is simple but arbitrary:

$$x[n] = \{ \dots 0 \quad \underline{1} \quad 2 \quad 3 \quad 4 \quad 0 \quad \dots \}$$

The filter responds to each element of $x[n]$ in succession. At $n = 0$, it responds to the "1" by outputting $\{1 \quad \frac{1}{2} \quad \frac{1}{4} \text{ etc.}\}$. The response to the "2" is doubled in size, and delayed by one cycle: $\{0 \quad 2 \quad 1 \quad \frac{1}{2} \text{ etc.}\}$. The response to the "3" is tripled in size and delayed by two cycles. Similarly for the "4", and then the input terminates.

This table (Fig 6) shows each response as a separate row. At the bottom, it adds up the responses by summing columns vertically. The output $y[n]$ on the last row is the sum of all the responses.

n	$x[n]$	$y[0]$	$y[1]$	$y[2]$	$y[3]$	$y[4]$	\dots
-1	0	0	0	0	0	0	\dots
0	1	1.000	0.500	0.250	0.125	0.0625	\dots
1	2		2.000	1.000	0.500	0.250	\dots
2	3			3.000	1.500	0.750	
3	4				4.000	2.000	\dots
4	0					0.000	\dots
$\Sigma = y[n]$		1.000	2.500	4.250	6.125	8.0625	\dots

← response to $x[0]$
 ← response to $x[3]$

This filter showed *linearity* by giving a response *directly proportional* to the size of the input sample. It showed *time invariance* by responding to successive samples (each arriving at a different time) with a scaled version of $h[n]$, all in the same unvarying manner. These are the features that make it an LTI filter.

To see how $y[n]$ is assembled, we look at the $y[3]$ column (inside the grey box) and, on close examination, we find that:

$$y[3] = h[3] \cdot x[0] + h[2] \cdot x[1] + h[1] \cdot x[2] + h[0] \cdot x[3]$$

We can generalise this result for any output sample $y[n]$:

$$y[n] = \sum_{m=0}^n h[n-m] \cdot x[m], \quad n = 0, 1, 2, 3, \dots$$

Notice how n is a *constant* while we do a summation over m to obtain $y[n]$. Then we increment n by one and do the next summation, resulting in $y[n+1]$. This equation sums up the filter action, and describes a process that we call *convolution*. Our tabular approach has shown this to be the natural result of linearity and time invariance. But, to better understand convolution, and to do convolution on paper, we recommend a different method, as follows.

7.3.2 Convolution of Sequences

This diagram shows the graphical method that we prefer (Fig 6):

$h[n-m] :$	0.125 0.250 0.500 <u>1.000</u>	→	<i>pictured at $n = 0$</i>				
$x[m] :$							
$y[n] :$							

The first row shows $h[n]$ in *time-reversed* form. The second row shows $x[n]$ in its normal form. We've underlined the $n = 0$ index positions and lined them up vertically, so that the first elements of each are overlapping. These are 1.000 and 1.0 on the diagram. The product of this vertical pair is $y[0] = 1.000$, the first output sample from the filter. Now we slide the $h[n]$ sequence to the right, one step at a time. After one step, we have two pairs overlapping, and the output $y[1]$ becomes *the sum*

of vertical product pairs $y[1] = 1.0 \times 2.0 + 0.5 \times 1.0 = 2.500$. After another step, we obtain $y[2]$ as $y[2] = 1.0 \times 3.0 + 0.5 \times 2.0 + 0.25 \times 1.0 = 4.250$. It's easy to check that this is equivalent to the tabular method, but it gives us a better insight into what convolution is about.

We prefer to think of sequences as covering all possible index values ($-\infty < n < \infty$). If we assume that $x[n]$ and $h[n]$ are both zero for all $n < 0$, we'll see from the diagram that we can extend the limits to infinity without affecting the result. Therefore:

$$y[n] = \sum_{m=-\infty}^{\infty} h[n-m] \cdot x[m], \quad (-\infty < n < \infty) \quad \text{digital convolution}$$

We use the special symbol $*$ to indicate convolution:

$$y[n] = h[n] * x[n] = \sum_{m=-\infty}^{\infty} h[n-m] \cdot x[m]$$

It also turns out that:

$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[n-m] \cdot h[m]$$

This means that we can time-reverse $x[n]$ rather than $h[n]$, then continue the process as normal, and the result $y[n]$ will be the same as before. Therefore:

$$y[n] = h[n] * x[n] = x[n] * h[n] \quad \text{digital convolution (or filtering)}$$

This is the *commutative* property of convolution.

A sequence is said to be *causal* if it is zero-valued for all $n < 0$. We convolved two causal sequences $x[n]$ and $h[n]$, and we found that the output $y[n]$ was causal too. This is true of filters in general. The output $y[n]$ is causal because we cannot get an output before we apply an input. It is simply the principle of cause and effect (hence the term *causal*). Filtering is most often performed in real time, meaning that the filter must wait for the next input sample before it can deliver the next output sample. It's not always like that. If the input data has been recorded in advance, we may have access to any input sample on demand. If we avail of that, we then have a *non-causal* filtering action.

7.3.3 Filter Frequency Response

If a digital filter is driven by a sampled sinusoidal sequence of frequency f (in cycles/sample), the output $y[n]$ develops into a sinusoidal sequence of the same frequency, but with different amplitude and phase, as we will now demonstrate. Because a sine wave is built from two phasors, we can use a *sampled phasor* as our input $x[n]$, and extend our result later to sine waves.

So, we have:

$$x[n] = e^{j2\pi f n} \quad \text{and} \quad y[n] = \sum_m x[n-m] \cdot h[m]$$

Then:

$$y[n] = \sum_m e^{j2\pi f(n-m)} \cdot h[m] = e^{j2\pi f n} \cdot \sum_m e^{-j2\pi f m} \cdot h[m]$$

Finally— for this specific input sequence:

$$y[n] = x[n] \cdot \left\{ \sum_m h[m] \cdot e^{-j2\pi f m} \right\} = x[n] \times (\text{DtFT of } h[n])$$

We recognise the bracketed quantity as the DtFT of $h[n]$, which is also the filter spectrum, a complex function of frequency called $H(f)$. Suppose that, at some specified f value, $H(f) = G\angle\theta$. (magnitude G at angle θ radians). For a phasor input at $+f$ we obtain:

$$x[n] = e^{j2\pi f n} \quad \text{and} \quad y[n] = e^{j2\pi f n} \cdot G \cdot e^{j\theta} = G \cdot e^{j(2\pi f n + \theta)}$$

For a phasor input at $-f$ (provided $h[n]$ are real-valued) we obtain:

$$x[n] = e^{-j2\pi f n} \quad \text{and} \quad y[n] = e^{-j2\pi f n} \cdot G \cdot e^{-j\theta} = G \cdot e^{-j(2\pi f n + \theta)}$$

For a cosine input $\cos(2\pi f n)$, we take one-half of the sum of these inputs, and similarly for the outputs, which gives us:

$$x[n] = \cos(2\pi f n) \quad \text{and} \quad y[n] = G \cdot \cos(2\pi f n + \theta)$$

Thus, if the filter spectrum is $H(f) = G \angle \theta$, the response $y[n]$ to a cosine input will be scaled in magnitude by G , and its angle will be altered by θ . Clearly, G and θ are functions of frequency, they are $|H(f)|$ and $\arg\{H(f)\}$. The plot of $|H(f)|$ versus f is the *filter magnitude spectrum (mH)*, and the plot of $\arg\{H(f)\}$ versus f is the *filter angle spectrum (aH)*. We'll be using these results very shortly.





7.4 FIR FILTERS AND FILTER STRUCTURES

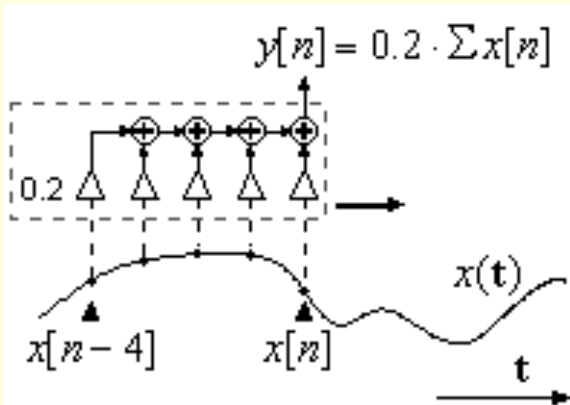
All of our filters do a convolution as we described, but they come in two major categories: FIR filters have a Finite-length Impulse Response, and we will deal with these now. IIR filters have an Infinite-length Impulse Response, and we will deal with those in the next chapter.

7.4.1 The FIR Averaging Filter

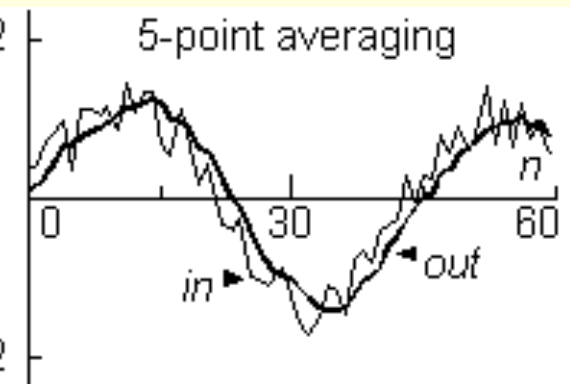
We'll describe a 5-point averaging filter that operates in real time. At any given time, it remembers the most recent incoming sample $x[n]$, and the four previous samples, $x[n-1]$ back to $x[n-4]$. The filtering action is the repeated calculation of the output $y[n]$, for $n = 0, 1, 2, 3, \dots$ etc, according to:

$$y[n] = \frac{1}{5} (x[n] + x[n-1] + x[n-2] + x[n-3] + x[n-4]) \quad n = 0, 1, 2, 3, \dots$$

This is a familiar averaging procedure. It tends to suppress rapid changes in the output. As such, it reduces the high-frequency



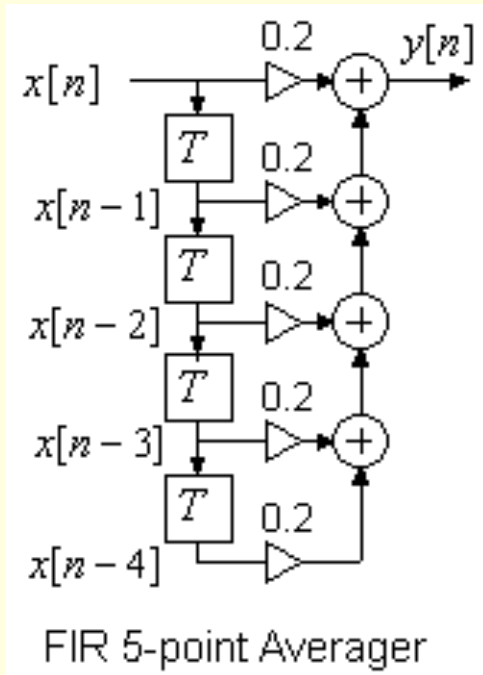
parts, but has little effect at low frequency. In this diagram (Fig c), the filtering is drawn to resemble the graphical convolution method. The filter is a set of "taps" or multiplier values (depicted as small triangles), all of them 0.2 in this case, and they slides from left to right, while monitoring the most recent five samples of the signal. Each sample is multiplied by the corresponding tap weight (0.2), and all five products are then added to form $y[n]$. After another step to the right, the process repeats, and it determines $y[n+1]$.



We took a noisy signal as our input, and performed 5–point averaging on it over 60 samples, with this result (Fig c). The output has lost most of the rapid input variations, as expected. But there is something else as well, namely, the appearance of a signal delay. In fact, the delay effect is quite real, and is exactly two sample intervals. The reasoning is that our computed average $y[n]$ is most representative of the *middle* of the filter window, which is at $x[n-2]$, whereas $y[n]$ appears *two* samples later. This reasoning applies to many FIR filters of length L (meaning L filter taps), and the delay is calculated as:

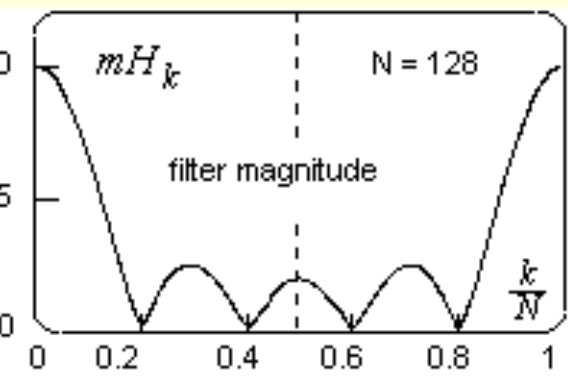
$$\text{filter delay in samples} = (L - 1)/2$$

The more usual kind of block diagram for this FIR filter looks like this (Fig c). It shows a chain of four delay blocks with the label "T" to indicate a T-second delay. Each block is just a data register which holds a data value for one filter cycle, and then passes it down the chain to the next register. The input data $x[n]$ enters at the top of the chain, and the older data $x[n-1]$, $x[n-2]$, etc resides at tapping points along the chain, as shown on the block diagram. These data values are multiplied by the filter coefficients, the triangles labelled 0.2, and added together to form the output $y[n]$. This output is a *weighted sum of past and present input values*. From the block diagram, one can immediately write the *difference equation* that describes the filter



as:

$$y[n] = 0.2x[n] + 0.2x[n-1] + 0.2x[n-2] + 0.2x[n-3] + 0.2x[n-4]$$

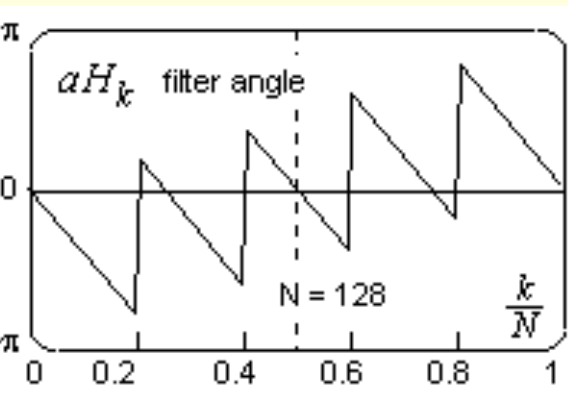


This is quite a simple description, and we can understand intuitively that higher frequencies are somewhat suppressed, but we need a more satisfactory statement of behaviour over frequency. Actually, we obtained it in the last chapter, and this (Fig 6.4.2) is the $|H(f)|$ that we found, the filter's gain magnitude plot (see [Ch 6.4.2](#)). It describes a gain that is close to 1.0 at low frequencies (below $f = 0.1$). Higher frequencies (up to $f = 0.5$, the top of the baseband) are more heavily attenuated.

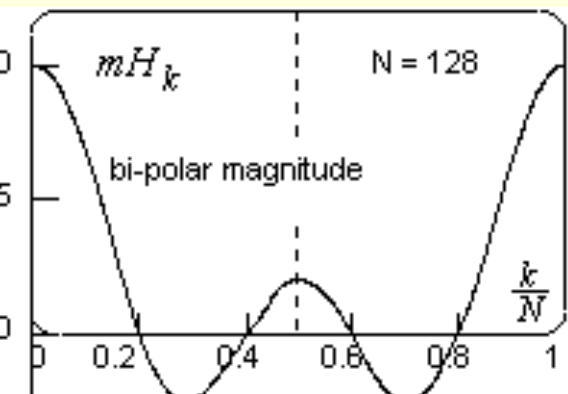
The gain is zero at $f = 0.2$. That's because $f = 0.2$ indicates $1/0.2 = 5$ samples per sine-wave period at this frequency. The 5-point filter "sees" exactly one sine-wave period, and the average over one period is zero. The filter averages the 5 samples, their average value is zero, and so the $y[n]$ values are zero every time. The same happens at $f = 0.4$ where the average is taken over 2 sine-wave periods, and the average is zero once again.

This 5-tap filter has a delay of $(5 - 1)/2 = 2$ samples. In terms of phase delay over frequency, this is a *linear phase lag*, expressed as:

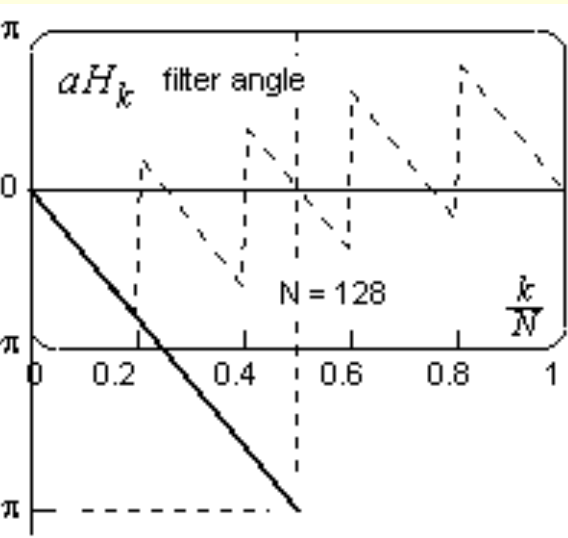
$$\text{lag angle} = -2\pi f n \text{ (radians), with } n = 2$$



The lag angle is greatest at $f = 0.5$, where it reaches -2π radians, which is one sine-wave period. This is indeed a 2-sample delay, because when $f = 0.5$ we have only 2 samples per sine-wave period. Our FFT spectrum gave an angle plot $\arg\{H(f)\}$ that seemed only "piece-wise linear" (Fig 6), but that is very misleading. It has several discontinuities, all with a step size of π radians. These steps actually describe a *change of sign* on the magnitude plot. We can re-draw both plots, making the magnitude plot into a *bi-polar* plot (Fig 7), and removing the steps from the angle plot (Fig 8). We now see the linear phase lag (the heavy line), and it reaches -2π when $f = 0.5$, as expected.



We can use these plots to find the gain and the phase at any frequency. For example, when $f = 0.1$, we find:



$$H(f) = 0.647 \angle (-1.257) \quad \text{when } f = 0.1$$

Suppose we now apply an input sinusoid at $f = 0.1$:

$$x[n] = \cos(2\pi \cdot 0.1 f_s \cdot nT) = \cos(2\pi f n) = \cos(2\pi 0.1 n)$$

$H(f)$ predicts that the output sequence $y[n]$ will be:

$$y[n] = 0.647 \cdot \cos(2\pi 0.1 n - 1.257)$$

smaller in amplitude by 0.647, and lagging by 1.257 radians. To demonstrate that this is what actually happens, we only have to run the difference equation over and over for $n = 0, 1, 2, 3, \dots$ etc.

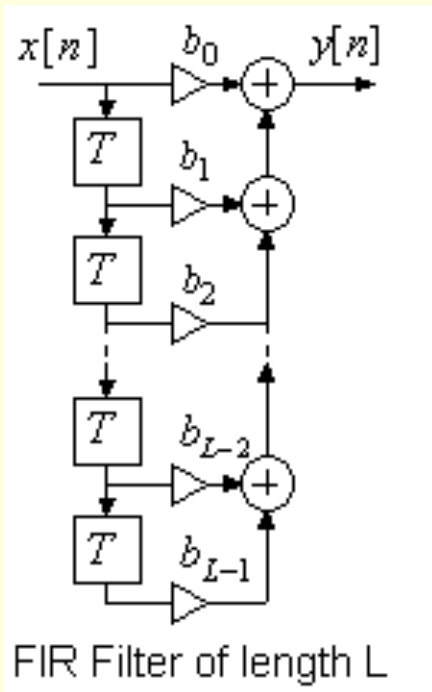
$$y[n] = 0.2x[n] + 0.2x[n-1] + 0.2x[n-2] + 0.2x[n-3] + 0.2x[n-4]$$

The result will be the same as we predicted. And we can do likewise for any other input frequency. That illustrates the importance of the $H(f)$ spectrum.

7.4.2 FIR Filter Structures

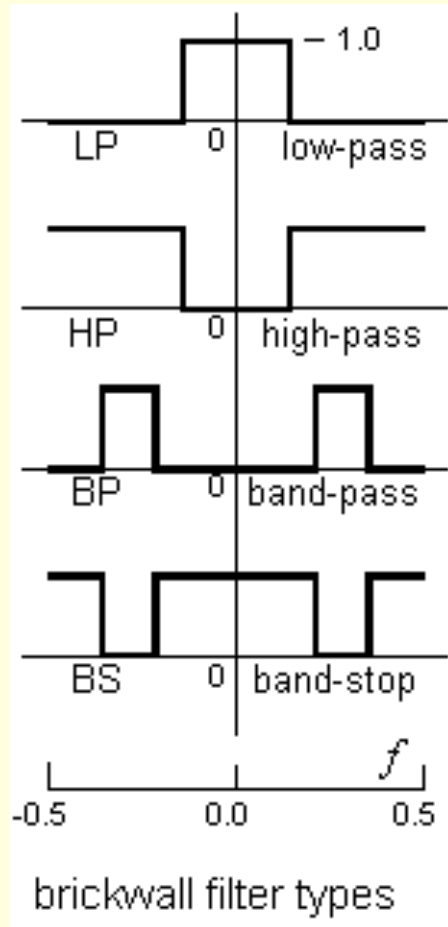
The averaging filter is just a particular case of the generalised FIR filter presented here (Fig 6). It can be of any length L , with L coefficients or "tap values", and with $L - 1$ delay blocks. The coefficients are named as b_0, b_1, b_2 , etc up to b_{L-1} . We operate the filter by repeated execution of its difference equation:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + \dots + b_{L-1}x[n-(L-1)]$$



FIR filter lengths range from $L = 2$ to $L =$ several thousand. With suitable choice of coefficients, we can *approximate* any of the "brick-wall" filters depicted here (Fig 1), but sharp transitions require longer filters.

If the input to this filter were $\delta[n]$, a solitary "1", it would ripple down the delay chain step by step. After the last delay, it would exit and be forgotten. Before exiting, it passes each filter tap in turn, gets scaled by the coefficient value, and then delivered to the output. It follows that the response to $\delta[n]$ will be:



$$h[n] = \{ \underline{b_0} \ b_1 \ b_2 \ b_3 \ .. \ b_{L-1} \ 0 \ 0 \ .. \}$$

In other words, the impulse response of an FIR filter is just the set of its coefficient values. The coefficients also decide the filter spectrum, but we normally handle this the other way round. We specify the spectrum that we want, and we then try to find the coefficients that will give it to us. This is the problem of filter *synthesis*, and we will leave this problem to a later chapter.

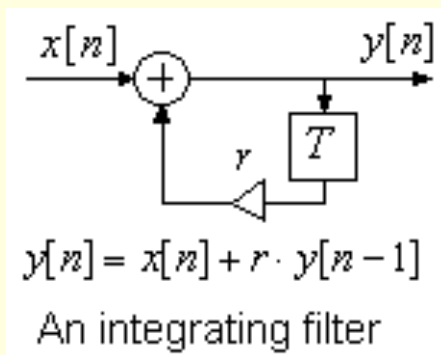




7.5 IIR FILTERS AND FILTER STRUCTURES

FIR filters are popular, but they are not the most efficient computationally. IIR filters can often do a similar job with far fewer coefficients. We will use this section to introduce them.

7.5.1 The IIR Integrator



This block diagram and difference equation (Fig 7.5.1) describe a very different kind of filter. The key difference is the appearance of $y[n-1]$ on the right-hand side. The output is no longer just a weighted sum of inputs. In this filter, earlier output samples are being fed back around a loop to the input. The use of *feedback*, and the presence of a *feedback loop*, are the distinguishing marks of an IIR filter.

The feedback includes a (multiplier) coefficient r . If we set $r = 1$, the difference equation reads:

$$y[n] = x[n] + y[n - 1]$$

All new values of $x[n]$ are added to $y[n]$, so that $y[n]$ becomes the sum of past and present $x[n]$ values. This filter is just an adding machine, but we call it an integrator. If we apply an impulse $\delta[n]$ at the input, the output becomes:

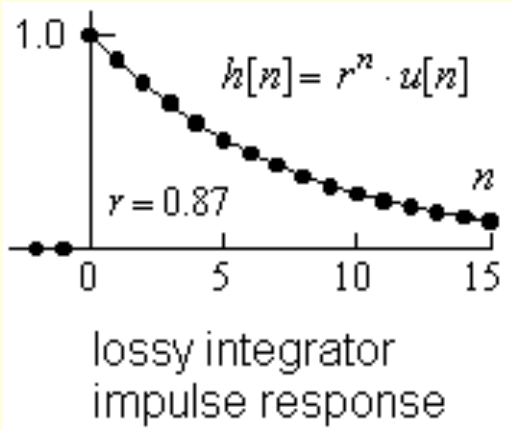
$$y[n] = \{0 \ 0 \ \underline{1} \ 1 \ 1 \ 1 \ 1 \ ..\} = u[n] \quad \text{a unit step sequence}$$

The output $y[n]$ goes from 0 to 1 when the impulse arrives at $n = 0$, and remembers this value thereafter. (It does this by re-circulating the "1" in the loop on every subsequent filter cycle). This assumes that $y[-1]$ was zero, it assumes *zero initial conditions*. We don't need such assumptions with an FIR filter.

Now suppose that $r < 1$. On every trip around the loop, the $y[n]$ value is multiplied by r . The new response to an impulse $\delta[n]$ becomes:

$$y[n] = h[n] = \{0 \ 0 \ \underline{1} \ r \ r^2 \ r^3 \ r^4 \ ..\} = r^n \cdot u[n]$$

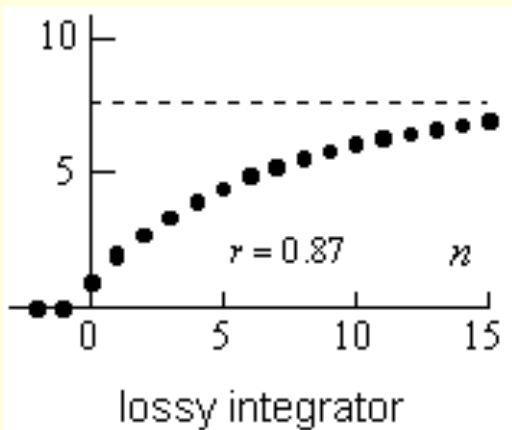
If we use $r = 0.87$, the impulse response looks like this (Fig ç), a decaying exponential sequence and, yes, we've seen this sequence before (i [Ch 7.2.2](#)). This is the response of a "lossy" integrator. It loses a fraction $(1-r)$ of its data on every cycle of the loop. The response decays away to zero, but never quite reaches zero. This $h[n]$ is of infinite length, making this an IIR filter. It is the re-circulation in the feedback loop that make IIR filter responses infinitely long.



If the input to the filter is $x[n] = u[n]$, a unit step sequence, the response will be:

$$y[n] = \sum_{m=0}^n r^m = \frac{1 - r^{n+1}}{1 - r}$$

We can understand this quite readily as the result of a graphical convolution involving $u[n]$ and $h[n]$. It looks like this (Fig c), and it tends to a limit of $1/(1 - r)$ as $n \rightarrow \infty$. This is fine as long as $r < 1$. But, if $r > 1$, the response to an impulse will grow without limit. That would be described as *unstable*, and it cannot be allowed to happen. We must guard against instability with IIR filters, whereas it cannot happen with FIR filters.



The DtFT gives us the spectrum:

Geometric Sums

$$\sum_{n=0}^{N-1} x^n = \frac{1-x^N}{1-x}$$

and ..

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$$

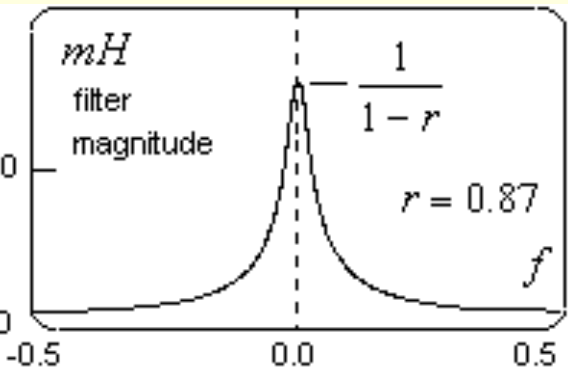
provided $|x| < 1$

$$H(f) = \sum_{n=-\infty}^{\infty} h[n] \cdot e^{-j2\pi f n}$$

normalised DtFT

With the help of this geometric sum (Fig ç), it becomes:

$$H(f) = \sum_{n=0}^{\infty} r^n \cdot e^{-j2\pi f n} = \sum_{n=0}^{\infty} (r \cdot e^{-j2\pi f})^n = \frac{1}{1 - (r \cdot e^{-j2\pi f})}$$

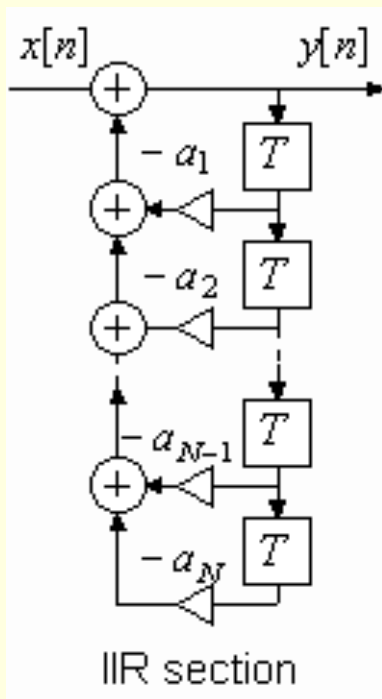


We've plotted the magnitude mH of this $H(f)$ over the base-band, with this result (Fig ç). It is clearly a low-pass effect, with a peak gain of $1/(1-r)$ at $f=0$. We could scale $x[n]$ by $(1-r)$ to give it a DC gain of 1.0 by writing:

$$y[n] = (1-r) \cdot x[n] + r \cdot y[n-1]$$

With this small modification, we have an IIR averaging filter. It gives a weighted combination of the new data $x[n]$ and the existing sum $y[n-1]$. When r is small, the new data is emphasised, and older data is more quickly forgotten.

This example illustrates the major role of geometric sums in the feedback loop of an IIR filter. The impulse response is a geometric series, but the closed-form description of the series sum uses just the feedback coefficient r to describe both the step response in the time domain, and the $H(f)$ of the filter as well.

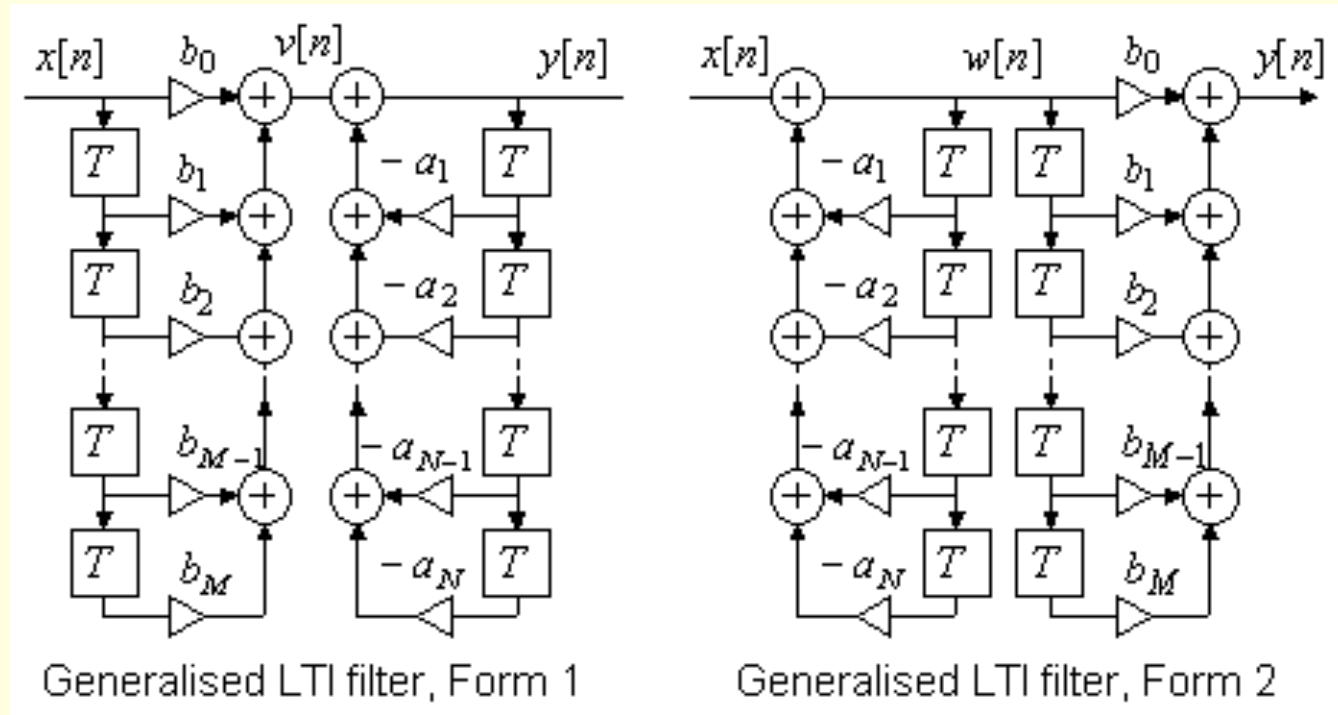


7.5.2 IIR Filter Structures

We've just seen a feedback section with only one feedback coefficient. More generally, we can have several feedback coefficients, like this (Fig ç). We've named them with a minus as $-a_1, -a_2$, etc, because that will give the algebra a cleaner appearance later. This is quite a complex filter, with a difference equation:

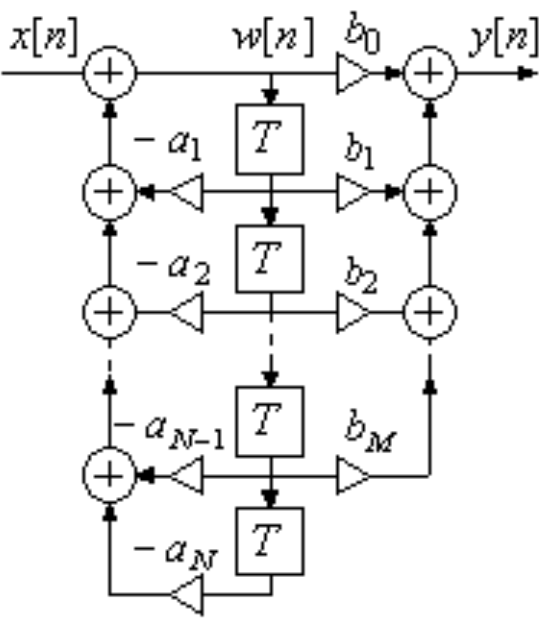
$$y[n] = x[n] - a_1y[n-1] - a_2y[n-2] \dots \dots - a_Ny[n-N]$$

More generally however, we allow an IIR filter to have an FIR section as well. Here (Fig ê) we see an FIR section followed by an IIR section. This is sometimes called the Generalised Form 1, and we can write its difference equation on inspection as:

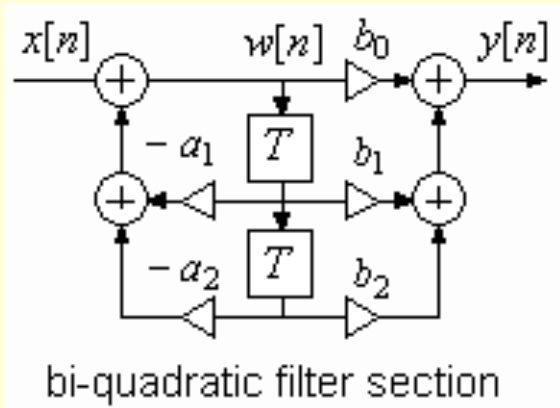


$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] \dots + b_Mx[n-M] - a_1y[n-1] - a_2y[n-2] - a_3y[n-3] \dots - a_Ny[n-N]$$

It turns out that we can place the IIR part in front, and still get the same difference equation, although that will not be obvious on inspection. The result is the Generalised Form 2, also shown (Fig 6). Notice how this form uses two delay chains, both handling the same signal $w[n]$. One of these chains is redundant. We can eliminate the shorter one and arrive at the simpler but equivalent version shown here (Fig 7). It still obeys the difference equation that we presented.



Efficient LTI filter, Form 2



When IIR filters use fixed–point arithmetic, the numeric truncation can give rise to complex and unwelcome non–ideal effects. That is much of the reason why long filters of this kind are seldom used. The IIR filter that has only two delays is a "second–order" filter, often called a "bi–quad" (or bi–quadratic section). It is a widely used building block (Fig ç) with the difference equation:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] - a_1y[n-1] - a_2y[n-2] \quad \text{bi–quad}$$

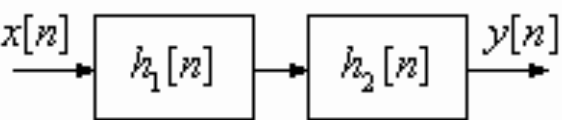
We will use this block quite frequently. It is all that we need to build IIR filters because we can combine several bi–quads in series or in parallel to build filters of higher order. That will be our next topic.

7.5.3 Filter in Series and in Parallel

Almost all of our filters are LTI filters (linear and time–invariant), and this allows us to easily combine them in series or in parallel. Very often, the individual filters will be bi–quads, but other LTI filters can be used as well.

The parallel connection is very simple (Fig ç). We're just adding the outputs of (two or more) individual filters. That's the same as adding their individual impulse responses. It also means that we can add the two filter spectra to get an equivalent filter spectrum $H(f)$.

LTI filters in parallel
 LTI filters in series cascade



$$h[n] = h_1[n] * h_2[n]$$

$$H(f) = H_1(f) \cdot H_2(f)$$

$$y[n] = y_1[n] + y_2[n]$$

$$h[n] = h_1[n] + h_2[n]$$

$$H(f) = H_1(f) + H_2(f)$$

The series connection is a "cascade" of two or more filter blocks as shown (Fig 1). It's easy to show that the overall filter impulse response is the *convolution* of the individual impulse responses. We also noted that convolution was commutative, and that means that we can interchange the order of the blocks without altering the result. We used this property when we replaced a Form 1 structure with a Form 2 structure.

In the next chapter, we will see how convolution in one domain is equivalent to multiplication in the other domain. For filters in series cascade, it means that we can multiply their spectral functions to get the overall filter spectrum.

Later on, we will learn how to design bi-quads for series or parallel connection, so as to implement filters that can meet some specified $H(f)$ description.





7.6 CORRELATION METHODS

Some filters have a purpose somewhat different from the norm. Their job is to recognise a certain known pattern of input samples $x[n]$. Recognition would be easy except that the known pattern $x[n]$ may be heavily corrupted by noise (as when we try to recognise radar pulses after they have been reflected from an aircraft). That calls for a filter which responds more strongly to the expected $x[n]$ pattern than to any other input sequence. It is known as a matched filter.

7.6.1 The Matched Filter Concept

The output from a filter is a sum of products. The idea of a matched filter is to incorporate the $x[n]$ "signature" in the detection filter in a way that will maximize the sum of products for the anticipated $x[n]$. A maximized sum, with $x[n]$ as one of the sequences, would be the sum of the squared $x[n]$ values (times an arbitrary scale factor). We can get this result from an FIR filter whose impulse response $h[n]$ is a *time-reversed* version of the expected input sequence $x[n]$. This example uses an arbitrary $x[n]$ of length six (Fig 6.1). To match this $x[n]$, we need a filter described by $h[n] = \{ 5 \ -4 \ 3 \ -1 \ 2 \ 1 \}$, a causal time-reversed copy of $x[n]$. To perform the convolution, we must flip $h[n]$ as shown, then slide it across the $x[n]$ sequence:

$h[n-m] :$	1 2 -1 3 -4 <u>5</u>	→	<i>pictured at $n = 0$</i>
$x[m] :$	<u>1</u> 2 -1 3 -4 5		..
$y[n] :$	<u>5</u> 6 -10 24 -35 56 -35 24		..

▲

The output $y[n]$ will be maximised when $n = 5$, with a value of:

$$y[5] = \sum_{m=0}^5 x[m]^2 = E_x = 56 \quad \text{the total energy in } x[n]$$

The peak output is also the energy of the $x[n]$ sequence. If there is added noise, the noise terms will tend to cancel rather than to accumulate. Whenever $y[n]$ reaches a peak that exceeds some agreed threshold, we conclude that the expected $x[n]$ has been detected.

7.6.2 Cross-correlation and Auto-correlation

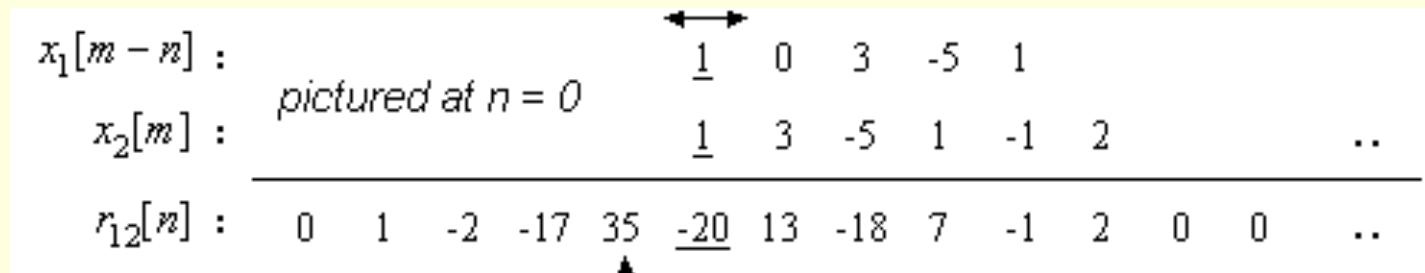
In the above illustration, we took two *identical* number sequences and we slid one across the other until we found a match (at $n = 5$), as indicated by the peaking of the output sequence $y[n]$. When the sequences are identical, this kind of action is called *auto-correlation*. But, in a practical detection filter, the input $x[n]$ will have added noise, and will not be identical to $h[n]$. We are then comparing two *different* sequences, and we are looking for some similarity between them, as indicated by a peaking of $y[n]$. This is called *cross-correlation*, and a matched filter is a cross-correlator that (usually) works in real-time.

For sequences $x_1[n]$ and $x_2[n]$, the cross-correlation sequence is:

$$r_{12}[n] = \sum_m x_1[m-n] \cdot x_2[m]$$

It looks very like a convolution. The only difference is that we used $x_1[m-n]$ rather than $x_1[n-m]$. It means that unlike convolution, when we do this operation on paper, we do *not* reverse one of the sequences beforehand.

Here is a simple example of a cross-correlation of two causal sequences of lengths 5 and 6 respectively (Fig 6). We write both sequences as given, then we slide the upper sequence to the left and to the right, computing sums of products until no overlap remains. The result $r_{12}[n]$ is of length 10, one less than the sum of the sequence lengths, and the result is not causal (it has non-zero values at negative indexes, and the maximum similarity is found at $n = -1$). If this were a convolution, the result $y[n]$ would again have length 10, but it would be a causal sequence.



Cross-correlation and convolution are similar in that the same numeric algorithm can be used to do both. (The difference is in whether we reverse one sequence before we enter it). But they are very different in other ways. Convolution is usually a real-time operation, one that uses new data as it becomes available. With correlation, we would often have all of the data to hand before doing a comparison. Cross-correlation is the fundamental operation that underlies identification methods as performed by a computer (as in fingerprint matching, voice recognition, etc).

7.6.3 Convolution versus Correlation

Unlike convolution, cross-correlation is not commutative. In fact:

$$r_{21}[n] = r_{12}[-n]$$

This means that, if we slide $x_2[n]$ instead of $x_1[n]$, we get the same answer but in time-reversed order. That is very easy to verify. On a direct comparison of correlation and convolution:

$$x_1[n] * x_2[n] = \sum_m x_1[n-m] \cdot x_2[m], \quad r_{12}[n] = \sum_m x_1[m-n] \cdot x_2[m]$$

we observe that:

$$r_{12}[n] = x_1[-n] * x_2[n]$$

It is the convolution of a time-reversed $x_1[n]$ with $x_2[n]$. In similar manner:

$$r_{21}[n] = x_2[-n] * x_1[n]$$

Finally, if $y[n] = x_1[n] * x_2[n]$, we also find that:

$$y[-n] = x_1[-n] * x_2[-n]$$

If we time-reverse *both* input sequences, we get the normal $y[n]$ obtained under convolution, except that this too will be reversed in time.

7.6.4 Auto-correlation of Sequences

Auto-correlation is the special case of cross-correlation where the two participating sequences are identical. So, it can hardly be for the purpose of discovering similarities ! But, the auto-correlation of a signal can tell us a lot about that signal, as we shall see. A sequence $x_1[n]$ has an auto-correlation sequence $r_{11}[n]$ obtained as:

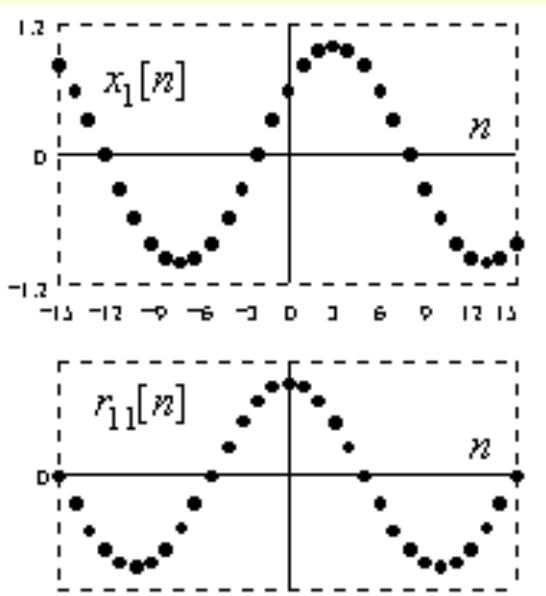
$$r_{11}[n] = \sum_m x_1[m-n] \cdot x_1[m]$$

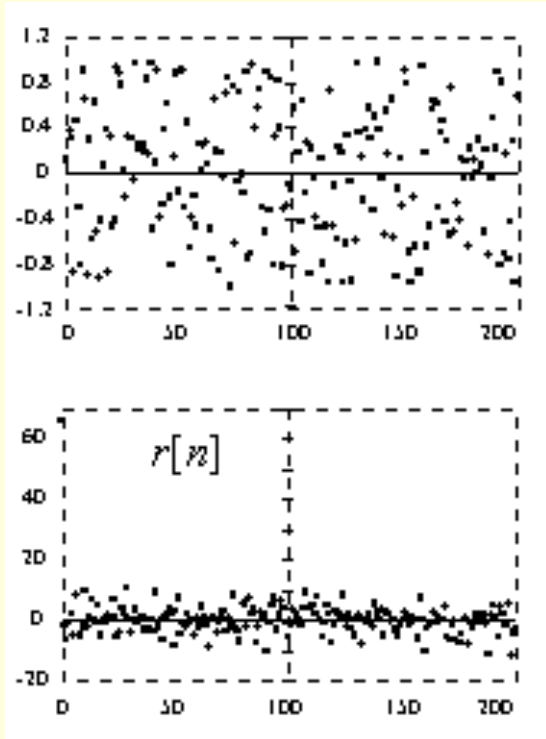
The auto-correlation $r_{11}[n]$ has its highest peak at $n = 0$, with a value of:

$$r_{11}[0] = \sum_m x_1[m]^2 = E_{x1} \quad \text{the signal energy}$$

This is the "perfect match" position. We can also see very quickly that $r_{11}[-n] = r_{11}[n]$, meaning that it has even symmetry about $n = 0$. This is true of auto-correlation sequences in general. (It was true of our matched filter when the input $x[n]$ was noise free, however, the peak came later at $n = 5$, because of the causality requirement under real-time working).

We'll use two very different sequences to illustrate some aspects of auto-correlation. This first sequence is a sampled sinusoid $x_1[n]$ (Fig ζ), with its auto-correlation function $r_{11}[n]$ shown underneath. The auto-correlation is a cosine, featuring even symmetry about $n = 0$ as expected. It also peaks at $n = 0$ (when the two copies coincide and are "in-phase"), *regardless* of the initial phase of $x_1[n]$. We also see that $r_{11}[n]$ is zero when the copies are in quadrature ($1/4$ cycle apart). This $r_{11}[n]$ has a *broad* peak region, and it repeats itself periodically.





The second example is a random noise sequence (Fig 1), with its auto-correlation sequence $r[n]$ shown underneath. We've not shown the $n < 0$ part, because the symmetry makes it unnecessary. It has one very large value (> 60) when $n = 0$, and quite small values everywhere else. This is easily explained. When $n = 0$, we are comparing two identical copies, and the result is the total energy (or sum of squares). But, with random noise, any sample is *unrelated* to the sample beside it, so that even for $n = 1$, we are comparing two entirely different sequences, and the sum of products will tend toward zero as an average (provided the sequence has no DC component). This $r[n]$ has a very *narrow* peak region (one sample wide), in fact, it tends toward $K \cdot \delta[n]$ in the limit as more and more input samples are included.

Based on these examples, we can appreciate the following. Noise sequences have little predictability from sample to sample, they have a wide range of frequencies, and they have a *narrow* auto-correlation peak. If the noise signal is then low-pass filtered, the higher frequencies are removed, it becomes more slowly varying, there is much greater predictability from sample to sample, and its auto-correlation peak becomes very much wider.

The auto-correlation $r[n]$ sequence says much about the frequency content of a signal. Later on, we will see how its Fourier transform is also the signal's power spectrum. As such, it tells us all about signal amplitudes over frequency, but it tells us nothing about the phase (or the timing information).

To sum up, convolution is about filtering a signal, cross-correlation is about comparing two signals, and auto-correlation is about the properties of a signal. This has been just a brief introduction, with much more to be said later on.





8.1 PREAMBLE

One way to design a digital filter is to have it mimic the action of a known analogue filter such that, as the sample interval $T \rightarrow 0$, they become more nearly alike. By letting $T \rightarrow 0$ with our digital filter, we will arrive at an analogue filter model, and at an integral that describes analogue convolution. We will also see the analogue equivalents of cross-correlation and auto-correlation.

To assist us in this effort, we will also need the spectral view that the Fourier Transform (FT) can provide. Although we've encountered the FT in several guises, we have yet to document the FT properties and the FT pairs that will be useful to us. We'll present these initially in terms of the CFT, but we'll extend them where appropriate to cover other useful forms.



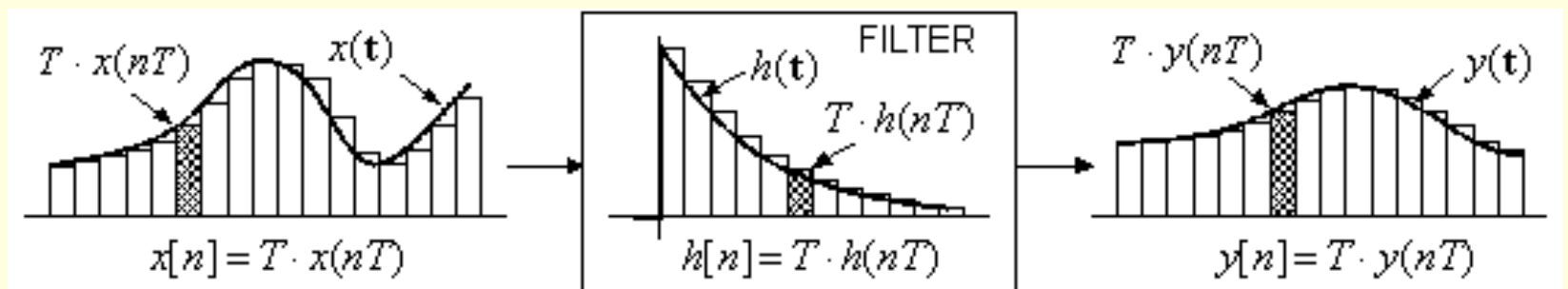


8.2 ANALOGUE FILTER DESCRIPTIONS

This section will take us from operations on sequences to the corresponding operations on analogue signals, and will give us the time-domain models that we use for linear analogue processing of signals.

8.2.1 From Digital to Analogue

This diagram (Fig 8.2) shows the digital filter as approximating the action of an analogue filter.



The filter generates the output sequence as $y[n] = x[n] * h[n]$. In order that the time scale is included, we treat these numbers as *area-samples* (rather than value-samples) of corresponding analogue signals, namely, $y(t)$, $x(t)$ and $h(t)$. We can therefore write the convolution as:

$$y[n] = \sum_m h[n-m] \cdot x[m]$$

or as:

$$T \cdot y(nT) = \sum_m T \cdot h(nT - mT) \cdot T \cdot x(mT)$$

We're free to divide across by T . That gives us the option to think of $x[n]$ and $y[n]$ as value-samples, if we prefer, but we *must* continue to think of $h[n]$ as area-samples of an analogue impulse response $h(t)$. We will go with this option, because value-samples are the norm for practical purposes, so we now have:

$$x[n] = x(nT), \quad y[n] = y(nT), \quad h[n] = T \cdot h(nT)$$

and

$$y(nT) = \sum_m T \cdot h(nT - mT) \cdot x(mT)$$

If we now let $T \rightarrow 0$, the summation becomes an integral with continuous time variables replacing the discrete time steps:

$$nT \rightarrow \mathbf{t} \quad mT \rightarrow \tau, \quad T \rightarrow d\tau$$

and resulting in:

$$y(\mathbf{t}) = \int_{-\infty}^{\infty} h(\mathbf{t} - \tau) \cdot x(\tau) \cdot d\tau$$

the convolution integral

We've seen how, with digital convolution, we could interchange the roles of $x[n]$ and $h[n]$. The same is true here:

$$y(\mathbf{t}) = \int_{-\infty}^{\infty} x(\mathbf{t} - \tau) \cdot h(\tau) \cdot d\tau$$

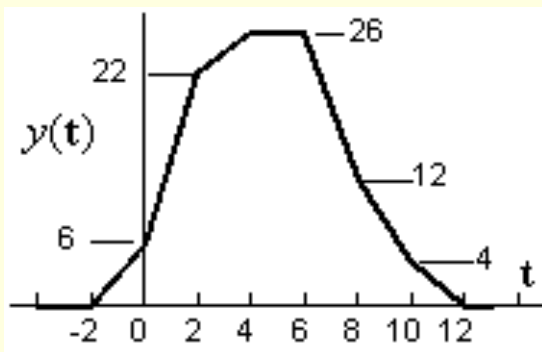
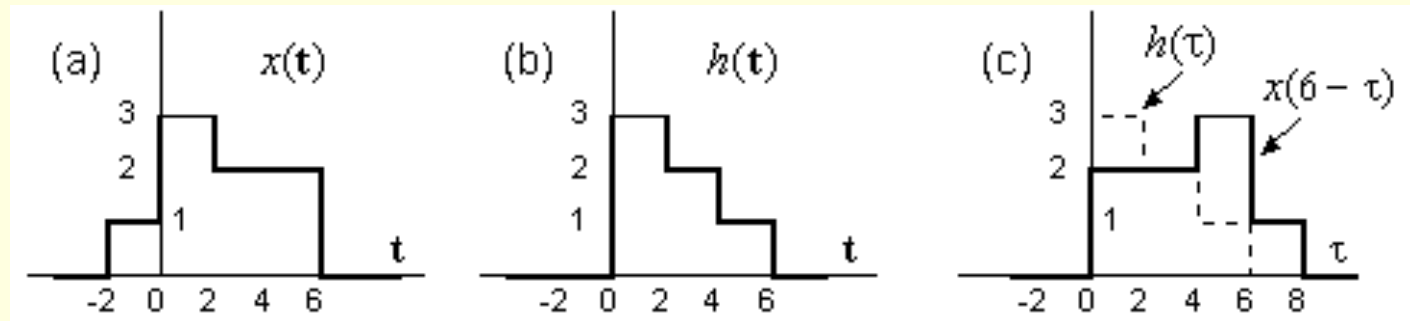
the convolution integral

In both these forms, \mathbf{t} is the real-time variable, and τ is a dummy time variable for the integration. For an integration in progress, \mathbf{t} is just a constant that sets the position of $h(-\tau)$, or of $x(-\tau)$, these being the time-reversed versions of h or

x respectively.

8.2.2 Analogue Convolution and Correlation

As an illustration of analogue convolution by the graphical method, we will convolve the $x(t)$ of plot (a) with the $h(t)$ of plot (b) as shown here (Fig 8.2.2).

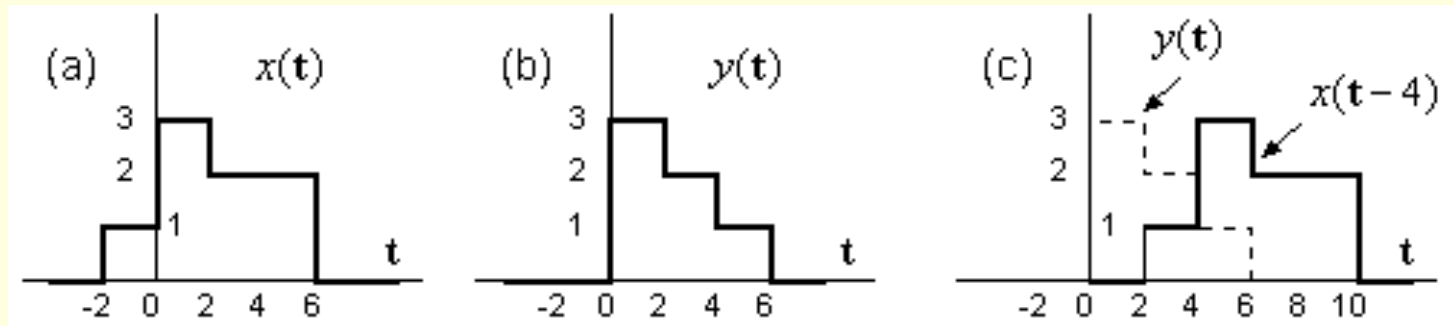


These regular shapes make it easy to do. In plot (c), we time-reverse $x(\tau)$ and we slide it past $h(\tau)$, computing the *the area under the product* as we go. This area will be the convolution result, $y(t) = x(t) * h(t)$. We've shown it at $t = 6$, where the area under the product is made up of three parts, namely, $(3 \times 2 \times 2) = 12$, plus $(2 \times 2 \times 2) = 8$, plus $(1 \times 3 \times 2) = 6$, for a total area of 26. This is the value of $y(6)$. After several other similar calculations, we get the result shown here (Fig 8.2.2). Careful checking of this example should result in a good understanding of analogue convolution.

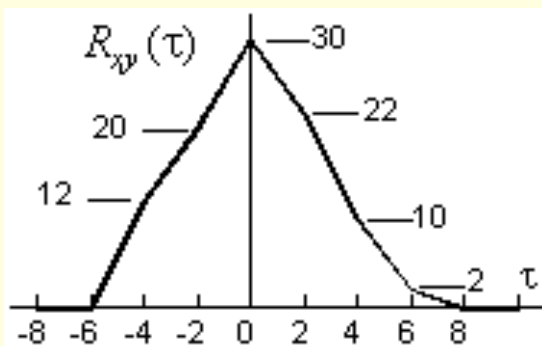
Analogue correlation is like analogue convolution except that we do not reverse one of the signals before we integrate. Using $R_{xy}(\tau)$ to mean the cross-correlation of $x(t)$ with $y(t)$, the correlation integral is:

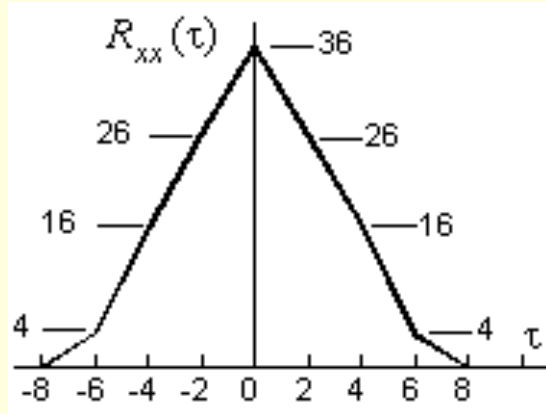
$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t-\tau) \cdot y(t) \cdot dt$$

As usual, t is the real-time variable. The time variable τ represents the *relative displacement* of $x(t)$ from $y(t)$ at which we are testing for similarity. For an integration in progress, τ is a constant, and is the lateral shift of $x(t)$ for this comparison. But $x(t)$ is *not* reversed in time. To illustrate, we will use the same two signals as we did for the convolution example, but we will re-name $h(t)$ as $y(t)$ to avoid any impression of filter action. These are just two arbitrary signals x and y that we wish to compare (Fig \hat{e}).



In plot (c), $x(t)$ has been moved to the right by 4, but not reversed. The sum of areas becomes $(2 \times 1 \times 2) = 4$, plus $(1 \times 3 \times 2) = 6$, a total of 10, which is $R_{xy}(4)$. After several other similar calculations, we get the result shown here (Fig ζ). Careful checking of this example should result in a good understanding of analogue cross-correlation.



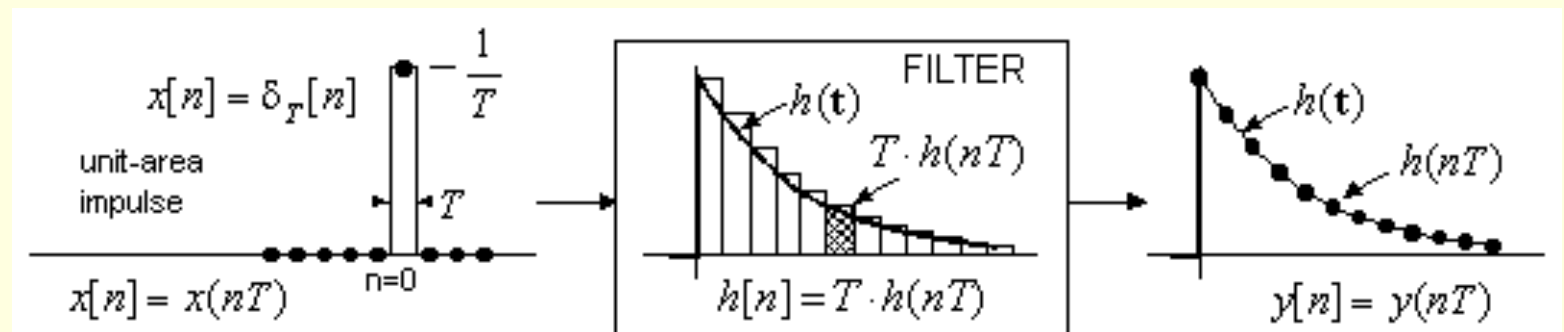


Analogue auto-correlation is no different, except that we use the same signal twice over. This (Fig ç) is a plot of $R_{xx}(\tau)$, and it has the expected peak at $\tau = 0$, and the expected even symmetry as well. Also, the information that $R_{xx}(\tau)$ gives us about $x(t)$ is the same as in the digital case.

8.2.3 Convolution with Impulses

The impulse response of a digital filter is a complete filter description. We can expect a similar result for analogue filters, which means that, when the input to the filter is the analogue impulse function $\delta(t)$, the filter output will be its analogue impulse response $h(t)$, and this too is a complete filter description.

The analogue impulse $\delta(t)$ is infinitely tall and infinitely narrow, and has unit area. The nearest approximation, for digital working, would be a pulse of width T and height $1/T$, where T is the sampling interval. The value-samples from this pulse would be $\delta T[n]$, with a sample of value $1/T$ at $n = 0$, and all zeros elsewhere.



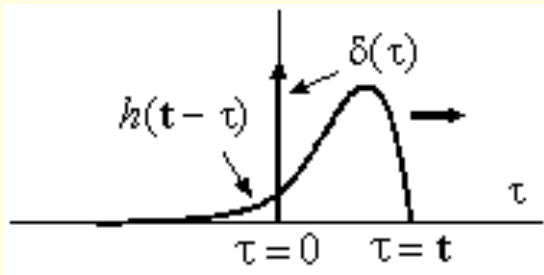
That is the situation shown here (Fig 6). The response is:

$$y(nT) = \sum_m T \cdot h(nT - mT) \cdot x(mT) = \sum_m T \cdot h(nT - mT) \cdot \delta_T(mT)$$

The summation over m is a trivial one, because only when $m = 0$ do we get a non-zero value, this value being $1/T$. The result is:

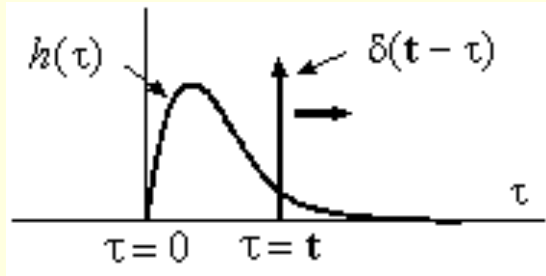
$$y(nT) = T \cdot h(nT) \cdot \frac{1}{T} = h(nT)$$

The output samples are value-samples of $h(t)$, the analogue filter's impulse response. In the limit, as $T \rightarrow 0$, our digital filter becomes an analogue filter which, when driven by $\delta(t)$, delivers $y(t) = h(t)$ to the output. There are two ways to visualise this:



or

$$y(t) = \int_{-\infty}^{\infty} h(t - \tau) \cdot \delta(\tau) \cdot d\tau = h(t) \quad \text{by sliding } h(-t) \text{ over } \delta(t)$$



$$y(t) = \int_{-\infty}^{\infty} \delta(t - \tau) h(\tau) d\tau = h(t)$$

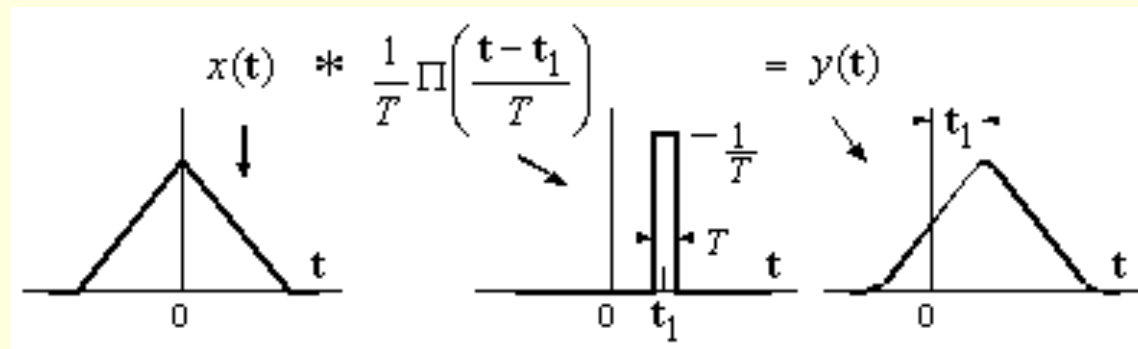
by sliding $\delta(-t)$ over $h(t)$

We can time-reverse $h(\tau)$ and slide it past $\delta(\tau)$ as shown (Fig e), or we can time-reverse $\delta(\tau)$ and slide it past $h(\tau)$, like this (Fig c). Notice, reversal of $\delta(\tau)$ has no effect. In both cases however, the product of h and δ yields an impulse of area $h(t)$ which, on integration, becomes the filter output, $y(t) = h(t)$. Thus:

$$y(t) = h(t) * \delta(t) = h(t)$$

The sliding action is a *scanning* of $h(t)$ by the impulse $\delta(t)$ which, under convolution, yields an exact copy of $h(t)$. Because $\delta(t)$ is the "perfect" impulse, the result of the scan is a "perfect" copy of $h(t)$.

We'll meet other situations where a signal is scanned by an *approximate* impulse shape, resulting in a less-than-perfect copy of the signal. This is illustrated here (Fig i).



The signal is a triangle, and is convolved with (or scanned by) a tall narrow rectangular pulse of unit area, located at time $t = t_1$. The result is a rather "blurred" triangle, right-shifted by t_1 seconds. Sharp corners have been smoothed, because of the finite resolving power of the approximate impulse. If we could let $T \rightarrow 0$, the impulse would become a perfect one, and the blurring would disappear. We can see all this intuitively from our understanding of analogue convolution. The concept is an important one, because it models many real-world measurement situations, where the result of the measurement is frequency-limited by the finite resolving power of the measuring device, which in turn is modelled by a less-than-perfect impulse shape. Indeed, a filter described as $h(t)$ could be the model for our measuring instrument, and the measure of its performance is the closeness of $h(t)$ to the ideal impulse shape. Ideally, we would have $h(t) = \delta(t)$, so that the measured response to an impulse at the input becomes $y(t) = \delta(t)$, a perfect reproduction of that impulse.





8.3 CFT PROPERTIES

To further our work with filters, and in other areas too, we will revisit the CFT and its properties, and we will tabulate some CFT pairs. We will later extend our findings to the DtFT, which we use to obtain the spectra of digital filters.

8.3.1 CFT Properties

We will list several properties (Fig 8.1) of the CFT. They are easily proved, and many of them simply mirror observations that we made earlier about sine waves and Fourier Series, so they should come as no surprise.

BASIC PROPERTIES OF THE CFT

#	Property	Time Domain	Frequency Domain
1	Linearity	$a_1x_1(\mathbf{t}) + a_2x_2(\mathbf{t})$	$a_1X_1(\mathbf{f}) + a_2X_2(\mathbf{f})$
2	Time Reversal	$x(-\mathbf{t})$	$X(-\mathbf{f})$
3	Scaling W real and positive	$x(\mathbf{t}/W)$	$W \cdot X(W\mathbf{f})$
4	Duality	$X(\mathbf{t})$ $X(-\mathbf{t})$	$x(-\mathbf{f})$ $x(\mathbf{f})$
5	Time Shift	$x(\mathbf{t} - \tau)$	$X(\mathbf{f}) \cdot e^{-j2\pi\mathbf{f}\tau}$
6	Frequency Shift	$x(\mathbf{t}) \cdot e^{j2\pi\mathbf{f}_0\mathbf{t}}$	$X(\mathbf{f} - \mathbf{f}_0)$
7	Convolution in Time	$x_1(\mathbf{t}) * x_2(\mathbf{t})$	$X_1(\mathbf{f}) \cdot X_2(\mathbf{f})$
8	Convolution in Frequency	$x_1(\mathbf{t}) \cdot x_2(\mathbf{t})$	$X_1(\mathbf{f}) * X_2(\mathbf{f})$
9	Correlation (Parseval's Theorem)	$\int_{-\infty}^{\infty} x_1(\mathbf{t}) \cdot x_2^*(\mathbf{t}) \cdot d\mathbf{t} = \int_{-\infty}^{\infty} X_1(\mathbf{f}) \cdot X_2^*(\mathbf{f}) \cdot d\mathbf{f}$	

We will comment briefly on each. Property 1, linearity, is vital. It says that the response is directly proportional to the stimulus, and it allows us to superimpose the results from two or more stimuli. Without linearity, attempts at analysis are greatly complicated. Linearity is a basic premise in nearly all that we do.

Property 2, time reversal, mirrors our earlier experiences with isolated sine waves, and with Fourier Series waveforms. If we "flip" the time pulse, then we "flip" the spectral description as well. For *real-valued* time signals, the flipped spectrum $X(-\mathbf{f})$ is the same as $X^*(\mathbf{f})$, because of the even-magnitude, odd-phase spectral symmetry.

Property 3, scaling, is one that we use a lot, and should be clearly understood. The scaling parameter W is assumed to be a positive real number. Let us suppose that W is greater than 1. Then $x(\mathbf{t}/W)$ is a version of $x(\mathbf{t})$ that has been *stretched horizontally* by a factor W . Its spectrum, $W \cdot X(W\mathbf{f})$, is a version of $X(\mathbf{f})$ that has been *compressed horizontally*, with scale factor $(1/W)$, but it has also been *stretched vertically* by a factor W . Overall, the area of $W \cdot X(W\mathbf{f})$ remains *unchanged* by the scaling. We can summarise the scaling action as follows:

- *When a pulse waveform undergoes horizontal expansion (or compression) in one domain, its counterpart in the other domain undergoes compression (or expansion), and also a height adjustment, such that its area does not change.*

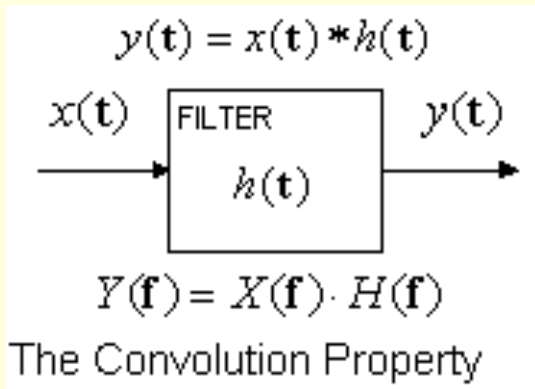
Scaling Rule

This is most important, because it allows us to define transform pairs from basic pulse *shapes*, and to adjust their dimensions later using the scaling rule. We will use this rule extensively. But property 1 (linearity) also has a role in re-shaping a pulse: if we change the pulse *height* in one domain, then from linearity, the *height* in the other domain follows suit.

Property 4, duality, exploits the symmetry that we have already noted. It uses the fact that the forward CFT and the inverse CFT are identical, except for the sign in the exponent. They differ only in the *direction of rotation* of the phasor term. Property 4 is the general result, which we have written in two alternative forms. But note also that, if a function $x(\cdot)$ has even symmetry, then $x(-\mathbf{f}) = x(\mathbf{f})$ and we can directly swap $x(\mathbf{t})$ with $X(\mathbf{f})$ to get a new transform pair. We will see examples of this shortly.

Property 5, time shift, should come as no surprise. It just says that, if we delay a signal $x(t)$ by t seconds, then its spectrum acquires a corresponding *linear phase lag*, amounting to $-2\pi ft$ radians. We made similar observations about Fourier Series.

Property 6, frequency shift, says that if we modulate (multiply) a signal $x(t)$ using a phasor of frequency f_0 , then we shift (translate) the signal spectrum horizontally by amount f_0 . This property will be important when we discuss band-pass signals for data communication. Properties 5 and 6, time shift and frequency shift, are closely similar in form. That is a consequence of duality, which was property 4.



Property 7, convolution in time, has great importance. The proof is not difficult, but the impact is large. We've seen how an analogue filter $h(t)$, shown here (Fig c), delivers an output $y(t)$ which is the *convolution* of $x(t)$ with $h(t)$. This property tells us that we get an equivalent effect by *multiplying* the spectra of $x(t)$ and $h(t)$, that is, $Y(f) = X(f) \cdot H(f)$. This result is important for a few reasons. First, multiplication is a far simpler (and faster) operation than convolution. Second, the spectral view is very useful in engineering. And in addition, a similar convolution property applies to digital signal processing, and we will use it extensively.

Property 8, convolution in frequency, is linked to property 7 by duality. It is useful in band-pass communication theory, for interpreting the spectra of time signals that have been modulated (meaning *multiplied*).

Property 9 differs from all the rest in that no transform pair is involved. The time-integral is a correlation test (or a similarity test) involving the time-signals $x_1(t)$ and $x_2(t)$. The frequency-integral is a correlation test (or a similarity test) involving the signal spectra $X_1(f)$ and $X_2(f)$. This property establishes that both tests give the same result. It means that similarity testing can be conducted in either domain. If $x_1(t)$ and $x_2(t)$ show strong similarity, then so also will $X_1(f)$ and $X_2(f)$. Seems very reasonable !

In the case where $x_1(t) = x_2(t) = x(t)$, the same property becomes a comment on the *energy* of $x(t)$, as we now demonstrate.

The integrals become:

$$\int_{-\infty}^{\infty} x(\mathbf{t}) \cdot x^*(\mathbf{t}) \cdot d\mathbf{t} = \int_{-\infty}^{\infty} X(\mathbf{f}) \cdot X^*(\mathbf{f}) \cdot d\mathbf{f}$$

For *any* vector v , the product $v \cdot v^*$ equals the squared magnitude, $|v|^2$. We have two instances of this here, so that the integrals reduce to:

$$\int_{-\infty}^{\infty} |x(\mathbf{t})|^2 \cdot d\mathbf{t} = \int_{-\infty}^{\infty} |X(\mathbf{f})|^2 \cdot d\mathbf{f} \quad = \text{total signal energy}$$

The left side is what we defined as the total energy of $x(\mathbf{t})$, and the right side now emerges as an *alternative* energy measure, expressed in terms of $X(\mathbf{f})$. Both these measures are of the same form. $|x(\mathbf{t})|^2$ is energy per unit time, better known as power. $|X(\mathbf{f})|^2$ is energy per unit bandwidth, in Hertz, better known as *energy spectral density*.

8.3.2 Basic CFT Pairs

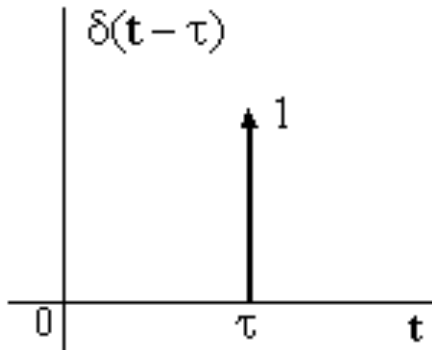
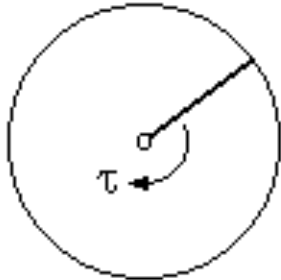
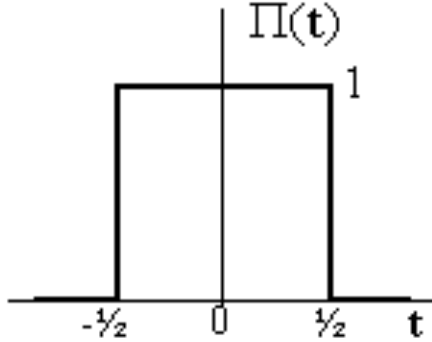
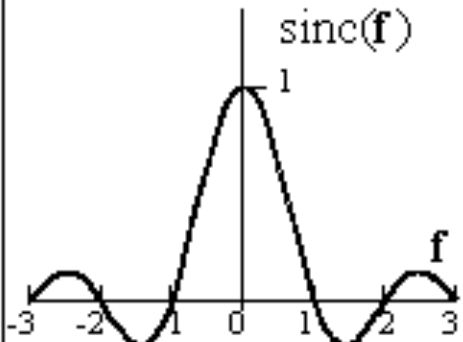
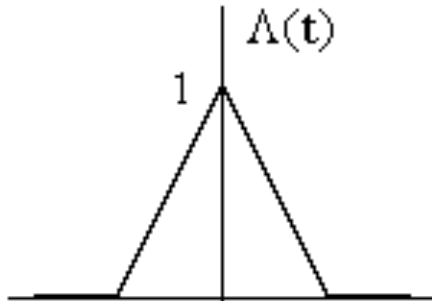
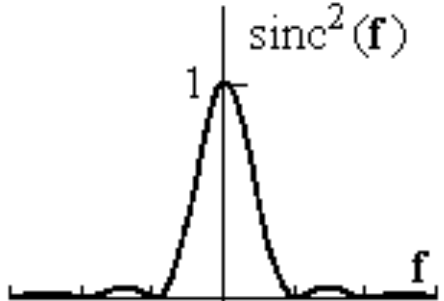
We will now assemble a small collection of "fundamental" CFT pairs. We will meet other pairs too, but those pairs are easily derived from the pairs shown here, with the help of the CFT properties which we listed.

The CFT pairs are tabulated below (Fig 6). Pair 1 connects an impulse in time to a phasor in frequency. We can prove this in one line, as follows:

$$X(\mathbf{f}) = \int_{-\infty}^{\infty} \delta(\mathbf{t} - \tau) \cdot e^{-j2\pi\mathbf{f}\mathbf{t}} \cdot d\mathbf{t} = e^{-j2\pi\mathbf{f}\tau} \cdot \int_{-\infty}^{\infty} \delta(\mathbf{t} - \tau) \cdot d\mathbf{t} = e^{-j2\pi\mathbf{f}\tau}$$

To explain this, if we take *any* continuous $x(\mathbf{t})$, in this case a phasor, and multiply it by the unit-impulse $\delta(\mathbf{t} - \tau)$, the product is an *impulse* at time $\mathbf{t} = \tau$, with a strength equal to $x(\tau)$. When we then integrate over the impulse, the answer is the impulse strength, $x(\tau)$.

BASIC CFT PAIRS

	$\{x(t), X(f)\}$ pair	$x(t)$ graph	$X(f)$ graph
1	Impulse to Phasor $\delta(t - \tau) \xleftrightarrow{\text{CFT}} e^{-j2\pi f\tau}$		
2	Rectangle to Sinc $\Pi(t) \xleftrightarrow{\text{CFT}} \text{sinc}(f)$		
3	Triangle to sinc ² $\Lambda(t) \xleftrightarrow{\text{CFT}} \text{sinc}^2(f)$		

Though the proof is easy, the interpretation may seem otherwise. But consider the special case which occurs if there is no delay. By setting $\tau = 0$ we get:

$$\delta(\mathbf{t}) \stackrel{\text{CFT}}{\leftrightarrow} 1.0$$

It says that an impulse has a flat frequency spectrum; it contains all frequencies in equal measure. For the more general case, we can read:

$$\delta(\mathbf{t} - \tau) \stackrel{\text{CFT}}{\leftrightarrow} 1.0 \quad \times \text{ (linear-phase lag of } -2\pi\mathbf{f}\tau \text{ radians)}$$

Thus the spectral amplitude is 1.0 at all frequencies and the phasor term merely accounts for the τ -second delay. We see that *a simple time delay translates into a phasor function of frequency.*

We gave a proof of Pair 2 much earlier (i [Ch 3.2.2](#)). The "sinc" shape of the spectrum $X(\mathbf{f})$ suggests that a rectangular pulse has predominantly low frequencies, but it has a noticeable content at much higher frequencies as well. The latter are necessary for the construction of fast changing edges, which are the step-wise boundaries of the time-pulse

Pair 3 is easily deduced from pair 2 using property 7, convolution in time, and it follows from the fact that $\Lambda(\mathbf{t})$ is the convolution of $\Pi(\mathbf{t})$ with itself, that is:

$$\Lambda(\mathbf{t}) = \Pi(\mathbf{t}) * \Pi(\mathbf{t})$$

This convolution is easy to visualise. Then, by property 7, the spectrum of $\Lambda(\mathbf{t})$ is $\text{sinc}(\mathbf{f}) \cdot \text{sinc}(\mathbf{f}) = \text{sinc}^2(\mathbf{f})$. This transform pair will later find use in some practical engineering applications. This pair also accounts for the $X(\mathbf{f})$ that we used earlier to describe a triangular pulse train, when we said (i [Ch 4.2.3](#)):

$$C_k = \varepsilon \cdot X(k\varepsilon) \quad \text{with} \quad X(\mathbf{f}) = \frac{1}{2}AW \cdot \text{sinc}^2(\frac{1}{2}W\mathbf{f})$$

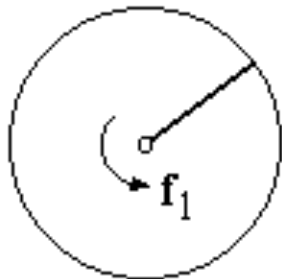
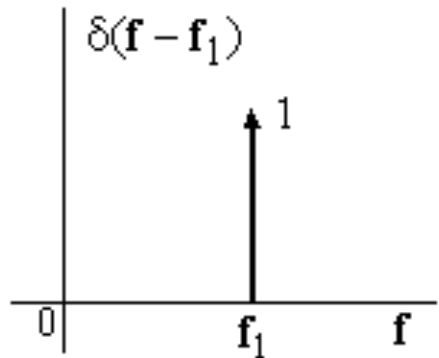
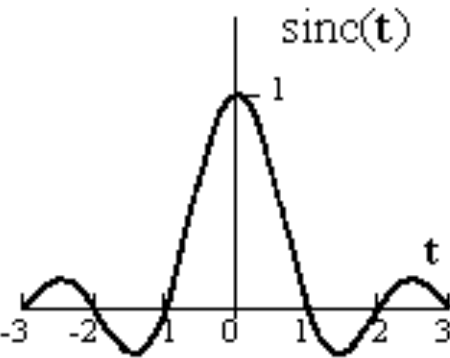
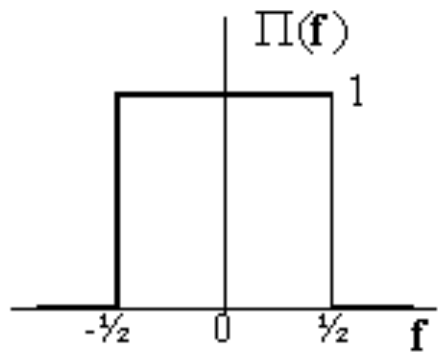
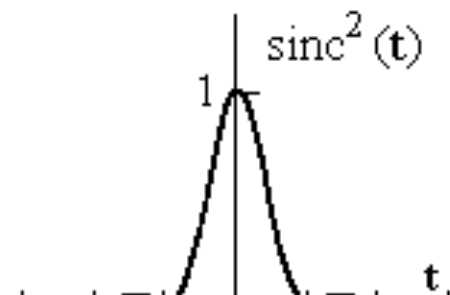
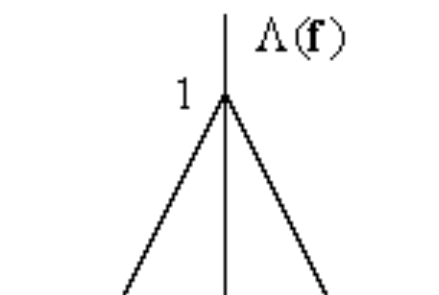
To arrive at this result, we first take the shape $\Lambda(\mathbf{t})$, we multiply it by A for a peak height of A , and then we *stretch* it by $\frac{1}{2}W$ for a pulse width of W . The scaling rule compresses the sinc^2 spectrum by $\frac{1}{2}W$, and also grows its height by $\frac{1}{2}W$. That accounts for the $X(\mathbf{f})$ given here. The factor A comes from linearity, because when we scale *any* $x(\mathbf{t})$ by A , the spectrum $X(\mathbf{f})$ is also scaled by A .

8.3.3 Derived CFT Pairs

From the Table of Basic CFT Pairs, and using the tabulated CFT *properties*, several new pairs are easily obtained. As a first example, we will apply the Duality property $X(-\mathbf{t}) \leftrightarrow x(\mathbf{f})$ to Pair 1 above, while also re-naming the constant τ as a frequency constant \mathbf{f}_1 . The result is Pair 4 in the Table of Derived CFT pairs below (Fig 8.3.3). This describes a familiar situation. A phasor time-function of frequency \mathbf{f}_1 is represented in frequency by a single *impulse* at frequency \mathbf{f}_1 . We used this concept for Fourier Series spectra, except that the impulse was then called a *line*. Both have the same meaning. The *length* of the *line* is the *strength* of the *impulse*, and they are both described in digital terms by a *number*.

Both of the CFT pairs numbered 2 and 3 use time functions $x(\mathbf{t})$ which have *even* symmetry. Thus, $x(-\mathbf{t}) = x(\mathbf{t})$, and the duality property then permits us to interchange the roles of $x(\mathbf{t})$ and $X(\mathbf{f})$ directly. The CFT pairs 5 and 6 then follow at once. Pair number 5 has wide application because the rectangular frequency function can be scaled to describe an ideal low-pass filter. We will see more of this later.

DERIVED CFT PAIRS

	$\{x(t), X(f)\}$ pair	$x(t)$ graph	$X(f)$ graph
4	Phasor to Impulse $e^{j2\pi f_1 t} \stackrel{\text{CFT}}{\leftrightarrow} \delta(f - f_1)$		
5	Sinc to Rectangle $\text{sinc}(t) \stackrel{\text{CFT}}{\leftrightarrow} \Pi(f)$		
6	sinc^2 to Triangle $\text{sinc}^2(t) \stackrel{\text{CFT}}{\leftrightarrow} \Lambda(f)$		

Pair 7 is easily derived from pair 4. We take two instances of pair 4. The first is at frequency \mathbf{f}_1 and is scaled by $\frac{1}{2}\angle\alpha$:

$$\frac{1}{2}e^{j\alpha} \cdot e^{j2\pi\mathbf{f}_1\mathbf{t}} \leftrightarrow \frac{1}{2}e^{j\alpha} \cdot \delta(\mathbf{f} - \mathbf{f}_1)$$

The second is at frequency $-\mathbf{f}_1$ and is scaled by $\frac{1}{2}\angle-\alpha$:

$$\frac{1}{2}e^{-j\alpha} \cdot e^{-j2\pi\mathbf{f}_1\mathbf{t}} \leftrightarrow \frac{1}{2}e^{-j\alpha} \cdot \delta(\mathbf{f} + \mathbf{f}_1)$$

Now we add these two pairs on both sides. The left side becomes a sinusoid of angle α . The right side becomes a pair of impulses at $\pm \mathbf{f}_1$ with strengths of $\frac{1}{2}\angle(\alpha)$ and $\frac{1}{2}\angle(-\alpha)$ respectively. It is identical to the line spectrum of a sinusoid, except that *line lengths* are replaced by *impulse strengths*. Notice, the *scaling* and *addition* used in this example are both dependent on the *linearity* of the CFT.

Pair 8 is a special case of pair 1. We can also arrive at it from a scaled version of pair 2 as follows:

$$\frac{1}{W} \Pi\left(\frac{\mathbf{t}}{W}\right) \leftrightarrow \text{sinc}(W\mathbf{f})$$

The time pulse on the left has width W , height $1/W$, and unit area. As we let $W \rightarrow 0$, this pulse approaches *impulsive* shape, while the sinc on the right spreads itself horizontally to approach a *constant* level of 1.0 in the limit. Pair 8 is the result after

limiting, that is, after $W \rightarrow 0$.

Pair 9 is a special case of pair 4, but it can also be derived from pair 5 in similar manner to that just described. Pairs that involve impulses are "transforms in the limit". They exist "just over the horizon" from the true CFT pairs that use only "regular" functions. But they are important for their ability to link analogue functions to digital sequences.

8.3.4 Special CFT Pairs

The next three pairs, pairs 10, 11 and 12 below (Fig 8.3.4), are listed here because they describe spectral functions that have special importance.

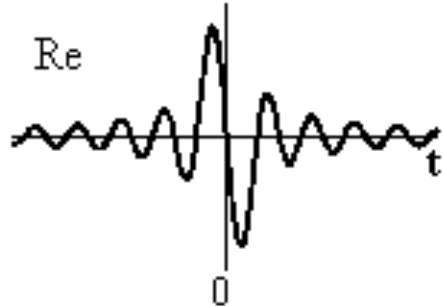
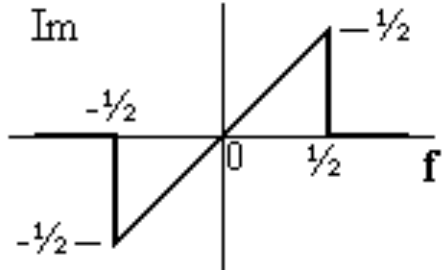
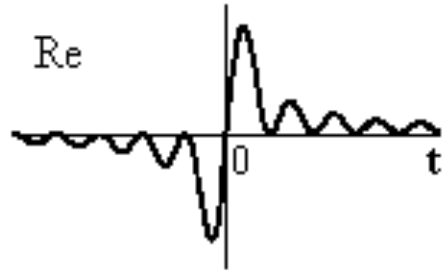
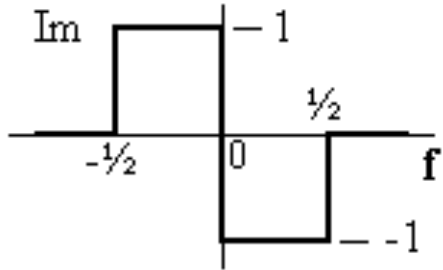
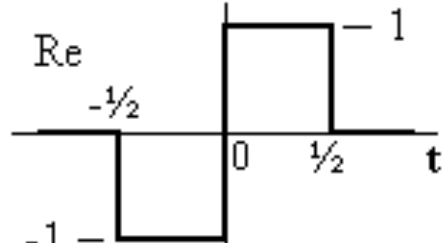
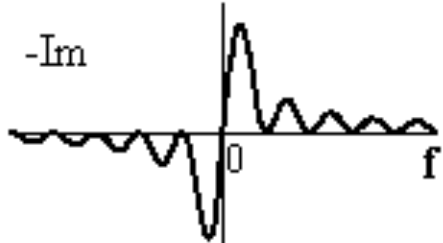
These are different from earlier pairs in that their time functions have *odd* symmetry. This means, in turn, that their spectral functions are *imaginary* and *odd*. The same is true of *any* time function that is real and odd, and is not hard to prove. The underlying reason is that odd time-functions are synthesised from sines, rather than from cosines, and a "-j" term in the spectrum describes the $-\frac{1}{2}\pi$ phase rotation of a sine wave, relative to the zero-angled cosine. Hence the "imaginary" spectral functions found in the Table.

Pair 10 is important because its spectrum mimics an ideal differentiator. After vertical scaling by 2π , it becomes the familiar $j2\pi f = j\omega$, valid up to a limit frequency of $\frac{1}{2}$, but this limit can be adjusted by frequency scaling. Later, this will help us to build *digital* differentiators as well. For verification of this pair, we can apply the inverse CFT to the spectral function $\Gamma(f)$, and the following indefinite integral (in which a can be complex) will be just what we need to complete the task:

$$\int f \cdot e^{af} \cdot df = e^{af} \cdot (af - 1) / a^2$$

Pair 11 is important because its spectrum mimics an ideal phase shifter. All frequency components up to a certain limit are phase-retarded by $\pi/2$, but their amplitudes are not altered. Phase shifters are especially useful in band-pass signal processing, and this result will help us to build *digital* phase shifters too. For verification of this pair, we can apply the inverse CFT to the spectral function $\Gamma_2(\mathbf{f})$, and the solution soon emerges without difficulty.

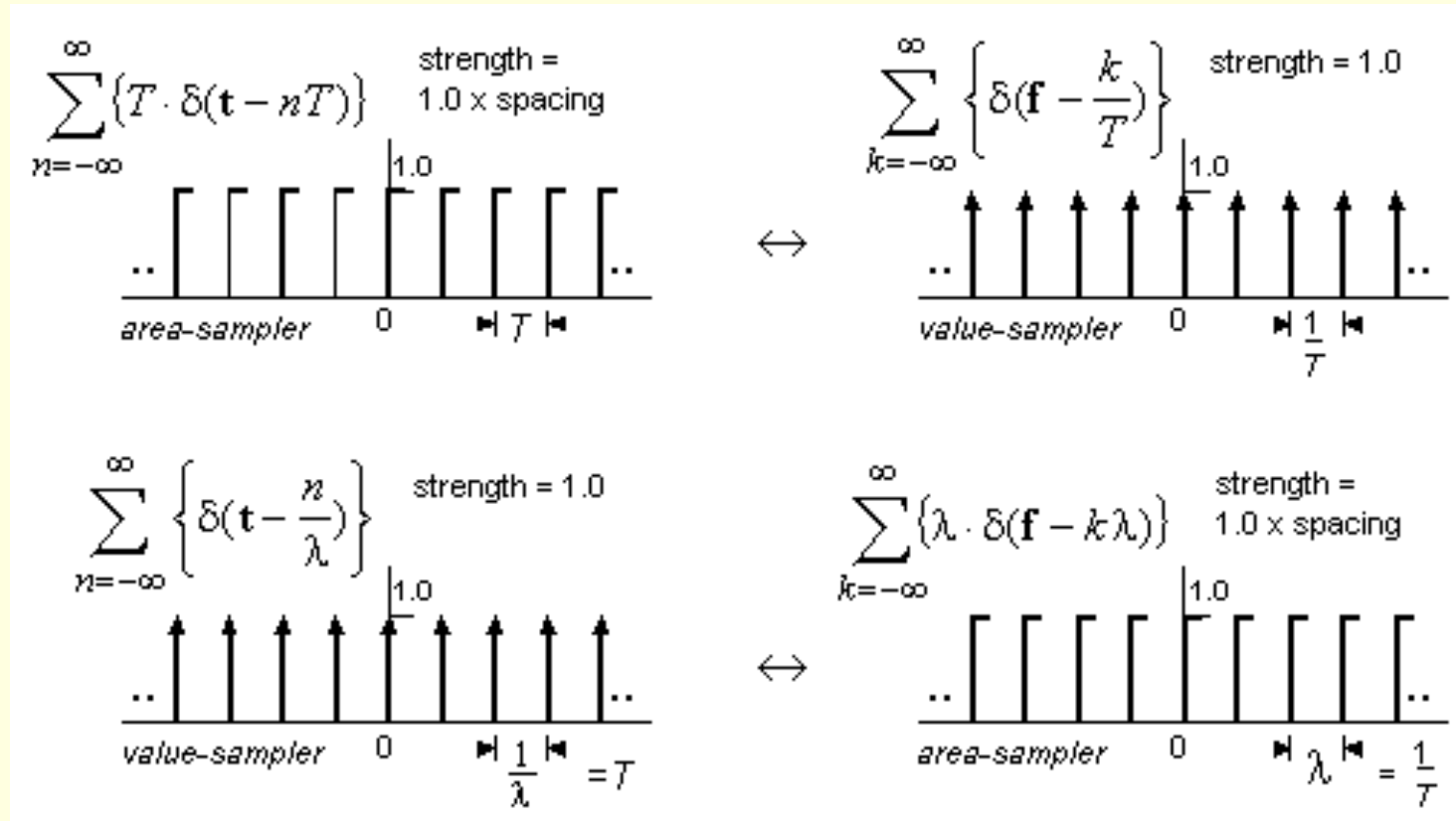
SPECIAL CFT PAIRS

	$\{x(\mathbf{t}), X(\mathbf{f})\}$ pair	$x(\mathbf{t})$ graph	$X(\mathbf{f})$ graph
10	Differentiator $\gamma_1(\mathbf{t}) = \frac{-1}{2\pi^2 \mathbf{t}^2} \left[\frac{\sin(\pi \mathbf{t})}{\pi \mathbf{t}} - \cos(\pi \mathbf{t}) \right]$ $\Gamma_1(\mathbf{f}) = j[\mathbf{f} \cdot \Pi(\mathbf{f})]$	Real Function $\gamma_1(\mathbf{t})$ Re 	Imag Function $\Gamma_1(\mathbf{f})$ Im 
11	Phase Shifter $\gamma_2(\mathbf{t}) = \frac{\sin^2(\frac{1}{2}\pi \mathbf{t})}{(\frac{1}{2}\pi \mathbf{t})}$ $\Gamma_2(\mathbf{f}) = j \cdot \left[\begin{array}{l} \Pi(2(\mathbf{f} + \frac{1}{4})) \\ -\Pi(2(\mathbf{f} - \frac{1}{4})) \end{array} \right]$	Real Function $\gamma_2(\mathbf{t})$ Re 	Imag Function $\Gamma_2(\mathbf{f})$ Im 
12	Bi-phase Pulse $\gamma_3(\mathbf{t}) = \left[\begin{array}{l} \Pi(2(\mathbf{t} - \frac{1}{4})) \\ -\Pi(2(\mathbf{t} + \frac{1}{4})) \end{array} \right]$ $\Gamma_3(\mathbf{f}) = -j \frac{\sin^2(\frac{1}{2}\pi \mathbf{f})}{\mathbf{f}}$	Real Function $\gamma_3(\mathbf{t})$ Re 	Imag Function $\Gamma_3(\mathbf{f})$ -Im 

Pair 12 is the dual of pair 11, and is obtained from pair 11 using CFT property 4. The time pulse, in this case, is in the shape of the bi-phase pulse that is used by Manchester line code. We'll see later how the square of its spectral pulse, $\Gamma_3(\mathbf{f})$, is also the power spectral density for Manchester-coded data streams.

8.3.5 Railings Transform Pairs

We arrived at the Railings Pairs (Fig 6) before our current work with the CFT (see [Ch 4.3.1](#)), but we can also regard them as CFT pairs in the limit.



They are neatly summarised by the statements:

- an area-sampler transforms to a value sampler
- a value-sampler transforms to an area-sampler

As previously noted, *multiplication* of a signal $x(t)$ by Railings serves to *sample* the signal, yielding an impulse sequence with impulse strengths $x(nT)$ for a value-sampler, or $T \cdot x(nT)$ for an area-sampler.

We also saw (in [Ch 8.2.3](#)) how *convolution* of a signal with an impulse $\delta(t)$ amounted to a scanning action which reproduced the signal exactly. Because Railings are a set of equally-spaced impulses, convolution of a signal with railings serves to *replicate* the signal, one replica for each impulse in the train. The replicas have the same spacing as the impulses, and this spacing becomes the period of the resulting periodic waveform. If we convolve a signal with a value-sampler, that signal is replicated, but its amplitude is not scaled.

When we use Railings with the Convolution Properties, 7 and 8 as tabulated, they assume added significance. We now see that *multiplication in one domain by an area-sampler is equivalent to convolution in the other domain with a value-sampler*. The result on one side is a set of area-samples. The result on the other side is replication without scaling. The outcome is a result that we already know, namely, that area-sampling in one domain equates to replication in the other domain. There's nothing new here, but the idea of Railings may help to re-inforce our pictorial vision of sampling and replication.





8.4 THE DtFT AND ITS PROPERTIES

Using Railings and the Convolution property, if the $x(\mathbf{t})$ of a CFT pair $\{x(\mathbf{t}), X(\mathbf{f})\}$ is *multiplied* by an area–sampler, with sample interval T , then $X(\mathbf{f})$ is *convolved* with a value–sampler to form a replicated $X(\mathbf{f})\sim$ of period $1/T$. The resulting DtFT pair becomes:

$$x(\mathbf{t}) \xrightarrow{\text{DtFT}} X(\mathbf{f})\sim \quad \text{or} \quad x[n] \xrightarrow{\text{DtFT}} X(f), \quad f = \frac{\mathbf{f}}{\mathbf{f}_s} = \mathbf{f}T$$

The $x[n]$ are area–samples, $= T \cdot x(nT)$. $X(f)$ and $X(\mathbf{f})\sim$ give identical function values, they just express the frequency differently. Through this process, all of our CFT pairs generate corresponding DtFT pairs. (We will use this later as a way of designing FIR digital filters). The DtFT is our transform for data sequences, and we will now look at some of its more important attributes.

8.4.1 Selected DtFT Properties

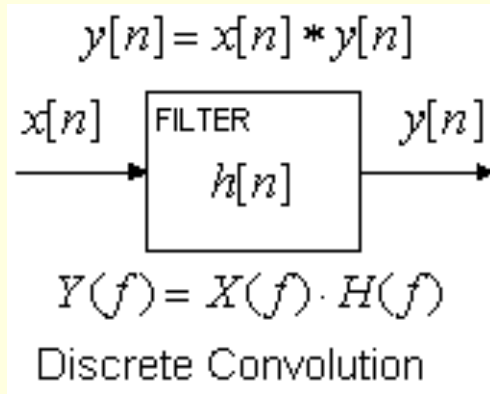
Some of the tabulated CFT properties (i [Ch 8.3.1](#)) are repeated here in DtFT form, and with corresponding CFT/DtFT reference numbers #, (Fig 8.1).

Time reversal, property 2, is as before. Time delay, property 5, is counted in samples (rather than seconds), and gives a linear phase lag in terms of f , for an m -sample time delay. Frequency shift, property 6, describes a phasor-modulation of the time sequence, and it shifts the spectrum by a normalised amount, f_0 . We can use this to see the effects of sinusoidal modulation also.

SOME PROPERTIES OF THE DtFT

#	Property	Time Domain	Frequency Domain
2	Time Reversal	$x[-n]$	$X(-f)$
5	Time Shift	$x[n-m]$	$X(f) \cdot e^{-j2\pi fm}$
6	Frequency Shift	$x[n] \cdot e^{j2\pi f_0 n}$	$X(f - f_0)$
7	Discrete Convolution	$x_1[n] * x_2[n]$	$X_1(f) \cdot X_2(f)$

The most important is property 7, discrete convolution. When we apply this result to a filter, where the input sequence $x[n]$ is convolved with the impulse response $h[n]$, it gives us an understanding of the filter (Fig 8.2) through the spectral relationship:



$$Y(f) = X(f) \cdot H(f)$$

The output–signal spectrum is the input–signal spectrum times the filter–spectrum. For most purposes, the action of the filter is best seen from this viewpoint. It is much more informative than the convolution sum. It is also much easier to compute. That's because we can get N values of $X(f)$ using only N complex multiplications, whereas the convolution that gives us N values of $y[n]$ requires NL multiplications, where L is the filter length.



Our set of CFT and DFT properties, and our collection of CFT pairs will prove very useful in the chapters that follow.





9.1 PREAMBLE

Much of our DSP work is about the processing of *analogue* signals, with audio and video signals as prime examples. The DSP work must be *preceded* by A/D conversion of analogue input signals, and it must be *followed* by D/A conversion to form the analogue output signals. We will cover some of the issues raised by these operations, and by the inherent *windowing* of the signals that we work with.



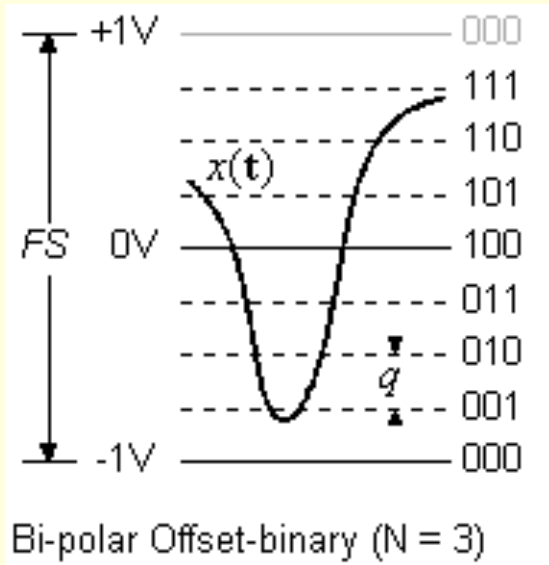


9.2 ANALOGUE TO DIGITAL (A/D) CONVERSION

The sections deals with A/D conversion from a signals perspective. We are not concerned with the technology of A/D conversion, only with the effects on the signal itself, and with particular emphasis on signal integrity.

9.2.1 Sampling Overview

An A/D converter has an analogue signal $x(t)$ as its input, and the output is a succession of sample values $x(nT)$, at regular sample intervals of T seconds, in the form of binary-coded data words. The arrangement shown here (Fig 9.2) is fairly typical. It describes a bi-polar analogue input signal in a Full-Scale (FS) range of 2 volts, with ± 1 volt maxima. The output is coded in an *offset-binary* format, which counts up directly from 000 to 111. We've used a 3-bit word ($N = 3$) for illustration, but word-lengths of 12 to 16 bits are typical. For an N -bit word, we get 2^N levels, with a *quantization* step-size of $q = FS/2^N$ volts between levels. The step-size q is also the *weighting* associated with the right-most bit of the word, better known as the LSB, or least-significant bit. The usable range goes from $-1V$ at the bottom to $(+1-LSB)$ volts at the top. The left-most bit is the MSB, or most-significant bit, with a weighting of $2^{(N-1)} \cdot q$ volts. Notice, the MSB is 0 for negative values of $x(t)$, and is otherwise equal to 1. To convert this offset-binary scale to a 2s-complement scale, which is popular in computer arithmetic, we only have to toggle the MSB.



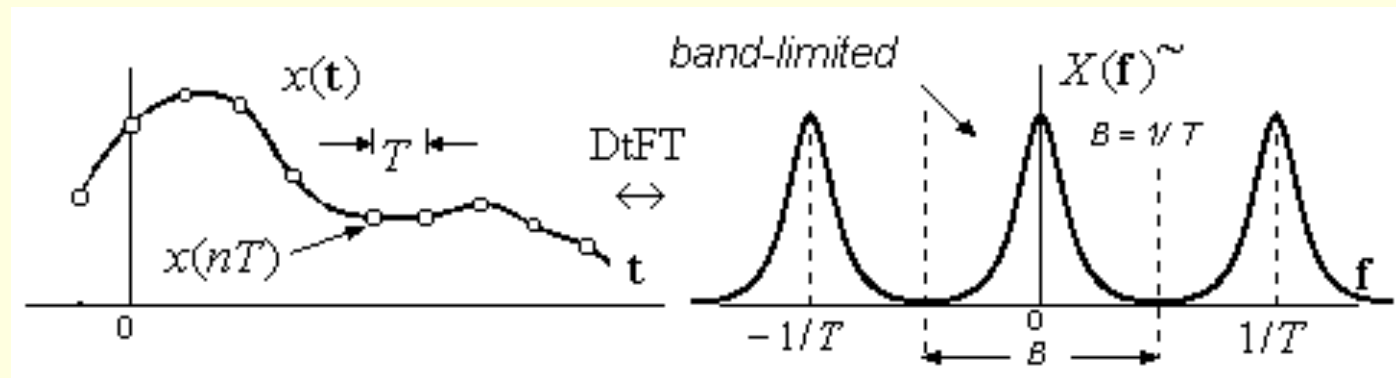
When $x(t)$ is between levels, there is a *quantization error* that separates the nearest digital level from the true $x(t)$ value. This error varies from sample to sample, with error limits of $\pm 1/2q$ volts (provided switching thresholds are set to give a *rounding effect*). We'll return to quantization a little later.

A practical concern with A/D converters is that the signal value is *changing* while the conversion is in progress. To get digital values that accurately represent $x(t)$ at a given instant, we need to *sample* $x(t)$ at time nT , and then *hold* the value $x(nT)$ in analogue form until the conversion is completed. This is accomplished by use of a sample-and-hold (or s/h) amplifier. If the signal is liable to change by more than $1/2$ LSB between samples, then an s/h amplifier may be needed. Most signals change much more rapidly than this, and so the s/h amplifier would normally be included. It has the important effect that, except for the quantization error, we can often treat the A/D conversion stage as an ideal one, that is, as an accurate generator of the desired $x(nT)$ values.

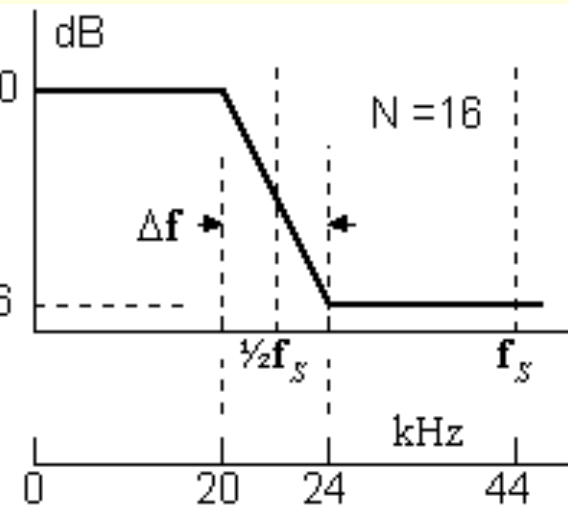
A/D converters differ mainly in their maximum sampling rate, and in their digital data-word length. Longer words mean smaller LSB values, which calls for higher precision in the analogue circuitry. Speed and precision will usually combine to decide the cost of the converter.

9.2.2 Specifying Anti-alias Filters

We're already well aware of the spectral replication that results from sampling a signal, and of the necessity that the signal be band-limited to $\pm f_s/2$ Hz, where $f_s = 1/T$, with sample interval T (Fig 6). This ensures that adjacent spectral replicas do not overlap, or that spectral *aliasing* is avoided.



A low-pass analogue filter, known as an anti-alias filter, is used to band-limit the analogue signal before A/D conversion. It's an unwelcome chore, but a necessary one. In practice, the filter must have a finite transition width Δf between the pass-band and the stop-band. This LP filter (Fig 7) has a transition band that is centered on $\frac{1}{2}f_s$. The figures underneath (not drawn to scale) are for a digital audio signal which is sampled at 44 kHz. (Compact Disk uses 44.1 kHz sampling). The transition band is centered on $\frac{1}{2}f_s$ which is 22 kHz, and has a width $\Delta f = 4$ kHz. The pass-band extends up to 20 kHz, which is considered satisfactory for high fidelity. Ideally, there should be nothing above 20 kHz. In practice, tones in the transition band are not sufficiently suppressed. A tone at 23 kHz will have an image at $(-23+44) = 21$ kHz, but this should be inaudible. A tone just over 24 kHz will have an image just below 20 kHz, but this tone is in the stop-band of the analogue filter, which makes it small enough to ignore. By centering the transition band at $\frac{1}{2}f_s$, we've ensured that any tones in the transition band, which are insufficiently suppressed, will not intrude into the passband.

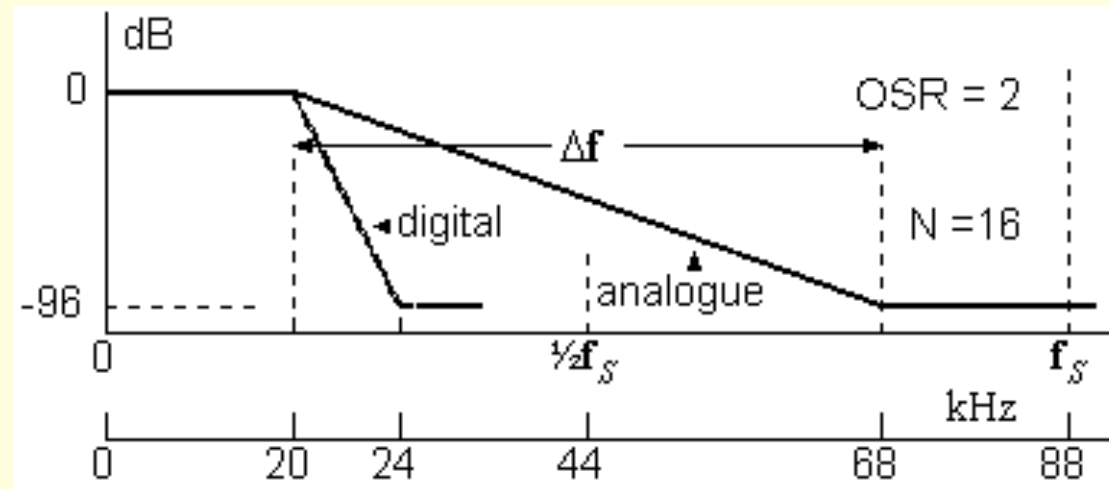


Other filter specifications include *pass-band ripple* and *stop-band rejection*. Pass-band ripple describes a small gain variation over frequency around a nominal value of 1.0, which is 0 dB. A ripple of 0.1 dB gives about 1% gain variation, which may not have a noticeable affect. We can give a figure for stop-band rejection as the dB attenuation needed to suppress a full-scale signal as far as the LSB level. For an N -bit word-length, this is $20 \cdot \log_{10}(2^{-N})$ in decibels, and for a 16-bit word-length, it becomes -96 dB, as the diagram suggests (Fig 9.2.3). This is a worst-case estimate, and some lesser attenuation may suffice.

A small transition width Δf requires a more elaborate analogue filter, but a wider Δf would force a higher sampling rate f_s , and thus increase the data storage requirement. The choice of $\Delta f = 4$ kHz in our example, combined with over 90 dB of stop-band rejection, demands a fairly complex anti-alias filter. As part of a general trend toward digital circuitry, and away from analogue circuitry, we have a strong motivation to simplify (or even eliminate) this anti-alias filtering requirement. We will now see how this might be accomplished.

9.2.3 The Over-sampling Option

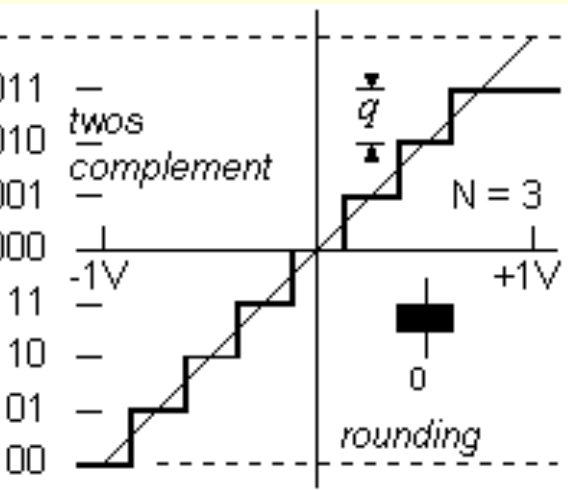
Over-sampling means sampling faster than is necessary. If we double the sampling rate for our audio signal, up from 44 kHz to 88 kHz, we will have an OSR (an Over-Sampling Ratio) of 2. Using the same method as before, the new specification for the anti-alias filter is as shown (Fig 6).



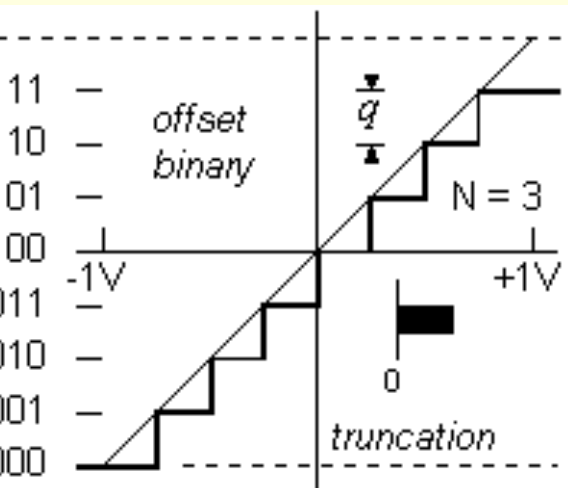
We now have a transition band $\Delta f = 48$ kHz in width, up from 4 kHz originally. This will vastly simplify the analogue anti-alias filter, but it also means processing the digital data at *twice* the rate that is needed. We don't have to do that. We can actually *halve* the data rate again by discarding every second sample of our digital data stream, but there is a condition. Before we discard any samples, we must *digitally* filter this data to the original specification, for a stop-band that commences at 24 kHz as shown on the diagram. The overall result is that a difficult *analogue* filtering job has become mostly a *digital* filtering job. But, this is generally quite acceptable, because the digital filtering can usually be done at little additional cost.

This OSR of 2.0 is only the beginning. Since the digital circuitry of today runs at speeds that are vastly higher than audio sampling rates, oversampling ratios of several hundred are realizable. These high ratios have become commonplace in "Delta-Sigma"-type A/D converters, and with them the analogue anti-alias filter problem is all but eliminated.

9.2.4 Quantization Issues



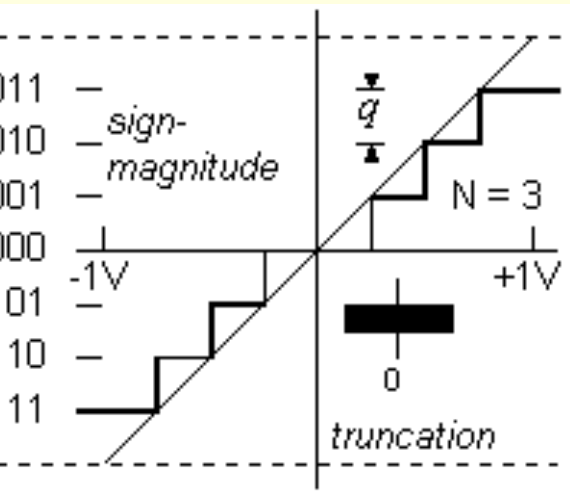
This diagram shows signal quantization (Fig 9.2.4) as a plot of binary codes versus analogue input voltage, in a range from $-1V$ to $+1V$. We've shown a two's complement code with a 3-bit word-length ($N = 3$). The 45° line shows true analogue levels, for comparison with the stepped code levels, such that the difference between them is the quantization error. This is an illustration of quantization by *rounding*, which always seeks to minimize the error magnitude. For a step-size q , it gives a quantization error range of $\pm \frac{1}{2}q$. This error range is also indicated by the small black rectangle symbol on the diagram.



Another option is to quantize by *truncation*. That means always using the nearest code level that is *smaller* than the signal level, as this diagram illustrates (Fig 9.2.5). In this case, the quantization error is in a range from 0 to q , and this range is indicated by the small black rectangle symbol. We could have used a two's complement code here too, but we've shown an offset binary code instead. The only difference is in the toggling of the MSB.

In some instrument applications, we could find sign-magnitude coding, where the MSB is a sign bit (0 for plus and 1 for minus), and the remaining bits count up the *magnitude* from zero. That would alter the meaning of truncation, as this diagram illustrates (Fig 9.2.6). Here, the truncation is in the direction of the horizontal 000 line. The error now ranges from $-q$ to $+q$, as the small black symbol indicates.

Quantization errors are constrained to the ranges that we indicated and, within those ranges, their sample-to-sample variations are mostly random. Later on, we'll use this information to characterize the noise in terms of its *mean noise power*.



One thing we can say right away : if the word length is reduced by one bit, then the step-size q is doubled, the quantization noise level is doubled, and the mean noise power is quadrupled. In decibels, that's a 6 dB increase in noise-power for a 1-bit reduction in word length.

We can use simulation methods and a random number generator to investigate quantization effects in a system. For that purpose, we also need functions that can quantize a signal in the ways that we have illustrated. Such functions are easily defined in Mathcad and in MATLAB.





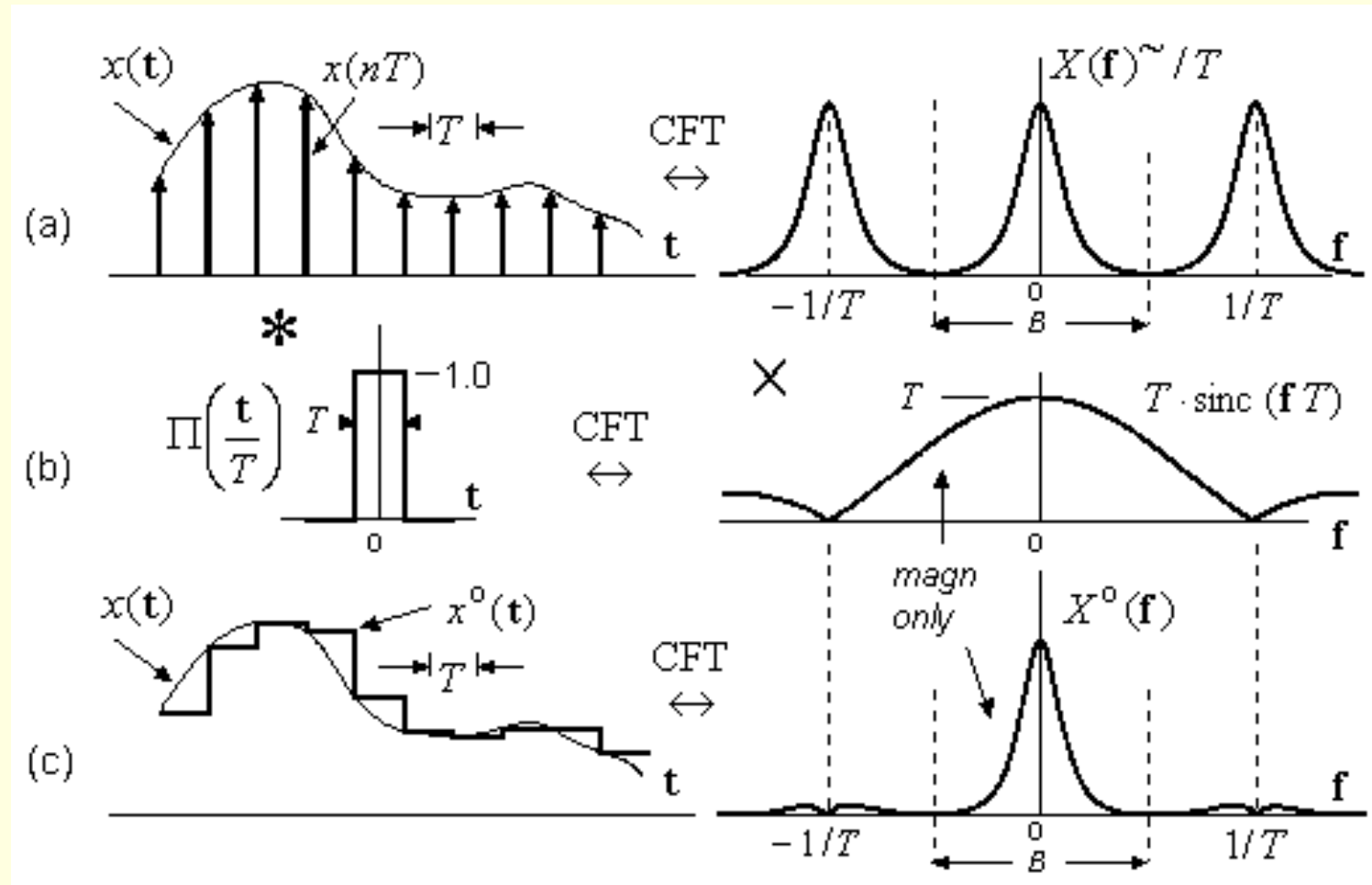
9.3 DIGITAL TO ANALOGUE (D/A) CONVERSION

The section deals with D/A conversion from a signals perspective. We are not concerned with the technology of D/A conversion, only with the effects on the signal itself, with emphasis on the faithful re-construction of a band-limited signal $x(t)$ from its samples $x(nT)$.

9.3.1 A Re-construction Model

We will now describe the conversion of a signal from a number-sequence in a computer (or DSP chip) into a continuous analogue output signal, with the help of a D/A converter. Part (a) below (Fig 9.1) depicts the signal as $x(nT)$, the sampled numeric form, as it exists before D/A conversion. The numbers are represented as impulses, and the impulse strengths are the numeric data values. On the right, we see the band-limited periodic spectrum of this data sequence. Because we consider the numbers to be value-samples of $x(t)$, the correct spectral description is the scaled $X(f) \sim T$, rather than just $X(f)$.

A D/A converter will convert samples sequentially, and will *hold* the current sample value until the next sample comes along. Stated otherwise, it *interpolates* between samples using a *zero-order hold* (or ZOH) operation, and the result is the stepped waveform $x^o(t)$ in part (c) of the diagram. This falls far short of the ideal sinc interpolation that would restore the true $x(t)$. It's not practical to attempt a sinc interpolation, but we can bring $x^o(t)$ much closer to $x(t)$ by doing a little analogue filtering of the D/A converter output. Before that, it will help if we understand in spectral terms what the zero-order hold does to the signal.

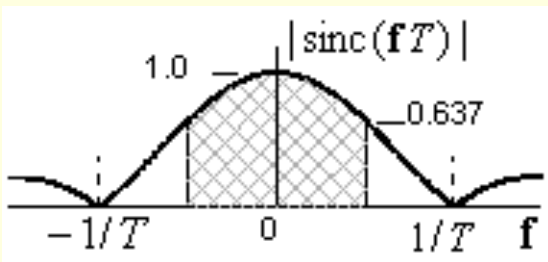


That is the purpose of part (b) in the diagram. It shows a pulse $\Pi(t/T)$ of width T and height 1.0. If we *convolve* this pulse with an impulse of strength $x(nT)$, we get a copy of the pulse adjusted to a new height of $x(nT)$. This action converts an impulse to a zero-order hold value, and holds it for T seconds. Consequently, convolution of the impulses $x(nT)$ in part (a) with $\Pi(t/T)$ in part (b) will generate the DAC output signal $x^o(t)$ of part (c). This will now enable us to see how the spectrum is altered by D/A conversion.

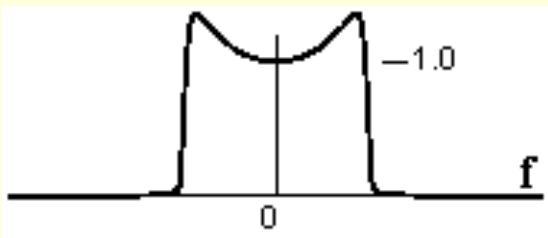
Using the scaling property of the CFT, the spectrum of $\Pi(t/T)$ becomes $T \cdot \text{sinc}(fT)$. The right side of (b) shows this as a spectral magnitude plot, as $|T \cdot \text{sinc}(fT)|$. We know that convolution in time equates to spectral multiplication, and therefore the spectrum of $x^o(t)$ must be :

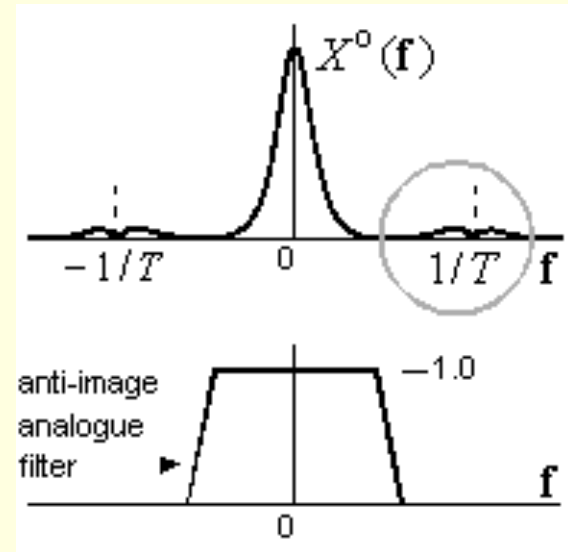
$$X^o(f) = \left(X(f) \sim / T \right) (T \cdot \text{sinc}(fT)) = X(f) \sim \cdot \text{sinc}(fT)$$

This ignores a small $T/2$ time delay which causality implies, but which has little effect. Because the output $x^o(t)$ is an *analogue* signal, its spectrum $X^o(f)$ is *not* periodic. Multiplication by the sinc has accomplished this. The "ideal" $X^o(f)$ would be $X(f)$, the CFT of $x(t)$, which is also an exact copy of the base-band period of $X(f) \sim$, with all of its images reduced to zero. (That is what we would get from a true sinc-t interpolation). The reality is the $X^o(f)$ of part (c), with significant shortcomings, as follows.



The shaded region here (Fig ç) is the sinc-baseband area, and it shows clearly how the higher baseband frequencies are attenuated by the ZOH action. At the top of the signal band, the reduction is by $\text{sinc}(1/2) = 0.637$, that's a 36 % reduction at the high end. In practice, this is easily remedied *before* D/A conversion, by use of a digital filter that *amplifies* the higher frequencies, in anticipation of this effect. The filter spectrum would look something like this (Fig ç). Analogue output devices, including audio-CD players, use this kind of compensation.





The other problem with $X^o(f)$ is the inadequate suppression of image frequencies, as shown in the encircled area here (Fig 1). This can be remedied by analogue post-filtering, after D/A conversion. We've also shown the outline of an analogue *anti-image filter* that would suppress the image but leave the base-band undisturbed. It requires a narrow transition band, which could result in a moderately complex analogue filtering task that we would very much like to avoid. In fact, we can again turn to over-sampling methods to turn this analogue task into a digital one.

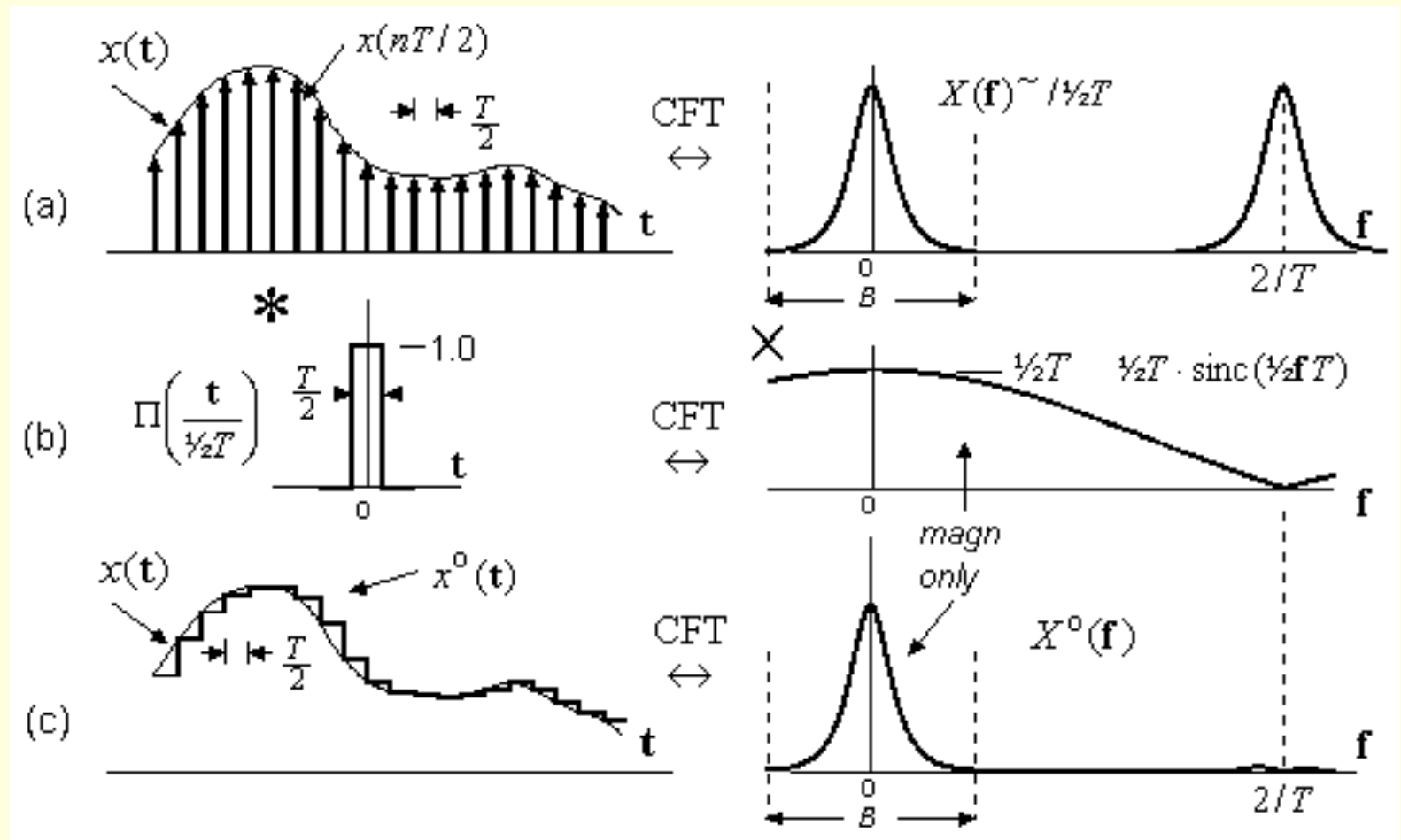
9.3.2 The Over-sampling Option

Quite often, the digital information is available at the normal rate, it is not oversampled. As an example, the audio data on a CD is at 44.1 kHz (because higher rates would take up too much space). So how can we over-sample ?

The answer is to *interpolate* the available data, and we've already seen how to do this using the DFT (i [Ch 6.3.4](#)), but we can also do it by other methods. To increase the sample-rate by $U = 4$, we would first insert 3 zeros between adjacent sample pairs, and then send this 4-times expanded sequence through a low-pass digital filter. The filter output is the 4-times oversampled signal that we require. We'll describe this technique in much greater detail later. For now, it's sufficient to know that we can easily increase the sample rate by an integer ratio of our choosing, and then send the up-sampled data to the D/A

converter. We'll pick up the story at that point, and we'll see how the higher rate has a beneficial effect on the output.

Our illustration is for oversampling by a modest $U = 2$, but even here the benefits of oversampling are quickly apparent (Fig 6).



We've used spectral symmetry to save space by showing only a little of the negative-frequency part. The new sample interval is $T/2$, but the base-band width is unchanged at a (double-sided) value of $B = 1/T$, as shown on the diagram. The images are moved out to $2/T$ and beyond, and this simplifies the post-filtering requirement in two ways. First, the anti-image analogue filter can have a very wide transition band. Second, the image in $X^o(\mathbf{f})$ at $2/T$ is already smaller than before, and this too relaxes the filter specification. We can see the improvement more intuitively in the time-domain plot (c), where the stepped output signal is already closer than before to the $x(\mathbf{t})$ that we want as our end-product.

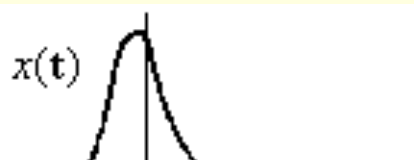
Higher values of U will give further improvement, with less need for analogue anti-image filtering, as for example in a CD player labelled "8 \times oversampling". Nowadays, far higher values of U , coupled with Delta-Sigma noise shaping, give us a solution that is almost totally digital. Delta-Sigma methods are another topic that we will take up in a later chapter.





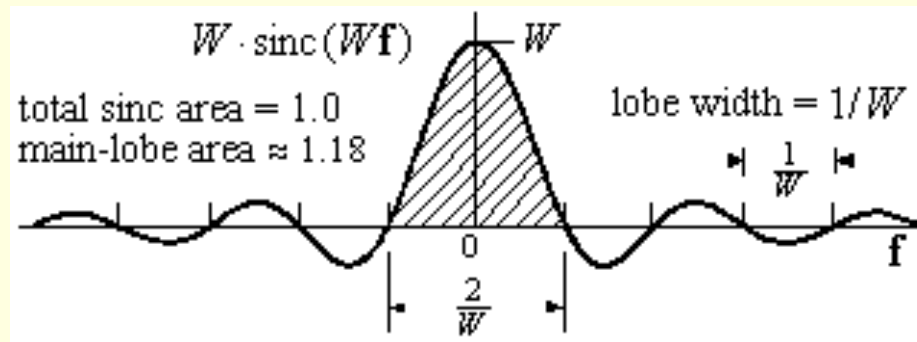
9.4 WINDOWING AND ITS EFFECTS

If we multiply a signal $x(t)$ by $\Pi(t/W)$, a unit-height rectangular pulse of width W seconds, we retain that part of $x(t)$ in the range $(-1/2W < t < 1/2W)$, and we set the rest to zero (Fig 1). It's equivalent to using a copy of $x(t)$ that ignores everything outside the "window" formed by $\Pi(t/W)$. In DSP we call it "windowing", it is something that we do frequently, and for various reasons. It may be because the available $x(t)$ data is less than complete, or because we can speed the work by neglecting the far-out parts that tail off gradually to zero. In some cases, the $x(t)$ is a *theoretical* shape that goes on forever (such as the sinc), but when we use it in computations we must limit the work-load by ignoring its outer regions. Whatever the reason for windowing, it amounts to an obvious distortion of a signal in time domain, whereas it is less obvious but may still be quite serious in the spectral domain. We need to understand the implications of windowing, and we must often take corrective action as well.



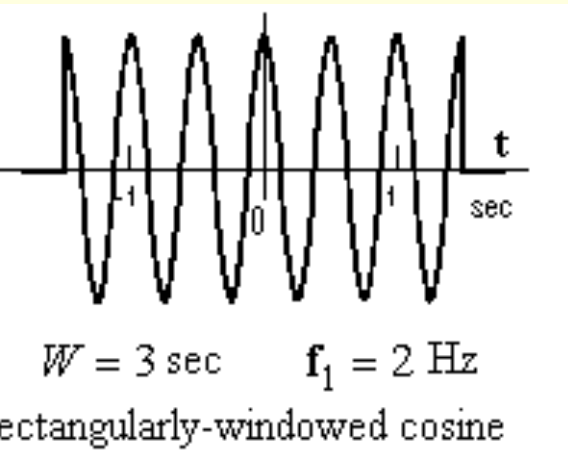
9.4.1 Rectangular Windowing

Windowing with $\Pi(t/W)$ (pronounced "rect of t/W ") is called Rectangular windowing. In practice, we would use $\Pi((t - t_0)/W)$ for a window that is centered on t_0 , so that we can place it where we please. But, to *understand* windowing, we'll find it easier to avoid the linear phase terms caused by time-shifting, so we'll continue to use $\Pi(t/W)$ for its real and even spectrum, which is $W \cdot \text{sinc}(Wf)$. There is no phase term to worry about, and it looks like this (Fig 6).



Because the "window" is a $\Pi(t)$ that's been *stretched* by W , then, by the CFT scaling property, this spectrum is a $\text{sinc}(f)$ that has been *compressed*, and made *taller*, by the same factor W . The shaded part known as the "main-lobe" is of height W , and of width $2/W$, that is two lobes wide. The main-lobe area is slightly over 1.0 and, if we let $W \rightarrow \infty$, it approaches true impulsive shape. To the left and right of the main-lobe we have "side-lobes" that oscillate and decay, inside a $1/f$ envelope extending all the way to infinity.

We'll use a windowed cosine to see what windowing does to a spectrum. A cosine goes on forever, and the spectrum of the cosine signal $\cos(2\pi f_1 t)$ is a pair of impulses of strength $1/2$ located at $\pm f_1$, as seen below (Fig 6). The windowed cosine is $x(t) = \cos(2\pi f_1 t) \cdot \Pi(t/W)$, and is shown here (Fig 7) for $f_1 = 2$ and $W = 3$, that is, for a 2 Hz cosine in a 3-second window. The effect of windowing on the cosine spectrum is to *convolve* it with the window spectrum, yielding:

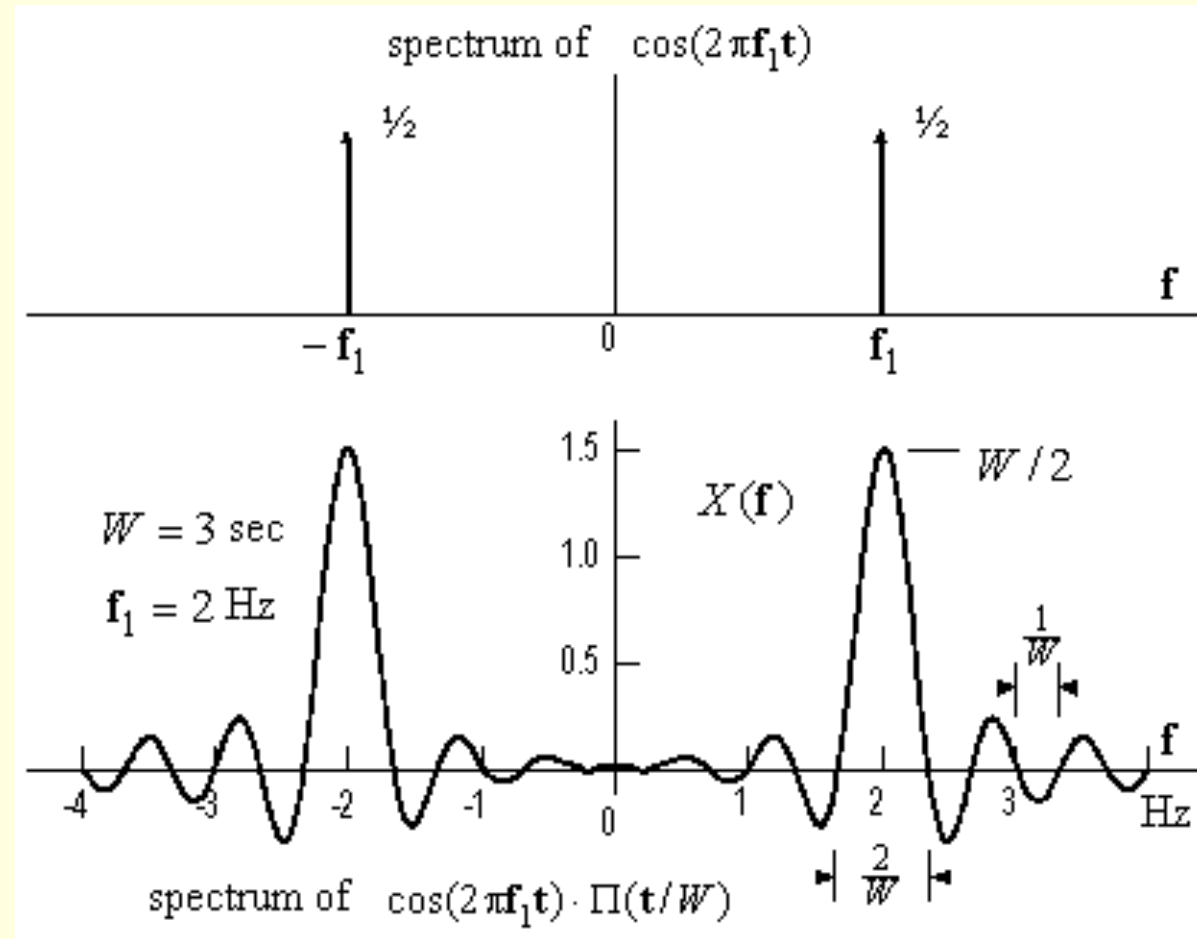


$$X(\mathbf{f}) = \left[\frac{1}{2} \delta(\mathbf{f} + \mathbf{f}_1) + \frac{1}{2} \delta(\mathbf{f} - \mathbf{f}_1) \right] * \left[W \cdot \text{sinc}(W\mathbf{f}) \right]$$

This generates two copies of the sinc, one at $-\mathbf{f}_1$, another at $+\mathbf{f}_1$, and both scaled by the impulse strength of $\frac{1}{2}$. The spectrum becomes:

$$X(\mathbf{f}) = \frac{1}{2} W \cdot \text{sinc}(W(\mathbf{f} + \mathbf{f}_1)) + \frac{1}{2} W \cdot \text{sinc}(W(\mathbf{f} - \mathbf{f}_1))$$

This is the $X(\mathbf{f})$ in the diagram (Fig 6), and it replaces the impulsive spectrum of an un-windowed cosine. It shows tall narrow pulses at the ± 2 Hz phasor frequencies, and a lot of smaller oscillations elsewhere.

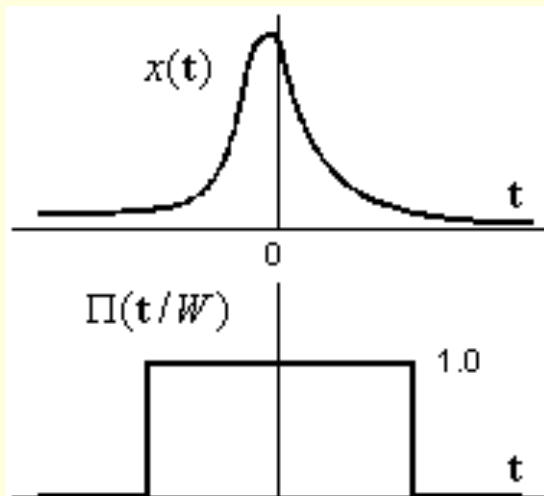


The new $X(f)$ tries to approximate the pair of impulses, and it does a better job as the window is made wider. As $W \rightarrow \infty$, the two main-lobes approach impulse shape, each with a strength close to $\frac{1}{2}$, and the oscillations decay more rapidly as we move away from the main-lobe frequencies.

The shortcomings of $X(f)$ are two-fold. The finite main-lobe width limits the *spectral resolution* to around $\Delta f = 2/W$ Hz. It means that on a spectrum of two cosines that are closer in frequency than this Δf , we will find it difficult to tell them apart.

Spectral resolution is inversely proportional to window width W , and we can always make it better by using a wider window, that means using more data, provided the data is available.

The second problem is the *side-lobes*. They suggest that we have signal content at frequencies where none exists, even at frequencies far removed from f_1 . In a signal that has *two* cosine frequencies, a strong tone and a weak tone, the side-lobes from the strong tone could swamp the weak tone altogether, such that we fail to recognise it. The side-lobe content is called *spectral leakage*. It is important that we minimise leakage, but reducing leakage is less easy than improving the spectral resolution. Intuitively, leakage describes all the frequencies that are needed to build the sudden steps that occur at the window edges (Fig 9.4.2). This in turn gives us a clue as to how the leakage can be reduced.

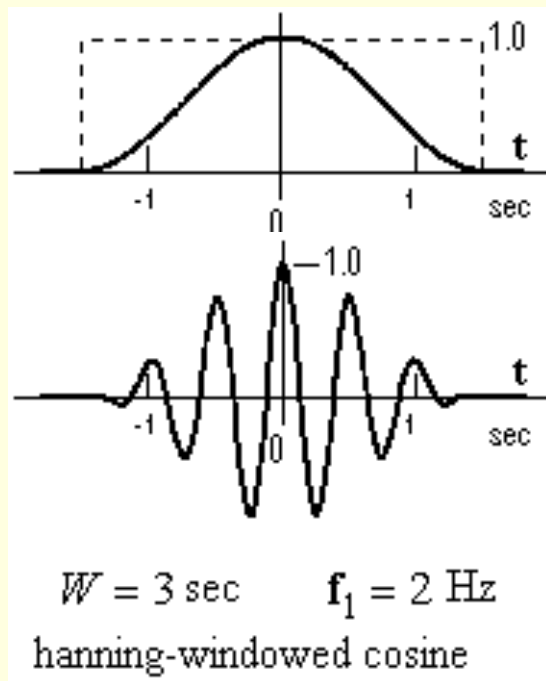


9.4.2 The Hanning Tapered Window

The basic approach to reducing leakage is to use a *tapered* window, such as this one (Fig 9.4.2), in place of the (broken-line) rectangular window. The taper eliminates the sharp edges that make leakage so severe, and we can expect a greatly-reduced

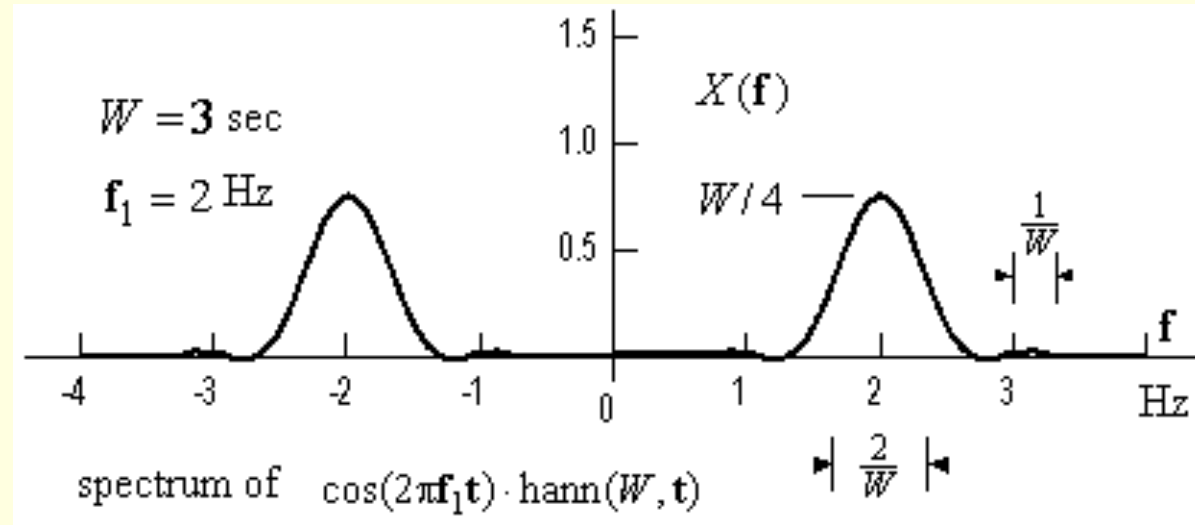
leakage effect in return. The price paid is that a tapered window distorts a signal, even *inside* the window region, but this distortion is often less serious in the spectral view than the distortion by rectangular windowing. We will use the same cosine signal, for a comparison of the spectral effects.

The tapered window in the diagram (Fig ç) is called a Hanning window, or a raised-cosine window, and it follows this description:



$$\text{hann}(W, t) = \left[\frac{1}{2} + \frac{1}{2} \cos\left(2\pi \frac{1}{W} t\right) \right] \cdot \Pi(t/W)$$

It's just one period of a cosine, of peak value $\frac{1}{2}$, and also raised by $\frac{1}{2}$. When we window our cosine signal in this way, the result is as shown (Fig ç), and the spectrum of this tapered cosine looks like this (Fig ê).



For ease of comparison, we've retained the scaling of the rectangular-windowed spectrum. Here, the main lobes are *half as tall and twice as wide*, with a main-lobe area that is again close to $\frac{1}{2}$. The greater main-lobe width *halves the spectral resolution*, but we can usually compensate for this unwelcome effect by using a wider time window. More importantly, the side lobes have been greatly reduced (in confirmation of our intuitive reasoning). The major benefit of a tapered window is this reduced spectral leakage.

9.4.3 Hanning versus Hamming

The Hanning window is just one of many tapered windows that we can choose from. Although the windowed cosine was a useful example, we can evaluate a window in isolation, with no particular signal in mind. A window $w(\mathbf{t})$ has a spectrum $W(\mathbf{f})$, and we can evaluate the window by inspection of its $W(\mathbf{f})$.

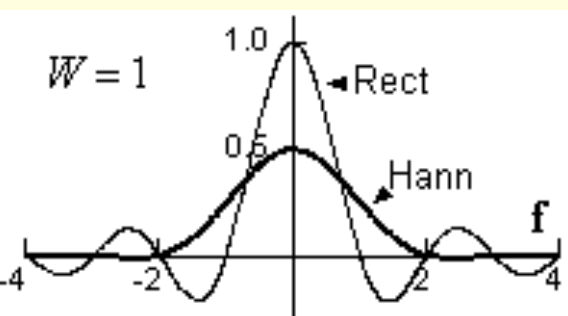
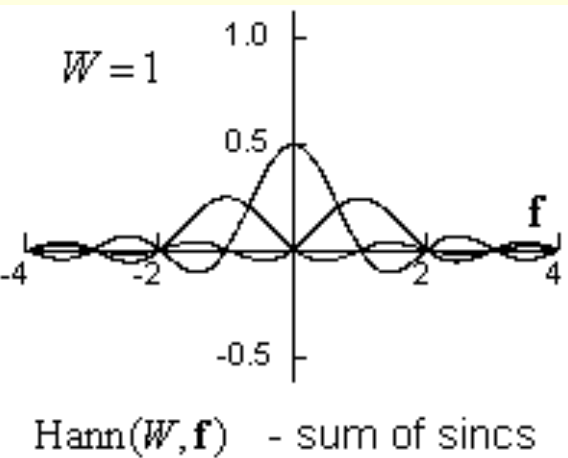
When $w(\mathbf{t}) = \text{hann}(W, \mathbf{t})$ as given above, we can use the CFT pairs numbered 7 and 9 ([Ch 8.3.3](#)) to see that it's spectrum is:

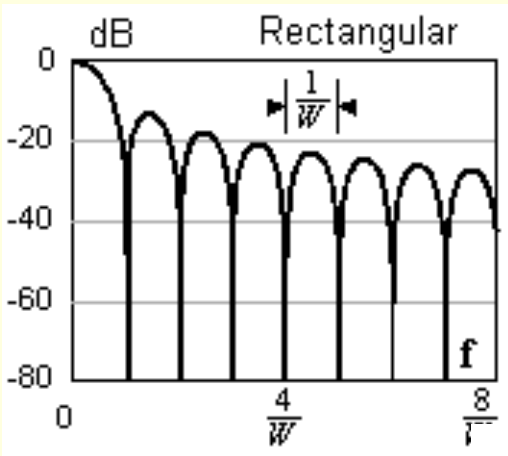
$$\text{Hann}(W, \mathbf{f}) = \left\{ \frac{1}{2} \delta(\mathbf{f}) + \frac{1}{4} \delta\left(\mathbf{f} + \frac{1}{W}\right) + \frac{1}{4} \delta\left(\mathbf{f} - \frac{1}{W}\right) \right\} * [W \cdot \text{sinc}(W\mathbf{f})]$$

This is a convolution of a sinc with three impulses, and the result is a sum of three scaled and shifted copies of the sinc. It becomes:

$$\text{Hann}(W, \mathbf{f}) = \frac{1}{2} W \cdot \text{sinc}(W\mathbf{f}) + \frac{1}{4} W \cdot \text{sinc}\left\{W\left(\mathbf{f} + \frac{1}{W}\right)\right\} + \frac{1}{4} W \cdot \text{sinc}\left\{W\left(\mathbf{f} - \frac{1}{W}\right)\right\}$$

Using a nominal width of $W = 1$, this spectral diagram shows the three sincs individually (Fig 9.4.3c). The next diagram shows the sum of these sincs, which is the Hanning window spectrum (Fig 9.4.3d). The Hanning has low leakage because the sincs add destructively in side-lobe regions. We've also shown the rectangular window spectrum for comparison. The Hanning main-lobe is twice as wide, but its side-lobes are very much smaller.



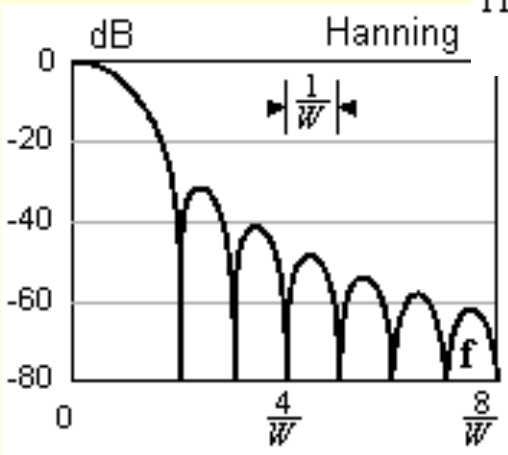


The Hamming window is just a small variation on the Hanning:

$$\text{hamm} (W, t) = \left[0.54 + 0.46 \cos\left(2\pi \frac{1}{W} t\right) \right] \cdot \Pi(t/W)$$

and

$$\text{Hamm}(W, f) = 0.54 W \cdot \text{sinc}(Wf) + 0.23 W \cdot \text{sinc}\left\{W\left(f + \frac{1}{W}\right)\right\} + 0.23 W \cdot \text{sinc}\left\{W\left(f - \frac{1}{W}\right)\right\}$$



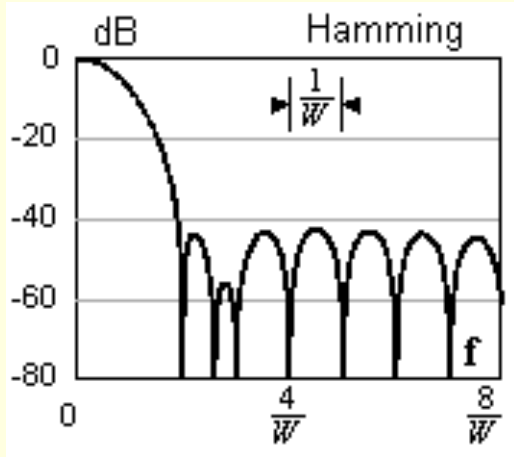
It makes small adjustments in DC level, and in the cosine amplitude, to obtain better suppression of the nearest side-lobes, but the side-lobes are hard to see on a linear scale. We will use a log scale to make the side lobes more visible, but we don't need to show the negative frequencies.

These (Fig c) are dB spectral plots for all three windows. Each plot is adjusted for 0 dB at $f = 0$. To illustrate this for the Hanning window, the function that we have plotted is:

$$20 \cdot \log_{10} |\text{Hann} (W, f)| - 20 \cdot \log_{10} |\text{Hann} (W, 0)|$$

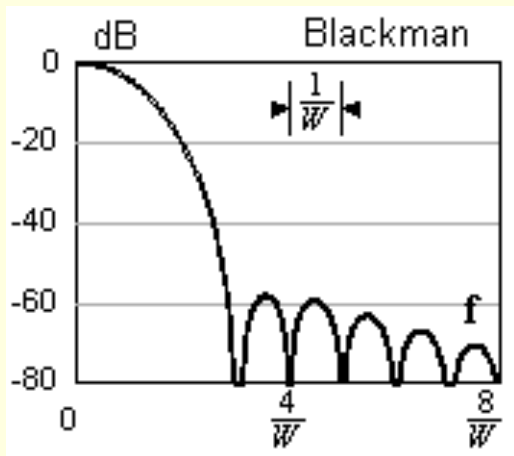
The side-lobes can now be compared. The rectangular window has large side-lobes, the largest being -13.5 dB for the nearest lobe. This compares with -32 dB for the Hanning window and -42 dB for the Hamming window. By this comparison, the Hamming window wins, but the Hanning side-lobes are better in the sense that they fall away rapidly with

frequency. In practice, the choice of a suitable window will depend on the context in which it is used.



9.4.4 The Blackman Window

The Blackman window achieves even greater side-lobe suppression, using broadly similar methods. It is defined by:



$$bk(W, t) = \left\{ 0.42 + 0.50 \cdot \cos\left(2\pi \frac{1}{W} t\right) + 0.08 \cdot \cos\left(2\pi \frac{2}{W} t\right) \right\} \cdot \Pi(t/W)$$

and by

$$\begin{aligned}
 Bk(W, \mathbf{f}) = & 0.42W \cdot \text{sinc}(W\mathbf{f}) \\
 & + 0.25W \cdot \text{sinc}\left\{W\left(\mathbf{f} + \frac{1}{W}\right)\right\} + 0.25W \cdot \text{sinc}\left\{W\left(\mathbf{f} - \frac{1}{W}\right)\right\} \\
 & + 0.04W \cdot \text{sinc}\left\{W\left(\mathbf{f} + \frac{2}{W}\right)\right\} + 0.04W \cdot \text{sinc}\left\{W\left(\mathbf{f} - \frac{2}{W}\right)\right\}
 \end{aligned}$$

Window Type	K_w
Rectangular	1
Hanning	2
Hamming	2
Blackman	3

Its spectrum is a sum of 5 sincs, rather than 3 sincs, giving even greater opportunity for side-lobe suppression by careful choice of weight factors. We can judge the result from its dB spectral plot (Fig 6). Side-lobes are the lowest we've seen, no more than -57 dB (that's only 0.14% of peak-value), and falling. This is a popular window, but it extends the main-lobe half-width to $3/W$, compared with $2/W$ for the Hanning and the Hamming, and just $1/W$ for the Rectangular window. The relative main-lobe widths are tabulated here (Fig 7) in terms of a width-factor that we've called K_w .

We'll use these windows in the next chapter to design better filters, and also in our subsequent work on the spectral evaluation of signals.





10.1 PREAMBLE

This chapter shows how we can build an FIR filter that approximates any specified spectral shape. Some important properties of FIR filters are identified and discussed. We also commence a gradual move toward a new " z -notation" which will later acquire more formal status as the z -transform. We show how advanced computer-based algorithms can give better performance than our (theoretically simpler) approach, and are also easy to use.



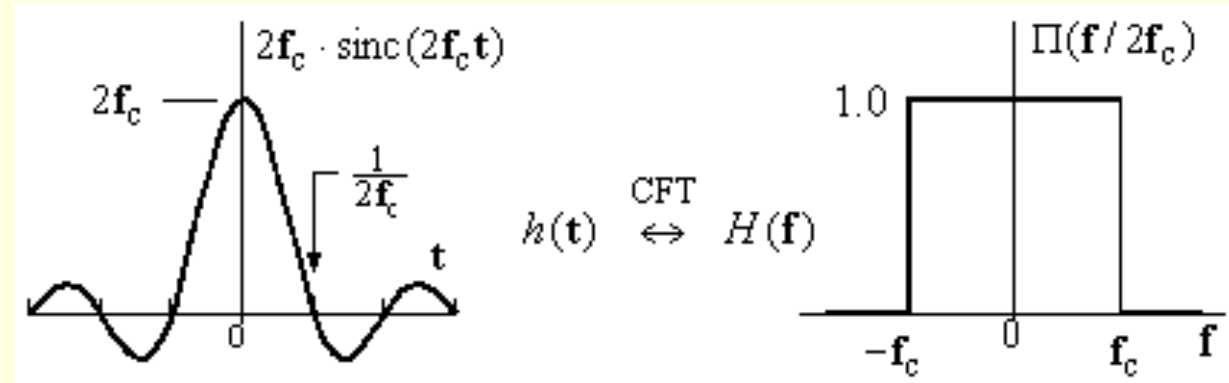


10.2 FIR FILTERS FROM FT PAIRS

An LTI filter is specified by its impulse response, $h[n]$, a discrete data sequence. This is a *signal* description as well as a filter description, and no distinction is necessary. The signal spectrum is $H(f)$, and this is also the filter spectrum. Fourier Transform (FT) theory provides the link between signals and their spectra, and forms the basis for the design method that we will present.

10.2.1 Filters based on CFT Pairs

A majority of filters are based on (idealised) brick-wall descriptions, and these are easily derived from CFT pairs. For example, a low-pass (LP) filter with a passband extending from DC to a cut-off frequency f_c (in Hz) can be based on the CFT pair shown here (Fig 6.1).



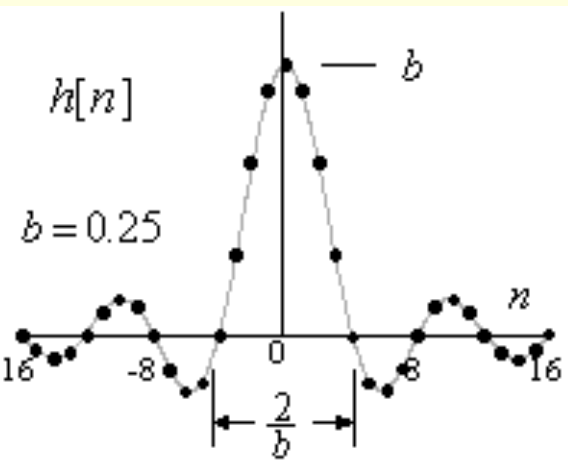
This $\{h(t), H(f)\}$ pair is a scaled version of CFT pair 5 in our Table (see [Ch 8.3.3](#)). We've *stretched* $\Pi(f)$ by $2f_c$, causing $\text{sinc}(t)$ to become *narrower* and *taller* by the same factor. The resulting $H(f)$ is the spectrum of an ideal analogue LP filter with cutoff frequency f_c Hz, and $h(t)$ is the filter's impulse response. To convert the filter to digital form, we must have a sampling frequency f_s which is suitably higher than $2f_c$. We then replicate $H(f)$ at intervals of f_s into a periodic $H(f)_{\sim}$ which is the spectrum of a set of area-samples $T \cdot h(nT)$ from $h(t)$, where $T = 1/f_s$. These area-samples become $h[n]$, the coefficients of our digital LP filter.

Our filter has a *two-sided* bandwidth of $2f_c$ Hertz. For digital working, we prefer the *normalised* bandwidth which we'll call b :

$$b = 2f_c/f_s = \text{bandwidth (cycles/sample) [two-sided, normalised]}$$

Remembering that $h(t) = 2f_c \cdot \text{sinc}(2f_c t)$, our filter coefficients become:

$$h[n] = T \cdot h(nT) = T \cdot 2f_c \cdot \text{sinc}(2f_c nT) = \frac{2f_c}{f_s} \cdot \text{sinc}\left(n \frac{2f_c}{f_s}\right) = b \cdot \text{sinc}(nb)$$



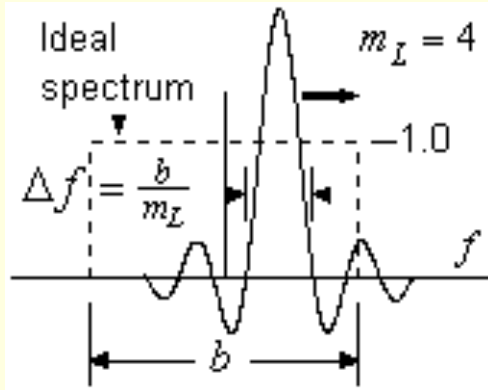
These (Fig c) are some of our coefficients when $b = 1/4$ (when the filter passband fills $1/4$ of the base-band). The largest coefficient is $h[0] = b = 0.25$, and we have $1/b = 4$ coefficients per lobe width, with $2/b = 8$ coefficients spanning the main lobe. There's just one problem: the sinc extends to $\pm \infty$, so it takes an infinity of coefficients to build the perfect brick-wall filter. We must make do with a finite length L , a set of L coefficients. Because each iteration of a filter requires L multiply operations, filter speed depends on keeping L as low as possible.

For a finite filter length L , the sinc must be windowed, but to get satisfactory results, the window should be several lobes wide. We will define window width W relative to the main-lobe width of $h(t)$, which is $1/f_c$ in seconds (or $2/b$ sample intervals), by saying that $W = m \cdot L \cdot (1/f_c)$ such that:

$$m_L = \frac{\text{window width}}{\text{main-lobe width}} = W \cdot f_c \quad (m = \text{window width factor})$$

L

The ideal filter spectrum of $\Pi(f/2f_c)$ is modified by windowing. If we call the window $w(t)$, with spectrum $W(f)$, the multiplication of $h(t)$ by $w(t)$ causes the filter spectrum $\Pi(f/2f_c)$ to be convolved with $W(f)$. For a simple rectangular window $w(t) = \Pi(t/W)$, the window spectrum is $W \cdot \text{sinc}(Wf)$, and the convolution then looks like this (Fig c). We've used a normalised f scale that shows the ideal filter in a bandwidth b , and the main-lobe of the sinc has a width of $2/W$ Hz, which, on the f scale, becomes $\Delta f = 2f_c / (mL f_s) = b/m \cdot L$. Under convolution, this sinc shape scans across the filter outline and



distorts it. The convolution in the diagram is for $m_L = 4$, indicating a window span that just covers the samples in the $h[n]$ plot above, and that yields a filter length of $L = 33$. We can view the resulting spectrum as the DtFT taken over these samples, that is:

$$H(f) = \sum_{n=-16}^{16} h[n] \cdot e^{-j2\pi fn}$$

Direct evaluation of this $H(f)$ gives the plot that we see here (Fig 6). We've shown a range of f that is twice the baseband width, just to emphasise the periodicity of $H(f)$, although the range $(0 < f < 0.5)$ would suffice.



The result of convolution with the window spectrum $W \cdot \text{sinc}(Wf)$ is evident in the oscillations that we see. Passband gain remains close to 1.0 because the area of $W \cdot \text{sinc}(Wf)$ is 1.0, but the ripple gives considerable variation. Stopband attenuation is poor, again because of the ripple. $\Delta f = b/m$ takes on special significance as *the width of the transition band*, as well as being the main-lobe width of $W \cdot \text{sinc}(Wf)$. We can see why this is so from the spectral convolution caused by windowing (Fig 9.4.3).

We defined the time-window width as $m_L \times$ (main-lobe width) which, expressed in sample intervals, becomes $m_L \cdot 2/b$, which is $2/\Delta f$. This is also the filter length, except that we treat $h[0]$ as one extra coefficient, and therefore:

$$L = 1 + \frac{2}{\Delta f} \quad \text{the filter length}$$

In practice, we round up the value of L to the nearest integer (often to the nearest *odd-valued* integer). The important point is that a sharp transition requires a long filter. Filter length is the price that we pay for a narrow transition band, Δf .

Window Type	K_w
Rectangular	1
Hanning	2
Hamming	2
Blackman	3

The ripple is an unwelcome aspect that needs attention. It arises from the sharp cutoff of a rectangular window, and can be greatly reduced by using a tapered window instead. All that we said about tapered windows is applicable here. The spectra of tapered windows have a main lobe that is wider by a factor K_w , as previously defined (see Ch 9.4.4), and the Table is shown again here (Fig 9.4.4). The Blackman window has the lowest side-lobes, but it also gives the greatest transition width for a given filter length. We must now modify our length estimate to read:

$$L = 1 + \frac{2K_w}{\Delta f}$$

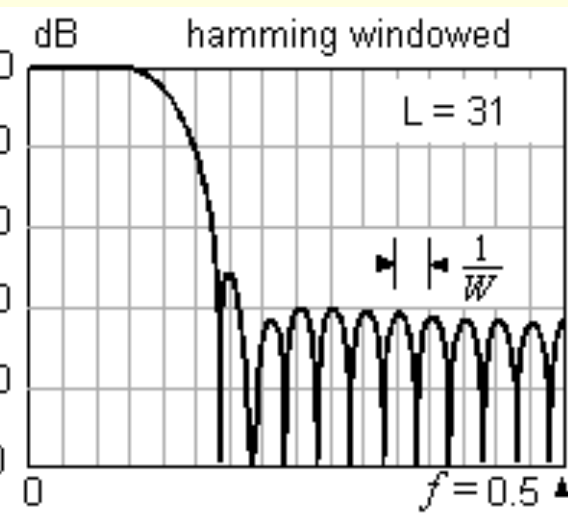
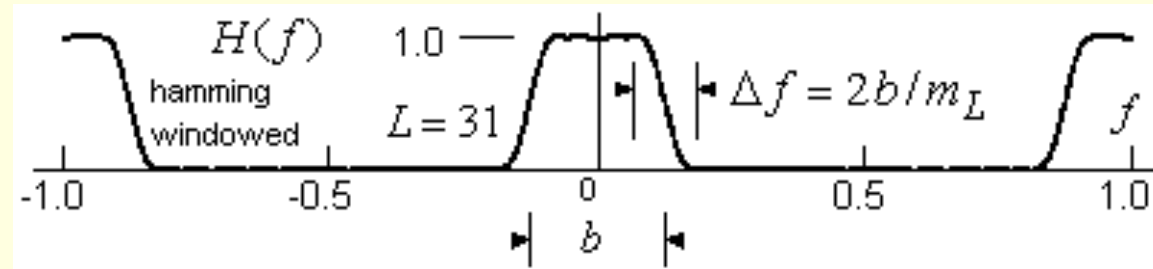
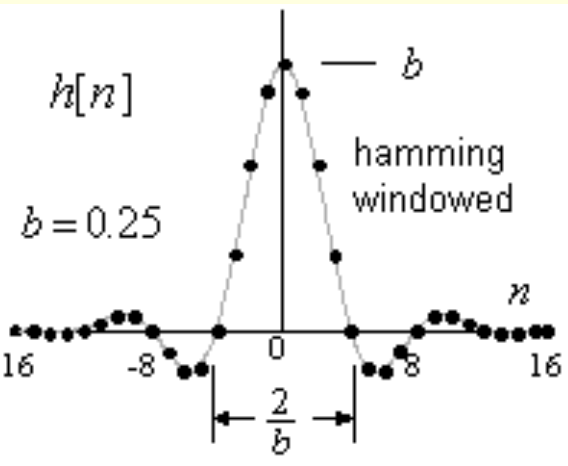
We'll use the Hamming window to show how coefficient values are altered. The window description is (i [Ch 9.4.3](#)):

$$\text{hamm}(W, t) = \left[0.54 + 0.46 \cos\left(2\pi \frac{t}{W}\right) \right] \cdot \Pi(t/W)$$

and our new set of coefficients becomes:

$$h[n] = b \cdot \text{sinc}(nb) \cdot \left[0.54 + 0.46 \cos\left(2\pi \frac{n}{L}\right) \right] \quad (-16 < n < 16)$$

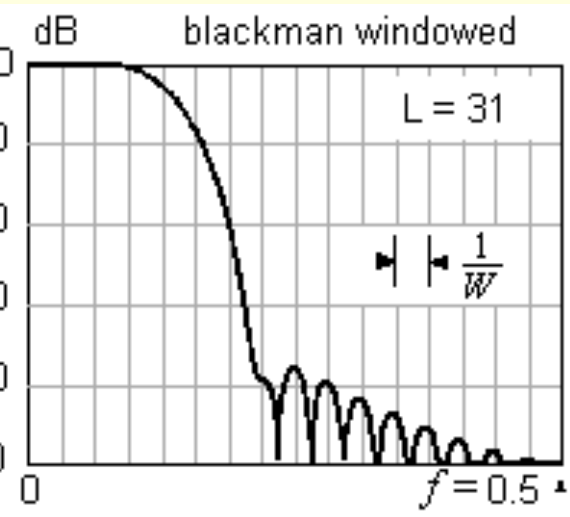
The indexing range shown is for our present example. We've also moved from seconds to sample-intervals by replacing t/W with n/L . The tapered coefficients reduce in value as we move away from centre, like this (Fig ç). The DtFT of the tapered coefficients gave us this new spectral diagram (Fig ê).



The ripple is much smaller, as expected, but the transition band is wider. Nominally, it is twice as wide, but this is a rather imprecise measure, and we'll introduce a better measure shortly. For a better appreciation of filter performance, a log scale is required, and only the positive baseband need be shown. This is the log plot of $H(f)$ for the same hamming-windowed filter (Fig ζ), and it shows almost 60 dB of stopband rejection. For comparison, this is the corresponding result when a Blackman window is used (Fig $\acute{1}$). The stopband rejection is much better, but the main lobe is wider, although not so wide as our K_w figures would suggest. Notice how the stopband ripple has a period of $1/W$ Hz, which is $1/2b/m_L$ on the f scale.

The following is a suitable design procedure:

1. Choose bandwidth $b (= 2 f_c \pm \Delta f)$, and transition width Δf .



- Choose a window to give the desired attenuation
- Choose filter length L close to $1+2K$
 $w/\Delta f$, even or odd.
- Compute coefficients $h[n]$ and spectrum $H(f)$.
- Examine $H(f)$ and iterate the solution if required.

One can use odd-length filters in general, and even-length filters where there is a specific requirement for same.

10.2.2 FIR Filters by Inverse DtFT

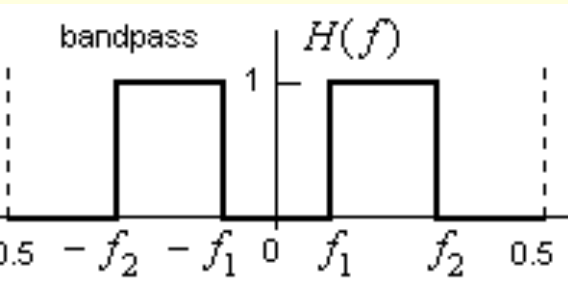
In the example just completed, we started with an analogue LP filter based on a CFT pair, and we moved to digital filter form by time-domain sampling. We did this to emphasise the analogue domain connection, but in fact we could have found the same filter more directly using the DtFT. The key to this approach is the inverse DtFT integral (i [Ch 6.4.1](#)):

$$h[n] = \int_{-1/2}^{1/2} H(f) \cdot e^{j2\pi f n} \cdot df \quad \text{normalised IdtFT integral}$$

We just specify the $H(f)$ that we require, and then use this integral to find $h[n]$. We deal with windowing issues in the same way as before. To illustrate the method, we will design a brickwall band-pass (BP) filter. The LP filter is a special case of the

BP design, and we will show that it gives the same set of coefficients as we have found already.

This is a brickwall bandpass filter (Fig 5), and it gives rise to the following integral for $h[n]$:



$$h[n] = \int_{-f_2}^{-f_1} 1.0 \cdot e^{j2\pi f n} \cdot df + \int_{f_1}^{f_2} 1.0 \cdot e^{j2\pi f n} \cdot df$$

This integrates to:

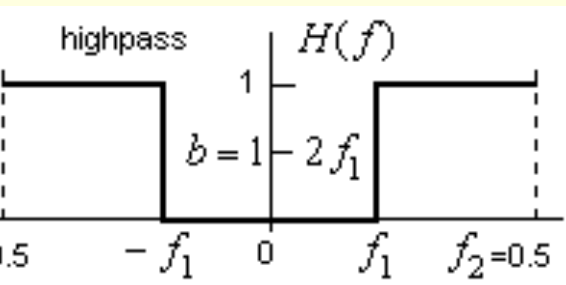
$$h[n] = \frac{1}{j2\pi n} \left[e^{j2\pi f n} \right]_{-f_2}^{-f_1} + \frac{1}{j2\pi n} \left[e^{j2\pi f n} \right]_{f_1}^{f_2}$$

and emerges as:

$$h[n] = \frac{1}{n\pi} [\sin(2\pi f_2 n) - \sin(2\pi f_1 n)]$$

The case of $n = 0$ must be integrated separately to yield $h[0] = 2(f_2 - f_1)$. We can also write this $h[n]$ in terms of the sinc function:

$$h[n] = 2f_2 \cdot \text{sinc}(2f_2 n) - 2f_1 \cdot \text{sinc}(2f_1 n)$$



We can check the low-pass (LP) case by setting $f_1 = 0$ and $f_2 = \frac{1}{2}b$. It yields $h[n] = b \cdot \text{sinc}(nb)$, the same result as before. For the high-pass (HP) case, we set $f_2 = 0.5$, (Fig c) and the pass-band, which now extends *circularly* from f_1 to $-f_1$, has a width which we can write as $b = 1 - 2f_1$. In this case:

$$h[n] = \frac{1}{n\pi} [0 - \sin(2\pi f_1 n)] = \frac{1}{n\pi} [\sin((b-1)n\pi)] = \frac{(-1)^n}{n\pi} [\sin(nb\pi)]$$

and it becomes:

$$h[n] = (-1)^n \cdot b \cdot \text{sinc}(nb)$$

Except for the $(-1)^n$ term, this is the same as the LP case. The only difference is that coefficients at odd values of n change sign.

We can now obtain coefficients for a brickwall filter of any specification. There is still the matter of ripple, and of transition-band width, and those issues are dealt with in the same way as before. But we are not restricted to brickwall filters. We can specify *any* $H(f)$ in the IdtFT integral. Quite often, the integral is an easy one, and we get an algebraic expression for $h[n]$. In more difficult cases, we can use numeric integration, and still get a result. We can even specify the desired $H(f)$ as a set of DFT spectral values, and then use the IDFT to get the desired $h[n]$, but we should note the possibility of time-domain aliasing if we use this approach.

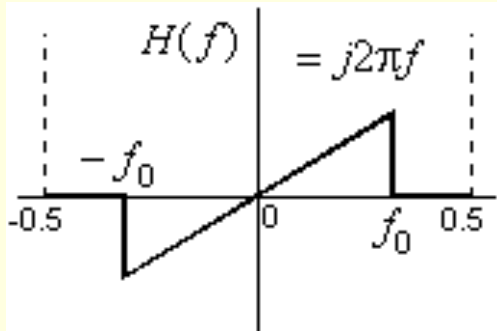
10.2.3 Differentiators by Inverse DtFT

A filter that differentiates a signal is very different from the brickwall type, and we'll use it as a further illustration of the IDtFT method. The output from such a filter should approximate the *slope* of the input signal. We may be familiar with the analogue case, where an input signal $x(t)$ is differentiated to give an output $y(t) = dx(t)/dt$, and the filter spectrum is $H(f) = j2\pi f$, or simply $j\omega$. To build a *simple* digital differentiator, we can *approximate* this action as:

$$\frac{x[n] - x[n-1]}{T} \leftrightarrow j2\pi f \quad \text{differentiation}$$

Substituting $f = \mathbf{f}/\mathbf{f}_s = \mathbf{f}T$, it becomes $(x[n] - x[n-1]) \leftrightarrow j2\pi f$. The filter that gives $y[n] = x[n] - x[n-1]$ has an impulse response $h[n] = \{1, -1\}$, and we will soon show that its spectrum $H(f)$ is a good approximation to $j2\pi f$ at low frequency (below $f = 1/4$).

The more formal way to design a differentiator is to set $H(f) = j2\pi f$ and insert this in the IdtFT integral. Since all $H(f)$ are periodic in $(-0.5 < f < 0.5)$, we can use these ± 0.5 limits, but we may prefer a lower limit $f = 0$ as suggested here (Fig 10.2.3). Using this in the IdtFT integral, we must find:



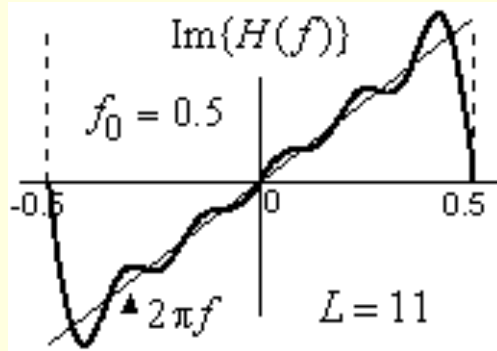
$$h[n] = j2\pi \int_{-f_0}^{f_0} f \cdot e^{j2\pi f n} \cdot df$$

We can apply the known integral:

$$\int x \cdot e^{ax} \cdot dx = \frac{e^{ax} \cdot (ax - 1)}{a^2}$$

with $a = j2\pi n$ and, after some algebra, we get the result:

$$h[n] = \frac{1}{\pi n^2} [2\pi f_0 n \cdot \cos(2\pi f_0 n) - \sin(2\pi f_0 n)]$$



The case of $n = 0$ is integrated separately to yield $h[0] = 0$. We used this result to generate a $h[n]$ of length $L = 11$ with $f_0 = 0.5$, and the DtFT of the sequence gave us this spectral shape (Fig c). The straight line marks the target value of $H(f) = j2\pi f$. The oscillation is due to simple rectangular windowing, but can be largely removed by use of a tapered window. We will then have a differentiator that performs well up to $f = 0.4$ approx. (The much simpler filter $h[n] = \{1 \ -1\}$ with only $L = 2$ coefficients has a poorer range, up to $f = 0.2$ or thereabouts).

If we substitute $f_0 = 0.5$ in our $h[n]$ expression, it simplifies to:

$$h[n] = \frac{\cos(n\pi)}{n} = \frac{(-1)^n}{n}$$

with values of $\{ \dots 1/3 \ -1/2 \ 1 \ 0 \ -1 \ 1/2 \ -1/3 \ \dots \}$. Notice that if we window this to a length of $L = 3$, it becomes $h[n] = \{1 \ 0 \ -1\}$, not unlike our intuitive filter $h[n] = \{1 \ -1\}$, except that the low-frequency gain of this 3-point filter is $4\pi f$ rather than $2\pi f$! That's a large error due to the very narrow window, but it diminishes with greater filter length. We'll revisit these examples in the next section, as part of our discussion on FIR filter attributes.





10.3 FIR FILTER ATTRIBUTES

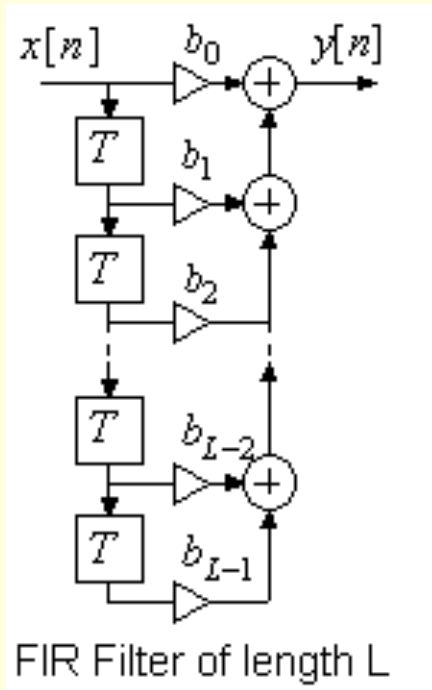
FIR filters are different in some fundamental ways from IIR filters. They require more computation than IIR filters of comparable performance, but this drawback is often compensated by other features which we will now explore.

10.3.1 Causal FIR Filters

Until now, we've used symmetric indexing (centered on $n = 0$) for our set of filter coefficients $h[n]$. For example, our LP filter was windowed to a length of $L = 33$ coefficients, which we described as:

$$h[n] = b \cdot \text{sinc}(nb) \quad \text{for } -16 < n < 16$$

But filters are generally *causal*, they operate in real-time. The standard causal FIR filter is redrawn here (Fig c), and it follows the difference equation:



$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + \dots + b_{L-1}x[n-(L-1)]$$

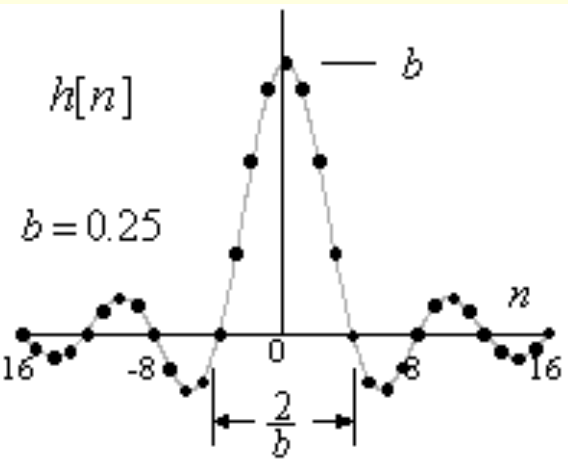
The output cannot precede the input, and this means that indexing cannot start before $n = 0$. The b -coefficients (not to be confused with bandwidth b) are just the $h[n]$ values, but the indexing is different. In our LP filter example:

$$b_n = h[n-16] \quad \text{for } n = 0, 1, 2 \dots 32$$

The only difference between b_n and $h[n]$ is a right-shift, or a time delay, of the $h[n]$ sequence by 16 samples, which is $\frac{1}{2}(L-1)$ samples in the general case.

10.3.2 The Linear Phase Attribute

Most of our FIR filters have important symmetries. A glance at the LP filter coefficients (Fig c) shows that they have *even* symmetry, that is:



$$h[-n] = h[n]$$

even symmetry

In fact, this is true of the BP and HP filters as well. If we turn to the differentiator example, we find that it has *odd* symmetry, that is:

$$h[0] = 0$$

and

$$h[-n] = -h[n]$$

odd symmetry

A majority of FIR filters have one or other of these symmetries, with important consequences for the filter spectrum, as follows. When we compute the spectrum by DtFT, the coefficients (except for $h[0]$) combine naturally in pairs. For a filter of length $L = 3$, we have:

$$H(f) = h[-1] \cdot e^{-j2\pi f(-1)} + h[0] \cdot e^{-j2\pi f(0)} + h[1] \cdot e^{-j2\pi f(1)}$$

For the even-symmetry case, $h[-1] = h[1]$, and then:

$$H(f) = h[0] + h[1] \cdot (e^{j2\pi f} + e^{-j2\pi f}) = h[0] + 2h[1] \cdot \cos(2\pi f)$$

The notable thing about this $H(f)$ is that it is *real-valued*. For the odd-symmetry case, $h[0] = 0$ and $h[-1] = -h[1]$, and the result is very different:

$$H(f) = -h[1] \cdot e^{j2\pi f} + h[1] \cdot e^{-j2\pi f} = -2j \cdot h[1] \cdot \sin(2\pi f)$$

The notable thing about this $H(f)$ is that its value is *imaginary*. As we add more coefficient pairs $\{h[-2], h[2]\}$, $\{h[-3], h[3]\}$ etc, these $H(f)$ continue to be *real* for even-symmetry filters and *imaginary* for odd-symmetry filters.

When we go from symmetric indexing to causal indexing, we delay the filter response by $\frac{1}{2}(L-1)$ samples. This applies a *linear-phase lag* term to $H(f)$:

$$H(f) \rightarrow = H(f) \cdot e^{-j2\pi f d} \quad d = \frac{1}{2}(L-1) \text{ sample intervals}$$

For an even-symmetry filter, $H(f)$ has *zero* phase at first, and when we make it causal, the linear-phase lag becomes its only phase term. It has *linear phase*, and this brings us to the question of distortion.

A brickwall filter removes certain frequencies from a signal, and tries to leave the remaining part *undistorted*. But it can still be distorted in two ways. If the gain magnitude varies across the passband, this imposes an *amplitude distortion* on the signal. We saw a ripple effect that could distort in this way, but we were able to minimise it by using tapered windows. The other kind of distortion is *phase distortion*. If the phase of $H(f)$ varies linearly over frequency, it delays all frequencies by the same amount, but that does not change the signal *shape*, it does not cause phase distortion. If the phase of $H(f)$ in the passband is *non-linear* over frequency, then different frequencies will be delayed by different amounts, and the signal *will* suffer from phase distortion.

The even-symmetry filter has only a linear-phase lag term, causing it to delay all frequencies equally by $\frac{1}{2}(L-1)$ samples, or by $\frac{1}{2}(L-1)T$ seconds. This is just the time difference between symmetric and causal responses, and it does not cause phase distortion. That is one of the big advantages that FIR filters can offer, all the more so because we will see that IIR filters *do* cause phase distortion, and to a degree that is often unacceptable.

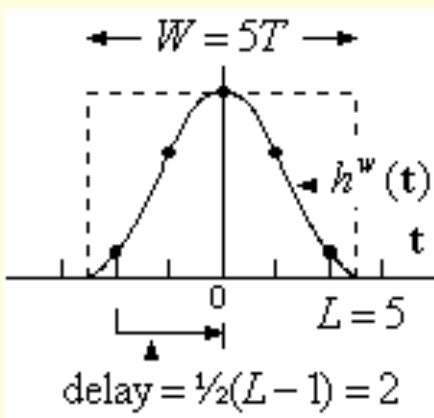
Odd-symmetry filters do not have strict linear phase, but they have a phase plot that consists of straight-line segments, so they do have *linear-phase slope*. That's not enough to eliminate shape distortion in general, but we'll see later that it is a sufficient protection for bandpass signals. That makes odd symmetry a useful property also, in some situations. We will soon see sample spectra to illustrate both linear phase and linear phase slope. Our final task in this section will be to show how we can ensure that these important symmetries are preserved.

FIR filters having even or odd symmetry are based on a windowed time response:

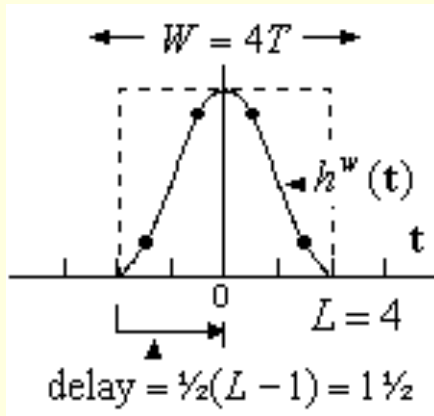
$$h^w(t) = w(t) \cdot h(t)$$

in which the window function $w(t)$ has even symmetry, and the filter function $h(t)$ has either even or odd symmetry. The resulting $h_w(t)$ has the same symmetry as $h(t)$. We want to maintain this symmetry in the samples from $h_w(t)$ that become our filter coefficients. For a filter of length L , we will use a window of length $W = LT$ seconds, for both even-length and odd-length filters.

The odd-length case is illustrated here (Fig c) when $L = 5$. The samples include a central value when $t = 0$, and the sequence is symmetric about the central value. To obtain the causal coefficients b_n , we must slide $h_w(t)$ to the right by $\frac{1}{2}(L-1)$ samples, or 2 samples in this case. The samples now commence at $t = 0$, and we can sample at instants $t = nT$ for $n = 0 \dots 4$. We then get:



$$b_n = T \cdot h^w\left(nT - \frac{1}{2}(L-1)T\right) \quad n = 0 \dots L-1$$



The even-length case is illustrated here (Fig 5) for $L = 4$. To maintain symmetry, samples are offset from $t = 0$ by $\pm 1/2T$. There are 4 samples, forming 2 symmetric pairs. To obtain the causal coefficients b_n , we must slide $h_w(t)$ to the right by $1/2(L-1)$ samples, or $1 1/2$ samples in this case. The samples now commence at $t = 0$, and we can sample at instants $t = nT$ for $n = 0 .. 3$. We then get:

$$b_n = T \cdot h^w\left(nT - \frac{1}{2}(L-1)T\right) \quad n = 0 .. L-1$$

This expression is the same as for the odd-length case, so we can use it to ensure symmetry, irrespective of the filter length. Notice that when L is even, we get a non-integer delay. This carries the inference that an *interpolation* is being performed, producing output samples that are mid-way in time between the input samples.

10.3.3 Filter Descriptions using "z".

The term $e^{-j2\pi f}$ appeared several times in the last paragraph. It's cumbersome to write, but it describes something quite simple. It's a linear-phase lag term, the frequency-domain equivalent of a T -second delay, or a 1-sample delay.

$$e^{-j2\pi fT} = e^{-j2\pi f/f_s} = e^{-j2\pi f}$$

a 1-sample delay

Not surprisingly then, it appears quite often, and it makes sense to have a simpler notation. The common practice is to define a new parameter z as:

$$z = e^{j2\pi f}$$

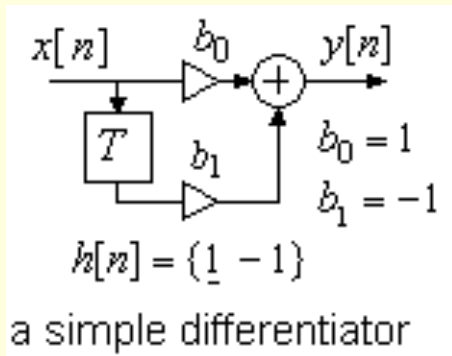
from which

$$z^{-1} = e^{-j2\pi f}$$

a 1-sample delay

We will use the new z parameter to help describe some filters that we know, and we will be surprised at how it lets us see these filters in an entirely new light. In all cases, the filter spectrum is:

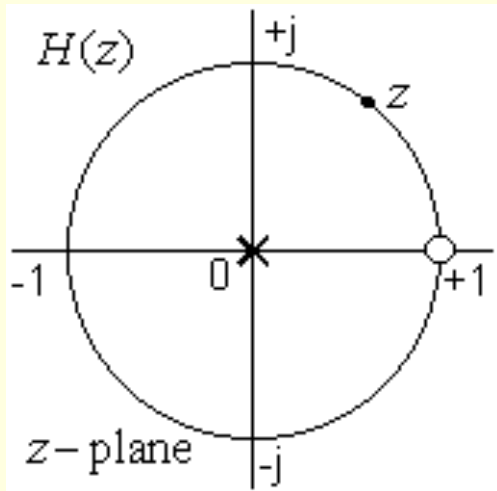
$$H(f) = \sum_n h[n] \cdot e^{-j2\pi f n} = \sum_n h[n] \cdot (e^{j2\pi f})^{-n} = \sum_n h[n] \cdot z^{-n}$$



For now, this is just the DtFT using a compact notation. Later on, we'll enlarge the concept and call it the "z-transform". To get an idea of where this can lead us, we'll apply the new notation to our "intuitive" differentiating filter.

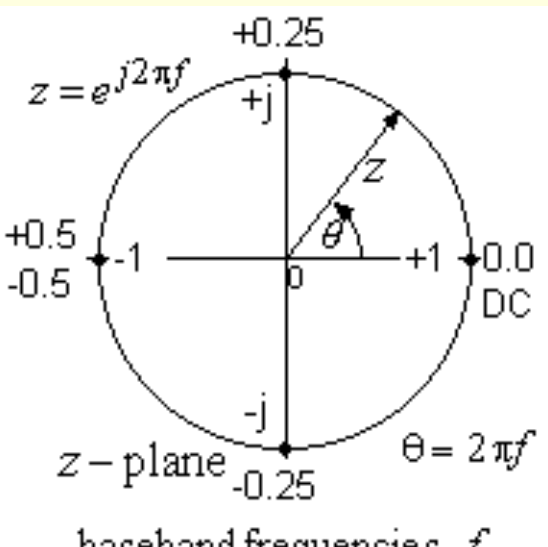
The filter is seen here (Fig 5) and, from its impulse response $h[n] = \{1 \ -1\}$, the spectrum is easily found:

$$H(z) = h[0] \cdot z^0 + h[1] \cdot z^{-1} = 1 - z^{-1} = \frac{z-1}{z}$$

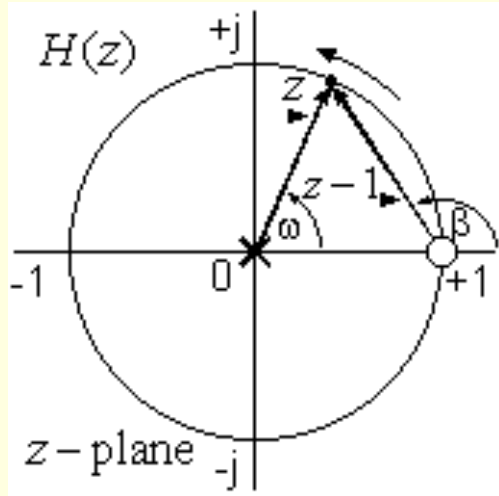


We've called it $H(z)$ rather than $H(f)$ because the frequency is expressed solely in terms of z . This $H(z)$ is a polynomial in z . $H(z)$ goes to zero when $z = 1$ and it goes infinite when $z = 0$. What we normally say is that $H(z)$ "has a zero" at $z = 1$ and it "has a pole" at $z = 0$. In all of this, z is a complex-valued frequency variable. Each value of z is a point on a so-called "z-plane". We can show $H(z)$ pictorially on the z -plane by marking in the pole and zero points.

This is a "pole-zero plot" for our simple differentiator (Fig ζ). We defined z as $e^{-j2\pi f}$ which is the same as $1.0 \angle (2\pi f)$. Its value is constrained to a *magnitude* of 1.0, at an *angle* determined by f . That's the significance of the circle on the diagram. It contains all possible values of z as we have defined it. It has a radius of 1.0 centered on $z = (0+j0)$. This "unit-circle" is our frequency axis. We've shown the "zero" of $H(z)$ as a small circle at $z = 1+j0$. We've shown the "pole" of $H(z)$ as an "x" at $z = 0+j0$. These two points are all that we need to say that $H(z) = K \cdot (z-1)/z$, but they don't give us a value for K . The pole-zero plot describes $H(z)$ completely in a pictorial way, *except for a scale factor*. These plots are a popular way of describing digital filters.



This unit-circle plot (Fig ζ) shows baseband frequencies at quadrant points. The numbers inside the circle are z values at the 4 corner points. The numbers outside the circle are corresponding f values, as computed from $z = e^{-j2\pi f}$ by noting that z is a vector of length 1.0 at an angle θ such that $\theta = 2\pi f$. The DC point $f = 0$ is at co-ordinates (1,0). If we go anti-clockwise from DC around the top-half of the circle, we move through the positive base-band range from 0.0 around to 0.25 and finally to 0.5. The bottom half-circle is the negative baseband range. The ± 0.5 points coincide because of the strictly periodic nature of digital signal spectra. If we keep going around the circle we cover the same ground again as we enter the higher image bands. That's because the images are inseparable from the baseband content, and the unit-circle is an expression of this effect.



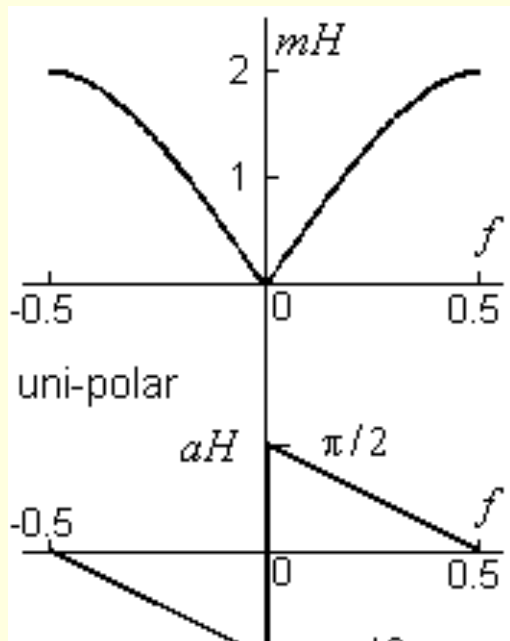
The pole-zero plot can actually tell us the shape of $H(f)$. This diagram (Fig 5) shows $H(z) = (z-1)/z$ as a ratio of two vectors which connect the zero and the pole to z , where z is both a unit-length vector and a mobile point on the circle that represents frequency. As we traverse the baseband, z moves around the circle and the vector lengths and vector angles change accordingly. We can find the spectral magnitude plot as:

$$mH(f) = \left| H\left(e^{j2\pi f}\right) \right| = \frac{|z-1|}{|z|}$$

and the angle plot as:

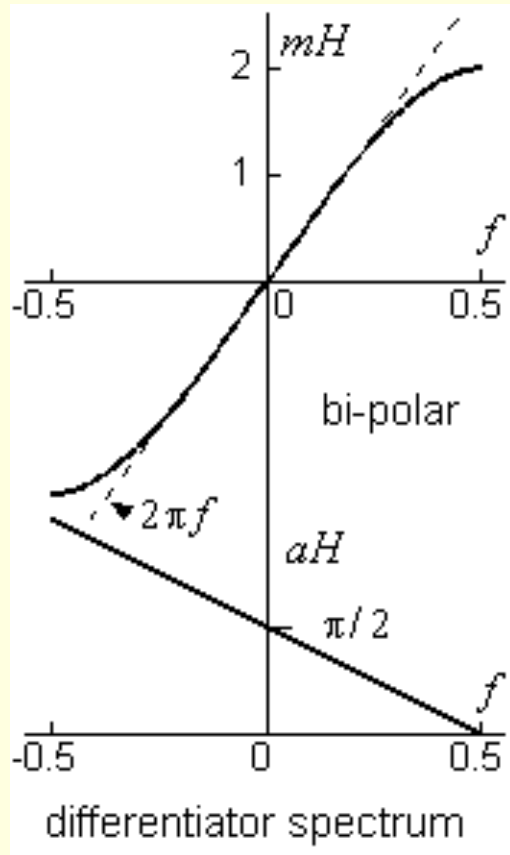
$$aH(f) = \arg\left\{ H\left(e^{j2\pi f}\right) \right\} = \angle(z-1) - \angle(z) = \beta - \omega$$

By visualising the motion of z on the circle, we can anticipate the shape of these plots. It's worth taking a few moments to try and do that now. The alternative is to compute mH as $|H(z)|$ using $e^{j2\pi f}$ in place of z , as in the expressions above, and similarly for aH . The results are shown here (Fig 6) and they should match your predictions ! This is a uni-polar plot, the kind that we would normally get from a computer, but the bi-polar plot shown here (Fig 7) has the very same meaning. We allow the magnitude to be positive or negative, and we adjust the phase plot by $\pm\pi$ radians to compensate. We draw bi-polar magnitude plots so that they reflect the underlying mathematics. In this case, $H(z) = 1 - z^{-1}$ and :



$$H(e^{j2\pi f}) = \left[1 - z^{-1} \right]_{z=e^{j2\pi f}} = 1 - e^{-j2\pi f}$$

Here's how we can separate the magnitude and the phase:



$$H(e^{j2\pi f}) = e^{-j\pi f} (e^{j\pi f} - e^{-j\pi f}) = 2j \cdot \sin(\pi f) \cdot e^{-j\pi f}$$

from which

$$mH(f) = 2 \sin(\pi f)$$

and

$$aH(f) = j \cdot e^{-j\pi f} = e^{j\pi/2} \cdot e^{-j\pi f} = 1.0 \angle (\pi/2 - \pi f)$$

The bi-polar mH plot was drawn to match the $mH(f)$ result seen here, and then we altered the phase accordingly. This might now look like linear phase, but *true* linear phase is linear through the origin, whereas this phase plot is offset by $\pi/2$. It has linear slope, but is not strictly linear. We saw a phase plot for an averaging filter (i [Ch 7.4.1](#)) and it *was* strictly linear. The same is true of the brickwall filters that we described.

The slope of the phase plot reflects the *delay* term $e^{-j\pi f}$, which describes a delay of $\frac{1}{2}(L-1)$ samples, or just a $\frac{1}{2}$ -sample delay. The offset of $\pi/2$ is the "j" of the ideal $j2\pi f$ transfer function. The dotted line on the mH plot marks the ideal $2\pi f$ response. This differentiator is close to the ideal as far as $f = \pm 0.25$, about half of the baseband.

10.3.4 FIR Transfer Functions using "z".

We can easily extend our z notation to cover FIR filters in general, according to the difference equation:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + \dots + b_{L-1}x[n-(L-1)]$$

We probably think of $x[n]$ as "the n^{th} sample" and of $x[n-1]$ as the one before it, and so forth. But, if consider the action *over all time*, we can also see $x[n]$ as an entire sequence, and we can see $x[n-1]$ as the same sequence, once delayed. The sequence $x[n]$ has a DtFT spectrum which we can write as:

$$X(z) = \sum_n x[n] \cdot z^{-n}$$

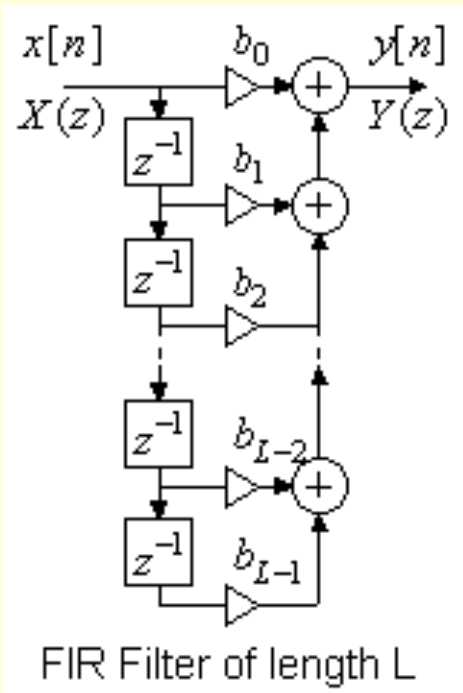
while the spectrum of $x[n-1]$ is $z^{-1} \cdot X(z)$ where z^{-1} describes the 1-sample delay. In similar fashion, the term z^{-2} describes a 2-sample delay, etc. If the difference equation is valid over *all time*, we can (by superposition) translate each and every term into a corresponding spectral function, with this result:

$$Y(z) = b_0X(z) + b_1z^{-1}X(z) + b_2z^{-2}X(z) + \dots + b_{L-1}z^{-(L-1)}X(z)$$

The filter Transfer Function (TF) is {output/input} = $Y(z)/X(z)$, which we like to call $H(z)$, and this becomes:

$$H(z) = \frac{Y(z)}{X(z)} = \left[b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_{L-1} z^{-(L-1)} \right]$$

It may not look much like a spectrum, but we can quickly extract the spectral information and plot it over frequency:



$$mH(f) = \left| H\left(e^{j2\pi f}\right) \right| \quad \text{and} \quad aH(f) = \arg\left\{ H\left(e^{j2\pi f}\right) \right\}$$

$(-0.5 < f < 0.5)$

The $H(z)$ expression covers FIR filters in general, and we feel prompted to use z^{-1} in place of T as our symbol for a 1-sample delay (Fig 10.3.5). But there is not very much new here. This new $H(z)$ is just the DtFT of the b_n coefficients (the causal version of $h[n]$), using the z -notation. However, the polynomial form of $H(z)$ is convenient for many purposes, and that includes what we're about to do next.

10.3.5 Linear-Phase FIR Zero Patterns

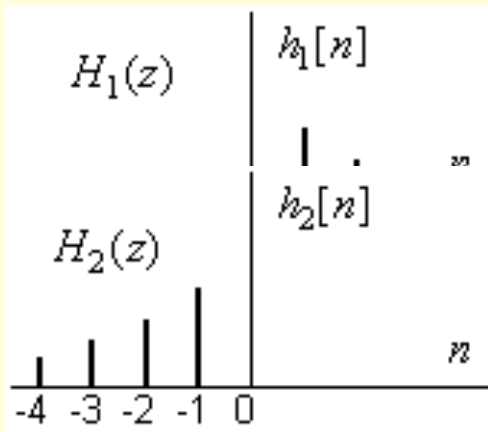
A polynomial $f(x)$ in x is of the form $f(x) = b_0 + b_1x + b_2x^2 + \dots + b_Nx^N$. It is of order N and it has N roots. That means that $f(x) = 0$ for N values of x . Finding the roots, if $N > 2$, is probably best done by computer. (Mathcad provides the "polyroots" function for this purpose). The roots are a useful alternative way to view some of the properties of $f(x)$.

Our $H(z)$ is a polynomial in z^{-1} of order $M = (L-1)$. We can write it in terms of its roots as:

$$H(z) = b_0 (1 - z_{01} \cdot z^{-1}) (1 - z_{02} \cdot z^{-1}) (1 - z_{03} \cdot z^{-1}) \dots (1 - z_{0M} \cdot z^{-1})$$

On a z -plane plot, the roots z_{01} , z_{02} , etc become the *zeros* of our filter. They are the values of z that cause $H(z)$ to become zero, and we've already seen how they can help us to visualise the filter spectrum. Notice, roots may be hard to find, but it's quite easy to multiply out this expression and to find the coefficients if the roots are already known. In general, we need the coefficients to run the filter, but the roots (that is, the zeros) are useful for the insight that they give us into the filter spectrum. The zeros of linear-phase filters come in groups that are easy to recognise, as we will now demonstrate.

This is a causal filter (Fig 5) described by the polynomial :



$$H_1(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4}$$

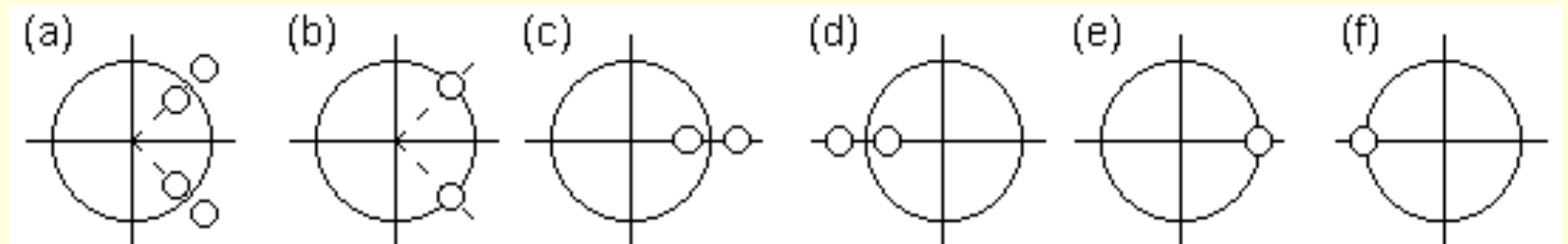
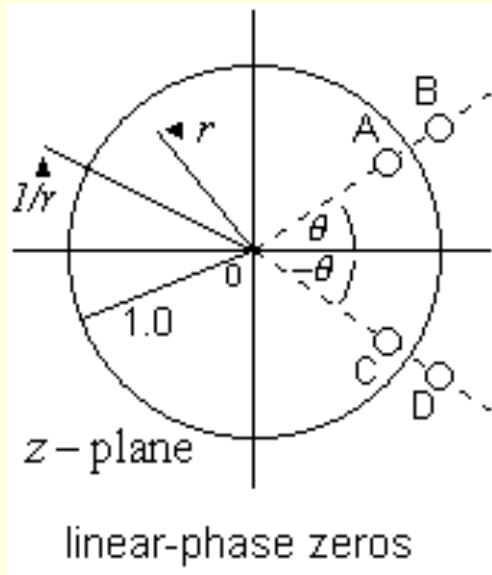
If z^{-1} is a time delay then z^{+1} is a time advance, so that a time-reversed version of the same filter (Fig 6) has this description:

$$H_2(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4}$$

$H_1(z)$ is a polynomial in z^{-1} . There are 4 values of z^{-1} that make $H_1(z) = 0$. $H_2(z)$ is a polynomial in z with *the same* coefficients. The same 4 numbers, when used as values of z , will make $H_2(z) = 0$. This means that the set of zeros for $H_1(z)$ and the set of zeros for $H_2(z)$ are at *reciprocal* locations on the z -plane. We admit that $H_2(z)$ is not causal, but that is easily remedied. A 4-sample right-shift makes it causal, and its spectrum becomes $z^{-4} \cdot H_2(z)$, but this does nothing to alter its zero locations. It follows that if we alter an FIR filter by reversing the order of its coefficients, then the zeros of the filter will move to *reciprocal* locations on the z -plane. We will now apply this idea to a linear-phase filter.

Linear phase comes from coefficient symmetry. If we time-reverse an even-symmetry set of coefficients, we end up with the same set as we started with (except for a time shift). This implies that reciprocation of the filter zeros has no overall effect on their placement. Only certain groups of zeros can meet this condition, primarily the group that we see here (Fig 1).

We have a group of 4 zeros labelled A, B, C, D. They are placed at angles $\pm\theta$ and at radii of $r (< 1)$ and $1/r$. The zero at A, for example, is at $r\angle\theta$ and, when we reciprocate it, we get $1/(r\angle\theta) = (1/r)\angle-\theta$, which is where D is located. The special property here is that when we reciprocate A it moves to D and when we reciprocate D it moves to A, and similarly for the (B,C) pair. Overall, these points just trade places, and we end up with the same set of zeros as we had before reciprocation. These zeros satisfy the linear-phase condition, but there are other special cases that we can include also. They are all shown here (Fig 2).



The general case is the (a) plot. Plot (b) shows two zeros at $1.0 \angle \pm\theta$. Plot (c) shows real zeros at r and at $1/r$. Plot (d) shows real zeros at $-r$ and at $-1/r$. Plot (e) has a zero at 1.0 . Plot (f) has a zero at -1.0 . It's easy to see that they all reciprocate without changing the overall zero placement. A linear-phase filter of length L has $L - 1$ zeros, often quite a large number, and these zeros can exist as any combination of the groupings in the diagram above. The same applies to filters that have only linear-phase slope, because time-reversal of an odd-symmetry sequence involves only a change of sign, and a possible time shift.



The zeros of an averaging filter are all on the unit circle, so they belong in the category of plot (b) above. When the zeros fall *on* the unit circle, the frequency response $H(f)$ goes to zero at the frequencies that those zeros represent.





