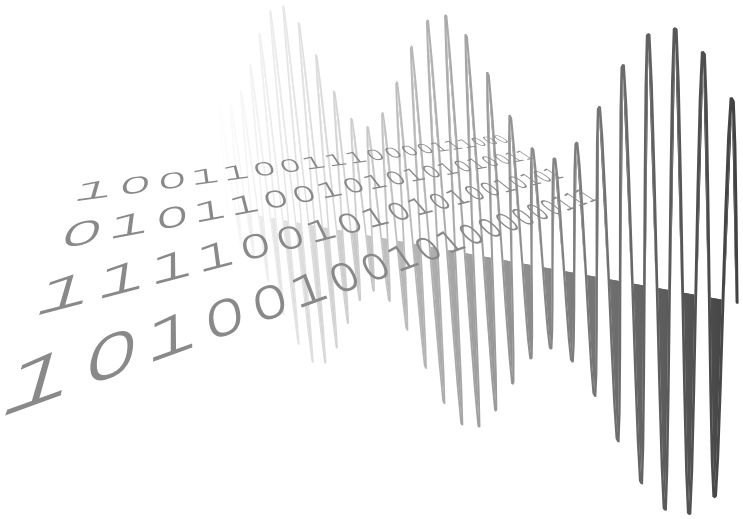
PH4026 Signals and Information



Paul Cruickshank and Graham Smith

Based on courses developed by Jim Lesurf

2020 Candlemas Semester v2

University of St Andrews

Contents

1 Introduction **8**

1.1 What is information? 8

1.1.1 What is an 'instrument'? 9

1.2 Accuracy and resolution 11

1.3 Chapter 1: Take-home messages 13

2 Sampling **14**

2.1 Describing signals: the time and frequency domains 14

2.2 Quantisation and sampling 16

2.3 Quantisation 17

2.3.1 How much information does a single value contain? 18

2.3.2 The required bit-depth 19

2.3.3 Dynamic range and signal-to-noise 20

2.4 Discretisation 22

2.4.1 Aliasing 22

2.4.2 Aliasing and spectral folding 24

2.5 The sampling theorem 26

2.6 Chapter 2: Take-home messages 27

3 Noise **29**

3.1 Johnson noise . . . 29
3.2 Shot noise . . . 36
3.3 One-over-f noise . . . 37
3.4 Quantisation noise and dither . . . 37

4 The (discrete) Fourier transform 41
4.1 Calculating the DFT . . . 44
4.1.1 The Fast Fourier Transform . . . 46
4.2 Some properties of the DFT . . . 49
4.3 Convolution and the convolution theorem . . . 50
4.3.1 The convolution theorem . . . 51
4.4 Fourier transform pairs . . . 52
4.4.1 The delta function . . . 52
4.4.2 The top-hat function . . . 53
4.4.3 The Gaussian . . . 54
4.5 Frequency resolution . . . 55
4.6 Spectral leakage and windowing . . . 55
4.7 Further reading . . . 59

5 Noisy messages, surprises and redundancy 61
5.1 Doubtful information and errors . . . 61
5.2 Surprises and redundancy . . . 67
5.3 Chapter 5: take-home messages . . . 71

6 Detecting and correcting mistakes 72
6.1 Errors and legality . . . 72
6.2 Parity and blocks . . . 74

6.3 Choosing a coding system 77
6.4 Chapter 6: take-home messages 79

7 The information carrying capacity of a channel 81
7.1 Signals look like noise 81
7.2 Shannon's equation 83
7.3 Choosing an efficient transmission system 84
7.4 Chapter 7: take-home messages 87

8 Example: the CD player 88
8.1 The CD encoding process 89
8.2 The CD player as a measurement system 93
8.3 The CD player as a digital signal processing system 95
8.3.1 Over-sampling 95
8.4 One bit more 97
8.5 Chapter 8: take-home messages 100

9 Data compression and reduction 102
9.1 Data compression 102
9.1.1 Run-length encoding 103
9.1.2 Huffman coding 105
9.2 Data reduction 110
9.2.1 JPEG encoding 110
9.2.2 Audio 'compression' ideas 115
9.3 Chapter 9: take-home messages 117

10 Sensors, amplifiers and noise 118
10.1 Basic properties of sensors 118

10.2 Amplifier noise . . . 120
10.3 Specifiying amplifier noise . . . 124
10.4 Optimising signal-to-noise ratio . . . 126
10.5 Behaviour of cascaded amplifiers and transmission lines . . . 128
10.6 Chapter 10: take-home messages . . . 131

11 Signal averaging **132**
11.1 Measuring signals in the presence of noise . . . 132
11.2 Limitations of simple averaging . . . 133
11.3 Periodic signals . . . 139
11.4 Chapter 11: take-home messages . . . 141

12 Coherence, mixers and heterodyning **142**
12.1 Mixer diodes . . . 142
12.2 Photoconductive detectors as mixers . . . 145
12.3 Use of heterodyne systems with complex signals . . . 148
12.4 Conversion gain . . . 150
12.5 Noise temperature . . . 152
12.5.1 Measurement of receiver noise temperature . . . 154
12.6 Minimum detectable temperature . . . 155
12.7 Chapter 12: take-home messages . . . 157

13 Spectrum analysis **158**
13.1 Simple filter-based system . . . 158
13.2 Simple mixer-based system . . . 159
13.3 Chapter 13: take-home messages . . . 160

14 Amplitude modulation and demodulation **161**

14.1 Basics of modulation . . . 161
14.2 An amplitude modulation circuit . . . 161
14.3 The envelope detector . . . 163
14.4 Characteristics of AM waves . . . 166
14.5 Chapter 14: take-home messages . . . 169

15 Frequency modulation and demodulation 170

15.1 FM and PM . . . 170
15.2 The spectrum of an FM signal . . . 172
15.3 FM sources . . . 174
15.4 FM/PM demodulation . . . 175
15.5 Chapter 15: take-home messages . . . 181

16 Phase sensitive detection 183

16.1 Chapter 16: take-home messages . . . 190

17 Digital modulation 191

17.1 Simple binary modulation . . . 191
17.2 Power and noise . . . 194
17.3 Multi-bit symbols . . . 196
17.4 Orthogonal multiplexing . . . 197
17.5 Multipath . . . 200
17.6 Chapter 17: take-home messages . . . 201

18 Antennae, link-gain and radar 202

18.1 Basic properties of coherent antennae . . . 202
18.1.1 Gain and directionality . . . 205
18.1.2 Radiation resistance . . . 208

18.1.3 The small magnetic loop antenna 209
18.2 More types of antenna 211
18.2.1 Antennae with dimensions on the order of wavelength 211
18.2.2 Antennae with dimensions much greater than wavelength 216
18.2.3 The link gain equation 217
18.3 Radar 219
18.3.1 Pulsed radar 219
18.3.2 FMCW radar 222
18.3.3 Radar on beam-filling targets 224
18.4 Chapter 18: take-home messages 225

Chapter 1

Introduction

This course aims to provide an explanation of the basic concepts of information collecting and processing and the use of coherent techniques, with examples ranging from digital audio to frequency modulation. We will provide an introduction to basic information theory, look at what gets in the way of signals (i.e., noise) and discuss what we mean when we talk about coherent signals.

The course does not take a very philosophical approach to the idea of what information is, but it is useful to define what we mean by information, which is addressed in the following brief section.

1.1 What is information?

This is in fact not a very straightforward question to answer (as alluded to above, it can leads us somewhat into philosophy) so for our purposes we'll regard it as something that initially comes from some *sensor* or *transducer*. It is this detection or measurement that 'creates' information. In fact, the sensor is reacting to the arrival of some input of energy or power due to some process, so it would be fairer to say that the sensor 'picks up' the information due to this energy pattern, but we'll ignore this fact and treat the information as being generated by the sensor. To make things a little more general, we'll consider things like data storage devices to be transducers as well, in the sense that they can 'produce' information.

This is obviously rather non-rigorous compared to the mathematical involvements of information theory, but does provide us with an advantage over approaching information theory purely as a branch of mathematics. Firstly, it helps us to understand information processing systems in terms of the physical properties of the real world. Secondly, it allows us to sidestep the more metaphysical side of things relating to the 'meaning' of information and simply think about what can happen to it as it gets moved around.

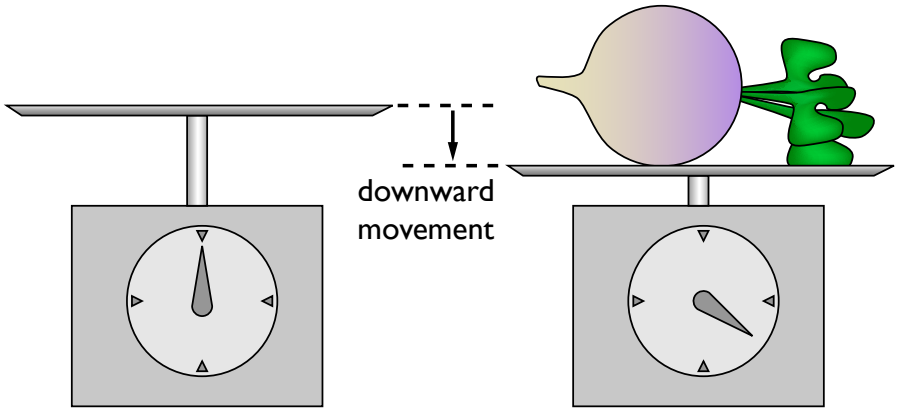


Figure 1.1: A simple kitchen scales as an example of a mechanical measurement system.

1.1.1 What is an 'instrument'?

At first glance, it can appear to some scientists that the sub-topic of *instrumentation* is obsessed merely with describing how voltmeters and oscilloscopes work. The subject, however, covers a far wider and more important area. Instrumentation for a scientific measurement, for example, may be a large set-up that allows us to 'poke' some physical system of interest in a controlled way then deal with the resulting response from the system and process the information contained in that response to understand processes going on. A voltmeter may form part of the overall setup, but instrumentation can cover a much broader picture than just the operation of some particular measurement device. In fact, we'll spend a lot more time considering general properties of signals than really getting into the nuts and bolts of how particular instruments work.

Many of the instrument and information examples we'll look at will be electronic or optical, simply because such methods are powerful, widely used and occur often in a physics context. It's important, however, to realise that the key points we'll make *aren't* only true in these areas. To emphasise this, the first example we'll look at is a simple mechanical system: a kitchen scales, as shown in figure 1.1. Odd as this may seem, it serves to make some fundamental points which apply to *all* measurement (information collecting) systems.

The scales has a pan or plate supported by a spring. When we place something on the pan the added weight presses down on the spring, compressing it. The pan moves downwards until the compression force from the squeezed spring balances the force of the increased weight. Our scales has a rotary dial with a pointer attached to the pan. The movement caused by the weight rotates the pointer to give us a reading of the weight.

The first point to note is that, like most measurement systems, this one is indirect. What we actually observe is a movement (rotation) of the pointer. We don't actually see the magnitude of the weight. If, for example, we put a turnip on the pan we might see the pointer move around through 120 degrees. If we liked, we could also use a ruler to find that the pan moved

down 2 cm. However, we don't usually quote weights in degrees or centimetres! In order to make sense of these observed values we have to *calibrate* the balance. To do this we can place two or three different known weights on the scales and make a note of how far the pointer goes around (or the pan falls) each time. We can then use these results to make a series of calibration marks on the face of the dial. Now, when we put our turnip on the scales we can read off its weight from the dial. This calibration process means that the scales provides us with a means to compare the weight of the turnip with a set of other 'standard' weights. In general, all measurements are comparisons with some defined standard.

Usually, we buy a kitchen balance which should already be calibrated (i.e. its dial is marked in kg, lb, etc, not degrees) and we don't bother to calibrate the weighing instrument for ourselves. However, when we consider the need for a calibration process an awkward question springs to mind - where did the known weights come from that were used to calibrate the readings? If all measurements are comparisons how were the values of those weights known? They, too, would need to have been weighed on some weight measurement system. If so, how was that system calibrated?

Any measurement we make is the last link in a chain of similar measurements. Each one calibrates a system or a *standard* (e.g. a known weight) which can be used for the next step. Right back at the beginning of this chain (at the National Physical Laboratory in the UK and other standards labs around the world) there will a primary reference system or standard which is used to define what we mean by 1 kg or 1 second or whatever. In effect, when we plonk something on the pan of a kitchen balance we're indirectly comparing it with the standard kg weight kept under a glass cover at the NPL.

When we place a turnip on the pan we have to wait a second or two to let the system to settle down and allow the pointer to stop moving. Similarly, when we remove the turnip the system will take a short time to recover. The second point we can make about the measurement system is, therefore, that it has a *finite response time* i.e. we have to wait for a specific time after any change in the weight before we can make a reliable reading. This limits our ability to measure any changes which take place too quickly for the system.

The third point to note is also an important one. If we put too large a weight on the pan the pointer will go right around and move *off scale*. (If the object is very heavy we may even break the scales!) No matter how well we search the shops, we can't find scales which can accurately measure any weight, no matter how big. Every real instrument is limited to operate over some finite range. Beyond this range it won't work properly and overloads or saturates to give a meaningless response.

The fourth and final basic point is something we won't usually notice using ordinary scales since the effect is relatively small. All of the atoms in the scales, including those in its spring, will be at room temperature. (We'll generally assume that room temperature is 293 K.) As a result, they'll be moving around with random thermal motions. Compared to the effect of placing a turnip on the scales these movements are quite small. But if we looked at the pointer very carefully with a powerful microscope we'd see its angle fluctuating randomly up and down a little bit because of the motions of the atoms in the spring. As a result, if we

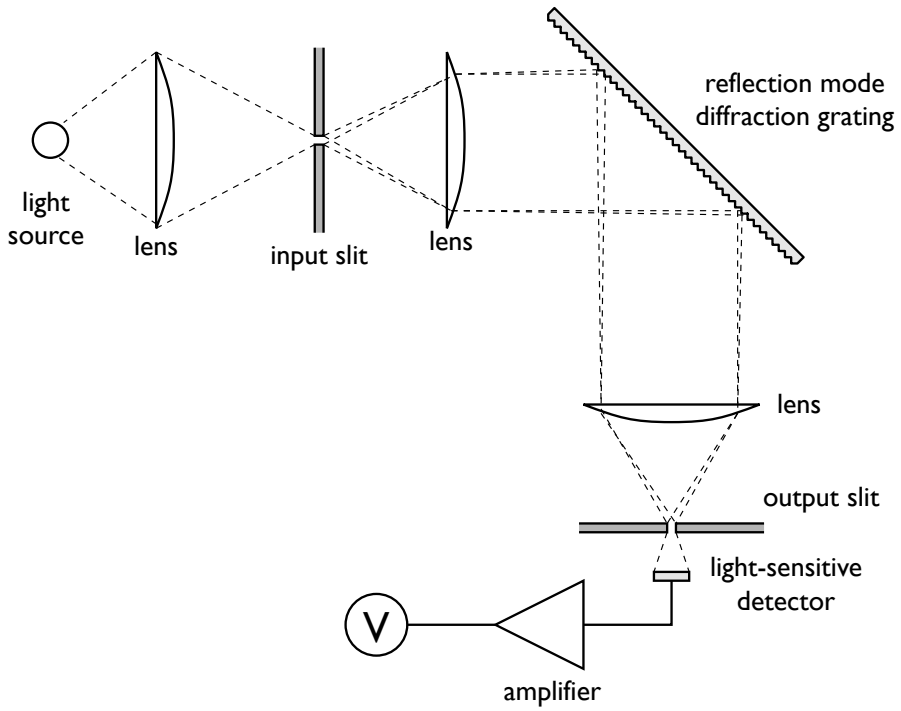


Figure 1.2: A simple reflection grating spectrometer.

wanted to measure the weight very accurately this thermal jittering would limit the precision of our reading. As a result, no matter how good the scales, our ability to make extremely accurate measurements is limited by thermal random effects or thermal noise.

In reality, you will *never* be limited by thermal noise when weighing turnips, but when measuring electrical signals, you may be limited in how precisely you can measure by thermally-generated electrical noise, or by other random processes such as shot noise.

1.2 Accuracy and resolution

It's important to realise that the amount of information we can collect is always finite. The example of kitchen scales has introduced us to the limiting effects of clipping, noise, and response time. It doesn't matter how clever we are, these problems occur in all physical systems since they are consequences of the way the real world works. To see some of the other problems which arise when we're collecting information, consider the system in figure 1.2. This diagram represents a diffraction grating being used to measure the spectrum produced by a light source, i.e. how much energy (or power) the light source produces as a function of wavelength.

The system is intended to provide us with information about how bright the light source is at

various light wavelengths. It relies upon the reflection properties of a surface patterned with a series of parallel ridges called a *reflection grating*. For an ordinary plane mirror, the angle of reflection equals the angle of incidence. For a grating, the angle of reflection also depends upon the *wavelength* of the light and the details of the grooved surface pattern. Hence the arrangement shown acts as a sort of adjustable filter. Only those light wavelengths which reflect at the appropriate angle will make their way through the output slit onto the detector.

As with the kitchen scales, the system provides an indirect way to measure the light's spectrum. We use the angle of the diffraction grating to tell us the wavelength being observed. The voltage displayed on the meter indicates the light power falling on the detector. To discover the light's spectrum we slowly rotate the grating (or move the output lens/slit/detector) and note how the voltmeter reading varies with the grating angle. To convert these angles and voltages into wavelengths and light powers we then need to know the sensitivity of the detector/amplifier system and the angles at which various wavelengths would be reflected by the grating. i.e. the system must be **calibrated**.

In most cases the instrument will be supplied with appropriate display scales. The voltmeter will have a dial marked in units of light power, not volts. The grating angle display will be marked in wavelengths, not degrees. These scales will have been produced by a *calibration process*. If the measurement we're making are important it will probably be sensible to check this calibration by making measurements on a *known* light source.

As with the kitchen scales, our ability to measure small changes in the light level will be limited by *random noise*: in this case random movements of the electrons in the measurement system and fluctuations in the rate at which photons strike the detector. The accuracy of the power measurement will depend upon the ratio of the light power level hitting the detector to the random noise. We could increase the light level and improve the precision of the power measurement by widening the slits and allowing more light through. However, this would have the disadvantage of allowing light reflected over a wider range of angles to reach the detector. Since the angle of reflection depends upon the light wavelength this means we are allowing through a wider range of wavelengths.

In fact, looking at the system we can see that it always allows through a range of wavelengths. Unless the slits are narrowed down to nothing (cutting off all the light!) it will always allow light reflected over some range of angles, $\Delta\theta$, (and hence having a range of wavelengths, $\Delta\lambda$) to get through. As a result there is an **unavoidable** trade off between the instrument's power sensitivity and its wavelength resolution or ability to distinguish variations in power confined to a narrow wavelength interval. This kind of trade off is very common in information collection systems. It stems from basic properties of the physical world and means that the amount of information we can collect is always finite i.e. we can never make perfect measurements with absolute accuracy or precision or certainty.

1.3 Chapter 1: Take-home messages

Measurement systems usually provide an indirect indication of whatever quantity is being measured, and that all measurements are comparisons which require some kind of calibration.

The amount of information we can collect is always *finite*, and will be limited by the effects of noise, saturation and response time. Information gathering involves a trade-off between various quantities. Such limitations may stem from the properties of the physical world, rather than just poor instrument design.

Chapter 2

Sampling

If you're doing much with experimental data in physics these days, then you are almost certainly doing it with a computer, and if you're modelling complex systems that cannot be addressed analytically[1], then you are *definitely* doing it with a computer. This means that the information that you're working with will have certain characteristics not possessed by the behaviour of the physical quantity that you are measuring. In this chapter we will look at the issues associated with **sampling** a signal.

2.1 Describing signals: the time and frequency domains

In this course, we will often be talking about signals that are a function of *time*. As a physicist dealing with data for lab instruments, for example, this is how you will most likely encounter them. So one way to describe a signal or a message is by listing its value as a function of time. This is the approach taken in figure 2.1 (a), which shows a periodic signal I generated by throwing together a handful of sinusoids (which is why it's periodic).

An alternative way to describe signals is in the *frequency domain*, rather than the time domain. Figure 2.1 (b) shows the amplitudes of the sinusoids in the same signal as a function of their frequency. Such a representation is often referred to as an amplitude spectrum. The notion of the **spectrum** of a signal is an extremely important one: in generic terms, it describes a signal in terms of how much of it there is at different frequencies.

For this particular signal, there are immediate advantages to looking at its spectrum rather than at the time-domain record. As it is in fact composed of a small number of sinusoids, its spectrum just has a handful of peaks, which are well defined. Please note that in this case, the spectrum shown is not complete, as it only shows the *amplitudes* of the different frequency components, and doesn't provide any information about the *phase* of each component.

[1] As far as I'm concerned, this includes anything involving integrals...

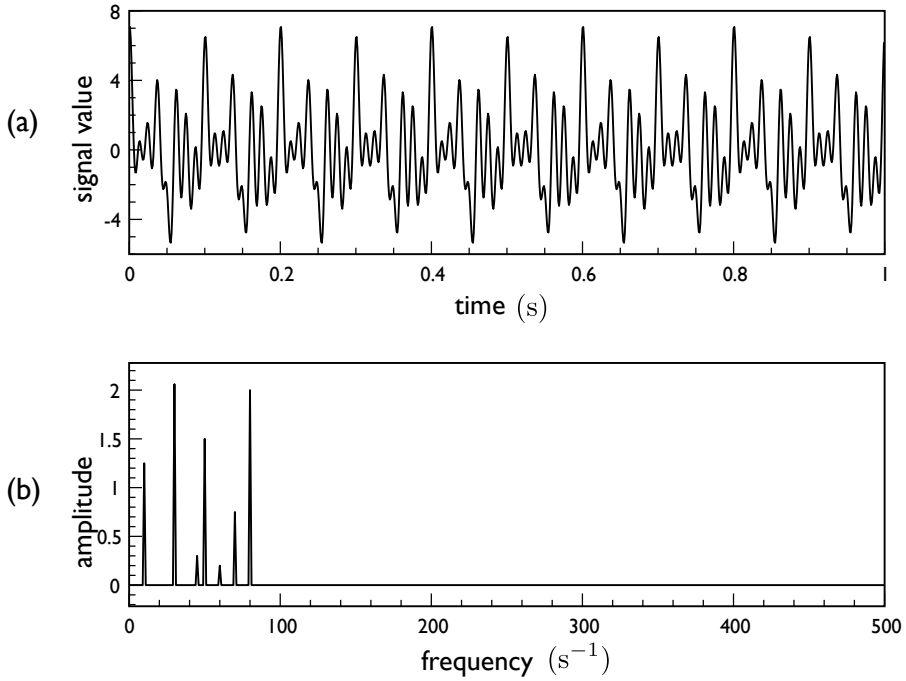


Figure 2.1: (a) A signal composed of a small number of sinusoids. (b) Magnitude amplitude spectrum of the signal in (a).

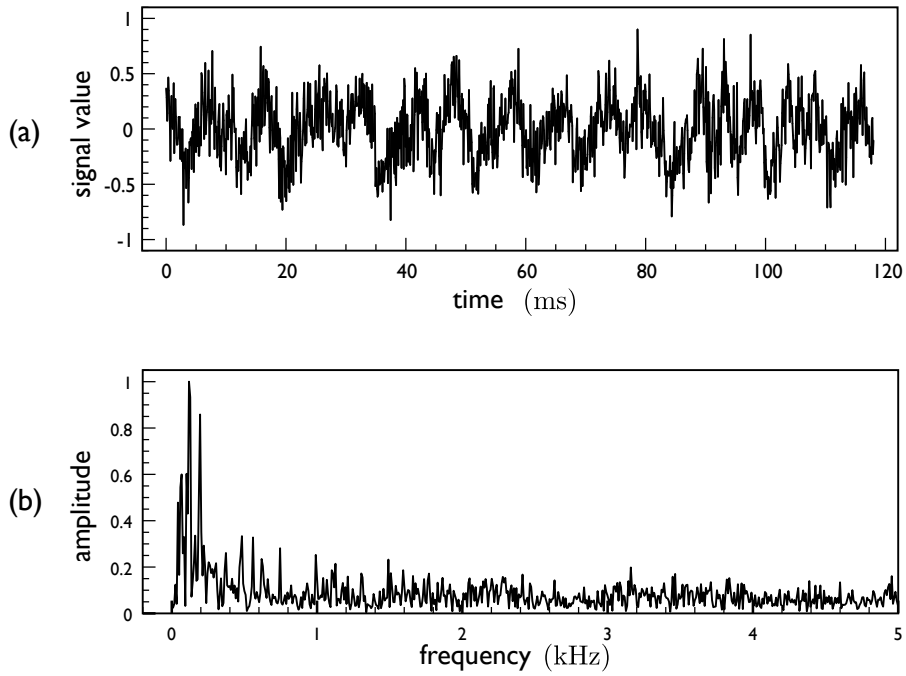


Figure 2.2: (a) A more complex signal, taken from the start of a song featuring heavily-distorted amplified instruments. (b) The magnitude amplitude spectrum (up to 5 kHz) of the signal in (a).

This example clearly shows that (at least in this case), the spectrum provides a simpler picture of the signal than the time domain version. It is, however, composed only of a few sinusoids. It will not come as a huge surprise that more complex signals will have more complex spectra.

Note the units used for the horizontal axis in figure 2.1 (b): I've deliberately used inverse seconds rather than hertz to emphasise that if a signal is a function of a particular dimension, then the frequency has the units of the inverse of that dimension.

Most of the time[2], we'll be talking about signals which are a function of time, but frequency representations apply to other systems as well. We will encounter the notion of spatial frequencies later, and you'll also spots lots of similarities between the ideas in this course and k-space in statistical or solid state physics. The arguments we'll make and conclusions we'll come to apply to all types of signals, even though most of the time I'll pretend that we're looking at voltages that change as functions of time.

Figure 2.2 shows the first tenth of a second (ish) of a song featuring heavily-distorted amplified instruments, along with the magnitude amplitude spectrum. In fact, the spectrum for this audio snippet extends up to even higher frequencies.

The analysis of signals by their frequency content is a useful and important tool, and we will frequently switch between these two pictures, making use of the one which best matches the context and purpose. You have already met the principal tool for making that jump, the Fourier transform. We will use a close cousin of that idea, but we'll deal with it in a later chapter: it's just worth nailing down the whole frequency/time thing before we get into a discussion of sampling.

2.2 Quantisation and sampling

Consider a signal which is the output voltage produced by some hypothetical experiment, which we wish to record and do something with it. I'm going to stick with the "signal-as-a-function-of-time-thing", but the arguments I'll make are general ones. In general, the signal produced by a sensor will be **continuous in voltage**: it could take any value within limits determined by the physical process that's producing it. It will also generally be **continuous in time**, i.e. it will exist at all instants of time between the start and the end of the experiment.

In order to get an analogue signal into a computer to do something with it, we need to **sample** it, i.e. measure its value at consecutive instants of time. The instrument that does this is an **analogue-to-digital converter** or **ADC**. There are two limitations that come in to play here:

- A given ADC will always be limited in how quickly it can take measurements, i.e., successive measurements (samples) will be spaced by a finite (i.e., non-zero) interval

[2] Yup, it's a pun

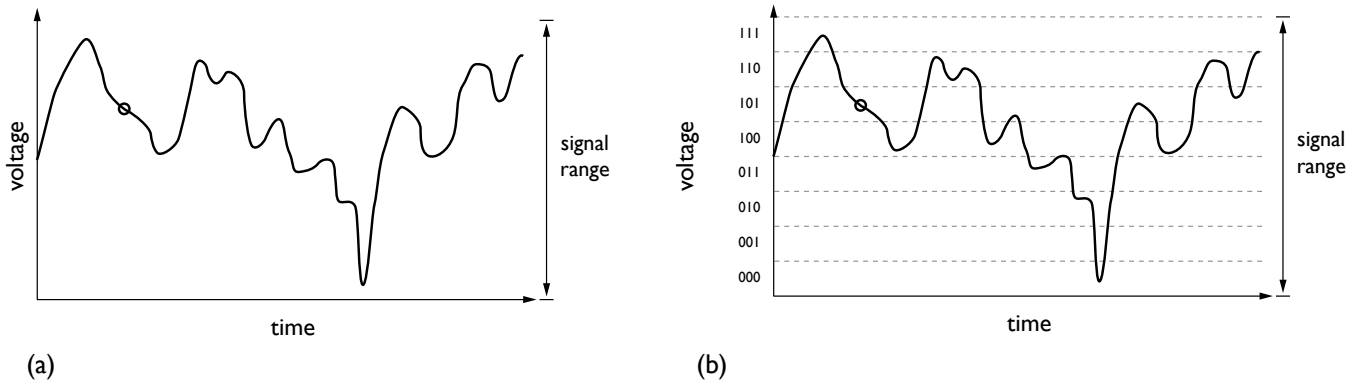


Figure 2.3: (a) A continuously varying signal voltage. (b) The same signal, with the voltage range divided up into smaller intervals with a symbol assigned to each interval.

of time. This means that our set of samples is no longer a continuous signal, it is now a **discrete** signal. The sampled version of the signal is only defined at the instants of time at which it has been sampled.

- A given ADC will only be able to measure a signal within some specified range. More importantly, it can only measure with a specified precision, i.e., once we choose an ADC, we can't measure with arbitrary resolution. This has the effect of **quantising** the signal. The measurements won't tells us exactly what the signal value at a given time is, only which of a number of voltage levels it was closest to. This quantisation is what makes digital signals digital, rather than analogue.

In real life, the quantisation of the original analogue signal and the discretisation[3] effectively occur together, i.e. the ADC does both at the same time.[4] We will, however, examine the effects of these two changes to the signal separately, as the issues that they give rise to are actually quite distinct. We will treat the ADC as a 'black box': we're more interested in *what* it does than *how* it does it.

2.3 Quantisation

We'll deal with the quantisation first. Consider figure 2.3 (a): this shows a section of a graph of some signal, in this case a voltage, which is varying as a function of time. It could be the output of some sensor, or a musical instrument, but it doesn't matter where it comes from or what it represents: what matters is that it's the details of the signal pattern, i.e., how it varies with time, that contains the information that we care about.

[3] This is actually a word.

[4] Inside the ADC itself, this statement isn't really true, but if we regard an ADC as a black box that spits out numbers, then it is.

To sample the voltage at a particular instant of time (the circle on the graph), we first need to decide on the **range** of input signal that we need to accept. The range that we specify needs to be large enough to accommodate any possible value of the signal. We can then look at the point on the waveform, and ask ourselves the question, "is the signal value here in the top half of the range?" If the answer is "yes", then we write down a '1'. If the answer is no, we write down a '0'. Then we define a new range consisting of the half of the original range that our value lies in, and ask the same question to get another yes/no answer, and write down another '1' or '0' accordingly. This gives us a two-digit number telling us in which *quarter* of the original range our signal value resides. In principle this process could repeat as often as we'd like. Figure 2.3 (b) shows the result for our circled value if we do this three times, which has divided the original range into eight ($= 2^3$) equally spaced intervals.

Note that if the signal moves out of our initially chosen range, we won't know what its actual value is, only that it's greater or less than the maximum or minimum value that we can record. When this happens, the signal is said to be **clipped**, and we have *lost* some information, which *cannot be recovered*. We are left with a signal that has suffered **distortion** as a result of the clipping.

2.3.1 How much information does a single value contain?

In the example discussed above and in figure 2.3, we asked three yes/no questions about the value of our signal at the chosen point. A yes/no question is essentially the simplest question that we can ask about something. Each answer in the process yielded a single digit, either 1 or 0, so in this case each question gave us an additional quantum of information. We call this quantum a **bit**, which is a contraction of 'binary digit'.[5]

The result of the successive narrowing down of where the signal values lies in the range yields a string of these bits, which forms a binary number which is proportional to the signal value. If we do the yes/no thing n times, then we'll get an n-bit binary number, which tells us which 2^n-th of the signal range that the value lies in. There are 2^n possible binary numbers, so we'd require 2^n distinct **symbols** to convey the information, assuming that over enough measurements the signal varied over the whole range. If we pick a value for n (more discussion on how to sensibly do this will follow!), then this means that we will have a finite set of discrete values that our now quantised signal can take, so rather than having to potentially use an infinite number of symbols to represent the infinite number of values that a continuous-in-voltage signal could take, we can use a finite set of symbols as an 'alphabet' to describe our signal or message.

Note that the number of symbols we'd require for our alphabet depends only on the precision with which we measure sample values, and doesn't tell us anything about how much resolution a particular case actually needs.

[5] It's worth noting that bit has two related, but actually separate meanings; in some contexts it beans a digit in a binary number, and in others it means a unit quantity of information.

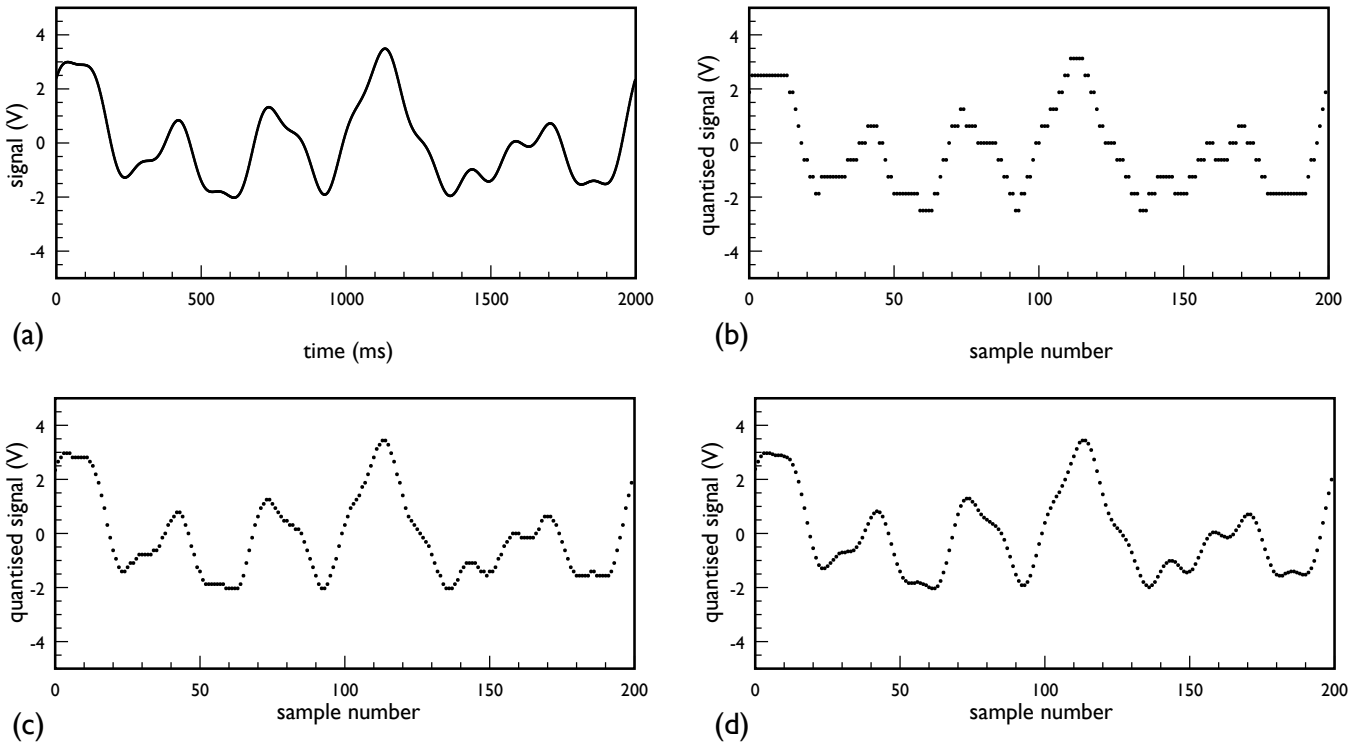


Figure 2.4: (a) A continuously varying signal voltage. (b) The same signal, but as it would appear if sampled by an ADC with a full range of ± 5 V, using 4 bits per sample. Note that the horizontal axis is labelled with the sample number, rather than time. (c) As (b), but with a six bits per sample. (d) as (c), but with 8 bits per sample.

2.3.2 The required bit-depth

The more bits we use for each sample, the greater the resolution of our measurement as the more symbols we have in our alphabet that we can use to describe our signal or message. If we use too few bits, then the quantised version of our signal is clearly distorted with respect to the original. Figure 2.4 (a) shows a signal voltage as a function of time. Part (b) shows the quantised signal we would have if we took samples with 4 bits per sample, with a voltage range of ± 5 V and the distortion to to quantisation is clearly visible. The results of using 6 and 8 bits per sample are shown in parts (c) and (d). I've deliberately *not* joined the dots in the graphs for the sampled and quantised signals, as a sampled signal is only defined at the times that it is sampled.

With 4 bits per sample, the distortion is very obvious, and the gaps between the quantisation levels are easy to see. This is less obvious when we move to 6 bits per sample (with the voltage interval now a factor of four smaller), but still visible. With 8 bits per sample, the quantisation is less obvious, but if the figure were enlarged, then it would be possible to see it. Note that the portions of the signal where it is changing slowly are those most visibly affected by the quantisation process.

2.3.3 Dynamic range and signal-to-noise

The number of bits per sample is often referred to as the **bit-depth** of the quantisation process. If we had enough money, we could buy an ADC that gave out values with maybe any bit depth that we wanted. More resolution gives us more information, right? This is true up to a point. However, any real signal will *always* be measured along with some random **noise**.

The noise determines the size of the smallest signal detail that we can expect to reliably observe: we can sample a signal as finely (with as many bits) as we want, but if there is a lot of noise on the signal then there becomes a point where adding more bits per sample doesn't really tell us any more as the random variations due to the noise will be larger than the resolution that extra bits will provide. There will also be a limit on how large a voltage can be transmitted without exceeding the range of the ADC (i.e., **'clipping'**).

For the sake of argument, assume that the channel by which our signal arrives has a noise level of around 0.2 mV and can handle a voltage range of ± 5 V. Ideally, the range of the ADC, V_{range} should equal that of the input channel (± 5 V in this case). An n-bit ADC could then determine the signal level at any instant with an accuracy of $V_{\mathrm{range}}/2^n$. For example, an 8-bit ADC would divide the 10 V range into $2^8 = 256$ intervals, each $10/2^8 = 0.039$ V wide, and determine which of those intervals the input was in at any instant. Adding two bits to the resolution will quadruple the number of voltage intervals, which would then be 9.7 mV wide. However, we *wouldn't obtain any extra information about the signal* as the interval the ADC could determine the voltage to within is already *way* less than the noise voltage level. Determining the voltage level more accurately than the noise level *yields no further information about the signal, it just tells us more about the characteristics of the noise*.[6]

The above discussion is illustrated in figure 2.5. This is a re-tread of figure 2.4, but this time the signal has had random noise of 0.2 V (rms) added to it. Again it shows the effect of quantising the signal with increasing bit-depth, but the presence of the noise means that (qualitatively) the 8-bit version (in which $\Delta V = 39$ mV) doesn't look terribly different to the 6-bit version (in which $\Delta V = 156$ mV). This shouldn't be a huge surprise, as with 6 bits per sample, we already can resolve a voltage interval which is smaller than the rms noise voltage.

This effect arises because the input signal will always have a finite **dynamic range**. The dynamic range is the ratio of the maximum possible signal size to the minimum detail detectable over the random noise. It is defined as a *power* ratio, of the maximum possible signal power to the mean noise power, and is usually quoted in decibels, or dB.

$$D_{\mathrm{dB}} = 10 \log_{10} \left(\frac{P_{\mathrm{max}}}{P_{\mathrm{min}}} \right) = 10 \log_{10} \left(\frac{V_{\mathrm{max}}^2}{V_{\mathrm{N}}^2} \right) \tag{2.1}$$

where V_{max} and V_{N} represent the rms value of the maximum signal voltage and the rms noise voltage respectively. The dynamic range should be distinguished from the actual **signal to**

[6] Of course, if we were using an ADC to try and characterise the noise on the signal as well as the signal itself, then the more bits the merrier...

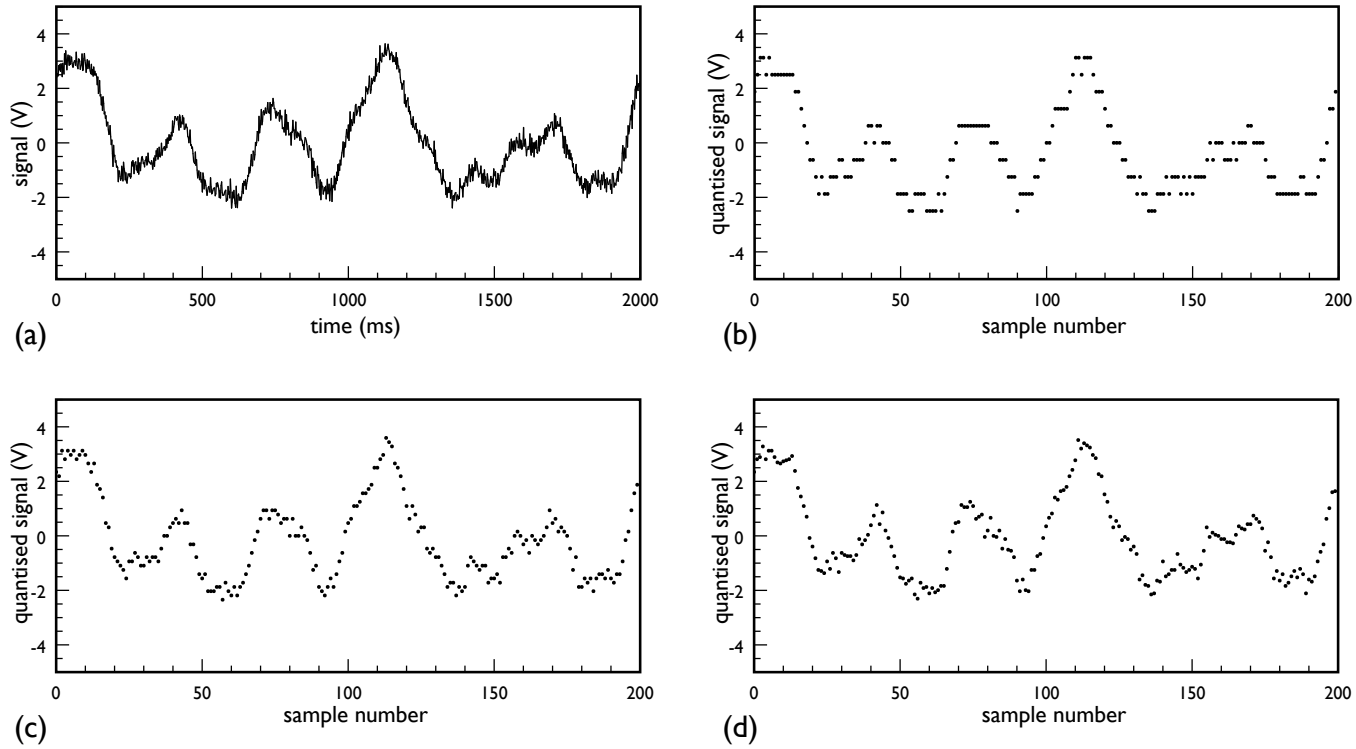


Figure 2.5: (a) The same continuously varying signal voltage presented in figure 2.4, but with random noise of 0.2 V (rms) added to it. (b) The same signal, but as it would appear if sampled by an ADC with a full range of ± 5 V, using 4 bits per sample. Note that the horizontal axis is labelled with the sample number, rather than time. (c) As (b), but with a six bits per sample. (d) as (c), but with 8 bits per sample.

noise ratio,

$$\mathrm{SNR}_{\mathrm{dB}} = 10 \log_{10} \frac{P_{\mathrm{S}}}{P_{\mathrm{N}}} \tag{2.2}$$

where P_S is the actual signal power, and P_N is the noise power: usually, $P_S < P_{\max}$ by definition. In the course, we will always use the convention that SNR refers to the ratio of the signal and noise *powers*, and *NOT* to the ratio of *amplitudes*. Remember, the power in a signal (or other types of wave) is proportional to the *square of the amplitude*.

2.4 Discretisation

As well as quantising a signal, our ADC will also convert it into a signal that is *discrete in time*. This means that, we can't pretend that our sampled signal is continuous. The act of sampling it at a series of particular instances means that what we're left with (the samples) is necessarily a record of what the signal is doing at a set of **discrete** times.

This distinction may seem like a pedantic one, but it's very important, as the mathematics of describing discrete signals differs from that used for continuous signals. In a signals context, we are often dealing with discrete signals, as it is common for information to be generated continuously by some sensor or transducer, and then be sampled by some analogue-to-digital conversion process.

In this course, the distinction between continuous and discrete signals will (as is common in the literature) be made by the brackets used: a continuous signal will be denoted by $x(t)$ and a discrete signal by $x[n]$, where n refers to the *sample number*. (This is why the sampled and quantised signals in figures 2.5 and 2.5 have sample number rather than time for their horizontal axes. We're going to assume that we're always dealing with sampled signals in which the samples are taken at equally spaced instants in time, so it is then trivial to relate the sample number to time (or position, or whatever it is that the signal is a function of.)

In this section, we'll address the issues that arise from discretisation, and sensible guidelines on how often we need to take samples if we want a decent picture of what our signal is doing.

2.4.1 Aliasing

Consider a continuous sinusoidal signal, $x(t) = \cos(2\pi f t)$. How often should we take samples?

Let's see what happens if we sample at a constant **sampling rate** of f_s samples per second (often abbreviated as $\mathrm{Sa\ s^{-1}}$. (We will always assume that our sampling rate is constant, i.e. the separation in time between adjacent samples is always the same.) If the sampling rate is f_s, then the time between samples will be

$$t_s = \frac{1}{f_s} \tag{2.3}$$

We will assume that we start measuring, i.e., take our first sample, at $t = 0$ s, so the time at which the n-th sample is taken will be nt_s and the value of the nth sample of our simple cosine will be

$$x[n] = \cos(2\pi f n t_s) \tag{2.4}$$

(I'm using square brackets to remind myself that our samples are only defined for integer values of n.) In this example, we know what the signal that we are sampling *is*. However, trig functions are periodic in 2π, so

$$x[n] = \cos(2\pi f n t_s) = \cos(2\pi f n t_s + 2\pi m) \tag{2.5}$$

where m is any integer. Following the treatment by Lyons[7], we'll let m be an integer multiple k of n, i.e. $m = kn$ and rearrange equation 2.6 a bit:

$$\begin{aligned} x[n] &= \cos(2\pi f n t_s + 2\pi m) \\ &= \cos\left(2\pi \left[f + \frac{k}{t_s}\right] n t_s\right) \\ &= \cos\left(2\pi \left[f + k f_s\right] n t_s\right) \end{aligned} \tag{2.6}$$

Thus, when we sample a signal at a rate of f_s Sa/s, w.e cannot tell the difference between a sinusoid at a frequency of f and a sinusoid at a frequency of $f + kf_s$, where k is *any* integer.

This is *really* important, and this phenomenon is referred to as **aliasing**. Figure 2.6 shows what would happen if we sampled a 2 Hz sinusoid at a sampling rate of 10 Sa/s. The samples produced are exactly the same as if the sinusoid being sampled had a frequency of 12 Hz ($f + f_s$) or -8 Hz ($f - f_s$). Overlapping sinewaves get messy pretty quick which is why there are only two curves per graph, but if we plotted sinusoids of $f \pm kf_s$ for any integer k, we'd get *exactly the same set of sample values*. Of course, these sinusoids are all behaving differently *between* the samples, but once we have a set of samples, then that is *all* that we have: we don't know what was going on in between.

Figure 2.6 also shows what would happen if we sampled 12 or -8 Hz signals at 10 Sa/s; we get exactly the same set of samples as if we sampled the 2 Hz wave. *If you have no prior knowledge of the signal, then you can't be certain of the frequency of what you're looking at!*

Aliasing means that you should never just join-the-dots and draw lines between a set of sampled values, as without some other information about the signal, you can't know that it's not doing something interesting between the sample values that you have.

[7] Understanding Digital Signal Processing, 3rd ed., Richard G. Lyons, Pearson, 2012

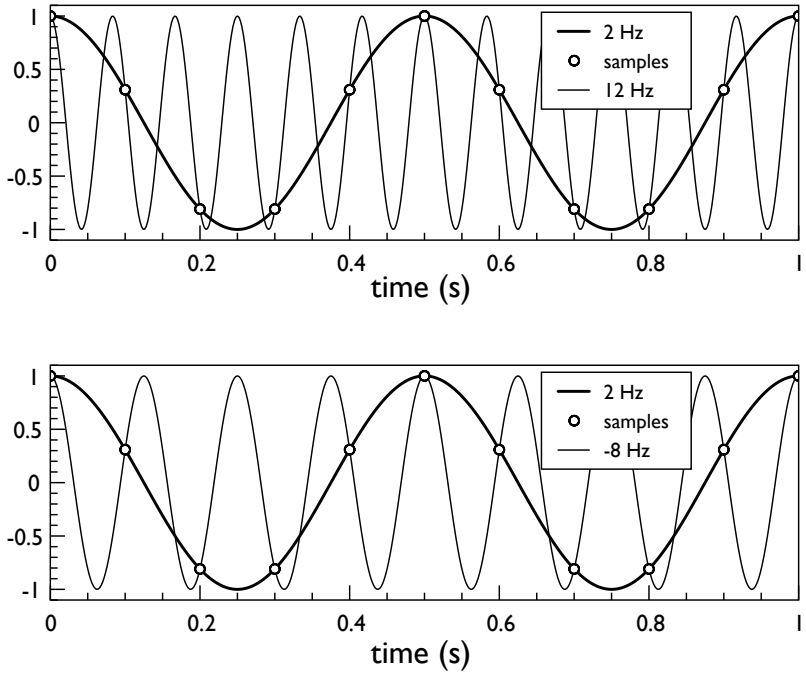


Figure 2.6: Aliasing in the time domain: a 2 Hz cosine (thick line) sampled at 10 Sa/s (circles), along with 12 Hz ($f + f_s$) and -8 Hz ($f - f_s$) cosines (thin lines).

2.4.2 Aliasing and spectral folding

Consider a signal the frequency of which is $f_s/2 < f < f_s$, and $f = f_s/2 + \Delta f$. To simplify matters we'll assume it's a cosine with zero phase constant:

$$x(t) = \cos\left(2\pi\left(\frac{f_s}{2} + \Delta f\right)t\right) \tag{2.7}$$

If we sample this at a rate of f_s, then we'll get sampled values

$$x[n] = \cos\left(2\pi\left(\frac{f_s}{2} + \Delta f\right)nt_s\right) \tag{2.8}$$

As the cosine function is symmetric about $t = 0$, we would get exactly the same set of samples if the frequency was equal to $-f$, i.e.,

$$\begin{aligned} x[n] &= \cos\left(2\pi\left(\frac{f_s}{2} + \Delta f\right)nt_s\right) \\ &= \cos\left(2\pi\left(\frac{-f_s}{2} - \Delta f\right)nt_s\right) \end{aligned} \tag{2.9}$$

Now consider the conclusion we were led to by equation 2.6: this tells us that we'd get the same set of samples if we were actually looking at a wave whose frequency differs from f by

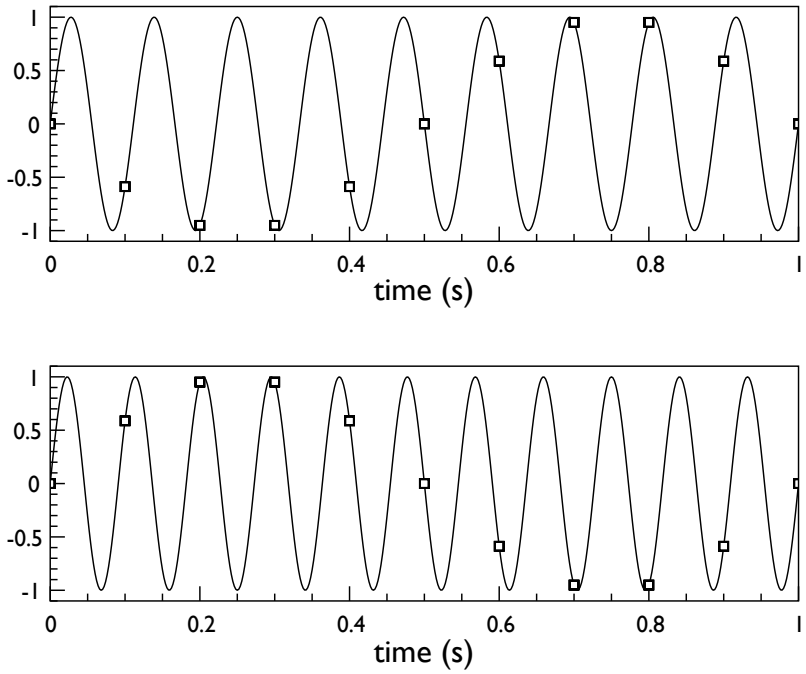


Figure 2.7: Aliasing in the time domain: samples (squares) obtained when sampling 9 Hz and 11 Hz sine waves (lines) at 10 Sa/s.

any multiple of f_s, so

$$\begin{aligned} x[n] &= \cos\left(2\pi\left(\frac{f_s}{2}+\Delta f\right)nt_s\right) \\ &= \cos\left(2\pi\left(\frac{-f_s}{2}-\Delta f\right)nt_s\right) \\ &= \cos\left(2\pi\left(\frac{-f_s}{2}-\Delta f+f_s\right)nt_s\right) \\ &= \cos\left(2\pi\left(\frac{f_s}{2}-\Delta f\right)nt_s\right) \end{aligned} \tag{2.10}$$

What's the point of this? Well, if we have a set of samples and we *do* just join the dots between them with straight lines, then we we get will look like the lowest possible frequency that goes through those points. Equation 2.10 tells us that when we sample at a rate of f_s, then the samples we'd get from a sinusoid of frequency $f_s/2+\Delta f$ will be the same as those that we'd get from a sinusoid of frequency $f_s/2-\Delta f$, which is of a lower frequency.

Note that although equation 2.7 defines a pure cosine, the argument applies to any sinusoid, although if it's phase is *not* zero, then the phase of the lower frequency alias will be different.

Figure 2.7 shows the samples that one would obtain when sampling 9 Hz and 11 Hz sine waves with a sampling rate of 10 Hz. In both cases, simply joining the dots between the samples with straight lines would result in what looked like a sine wave with a frequency of

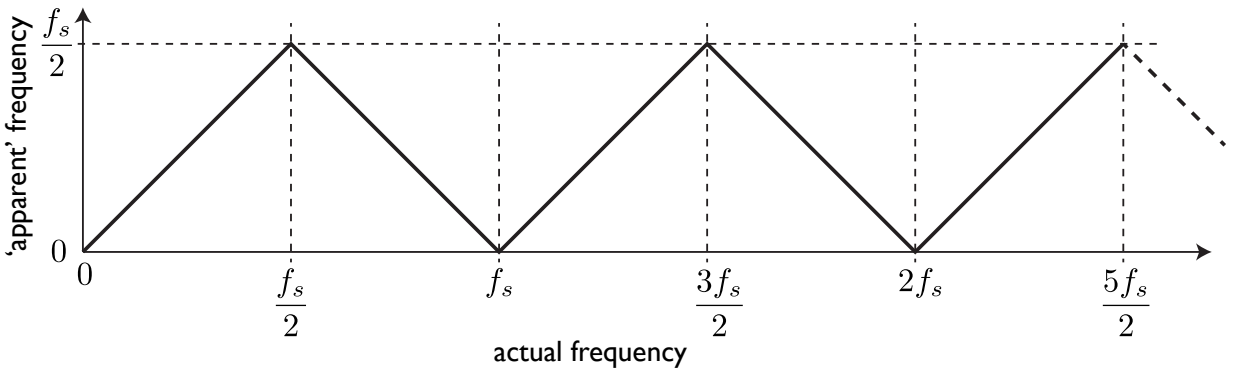


Figure 2.8: Spectral folding: the graph illustrates what frequency a sinusoid would appear to have as a function of it's actual frequency, when sampled at a rate of f_s samples/second.

$|f_s/2 - f| = 1$ Hz, but the phase is inverted between the two cases.

When we are sampling a signal at a rate of f_s, then any frequency components *above* $f_s/2$ will not be trivially recognisable, i.e., with a join-the-dots approach. A frequency component Δf above $f_s/2$ will appear to be 'reflected' or *folded* back down in frequency about $f_s/2$, which is often referred to (amongst its other names) as the *folding frequency*.

This is illustrated in figure 2.8.

Whereas figure 2.6 and 2.7 illustrate aliasing in the time domain, figure 2.9 shows it in the *frequency domain*: the solid line in that figure represents the power spectrum of a signal that we wish to sample. Let's say the sampling is done at a rate of f_s sample per second. One of the effects of discretisation is that the resulting spectrum becomes *periodic*, with copies of the 'actual' spectrum being duplicated all along the frequency axis at intervals of f_s. What this means is that a signal with a spectrum consisting of one of the dashed copies produces the same set of sample values as the signal whose spectrum is shown by the solid line.

In the particular case of figure 2.9, f_s is less than twice the highest frequency that the signal contains, so the duplicates of the spectrum overlap with each other.

When we are sampling a signal, it is important to make sure that we sample often enough in order to avoid aliasing. It is common for systems that convert analogue information to digital to precede the ADC itself by an *anti-aliasing filter*: an analogue device which does not pass frequencies above half the sampling rate of the ADC.

2.5 The sampling theorem

It's worth introducing here formal statements of a very important principle in information theory and signal processing: the **sampling theorem**. You'll find this phrased in slightly different

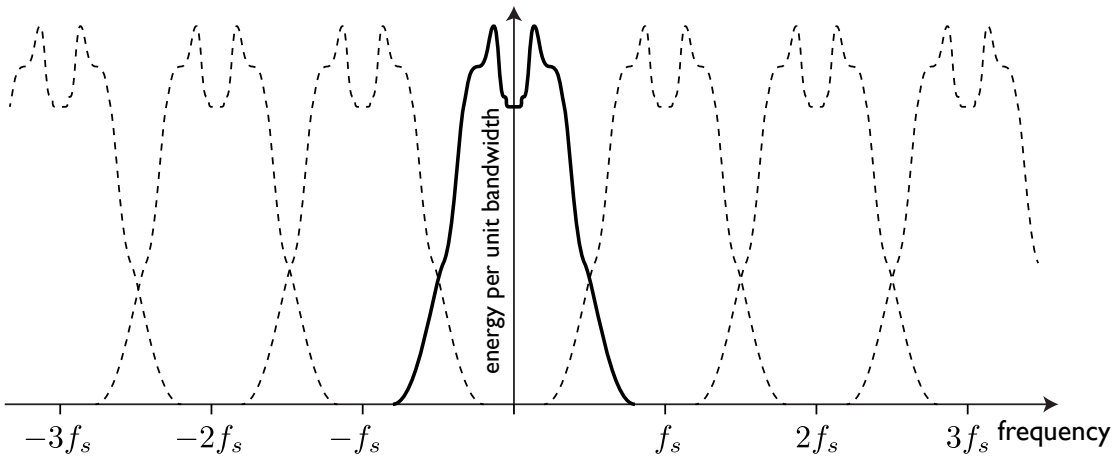


Figure 2.9: Aliasing in the frequency domain: the thick solid line in the middle represents the power spectrum of a signal that we are sampling at a rate of f_s samples/second.

ways in various sources, so I'll use a direct quote from Claude Shannon, who was instrumental in the development of information theory: "If a function $f(t)$ contains no frequencies higher than W cps, it is completely determined by giving its ordinates at a series of points spaced $1/2W$ seconds apart."[8] Note that 'cps' stands for cycles per second.[9] Additionally, quoting Shannon again (from the same paper): "A similar result is true if the band W does not start at zero frequency but at some higher value", so it's the *range* of frequencies that matter, rather than just the maximum frequency.

Signals which have a frequency content which is only non-zero within some finite range, and zero outside that, are said to be **band limited**. There is a minor hiccup here, however: and signal which is *truly* band-limited must have frequency components that extend over an infinite range.

If we combine the sampling theorem with the notion that the bit-depth at which it makes sense to sample a signal is limited by the amount of noise on that signal, then we can see that an analogue signal with a finite bandwidth, finite SNR and finite duration contains only a finite amount of information.

2.6 Chapter 2: Take-home messages

Signals that are functions of time (or space) also have accompanying representations as a function of (spatial) frequency. Such a representation is called the **spectrum** of the signal. Depending on the problem at hand, one of these representations may have advantages over

[8] Shannon, Claude E., "Communication in the Presence of Noise", Proc. IRE **37** (1), 1949.
[9] The Hertz was only adopted as an SI unit in 1960.

the other.

An analogue (continuously varying) signal can be **sampled** resulting in a **discrete** signal. This can be done in such a way as to recover *all* the information it contains. The **sampling theorem** theorem sets the fundamental limit on how often we need to sample the signal, and if we do not sample often enough, we will run into the problem of **aliasing**.

The amount of information that a signal can convey is *finite*, limited by the allowable range of signal level, the amount of **noise** and its frequency range or **bandwidth**. Signals that occupy a finite frequency range are said to be **band-limited**.

Chapter 3

Noise

Whenever we try to make accurate measurements we discover that the quantities we are observing appear to fluctuate randomly by a small amount. This limits our ability to make quick, accurate measurements and ensures that the amount of information we can collect or communicate is always finite. These random fluctuations are called **noise**. A common question when designing or using information systems is, "Can we do any better?" In some cases it's possible to improve a system by choosing a better design or using it in a different way. In other cases we're up against fundamental limits set by unavoidable noise effects. To decide whether its worth trying to build a better system we need to understand how noise arises and behaves. Here we will concentrate on electronic examples, but bear in mind that similar results arise when we consider information carried in other ways (e.g. by photons in optical systems).

3.1 Johnson noise

In 1927 John. B. Johnson observed random fluctuations in the voltages across resistors[1]. A year later Harry Nyquist published a theoretical analysis of this noise which is thermal in origin[2]. This type of noise is thus variously called *Johnson* noise, *Nyquist* noise or *thermal noise*.

A resistor consists of a piece of electrically conducting material, the geometry and properties of which have been adjusted to make it a less-than-perfect conductor. We can treat it as a 'box' of material containing conduction electrons which move around, interacting with each other and the atoms of the material. At any non-zero temperature we can think of the electrons as an electron gas in the resistor. The electrons move in a random manner, similar to Brownian motion, scattering off each other and the atoms in the resistor. At any particular

[1] Johnson, J. B., "Thermal agitation of electricity in conductors", Phys. Rev. **32**, p97, 1928.
[2] Nyquist, H., "Thermal agitation of electric charge in conductors", Phys. Rev. **32**, p110, 1928.

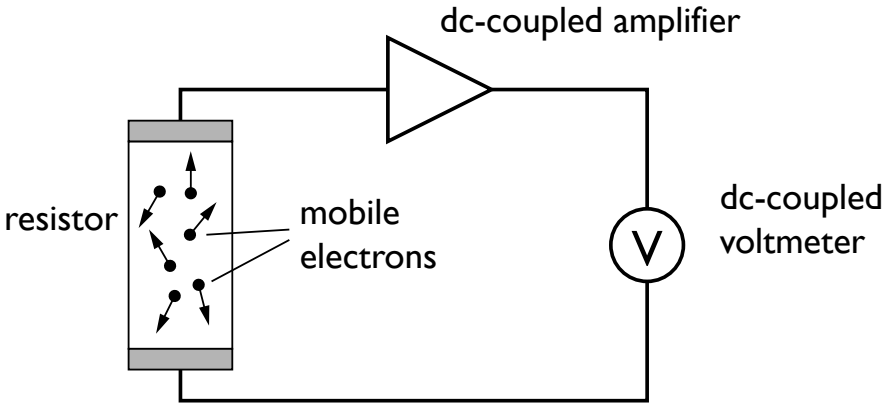


Figure 3.1: Measurement of the fluctuating voltage produced by the random motions of charge carriers in a resistor at a finite temperature.

moment, the instantaneous distribution of electrons may (will!) be uneven along the resistor: this means that there will be a potential difference across the ends of the resistor (i.e. the non-uniform charge distribution produces a voltage across the resistor). As the distribution fluctuates from instant to instant the resulting voltage will also vary randomly.

Figure 3.1 shows a resistor connected via an amplifier to a dc voltmeter. Provided that the gain of the amplifier is high enough and the meter is sufficiently sensitive (and can respond fast enough) we'll see the meter reading alter randomly from moment to moment in response to the thermal movements of the charges in the resistor. As the fluctuations are random, we can't predict what the precise noise voltage will be at any future moment, but we can make some *statistical* predictions after observing the fluctuations for some time. If we note the random voltages at regular intervals for a long period, we can plot a histogram of these results: we choose a *bin width* $\mathrm{d}V$ and divide up the range of possible voltages into bins of this size, as shown in figure 3.2. The histogram (once normalised) indicates the probability $p(V)\mathrm{d}V$ that any future voltage measurement of the voltage will give a result in a particular range $V \to V + \mathrm{d}V$. The histogram thus displays the *probability density distribution* of the fluctuations. From the form of the results, two conclusions become apparent.

Firstly, the *average* of the all the voltage measurements in the limit of an infinite number of measurements, $\langle v \rangle$ will be zero: this shouldn't come as surprise as the voltage fluctuations due to random carrier movements are random. (There's no reason for the electrons to prefer one end of the resistor to the other!) The average thus doesn't tell us anything about how large the noise fluctuations are.

A more useful number for specifying the amount of random noise is a *root mean square* or rms quantity, which for m measurements is defined as

$$V_{\mathrm{rms}} \equiv \sqrt{\frac{1}{N}\sum_{j=1}^{N} V_j^2} = \sqrt{\overline{V_j^2}} \tag{3.1}$$

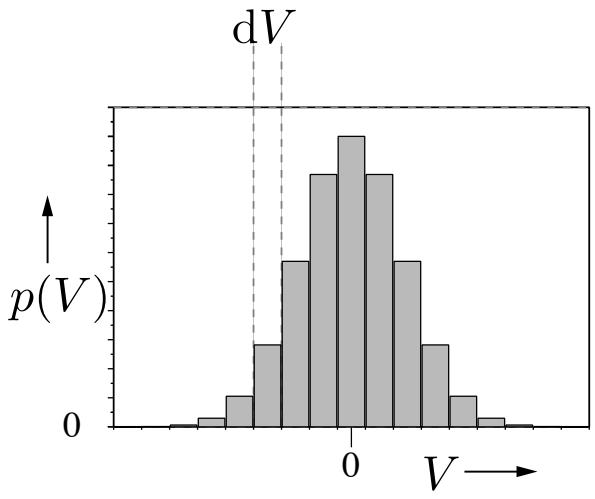


Figure 3.2: A cartoon histogram of some noise measurements showing a distribution of measured voltages around the nominal 'actual' value.

where the bar over V_j^2 denotes that we've taken the average of V_j^2. Since V_j^2 will always be positive, $V_{\rm rms}$ will always be positive (except for the trivial case when the width of the Gaussian distribution is zero, in which case $V_{\rm rms}$ will be zero.) The larger σ is, the larger the rms voltage level, so unlike the mean voltage, the rms voltage is a useful indicator of the noise level. It is also useful in practical situations as the power associated with a given voltage goes as the *square* of the voltage, so the average *power* level of some noise fluctuations can be expected to be proportional to $V_{\rm rms}^2$.

The histogram that we get will approximately fit a *normal* or *Gaussian* distribution of the form

$$p(V) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-V^2}{2\sigma^2}\right) \tag{3.2}$$

This expression is normalised, i.e. has an area of unity. This has to be the case, as our random voltage has to have *some* value.

The value of σ which fits the observed distribution indicates its width, so is a useful measure of the amount of noise. It also happens (hurrah!) that the rms value of the noise voltage will be equal to σ in equation 3.2. Which is nice.

Note that this distribution will only result after a *very* large number of measurements: the chances of any sensible number of measurements returning a histogram as symmetrical as that in figure 3.2 are vanishingly small. Figure 3.3 shows histograms of 'noise'[3] values I generated on my computer. In all three cases, the noise has an rms value of one (so σ for the underlying probability function described by equation 3.2 will also have a value of one) and a mean of zero, and the histograms all have 100 bins between -5 and 5. As the number of

[3]The inverted commas are there because truly random number generation is a tricky thing.

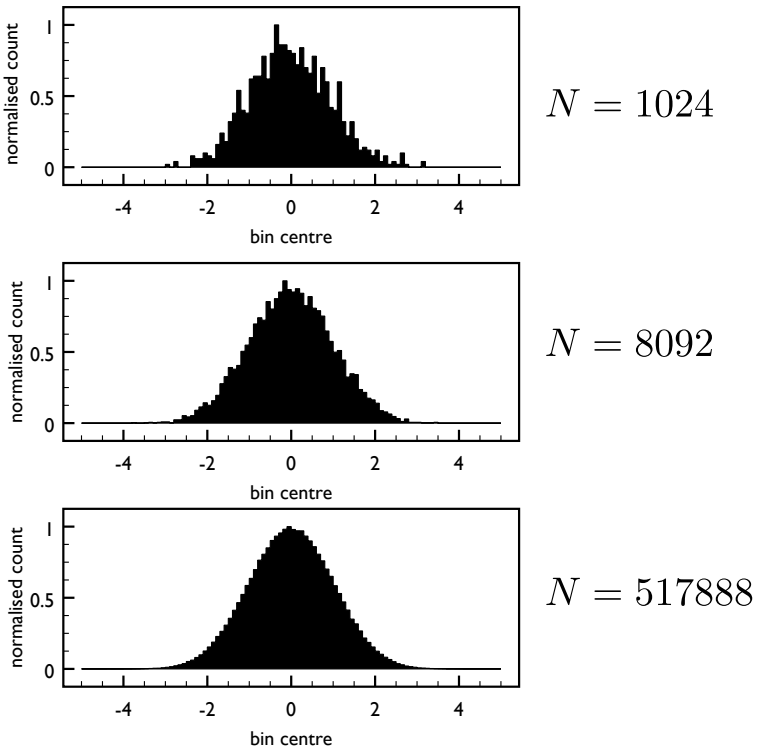


Figure 3.3: Histograms of three sets of computer-generated Gaussian random noise, normalised to have a maximum value of unity. The standard deviation of the underlying noise in all three cases is unity, and there are 100 bins in each of the histograms.

samples gets larger, the histogram gets more symmetrical as it comes closer to the underlying (continuous) Gaussian distribution[4].

Since thermal noise arises from the thermal motions of the electrons in a resistor, we can only get rid of it by cooling the resistor to absolute zero (which is, of course, non-physical)[5]. More generally, we can expect the thermal noise power level to vary in proportion with temperature.

An explanation of *why* the noise voltage across a resistor has these Gaussian characteristics is beyond the scope of this course[6], but is a consequence of the *central limit theorem*, which states that a process involving the sum of many independent random variables will result in a normal (i.e., Gaussian) distribution, even if the random processes themselves don't have

[4]These histograms were made by a home-grown method of getting approximately Gaussian noise by combining the results of lots of calls to my C compiler's standard random number generation function, so I do not have a terrible amount of confidence in it's underlying 'Gaussian-ness' in the limit of an infinite number of samples.

[5]Some experiments do reduce the amount of noise by cooling detectors or amplifiers as close to absolute zero as is affordable.

[6]Secret lecturer code for "I don't really understand it"...

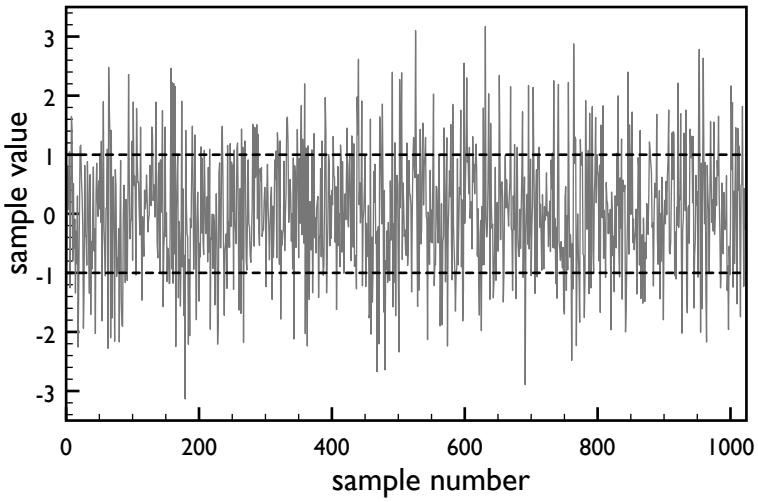


Figure 3.4: Computer generated Gaussian noise, with an rms value (σ) of unity. The dashed black lines are plotted at $\pm\sigma$ about the horizontal axis.

an underlying normal distribution[7]. Conduction electrons in a resistor will experience zillions[8] of scattering events in even the shortest measurement time, so the central limit theorem is appropriate to invoke here.

What about the *amplitude* of Gaussian noise? Well, this is quite a hard thing to pin down: a random signal clearly cannot have a well defined amplitude in the way that (say) a square-wave or sine-wave does.

This can be seen in figure 3.4, where I've plotted 1024 samples of computer-generated random noise, with a target rms value ($= \sigma$) of one. The peak-to-peak value over these 1024 samples is ≈ 6.3: it would, of course, be slightly different if I generated another set of 1024 random numbers with the same underlying distribution. A sensible rule of thumb is that the peak-to-peak amplitude of random Gaussian noise will be between 6 and 8 times greater than the rms value.

So far we've considered the statistical properties of noise in terms of its overall rms level and probability density function. This is not the only way to quantify noise and figure 3.5 shows an often more convenient alternative. In figure 3.5, we're examining the Johnson noise produced by a resistor: the voltage fluctuations are amplified and passed through a **band-pass filter** to an rms voltmeter. The filter only passes frequencies in some range, $f_{\rm min} < f < f_{\rm max}$, i.e. the **bandwidth**, B, of the filter is $f_{\rm max} - f_{\rm min}$. We will assume that the filter is a perfect band-pass filter, i.e., that it perfectly passes signals in the pass-band, and completely blocks

[7] Indeed, it is the central limit theorem which means that I can approximate Gaussian noise by adding lots of random numbers generated by a software routine which does not produce them according to a random distribution.

[8] It's a technical term.

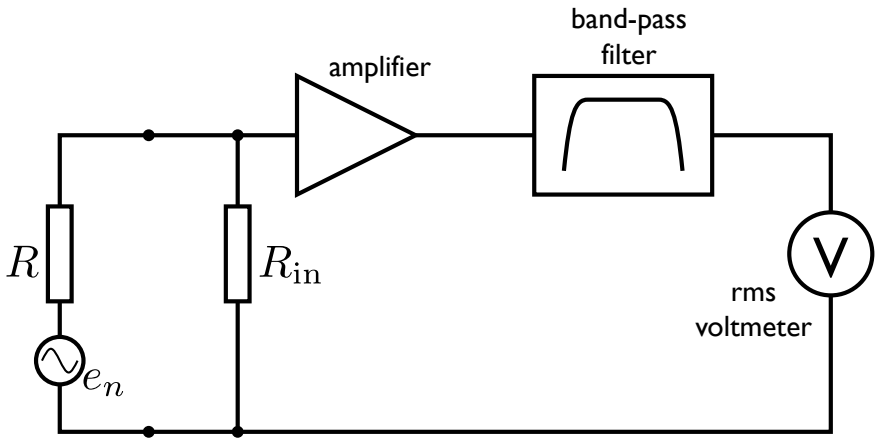


Figure 3.5: A simple spectral noise measurement system. The resistor is as some finite temperature, T.

signals outside this range[9]. R_{in} in the figure represents the input resistance of the amplifier. We're using the fairly common approach of pretending that the noise generated in the resistor is actually coming from a random voltage generator, v_n, in series with a noise-free resistor. If we constructed a system such as this, we would find that the rms fluctuations seen by the meter imply that the (fictitious) noise generator produces

$$\left\langle v_n^2 \right\rangle = 4k_B TBR \tag{3.3}$$

where k_B is Boltzmann's constant, T is the resistor temperature (in K), R is its resistance (in Ω) and B is the bandwidth (in Hz) over which the noise voltage is observed. *Note that we're not offering any proof of this statement, but presenting it as a matter of experimental fact.* In practice, all the other items in the circuit will also generate some noise, but we'll assume that the noise produced by the resistor will be large compared to any other source of random fluctuations.

Not all of the noise voltage will be dropped across the input of the amplifier: the resistor R and the input resistance of the amplifier R_{in} together form a potential divider. The voltage that we're actually measuring is that dropped across R_{in}, so will always be less than v_n. The current entering the amplifier (i.e. flowing in R_{in}) will be

$$i = \frac{v_n}{R + R_{\text{in}}} \tag{3.4}$$

The corresponding voltage seen at the amplifier input (i.e. across R_{in}) will be

$$\begin{aligned} v &= iR_{\text{in}} \\ &= \frac{v_n R_{\text{in}}}{R + R_{\text{in}}} \end{aligned} \tag{3.5}$$

[9] The fact that we can't actually build a filter that does this doesn't need to put us off!

so the mean noise *power* entering the amplifier will be

$$\begin{aligned} P_n &= \langle iv \rangle \\ &= \frac{\langle v_n^2 \rangle R_{\text{in}}}{(R + R_{\text{in}})^2} \end{aligned} \tag{3.6}$$

For a given resistor, R, we can maximise this by arranging for $R_{\text{in}} = R$, when we obtain the *maximum available noise power*:

$$P_{n,\max} = \frac{\langle v_n^2 \rangle}{4R} \tag{3.7}$$

and from equation 3.3, this can be seen to be

$$P_{n,\max} = k_B B T \tag{3.8}$$

The represents the maximum thermal noise power that we can get into an amplifier from the resistor. To achieve this we need to *match* the source and amplifier input resistances, i.e. make them the same. The maximum available noise power thus *does not depend upon the resistor value.*

We can define another useful quantity: the **noise power spectral density** (NPSD) at any frequency is defined as the noise power in a 1 Hz bandwidth centred on that frequency. Setting $B = 1$ Hz in equation 3.8 we can see that the Johnson noise has a maximum available NPSD of $k_B T$, i.e. it depends only on the temperature. This means that Johnson noise has an NPSD which *doesn't* depend on the fluctuation frequency. The same is true of shot noise. Noise with this character is said to be **white** since we see the same power level in a fixed bandwidth at every frequency.

It should be immediately obvious that, strictly speaking, no power spectrum can be truly white over an infinite frequency range, as the total power integrated over an infinite bandwidth would be infinite, which is non-physical. In any *real* situation, the noise generating processes will be subject to some inherent mechanism which results in a finite noise bandwidth.

In practice, most systems we devise to observe noise fluctuations will only be able to respond to a range of frequencies which is much less than the actual bandwidth of the noise being generated, which will in itself limit any measured value of the total noise power. Hence for most purposes we can consider thermal and shot noise as white over any frequency range of interest. However, the NPSD *does* fall away at extremely high frequencies, and this ensures that the total noise power is always finite. (Phew!)

Another point to note is that electronic noise levels are often quoted in units of $\mathrm{VHz}^{-1/2}$ or $\mathrm{AHz}^{-1/2}$, more practical units for low noise levels may be $\mathrm{nVHz}^{-1/2}$ or $\mathrm{pAHz}^{-1/2}$. These figures are sometimes referred to as the NPSD, even though the units are not those of power, because most measurement instruments are normally calibrated in terms of voltage or current. For white noise we can expect the total noise level to be proportional to the measurement bandwidth. The 'root Hz' reflects the fact that power is proportional to voltage (or current) *squared*, and a noise level specified as an rms voltage or current will increase as the *square root* of the measurement bandwidth.

Johnson noise doesn't just appear in actual resistors, but in anything that has a resistance. This includes semiconductor devices and interconnecting wires.

3.2 Shot noise

Another form of noise which is, to all intents and purposes, also unavoidable, is **shot noise**. If we consider a current I flowing in a wire, we know that it is produced by a stream of discrete charge carriers, with charge $q_e = 1.6^{-19}$ C. The current is related to the number of charges, n passing a point in time t by

$$I = \frac{nq_e}{t} \tag{3.9}$$

In reality, the carriers will not pass at an absolutely constant rate, as each will have a random component to its velocity and separation from its neighbours. When we repeatedly count the number of carriers passing in a series of m successive measurements of equal duration t, we'll find that the counts will fluctuate randomly from one interval to the next.

The effect of these variations is therefore to make it look like there is a randomly fluctuating noise current i_n superimposed on the nominally steady current I. If we made enough (very sensitive!) measurements to characterise these fluctuations, we would find that the noise current amplitude is related to the value of the current itself:

$$\left\langle i_n^2 \right\rangle = 2q_e IB \tag{3.10}$$

where B is the measurement bandwidth. Since some current and voltage is always necessary to carry a signal, this noise is *unavoidable*.

As the energy of a light field is quantised in units of the photon energy, shot noise is also an issue in systems that detect light levels.

Like Johnson noise, shot noise is also Gaussian and spectrally white, but any real process will have a limited bandwidth, so the shot noise amplitude may be estimated using equation 3.10. It is worth explicitly noting that although the noise amplitude gets larger with increasing current, the ratio of the current to the noise current will increase. This means that shot noise becomes a bigger problem when one is tying to measure very small currents (or light levels, etc): if one has a current of 1 A, then the effects of shot noise are negligible, but if you're trying to measure pA (and some people are!), then it's a much bigger concern. Similarly, if you're taking a photograph of the sun, it won't be an issue (although blowing up the camera sensor might be!), but when you are measuring light levels that correspond to only thousands of photons hitting a detector, then it really will be.

3.3 One-over-f noise

There are a great deal of physical process which produce noise, we've only touched on two of them. Another important type of noise we'll briefly look at is really a catch-all term for a whole host of different processes, all characterised by the fact that the noise the produce is not spectrally flat, but has a frequency dependence, namely that as the frequency increases, the noise spectral density decreases, i.e., the noise problem is worse at low frequencies. The name comes from the fact that the NPSD associated with it drops with increasing frequency, i.e.,

$$\text{NPSD} \propto f^{-n} \tag{3.11}$$

where n is a positive number typically around unity. $1/f$ noise is a particular nuisance because it impedes efforts to reduce noise by simple averaging. Sometimes the best way to avoid it is to alter the measurement system so that the measurement takes place at a frequency high enough that the $1/f$ noise is no longer significant.

3.4 Quantisation noise and dither

The last thing we'll discuss is **quantisation noise**: this is very different from the other issues we've discussed as it only applies to sampled data.

As should have been apparent from considering the contents of section 2.3, when sampling data with a finite bit-depth, then it will be incredible blind luck if the sample value is *exactly* equal to the underlying signal value at that instant, so there will (almost) always be an error. Once a signal has been sampled, then all we are left with is the quantised values, so we will have lost some information about the signal. The simplest way to model this is to consider the signal to have had some random noise added to it.

This is a reasonable approach because in general there should be no correlation between the signal value and the error between it and the resulting (quantised) sample value. Quantisation noise, however, is definitely *not* Gaussian: the probability density function of quantisation noise is *flat* between its minimum and maximum values, and zero outside this.

Consider the case in which the quantisation system works (somehow!) by spitting out the quantised value which is *closest* to the actual signal value at the instant that the signal is being sampled. If we define the error e as being the difference between the actual value and the quantised value, then it will be constrained to be between

$$-\frac{\Delta V}{2} \leq e \leq \frac{\Delta V}{2} \tag{3.12}$$

where ΔV is the value that corresponds to one **least-significant bit** (LSB), i.e.,

$$\Delta V = \frac{V_{\text{range}}}{2^b} \tag{3.13}$$

where b is the number of bits per sample.

Provided that the signal value fluctuates from sample-to-sample by amounts greater than the LSB, and that the signal is not correlated with the clock that is controlling the sample timing, then the actual value of e will vary randomly within the constraints specified in equation 3.12. This means that for many purposes, we can model the degradation to the SNR by assuming that we have added some random noise to our signal. The random noise 'added' by quantisation, however, is certainly *not* Gaussian: there is no reason for any allowed value of e to be preferred over any other allowed value of e, so the probability density function of the quantisation noise is constant between $\pm$LSB/2 and zero outside this range. The probability density must be normalised, so between $\pm$LSB/2 has a value of $(1/\mathrm{LSB})\ \mathrm{V}^{-1}$.[10]

The rms value (standard deviation) of the quantisation noise will be

$$e_{\mathrm{quant}} = \frac{1}{\sqrt{12}}\mathrm{LSB} \tag{3.14}$$

The quantisation noise is added to the noise level that is already on the signal (i.e., before sampling it) to determine the overall noise level. Because the two noise contributions should be completely uncorrelated, it is the two noise *powers* that we should add, not the noise *voltages*. This matters when determining SNR because power is always proportional to the *square* of the amplitude.

How significant quantisation noise is depends on the noise level on the signal before it is quantised, and on the bit-depth of the ADC.

The simple quantisation-as-added-noise model breaks down when the error signal *is not* random. This can occur when the signal variations are small compared to the LSB interval of the ADC, and then the ADC output value spends many successive samples on the same value. This is illustrated in figure 3.6: part (a) of the figure shows a continuous signal consisting of a cosine with an exponentially decaying amplitude and a peak value of one. Part (b) shows the effect of quantising this signal with a vertical resolution of 0.5, and the result is clearly distorted. More than half the samples after quantisation have a value of zero.

One way around this problem (if it matters) is to deliberately add some noise to the signal before sampling, as shown in part (c). The resulting set of samples (shown in (d)) is not stuck on zero over the whole low-amplitude tail. It is, however, still an unholy mess, but when filtered (digitally, in this case), we can see that we've recovered at least some of the low-level behaviour that was completely obscured in (b).[11]

Dither can also help us with some problems with quantisation that are more apparent in the frequency domain. If we have a periodic signal of well-defined frequency (let's say a

[10] The units here assume that we're measuring a voltage: they will be the inverse of whatever unit is being measured.

[11] You may notice that the filtered case appears to be delayed in phase: this is a side effect of the way the filtering was done: convolution was used to filter with the same effect that using a simple RC filter would have. Such a detail can be ignored for the purposes of the present discussion.

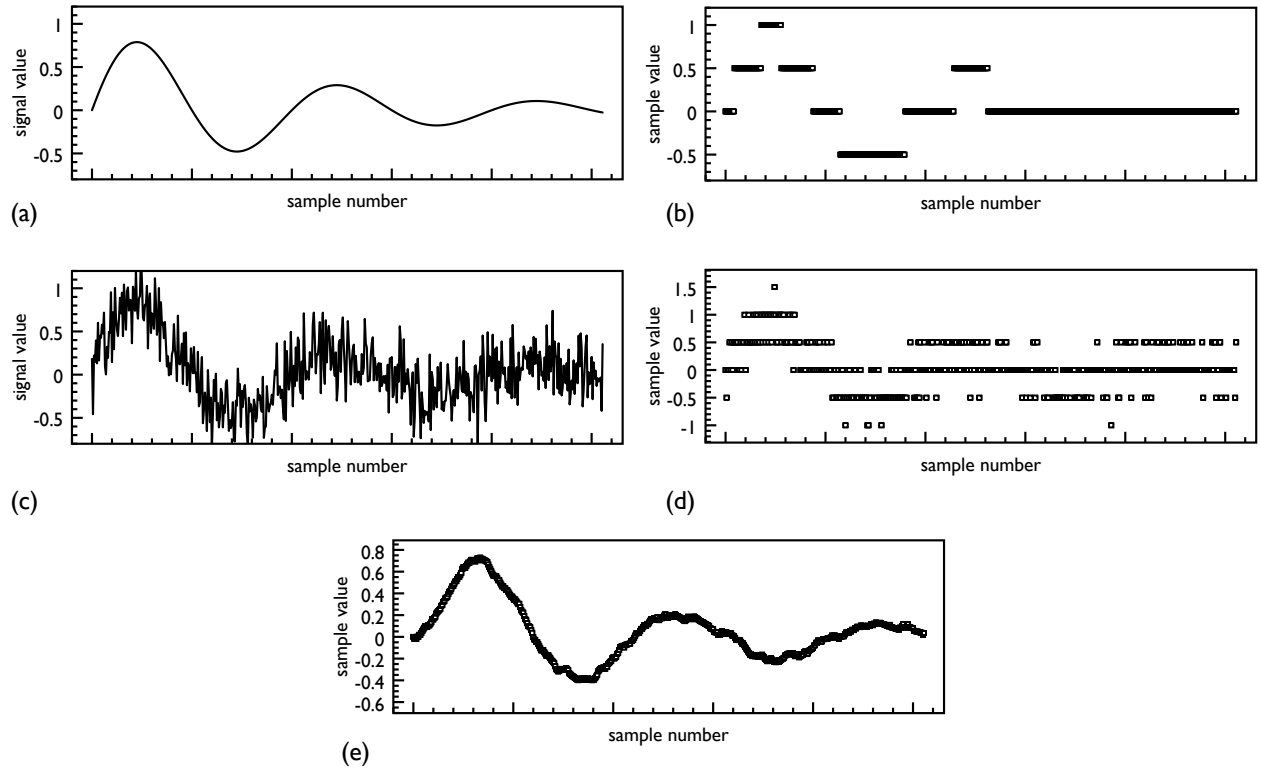


Figure 3.6: (a) A signal consisting of a sine-wave with an exponentially decaying amplitude. (b) The signal in (a) after quantisation by an ADC with a resolution of 0.5. (c) The signal in (a) with added Gaussian noise with an rms of 0.25. (d) The signal in (b) after quantisation with a resolution of 0.5. (e) The signal in (d) after filtering.

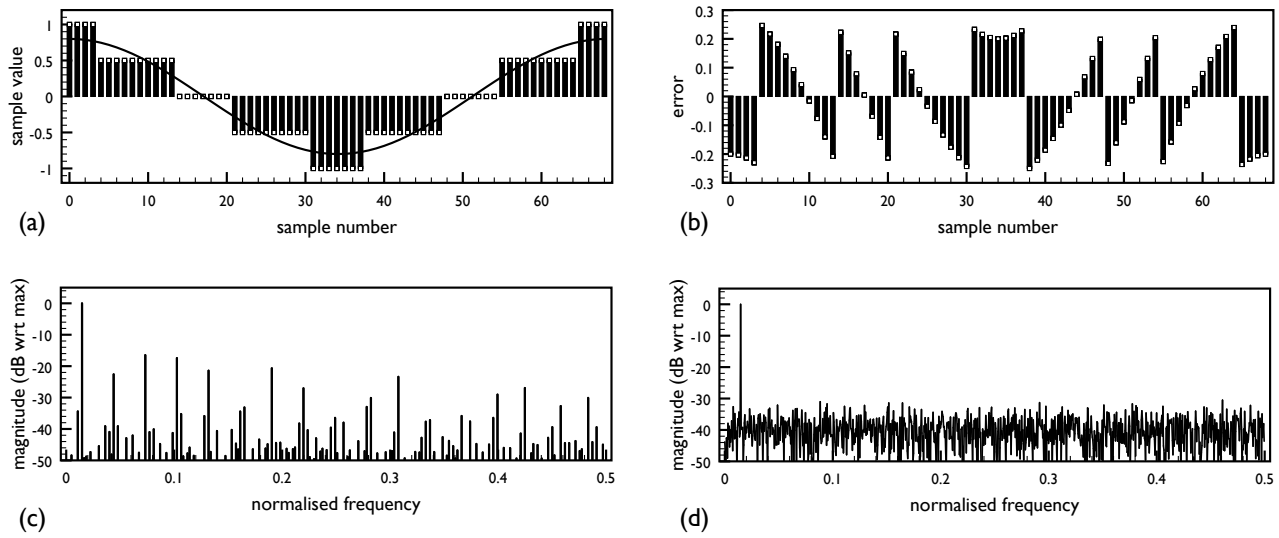


Figure 3.7: (a) The first 68 samples of a cosine signal with amplitude 0.8 and normalised frequency of 60/4096: the solid line shows the cosine signal and the bars and squares show the values obtained after quantisation by an ADC with a resolution of 0.5. (b) The error between the underlying value and the sampled value. (c) The magnitude spectrum (positive frequencies only) of the signal after quantisation. Note that the scale is in dB, normalised to the maximum value, and that many values lie far below the lowest limit of the vertical axis. (d) The magnitude spectrum (positive frequencies only) obtained with the same signal, but with Gaussian noise with an rms value of 0.5LSB added *before* quantisation.

sinusoid[12], then the quantisation error may *not* be random and the quantised waveform will feature repeated structure due to the quantisation.

An example of this is shown in figure 3.7: part (a) shows one cycle of a cosine wave with amplitude of 0.8 and the samples produced when quantising with a vertical resolution of 0.5 and part (b) shows the error in the sampled values. If we use the **discrete Fourier transform** (DFT, available in all its glory in chapter 4) to give us the *spectrum* then we get another (useful!) way of looking at the effects of quantisation: part (c) shows the first 2048 points of the DFT of a 4096-long record consisting of the quantised cosine wave as shown in (a). Note that the vertical scale is in dB, relative to the maximum value in the spectrum. Our normalised signal frequency is 60/4096, and is the largest spike in figure 3.7 (c). We can see, however, that there are many other relatively prominent spurious peaks, which are due to the quantisation noise. The SNR in this case is about 17 dB.

Figure 3.7 (d) shows the spectrum that results from adding Gaussian noise with an RMS value of 0.5 LSB to the cosine before quantising it. This has had the effect of lifting the noise floor (not surprising, given that we added noise!), but also it has removed the large forest of spurious peaks that were due to the quantisation noise, and the SNR is increased to around 30 dB. It is rather remarkable that in these circumstances, adding noise actually improves the SNR!

[12] By now, you shouldn't be surprised!

Chapter 4

The (discrete) Fourier transform

It's extremely useful to be able to go from a time-domain description of a signal to the frequency-domain (and back again!) The mathematical tool for this job is the **Fourier transform** which (most of) you have already encountered in PH3081.

For a continuous signal $x(t)$, its Fourier transform $X(f)$ is

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} \mathrm{d}t \tag{4.1}$$

j here is $\sqrt{-1}$: j is used rather than i as we will occasionally use i to denote a current.[1] You may also encounter these expressions written in terms of the angular frequency, $\omega = 2\pi f$, and then there are additional factors of 2π to worry about. An advantage of writing in terms of f rather than ω is that we don't need to remember where to stick factors of 2π in front of the integrals.

We can get from the FT $X(f)$ back to the function of time $x(t)$ by using the **inverse** Fourier transform:

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft} \mathrm{d}f \tag{4.2}$$

We say that $x(t)$ and $X(f)$ are **Fourier transform pairs** and denote this by

$$x(t) \rightleftharpoons X(f) \tag{4.3}$$

The '$\rightleftharpoons$' symbol is a common (but not the only) way of denoting FT pairs. Something else that varies from text to text is the sign of the exponents in the forward and inverse transforms, equations 4.1 and 4.2: here we're using the convention that in the forward transform the complex exponential in the integral has a *negative* exponent, whereas that in the inverse transform is *positive*. It doesn't matter what convention one chooses, as long as the signs

[1] Come to think if it, I'm not that certain we will. But in the communications/signal processing world, the j convention is pretty common.

of the exponents in the forward and inverse transforms are *different*, and once you pick a convention for a problem you should stick to it!

When dealing with *discrete* signals (i.e., sets of samples) however, The Fourier transform is not really appropriate as the signal doesn't exist between the sample. In this case, we make use of a close relative (indeed, a special case, really) of the FT called the **discrete Fourier transform** or **DFT**. The DFT is similar in form, and it won't be a huge surprise that it features a *sum* rather than an integral. Rather than $x(t)$, the DFT is performed on a set of samples $x[n]$, where n indicates that x is a function of the sample number. If we have N equally spaced samples, then the DFT is given by

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \tag{4.4}$$

where $k = 0, 1, 2, ..., N-1$. (Note: k here is not a wavenumber as it was in PH3080: it is just an index.[2]) You will often find this without the factor of $1/N$.

We can get back to our $x[n]$ using the **inverse DFT**:

$$x[n] = \sum_{k}^{N-1} X[k]e^{j2\pi kn/N} \tag{4.5}$$

where the only difference between 4.4 and 4.5 is the sign of the exponent and the fact that the inverse transform doesn't have its values divided by the factor of N. Again, there are different conventions on where the factor of N goes between the forward and inverse transforms. I prefer the chosen convention as it makes it easier to relate amplitudes of sinusoids in the n domain to the amplitudes of the components in the k domain. (If the forward transform did not have the factor of $1/N$, then the inverse transform would need to have it.[3])

$x[n]$ and $X[k]$ are thus a **transform pair**:

$$x[n] = \sum_{k}^{N-1} X[k]e^{j2\pi kn/N} \rightleftharpoons X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \tag{4.6}$$

n and k are the indices to sets of **conjugate variables**. Often, we start with a set of samples at different instants of *time*, so the conjugate variable is *frequency*.

Note that (just like the Fourier transform), the DFT is complex. If you want to write a program to calculate the DFT of a signal, then you will probably make use of Euler's relation, $e^{j\theta} = \cos\theta + j\sin\theta$. As far as we're concerned in the course, our $x[n]$ (which will be a sampled

[2] I considered trying a nomenclature that would avoid this possible source of confusion, but there are only so many letters in the alphabet...

[3] Here, Mathematica hedges its bets, and by default has a factor of $1/\sqrt{N}$ preceding the sums for both the forward and the inverse transforms, a convention that maximises the symmetry of the maths between the two directions.

version of some $x(t)$) will always be real, as it will be the result of some measurement that's spitting out a voltage (or a current (or a pixel brightness, etc, etc)), but equation 4.4 is fine for complex $x[n]$ as well. **Note that even though for us $x[n]$ will probably always be real, $X[k]$ will in general be complex** (except for a few trivial cases which correspond to pretty boring signals). (There is a formulation of the DFT specific to real signals, but we won't bother as it doesn't have the same elegant symmetry between the forward and inverse transforms as is the case for the full-fat complex version.)

It's worth dissecting equation 4.4 to see what its actually doing: for each value of k, it gives us the amplitude of the signal at a frequency proportional to k, and it all boils down to the mutual orthogonality of complex exponentials. If our signal contains no frequency component at k/N, then the sum for that particular k will be zero, just like extracting the coefficients for a trigonometric Fourier series. $X[k]$ is the amplitude of the *complex exponential* that has a *normalised* frequency of k/N, normalised to the sampling rate. The sum of equation 4.4 is the correlation of our signal $x[n]$ with the complex exponential of frequency k/N. The reason that the frequency is normalised is that n is just the sample number.

Normalised frequency can take a bit of getting used to as an idea, but for a given record length (i.e., N), it's straightforward to relate k to the frequency it tells us about:

$$f_k = \frac{k}{N} f_s \tag{4.7}$$

Now, I've said that k runs from 0 to $N-1$, which tells us that the maximum value for f_k will be a wee bit less than f_s (depending on how large N is). However, in section 2.4.2, we saw that if we sample a sinusoid whose frequency is greater than $f_s/2$, then it is no longer obvious just by looking at the samples what the frequency of the signal is. So how does that square with a maximum f_k of $\approx f_s$?

The answer lies in the fact that we treat the frequencies above $f_s/2$ in the sum as aliases of frequencies between $-f_s/2$ and zero: it's straightforward to show that

$$e^{-j2\pi kn/N} = e^{-j2\pi(k-N)n/N} \tag{4.8}$$

Now the concept of a negative frequency may feel a little odd. If we are representing a signal just by a sum of real sines and real cosines, then all that are required are positive frequencies. If we want to represent a signal by a sum of complex exponentials, however, then we need negative frequencies. For example, let's take a simple real sinusoidal wave, $x = \cos(\omega t)$. We can use Euler's relation to show that this is the same as

$$\cos(\omega t) = \frac{1}{2}\left(e^{j\omega t} + e^{-j\omega t}\right) \tag{4.9}$$

i.e., we need terms in both ω *and* $-\omega$ to reconstruct our real cosine from complex exponentials.

Figure 4.1 shows an example set of samples separated in time by 1 ms, i.e., with a sampling rate of $f_s = 1$ kHz. Part (a) shows the samples, and parts (b) through (d) show the real and

imaginary parts of the spectrum obtained using the DFT. It presents a number of different ways of presenting the spectrum, which are all equivalent, and you should get used to relating these to each other.

4.1 Calculating the DFT

The spectrum showed in figure 4.1 is all well and good, but how in practice do we go about calculating the DFT? Well, we just need a computer program to implement equation 4.4. If you were going to write your own, then you'd want to decompose the complex exponential using Euler's relation to get expressions for the real and imaginary parts of $X[k]$:

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]\left(\cos\left(2\pi k\frac{n}{N}\right) - j\sin\left(2\pi k\frac{n}{N}\right)\right) \tag{4.10}$$

Now, our signals will always be real by definition, but in the most general case $x[n]$ can be complex, i.e.,

$$x[n] = x_R[n] + jx_I[n] \tag{4.11}$$

and we have to account for this. Writing $x[n]$ in this way, then we have

$$\begin{aligned} X[k] &= \frac{1}{N}\sum_{n=0}^{N-1} (x_R[n] + jx_I[n])\left(\cos\left(2\pi k\frac{n}{N}\right) - j\sin\left(2\pi k\frac{n}{N}\right)\right) \\ &= \frac{1}{N}\sum_{n=0}^{N-1} x_R[n]\cos\left(2\pi k\frac{n}{N}\right) - jx_R[n]\sin\left(2\pi k\frac{n}{N}\right) \\ &\quad + jx_I[n]\cos\left(2\pi k\frac{n}{N}\right) + x_I[n]\sin\left(2\pi k\frac{n}{N}\right) \end{aligned} \tag{4.12}$$

so

$$X_R[k] = \frac{1}{N}\sum_{n=0}^{N-1} x_R[n]\cos\left(2\pi k\frac{n}{N}\right) + x_I[n]\sin\left(2\pi k\frac{n}{N}\right) \tag{4.13}$$

and

$$X_I[k] = \frac{1}{N}\sum_{n=0}^{N-1} x_I[n]\cos\left(2\pi k\frac{n}{N}\right) - x_R[n]\sin\left(2\pi k\frac{n}{N}\right) \tag{4.14}$$

Even if our signal is real (which measured signals usually are), the DFT will be complex. Equations 4.13 and 4.14 form the core of simple DFT programs. You maybe wouldn't want to memorise them, but they're easy to derive. If our sampled signal $x[n]$ *is* real, then they are simplified because all the x_I will be zero, i.e.

$$X_R[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]\cos\left(2\pi k\frac{n}{N}\right) \quad (x[n] \text{ real}) \tag{4.15}$$

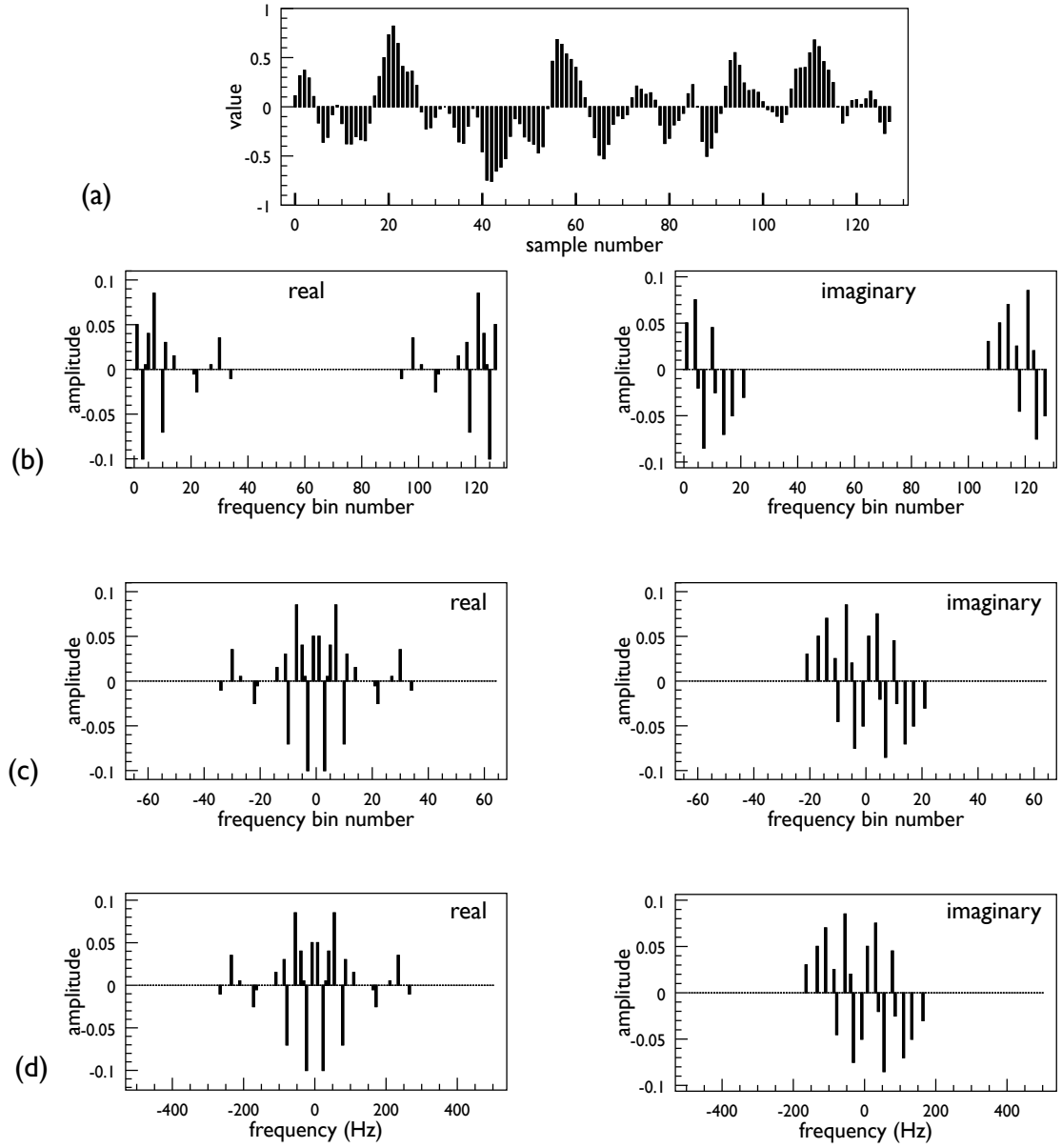


Figure 4.1: (a) A set of 128 samples of a signal. The sampling rate is $f_s = 1$ kHz. (b) The result of the DFT, with the horizontal axis running from 0 to $N-1$. (c) The same DFT result, but with the DFT amplitudes from $N/2+1$ to $N-1$ shifted leftwards by N. (d) The same data shown in (c), but now with the horizontal scale labelled in terms of frequency, from -500 Hz to 500 Hz.

and

$$X_I[k] = \frac{1}{N}\sum_{n=0}^{N-1} -x[n]\sin\left(2\pi k\frac{n}{N}\right) \quad (x[n] \text{ real}) \tag{4.16}$$

Equations 4.15 and 4.16 are of great interest to us as we'll almost always be interested in real signals, but remember that they are not general.

For each value of k, we are multiplying the samples point-by-point by a sine or cosine function, then summing all the values. Each complex exponential $\exp(-j2\pi kn/N)$ is a basis function, so we are projecting our set of samples $x[n]$ onto each basis function in turn (i.e., $k = 0, 1, 2, ..., N-1$), to find the amplitude that basis function has in the overall superposition of basis functions required to reproduce our $x[n]$.

This process is illustrated in some detail by figures 4.3 and 4.2. These figures illustrate what is going on in the calculation of the first four amplitudes of $X[k]$, i.e., for $k = 0, 1, 2, 3$. For each value of k, the nth sample is multiplied by $\sin(2\pi nk/N)$, and the values summed over n to give $X_I[k]$, and by $\cos(2\pi nk/N)$ for $X_R[k]$. Figure 4.3 shows the process for the first four sine basis functions, which yields the imaginary parts of the first four values of X and figure 4.2 shows the process for the first four cosine basis functions, which yields the real parts of the first four values of X.

4.1.1 The Fast Fourier Transform

If you squint at equation 4.4 for long enough and think a wee bit, you will see that for each point in a DFT (i.e., each value of k), we need to do N complex multiplications, then add all the results together. Given that there will be N values of k that you do this for, that means N^2 complex multiplications to do a DFT of N samples.

This means that the amount of computation effort required to DFT a data set scales as the *square* length of that data set. So, for example, if taking the DFT of a record consisting of 4096 complex points takes about 1 second[4], then doing one that is a million points long would take 18 hours![5]

This is bad news from a practical point of view. We are saved (hurrah!) by the fact that one can split an N-point DFT into *two* DFTs that are only $N/2$ points long, calculate them separately and then combine them to get our N-point DFT. The combination stage introduces a little more computational expense, but not much compared to the effort of calculating the two (now shorter) sub-DFTs. If we always choose N to be an integer power of 2 (i.e., $N = 2^\gamma$, where γ in an integer) then this process can be repeated until we have a problem that consists of doing $N/2$ individual 2-point DFTs (which are mathematically trivial), then doing $N/4$ 4-point DFTs on the result, then $N/8$ 8-point DFTs on the result, and so on, to end up with the N-point DFT that we require. This process takes a length of

[4] 1.2 seconds on my machine with a rather clunky DFT routine that I wrote.

[5] I didn't try one this long, as it would get in the way of watching cat videos...

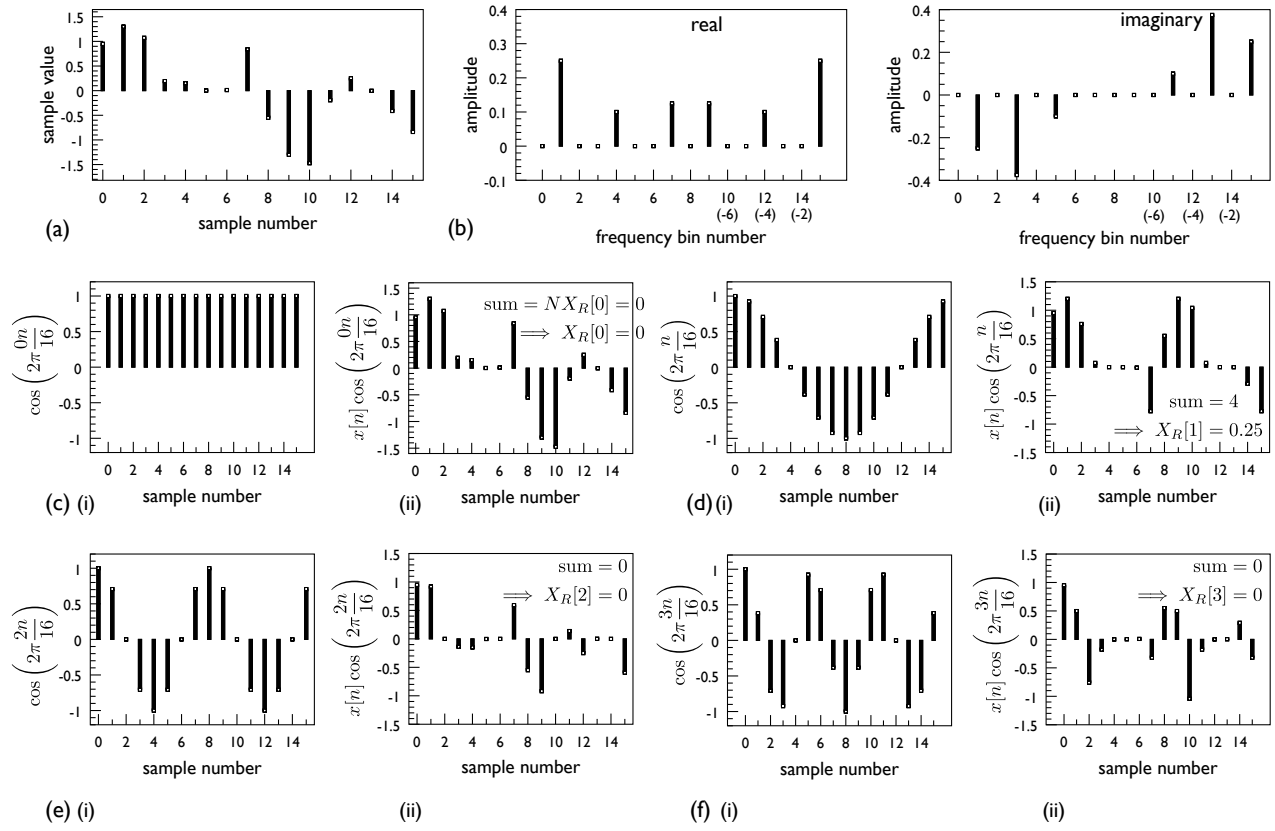


Figure 4.2: (a) A set of 16 samples. (b) The 16 point DFT of the samples in (a). The frequency bin number runs from 0 to 15, but the values for $N/2+1$ and greater could be shifted to the left by N. (c)(i) The cosine basis function for $k = 0$ and (ii) the point-by-point multiplication of this basis function by the sample values. (d) As for c, but for $k = 1$. (e) As for c, but for $k = 2$. (f) As for c, but for $k = 3$.

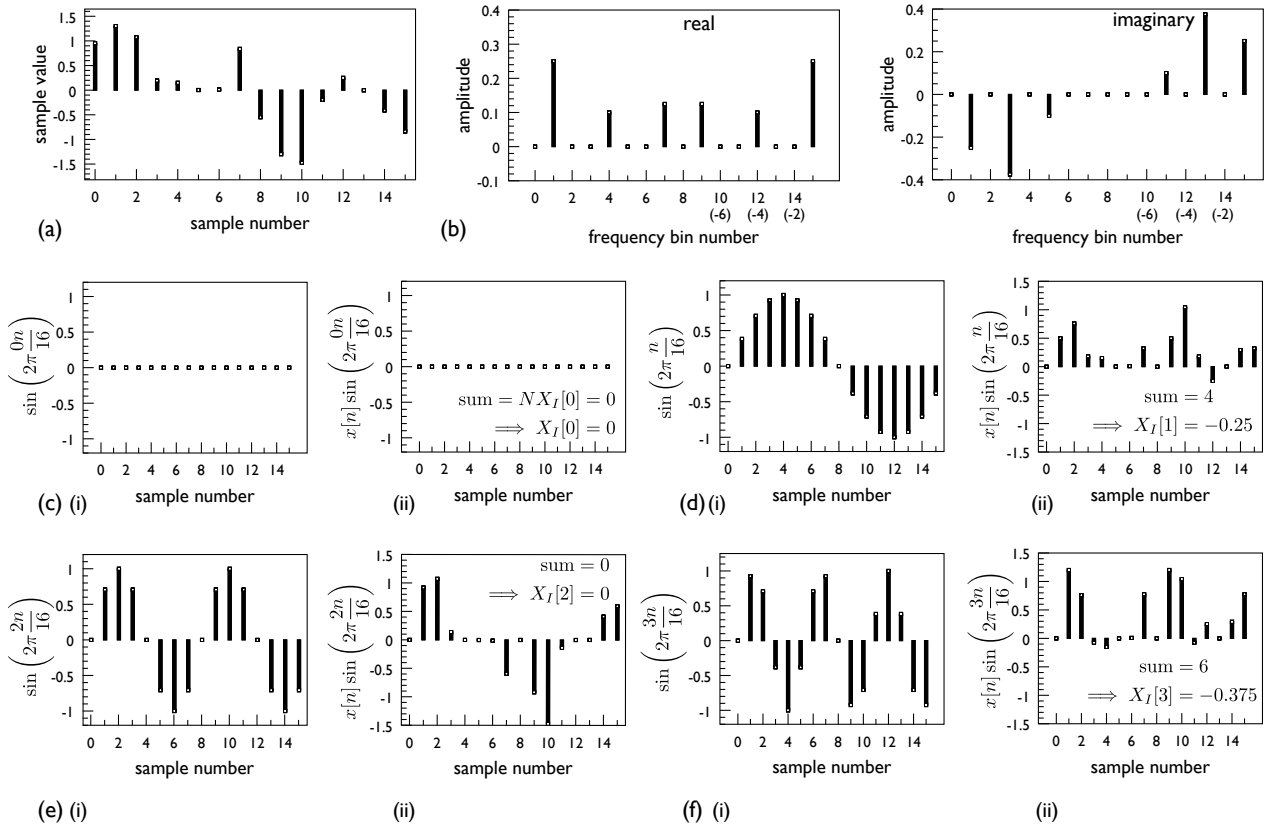


Figure 4.3: (a) A set of 16 samples. (b) The 16 point DFT of the samples in (a). The frequency bin number runs from 0 to 15, but the values for $N/2+1$ and greater could be shifted to the left by N. (c)(i) The sine basis function for $k = 0$ and (ii) the point-by-point multiplication of this basis function by the sample values. (d) As for c, but for $k = 1$. (e) As for c, but for $k = 2$. (f) As for c, but for $k = 3$.

time which does not scale with N^2, but rather goes as $N \log_2(N)$. So if a 4096-point DFT using this new method took a second (but it wouldn't: it'd be a lot faster[6]), then a million points would take half an hour.

Such a method of calculating the DFT is known as a **fast Fourier transform**, or **FFT**. The FFT is not really a single *thing*, rather the term refers to any of many methods of calculating the DFT that is (much!) faster than the simplest method of following the recipe described by equations 4.13 and 4.14. The particular scheme just outlined above is a *radix-2 decimation-in-time* FFT, but the details of how an FFT operates is really irrelevant to this course: it gives the *actual* DFT, and *not* just an approximation to it, and it's really the DFT, what it does and why it leads to results that it does that we care about. I don't care whether you know anything about the FFT: this section is just here because you will (have) encounter(ed) it in LabView, MATLAB, oscilloscopes, etc, and it's important to realise that it really means the DFT.

4.2 Some properties of the DFT

The DFT has lots of mathematical properties: we'll look at a very small hand-full. (As the DFT is just a special case of the Fourier transform, then it inherits some of these properties from the Fourier transform, and others arise due to the discrete nature of the signal.)

- **Linearity**: the DFT is **linear**: if we have two signals and take the DFT of both of them, then the sum of the DFTs is the same as what we'd get of we took the DFT of the sum of the two signals, i.e. if the DFT of a signal $x[n]$ is $X[k]$ and the DFT of signal $y[n]$ is $Y[k]$, then the DFT of the sum, $S[k]$ (where the sum is $s[n] = x[n] + y[n]$) will be
$$S[k] = X[k] + Y[k] \tag{4.17}$$
(If this was not the case, then the DFT would be pretty useless to us, as we'd only be able to use it for looking at pure sinusoids, and we'll see later that they (practically by definition!) can't carry any information.) Proving equation 4.17 is pretty straightforward.

  Linearity also means that that if $X[k]$ is the DFT of $x[n]$, then the DFT of $ax[n]$, where a is a scalar, will be $aX[k]$.

- **Discreteness**: one of the fundamental differences between the (continuous) Fourier transform and the DFT is in the name (sort-of): the result of a DFT is a *discrete* spectrum. The spectrum is *not* discrete *just* because the signal is discrete, however: it is discrete because we only have a finite set of basis functions. (These two things *are* of course related, but it's worth being pedantic in this case. For example, a continuous *periodic* function will have a (continuous) Fourier transform that consists of a set

[6] 1.3 ms on my machine with a really badly-written routine.

of equally-spaced (continuous) delta functions (of varying amplitudes), separated by regions in which it is zero.) The fact that the time domain signal is periodic is what make the DFT discrete. This doesn't seem to make much sense: nothing we've said so far has restricted us to periodic time-domain signals. However, as soon as we only have a finite set of terms in a Fourier series, which are all harmonically related (like in this case), then the signal produced by that series *is* periodic.

- **Periodicity**: The other fundamental difference between the DFT and the continuous Fourier transform is that the DFT is **periodic**: this arises due to the fact that there are an infinite set of possible signals that could have given rise to a particular set of samples, as discussed in sections 2.4.1 and 2.4.2. Indeed, one can tell just by looking at equation 4.4 that it will be periodic with period N. There are two equivalent ways to view this periodicity: one is to duplicate the result of the DFT by copying and pasting it at intervals of mf_s where m is an integer. Another way is to view the DFT represented such that the frequency axis is *circular*: if we have an N-point DFT, then $X[N] = X[0]$.

  Another way of explaining this is that the DFT is periodic because the input signal itself is discrete. There is a pleasing symmetry between this and the above bullet point: if a signal is discrete in one domain, then it's periodic in the conjugate domain.

- **Symmetry**: The DFT basis functions are complex exponentials. This may look like additional bad news (complex means more complicated, right?) but the good thing is that a complex spectrum tells us about the relative **phases** of signals as well as magnitudes. When we are dealing with real-valued signals (i.e., $\mathrm{Im}(x[n]) = 0$ for all n), then things are simplified somewhat.

  If a signal is real, then the real part of the spectrum is *always* **symmetrical** about zero frequency and the imaginary part is *always* **anti-symmetic** about zero frequency. This is consistent with the example shown in figure 4.1. This means that there is some redundancy in the DFT output, as if we know what the first $N/2$ values of the DFT are, then we can work out trivially what the remaining $N/2$ values are.

4.3 Convolution and the convolution theorem

If we have two functions, $x(t)$ and $g(t)$, then the **convolution** of those two functions, $y(t)$ is defined as:

$$y(t) = x(t) * g(t) \equiv \int_{-\infty}^{\infty} x(\tau)g(t-\tau)\mathrm{d}\tau \tag{4.18}$$

Note that "$*$" here denotes convolution, and not multiplication!

You may be forgiven if your reaction to this is "so what?": convolution is not terribly easy to visualise in our heads and you may not have used it much before.

Convolution is useful to us for two reasons: the first, which applies to the immediate context of the Fourier transform. We can use the **convolution theorem** in many cases to determine what the spectrum of a modulated signal will be.[7] The second reason is that convolution can be used to model the effects on a signal of passing through some processing element, such as a filter or transmission line, and convolution is itself used extensively in digital signal processing to implement various filtering and mathematical operations. We won't deal with the latter a great deal, but it is good to at least know about it, so we'll discuss it a bit here, then move on to the former.

Expression 4.21 needs to be modified if we want a version that applies to discrete signals. Let's consider two discrete signals: $x[n]$ is N samples long, i.e., n runs from 0 to $N-1$. $g[n]$ is M samples long, i.e., n runs from 0 to $M-1$. Note that n is just an index The discrete convolution of these two signals is $y[k]$, where k is another index, which runs from 0 to $N+M-2$, i.e., the length of $y[k]$ is $N+M-1$. Then,

$$y[k] = \sum_{a=0}^{M-1} g[a]x[k-a] \tag{4.19}$$

(Note that if you tried to implement this in a computer program exactly as it is, it would crash as there will be values of k and a during the calculating for which $k-a<0$: you would need to trap this. One possible way to handle this (and the one taken in the examples we'll look at) is to act as if x has a value of 0 for $i<0$ and $i>N$.)

Convolution is a linear operation, and it is also commutative, i.e.,

$$x * g = g * x \tag{4.20}$$

4.3.1 The convolution theorem

The **convolution theorem** states that the Fourier transform of the convolution of two functions is the product of the Fourier transforms of the two functions that have been convolved, so if $y(t) = x(t) * g(t)$, $G(f)$ is the Fourier transform of $g(t)$ and $X(f)$ is the Fourier transform of $x(t)$, then $Y(f)$, the Fourier transform of $y(t)$ is

$$Y(f) = X(f)G(f) \tag{4.21}$$

This is surprisingly important in signal processing, due to how common it is to model the effects of a system in terms of convolution. A signal processing element, such as a filter or amplifier, provided that it is linear (and obeys some other more obscure principles, that we'll just take for granted), can be completely characterised by its response to an impulse. The response to an arbitrary signal is just the the sum of its responses to a series of scaled and shifted impulses, i.e., the convolution of the input signal and the impulse response.

[7] A formal discussion of the spectrum of a sampled signal, for example, makes use of it.

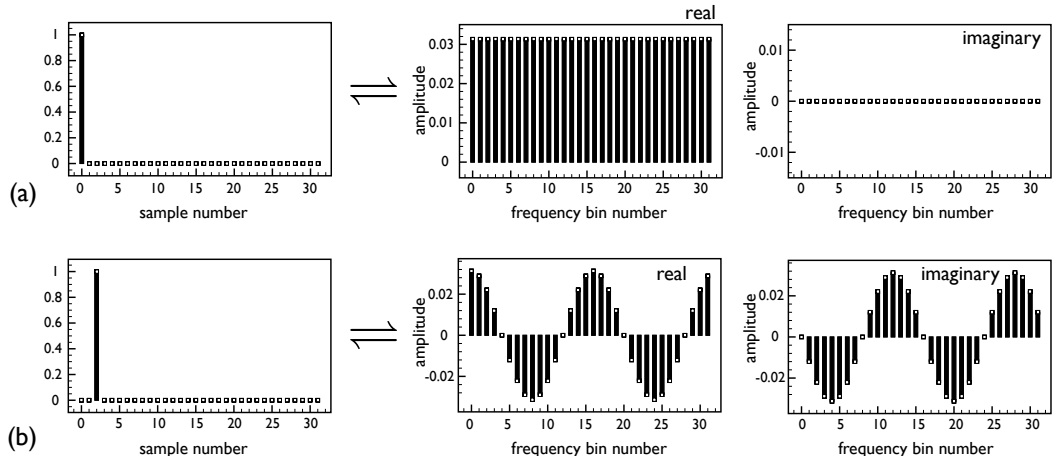


Figure 4.4: Discrete delta functions and their 32-point DFTs: (a) delta function located at sample zero and (b) at sample 2.

4.4 Fourier transform pairs

Any set of samples will have a corresponding DFT. There are some common ones that we will encounter (and, indeed, that occur in other contexts in physics), and we'll look at a few here. We're not going to derive the examples shown here (life is too short), but it's important to know at least the basic form of some key results.

4.4.1 The delta function

An important signal when discussing discrete signals is the **delta function**, δ. This is a bit different from the delta function you are used to, which only makes sense for continuous signals, but has similarities as well.

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} \tag{4.22}$$

This is also known as the **unit impulse**. If the impulse occurs at a value of $n \neq 0$, then it is shifted, i.e., a unit impulse occurring at the 4th sample would be $\delta[n-4]$.

Thus as the DFT of a sinusoid yields an impulse (a pair, in fact, and only if the sinusoid is one of the DFT basis functions), so the DFT of an impulse yields a sinusoid. Two examples are shown in figure 4.4.

Figure 4.4 (a) shows the 32-point DFT of $\delta[n]$ and part (b) shows the DFT of $\delta[n-2]$. The number of cycles of the DFT goes through in the frequency domain is equal to the number of samples by which the impulse is shifted from zero. The discrete delta function is perhaps

more useful that you would think, chiefly because *any* discrete signals can be composed of a series of scaled-and-shifted delta functions.

Note that although parts (a) and (b) look rather different, in both cases the *magnitude* is the same over all the bins in the DFT. In this course we are often more interested in the magnitudes of signal spectra (i.e., how much bandwidth does something need?) than the phases, but there are instances where the phases do matter to spectra.

4.4.2 The top-hat function

The proper name for this function is *rectangular pulse*, but I think *top-hat* sounds cooler. Figure 4.5 the shows 64-point DFTs of 2 discrete top-hat functions, one which is 8 samples long and one which is 16 samples long. If we took these two top-hats and translated them to the right (i.e., shifted them to later samples), then the real and imaginary parts would look a complete mess, but the *magnitude* would be left unchanged.

If we have a set of N samples containing a top-hat which is M samples wide, then the *magnitude* of the DFT is given by

$$|X[k]| = \left| \frac{\sin(k\pi M/N)}{\sin(k\pi/N)} \right| \tag{4.23}$$

This is derived in Lyons if you're really keen!

Consider the case where $k << N$: then

$$\begin{aligned} |X[k]| &= \left| \frac{\sin(k\pi M/N)}{\sin(k\pi/N)} \right| \\ &\approx \left| \frac{\sin(k\pi M/N)}{k\pi/N} \right| \\ &\approx M \left| \frac{\sin(k\pi M/N)}{k\pi M/N} \right| \\ &\approx M \left| \text{ sinc } \left(\frac{k\pi M}{N} \right) \right| \end{aligned} \tag{4.24}$$

where $\text{sinc }(x) = \sin(x)/x$. This makes a bit of sense (hopefully!), as the continuous Fourier transform of a continuous top-hat function is a sinc fuction. Note that this approximation is only valued for small k (or small $N - k$ due to the DFT periodicity). Given that our top-hat is just a discrete representation of a continuous top-hat, why is the approximation to a sinc function only valid for low normalised frequency? In a word, aliasing.

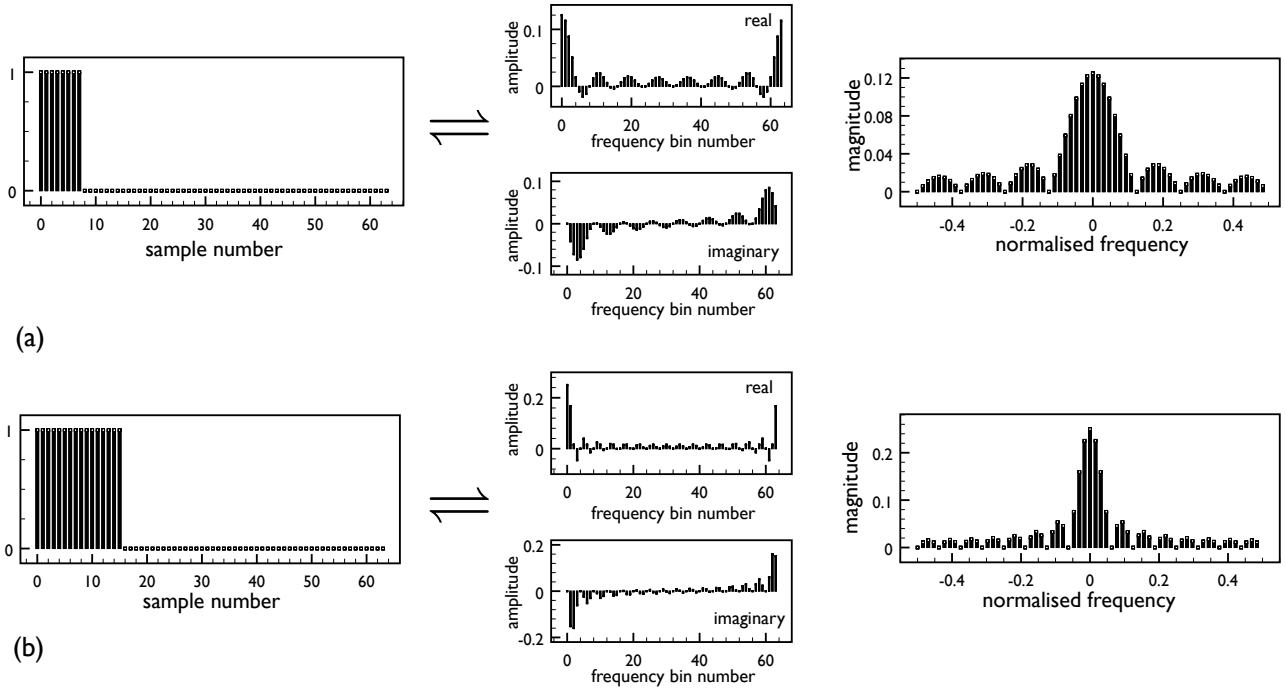


Figure 4.5: Top-hat functions and their 64-point DFTs: (a) 8 samples wide, (b) 16 samples wide.

4.4.3 The Gaussian

A Gaussian centred at x_0 with a full-width at half-maximum (FWHM) of w and an amplitude A is described by the equation

$$g(x) = A \exp\left(-4 \ln 2 \left(\frac{x - x_0}{w}\right)^2\right) \tag{4.25}$$

(If this looks odd compare to Gaussians you have seen before, it's because they are not usually specified by both an amplitude and the FWHM together. I've chosen this way of doing it here because it means that when I plot it, it's easy to check that I've got the equation right!)

A discrete version of this could be described by something like

$$g[n] = A \exp\left(-4 \ln 2 \left(\frac{n - n_0}{w}\right)^2\right) \tag{4.26}$$

where n_0 is where it peaks (in samples) and w is the FWHM, also in samples.

If we calculate the DFT of this, then we find that the result is *also* Gaussian in shape, i.e., a Gaussian has the rather remarkable property of being its own Fourier transform, (albeit with scaling factors, etc.), although we will get odd effects (again due to aliasing) if the Gaussian is truncated.

The FWHM of the time-domain Gaussian Δt (in seconds) and the FWHM in the frequency domain Δf (in Hz) of its Fourier transform are related by

$$\Delta f = \frac{4 \ln 2}{\pi \Delta t} \tag{4.27}$$

When we have a width specified *in samples*, then we need to account for the sampling rate and the length of the DFT: if the FWHM of the Gaussian in the time domain *in time-domain samples* is w_t then the FWHM *in frequency-domain samples* of the DFT is

$$w_f = \frac{4 \ln 2}{\pi} \frac{N}{w_t} \tag{4.28}$$

4.5 Frequency resolution

Equation 4.7 tells us how to relate the bin number in the DFT to the frequency it corresponds to, repeated here:

$$f_k = \frac{k}{N} f_s \tag{4.7}$$

so the frequency resolution of the DFT is

$$\Delta f = \frac{f_s}{N} \tag{4.29}$$

So what? Well, a bit of thought and this might surprise you: it means that if we (say) double the sampling rate, but keep the number of samples (i.e., N) constant, then the frequency resolution gets *worse*. If we double the sampling rate and double the number of samples, then the frequency resolution stays the same. What matters for the best frequency resolution is the total length of time that our record lasts for.

If more frequency resolution is desired at the same sampling rate, we have to take more samples. If we want to increase the frequency resolution of a DFT using only a set of samples that we already have, then we can **pad** the set of samples by adding extra zeros to result in a longer record. Note that this cannot increase the amount of information we actually end up with: we just end up sampling the spectrum at more closely spaced intervals.

4.6 Spectral leakage and windowing

Figure 4.6 shows the results of performing a DFT on a 32-point signal, consisting of just a single cosine with unity amplitude. Only the first 16 points of the DFT are plotted, and the figures show the *magnitude*. The 11 different graphs show the results as the normalised frequency is changed from $1/4$ to $9/32$ in steps of 0.1.

You will notice that the result in the first graph is exactly what we'd expect: a single spike at frequency bin 8, which corresponds to a normalised frequency of $8/32 = 1/4$. As the frequency is increased, however, we start to see non-zero components emerging in the other bins, which becomes extremely noticeable at a frequency of $8.5/32$, then reduces again until at $f = 9/32$ we are left with a single spike at (not unexpectedly) $f = 9/32$.

So, what's going on here? In the case where $f = 8/32$ or $f = 9/32$, then our cosine signal exactly matches one of the DFT basis functions. Any normalised frequency with a value of $n/32$ where $n = 0, 1, 2, ...31$ will exactly match one of the basis functions, so the result of the DFT is just a peak at n (and a corresponding one at $-n$ or $N/2 - n$ depending on which convention you prefer). If the signal does *not* match one of the DFT basis functions, then it will need a contribution of *all* the basis functions to make up the signal.

There are a couple of reasons this manifests in the way in does. We need to remember that just as our sampled signal $x[n]$ is discrete, so is its DFT, $X[k]$. In addition, the signal does not consist of an actual cosine (which begins forever ago and lasts forever), it consists of only *a part* of a cosine. What we are taking the DFT of is a set of samples produced by sampling the *product* of a ideal sinusoid by a function which has a value of unity for the duration of our signal, and a value of zero everywhere else.

According to the convolution theorem, this means that the Fourier transform will be the *convolution* of that of a pure sinusoid with that of a top-hat. The FT of a sinusoid is a pair of delta functions and that of a sinusoid is a sinc function, so the FT of (sinusoid multiplied by top-hat) is the convolution of a sinc function with a delta function, i.e., another sinc function.

When we are using the DFT for spectral analysis, then leakage is important (i.e., bad) because it means that a strong signal can mask a weak one that is nearby in frequency.

One method of reducing the leakage is to *window* the time-domain[8] signal, i.e., multiply our set set of samples point-by-point by a **windowing function**. There are tons of different windowing functions available, but what they all have in common is that their value is smaller at the edges of the sampling period than in the centre. This is illustrated with one example in 4.7.

The use of a window means that the signals spectrum is convolved with that of the window function, rather than that of the top-hat. In figure 4.7, the window function being used is called a **Hamming window**, which is just a cosine with an offset. Because the amplitude of the window function is much lower at the edges of the sample period, the leakage is reduced. The down side is that the main lobe of the spectrum is widened, so we get less leakage at the expense of poorer spectral resolution. This is a general point: windowing the data will always result in a wider spectral peak than no (i.e., rectangular) windowing.

There are *lots* of different windowing functions, all with different tradeoffs, usually of peak sharpness against leakage and amplitude response. The details don't matter, but you should

[8]Or whatever the domain is that our signal is acquired in...

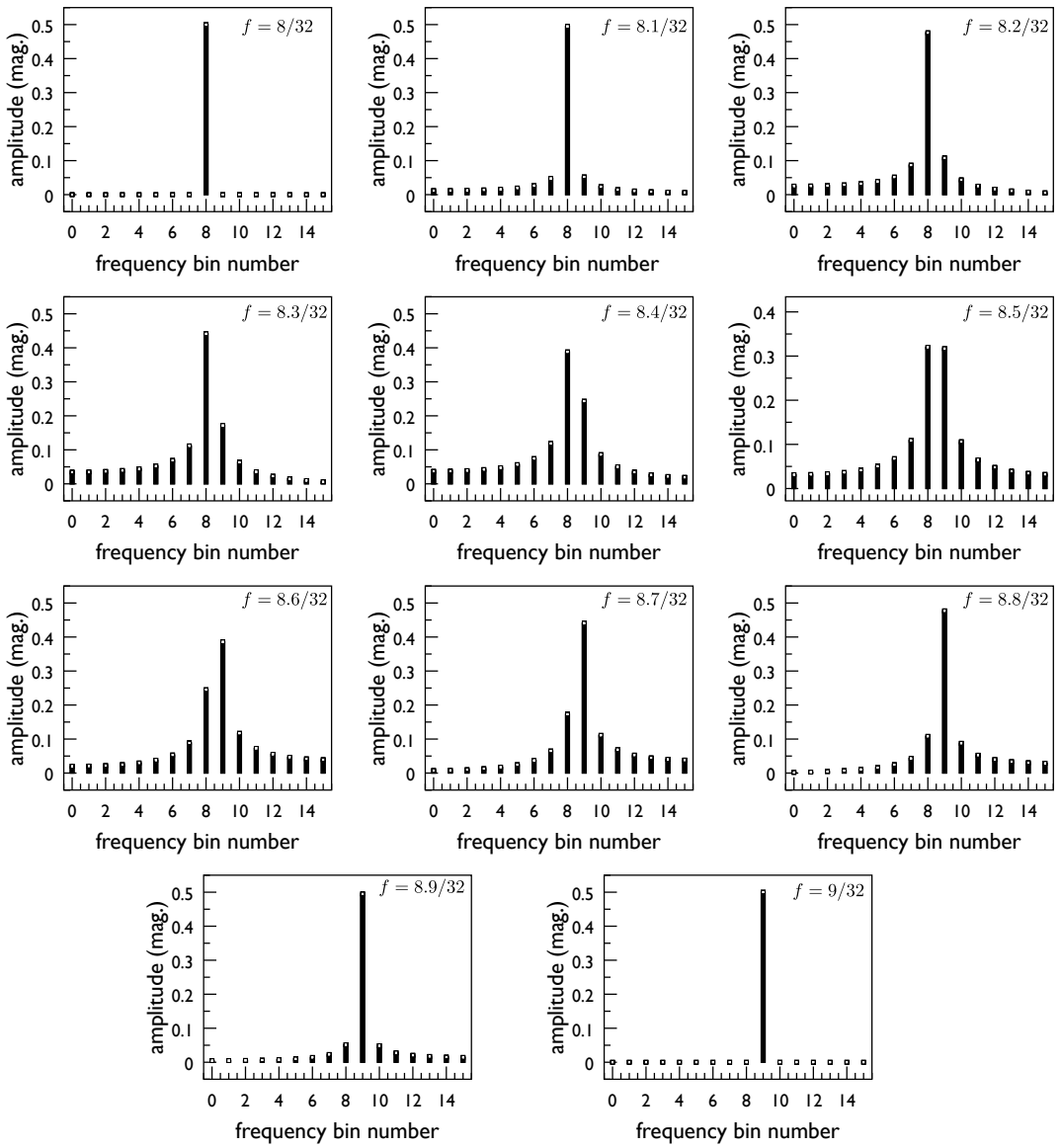


Figure 4.6: The amplitude magnitude of the first 16 ($k = 0...15$) points of a 32-point DFT of a real signal of unity amplitude as its frequency varies from 8/32 to 9/32. This shows how the amplitude of the signal is spread over the whole frequency spectrum when the frequency doesn't exactly match one of the DFT basis functions.

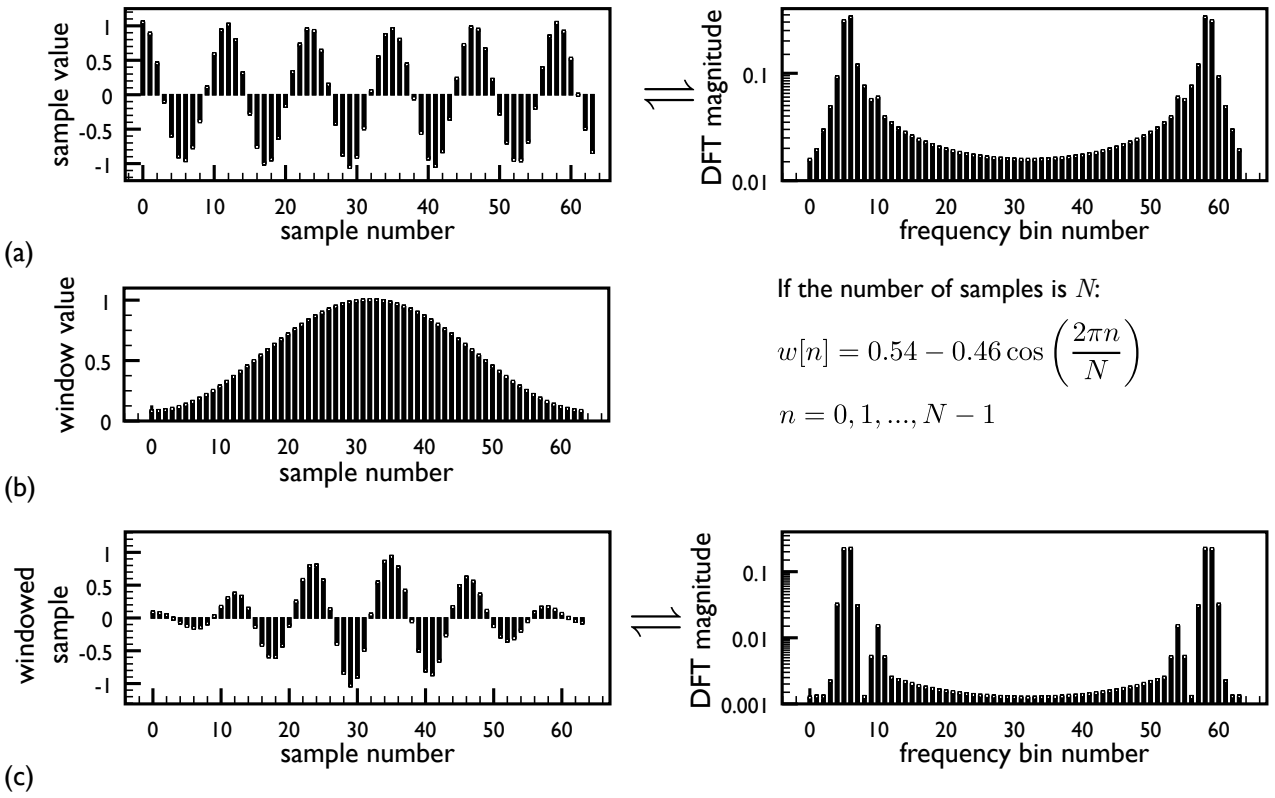


Figure 4.7: (a) A 64 samples long signal consisting of the sum of two cosines with normalised frequencies $f_1 = 5.5/64$ and $f_2 = 10/64$ and amplitudes $A_1 = 1$ and $A_2 = 0.05$, and its DFT. (b) A *Hamming* windowing function. (c) The point-by-point multiplication of the signal and the Hamming window function, and the DFT of the windowed signal.

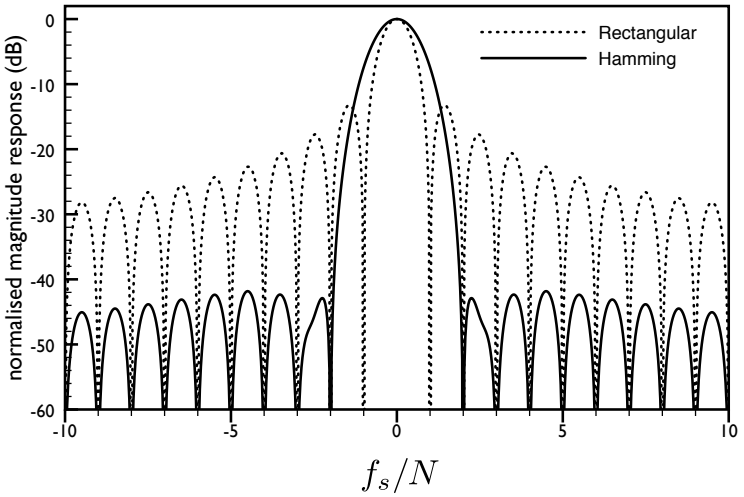


Figure 4.8: The DFT of (dashed) a rectangular and (solid) a Hamming window of the same width. If a signal's frequency does not match one of the DFT basis functions, then the samples that form its DFT will not lie in the deep nulls.

know that the shape of a spectral peak after windowing is the **convolution** of its underlying shape and the Fourier transform of the windowing function.

As an example, figure 4.8 shows the DFTs of a rectangular window and a Hamming window of the same with. The 'time' domain signals were *padded* to increase the frequency resolution of the DFT, allowing us to get close to an approximation of what the actual Fourier transform would look like.

The upper half of the spectrum has been shifted down to the negative frequencies to give a graph that shows what the shape of a spectral line would look like if the underlying (i.e., un-windowed) signal was a perfect sinusoid.

Figure 4.8 also rather nicely illustrates the effect of **padding** to increase the frequency resolution of the DFT. The original functions (of the rectangular and hamming windows) used to generate the figure are 32 samples long. Padding these out to a total length of 1024 samples (simply by adding zeroes) before taking the DFT means that we have 32 points in the DFT for every multiple of f_s/N, rather than just 1, which is why they are plotted as lines rather than discrete points, which would be too close to resolve in a diagram anyway.

4.7 Further reading

Brigham, E. Oran, *The Fast Fourier Transform*, Prentice-Hall, 1974. (Available in the JF Allen Library, classmark QA404.B8.)
Smith, Stephen W., *Digital Signal Processing - A Practical Guide for Scientists and Engineers*,

Newnes/Elsevier, 2003. Rather remarkably, the whole book is available free at dspguide.com: the site does carry adds, but they are discreet, and tend to be for things like spectrum analysers, so at least related to the course topic!
Lyons, Richard G., *Understanding Digital Signal Processing*, 3rd Ed., Pearson, 2011
James, J. F., *A Student's Guide to Fourier Transforms*, 3rd Ed., Cambridge, 2011. (Available in the JF Allen Library, classmark QC20.7F67J2G11.)

Chapter 5

Noisy messages, surprises and redundancy

5.1 Doubtful information and errors

Random noise makes the result of any quantitative measurement uncertain to some extent. This lack of perfect precision is often referred to in terms of producing a given level of **error** in any result. Here, we'll look at how noise affects our ability to communicate information. To see how noise affects information transmission, consider the situation shown in figure 5.1. This shows a pretty bare-bones cartoon of an information communication channel: we have a transmitter, a receiver, a channel that connects them and noise somehow getting added to the signal. In this case a message is being sent as a stream of binary digits, i.e. it is in the form of a *serial digital* signal. The transmitter uses one voltage level, V_1 to signal a binary '1' and another, V_0, to signal a binary '0'. The information is therefore carried by the voltage

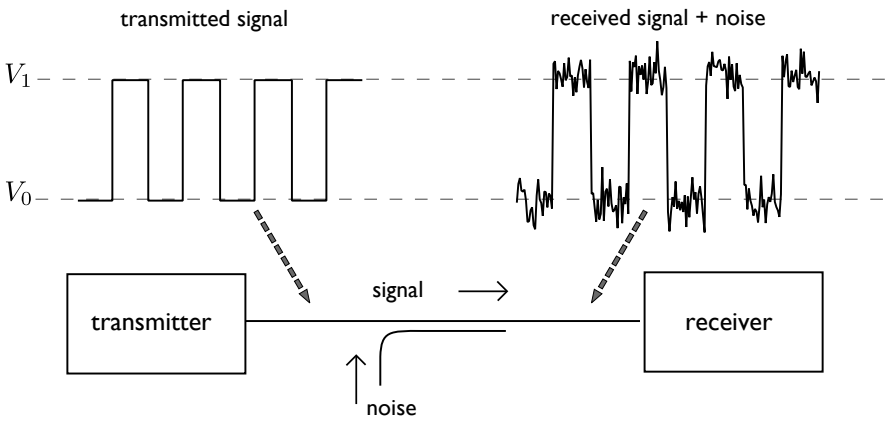


Figure 5.1: Simple digital communication over a noisy channel.

pattern. Some random noise is introduced during transmission. As a result, the received signal is a combination of the intended signal voltage pattern and this added noise. For simplicity, we assume that the transmitter and receiver are in themselves perfect and don't generate any noise of their own. In reality this won't be true, but for our purposes it doesn't really matter where the noise comes from. Any actual noise coming from the transmitter or receiver circuits would have an identical effect to the same total noise voltage injected onto the channel from an external source.

In the absence of any noise, the receiver could repeatedly measure the input it sees and decide it has received a '1' if the voltage is V_1, or a '0' if the voltage is V_0. The noise, however, means that the receiver will hardly ever see a voltage exactly equal to *either* V_1 or V_0. We need some other recipe for deciding what we've received and the simplest way is to define a sensible *decision level*, V'. Here we've chosen V' to be mid-way between V_0 and V_1. (More sophisticated schemes that further restrict the legal voltage range for the two symbols are more widely used.) The receiver can now operate by regarding any voltage *greater* than V' as representing a '1' and any voltage *less* than V' as representing a '0'. The results of decoding a noisy digital signal in this way can be understood by looking at figure 5.2. Consider the situation where the receiver sees a combination of the actual transmitted voltage level, plus some noise. (It doesn't really matter where the noise gets it: it could be at the point of transmission, or interference picked up as the signal propagates, or noise generated in the receiver itself. All that matters is that the voltage about which we have to make a decision has noise on it.)

We'll assume that the noise is Gaussian, and that the rms value of the noise voltage is σ. (Remember, the *average* value of the noise voltage will be zero, which doesn't tell us very much.)

The effects of the noise can be assessed by making a very large number of measurements of the received voltage levels and plotting a probability distribution of the results. The two plots in figure 5.2 show the distribution of voltages seen by the receiver when the transmitter is sending V_0 and when the transmitter is sending V_1. For each case, the the resulting spread of voltages has a mean at the intended signal voltage. The shape of these two curves is Gaussian, as we've assumed that the noise has a Gaussian PDF.

Given how the receiver decides whether it has seen a '1' or a '0', we can predict the frequency of mistakes by calculating the fraction of each plot which is on the wrong side of V', i.e. in the shaded areas of the two curves in figure 5.2.

When the transmitter is trying to send V_1 the probability, $p(V)$, with which the received voltage is seen to be in some interval, $\mathrm{d}V$, centred at some voltage V will be

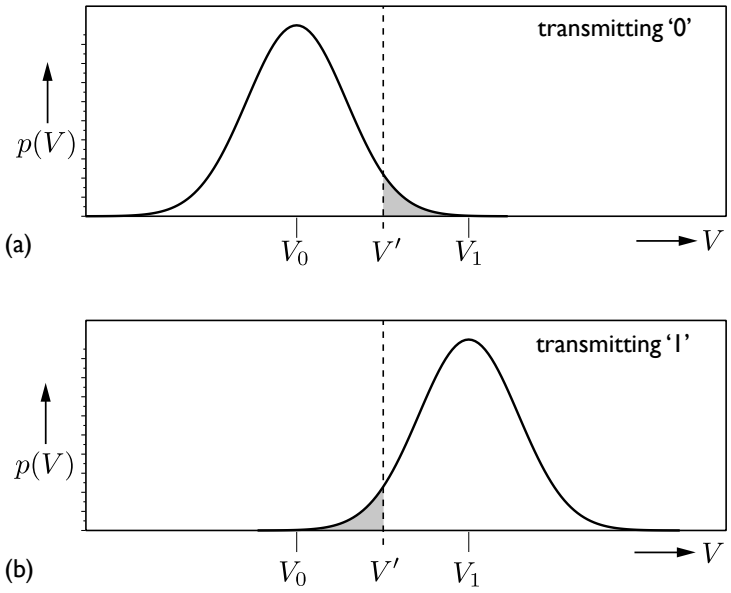


Figure 5.2: The effect of noise on the probability density of voltages seen by a receiver when a (a) '0' or (b) '1' is being sent by the transmitter.

$$\begin{aligned} p(V) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-(V-V_1)^2}{2\sigma^2}\right] \mathrm{d}V \\ &= A \exp\left[\frac{-(V-V_1)^2}{2\sigma^2}\right] \mathrm{d}V \end{aligned} \tag{5.1}$$

which is just the same as equation 3.2, shifted along the voltage axis so that the distribution is centred on V_1, and I'm using A as I'm too lazy to re-write all the constants all the time.

When V_1 is being sent, the chance, C_1, that it will be correctly received is determined by the fraction of the distribution which lies *above* V'. This may be obtained by integrating over the appropriate part of the curve to obtain

$$\begin{aligned} C_1 &= \int_{V'}^{\infty} p(V)\mathrm{d}V \\ &= \int_{V'}^{\infty} A \exp\left[\frac{-(V-V_1)^2}{2\sigma^2}\right] \mathrm{d}V \end{aligned} \tag{5.2}$$

Similarly, the chance that V_0 will be correctly received (C_0) is determined by the fraction of

the distribution which lies *below* V' when V_0 is being sent:

$$\begin{aligned} C_0 &= \int_{-\infty}^{V'} p(V)\mathrm{d}V \\ &= \int_{-\infty}^{V'} A\exp\left[\frac{-(V_0-V)^2}{2\sigma^2}\right]\mathrm{d}V \end{aligned} \tag{5.3}$$

In statistics and probability, this is a routine problem, so we inherit some convenient shorthand for these integrals: the **error function** is defined as

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-z^2}\mathrm{d}z \tag{5.4}$$

and after a bit of algebraic shenanigans, we can write our expressions for C_0 and C_1 as

$$C_1 = \frac{1}{2}\left[1+\mathrm{erf}\left(\frac{V_1-V'}{\sqrt{2}\sigma}\right)\right] \tag{5.5}$$

and

$$C_0 = \frac{1}{2}\left[1+\mathrm{erf}\left(\frac{V'-V_0}{\sqrt{2}\sigma}\right)\right] \tag{5.6}$$

The useful bit here is that erf is a function that we can look up (hurrah!) and in order to evaluate the necessary integrals. erf can't be evaluated analytically, but it can be simply looked-up. It is readily available in computer languages[1]. We can now use the above expressions to see how often the receiver will pick up the correct signal level in the presence of some noise.

The error function is plotted in figure 5.3.

Since we defined V' to be midway between V_0 and V_1, $C_1 = C_0$, and we only need to look at how *one* of C_1 and C_0 depends on the noise level and the chosen voltages. The amplitude of the transmitted signal is

$$V_S = \frac{V_1 - V_0}{2} \tag{5.7}$$

The rms value of the typical noise voltage is σ. Since $V' = (V_1 - V_0)/2$, we can say that the fractional chance of each '1' *or* '0' being correctly received will be

$$C = \frac{1}{2}\left[1+\mathrm{erf}\left(\frac{V_S}{\sqrt{2}\sigma}\right)\right] \tag{5.8}$$

As we'd expect, we can see that C approaches unity when the signal amplitude is a lot greater than that of the noise. This makes sense.

[1] For example, in Excel it's `ERF()`, in Mathematica it's `Erf[]` and in C it's `erf()`.

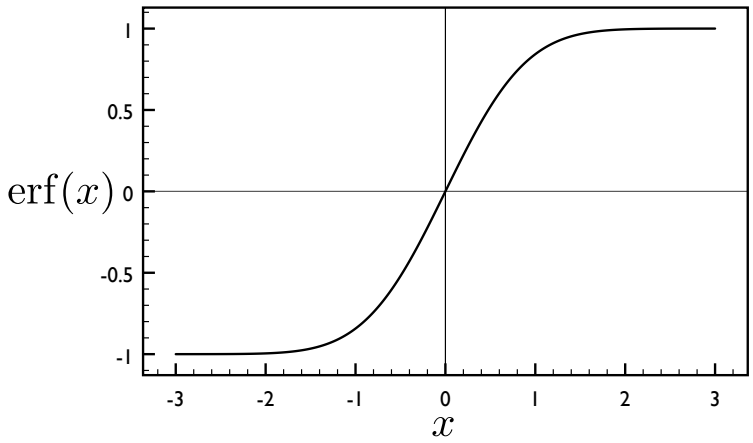


Figure 5.3: The error function $\mathrm{erf}(x)$ as a function of x. For $|x| > 3$, it is so close to the limiting values of -1 and 1 that the difference from these values would be imperceptible on the scale used.

Somewhat more curious is that when the signal amplitude is *zero*, $C = 1/2$, so even when no signal is transmitted, the receiver will correctly pick up 50% of the '1's and '0's in the signal pattern.

This is not as odd as it may seem at first sight, but can be explained by the following analogy. Imagine that we didn't bother with a receiver at all, but just kept tossing a coin. Every time the coin comes up heads, we decide that we the message contains a '1', and similarly every tail means a '0'. We can thus build up a pattern of '1's and '0's without bothering to look at the actual signal. Since there are only two possibilities, whenever we throw the coin we've got a 50% chance of getting the correct result, so 50% of the '1's and '0's in our coin generated version of the message will be correct.

The key point to realise here, however, is that this *doesn't* mean that we've received 50% of the actual information as we have no way of knowing *which* 50% of the coin generated '1's and '0's are the correct ones. This is just the same as if we'd used random noise to make the receiver perform the equivalent of our coin tossing to generate a random stream of bits.

This also demonstrates an extremely important feature of the way information is communicated and processed. The amount of information we get when we pick up a signal doesn't just depend upon how may bits we've gathered, but also upon *how confident we can be that each bit is correct*. The amount of information received depends on how certain we are that the pattern is correct. If we're only 50% certain and there are only two possibilities, we don't actually have *any* real information since any other outcome is just as likely to be the correct one.

Equation 5.8 is all well and good, but in practice in most real situations it is more convenient to deal with signal and noise *powers*.

A square wave of amplitude V_S will always have an instantaneous voltage of either $+V_S$ or

$-V_S$, so has a mean power, S, given by

$$S = \frac{V_S^2}{R} \tag{5.9}$$

where R is the resistance across which the signal appears. (Note: the relationship between average power and amplitude depends on the *shape* of the signal, i.e. the relationship is different for a square wave and a sine wave.) Thus, we can use

$$V_S = \sqrt{SR} \tag{5.10}$$

to replace the signal voltage in the above expression. We already know that σ is the rms value of our noise, so the noise power level will be

$$N = \frac{\sigma^2}{R} \tag{5.11}$$

so we can say that

$$\sigma = \sqrt{NR} \tag{5.12}$$

We can now express C in terms of the signal and noise powers:

$$\begin{aligned} C &= \frac{1}{2}\left[1 + \mathrm{erf}\left(\sqrt{\frac{S}{2N}}\right)\right] \\ &= \frac{1}{2}\left[1 + \mathrm{erf}\left(\sqrt{\frac{\mathrm{SNR}}{2}}\right)\right] \end{aligned} \tag{5.13}$$

Thus, we have an expression that tells us the chance that the bits will be received correctly in terms of two easily measurable quantities, the signal power, S, and the noise power, N. Note that while useful for illustration of this particular example, equation 5.13 is not general: it only applies to the case where the signal is a square wave wave and the decision level is halfway between the two defined voltage levels for 0 and 1.

The fraction of bits correctly received is plotted against the SNR (in dB) in figure 5.4.

Whenever possible, we should make the SNR as large as we can to minimise the possibility of errors. If this is done, we can often neglect the information loss produced by random noise. The SNR must always, however, be *finite*, so we can never get rid of the problem altogether. Despite this, an SNR of just 10 (equivalent to 10 dB) gives $C = 0.9992$, i.e. 99.92% of the bits in a typical signal with this signal to noise will be correctly received. A slightly better SNR of 25 (equivalent to 14 dB) would give us an **error rate** of just 3 bits in every 10^7.[2] 3 bits wrong in 10 million can still be a problem if we don't know about them, and depending on the context in which they occur, it couls be a complete disaster.

[2]Other amusing calculations: if we used such a scheme with an SNR of 20 dB to transmit data at a rate of 1 Gbits/sec, then the typical error rate would be one in four million years...

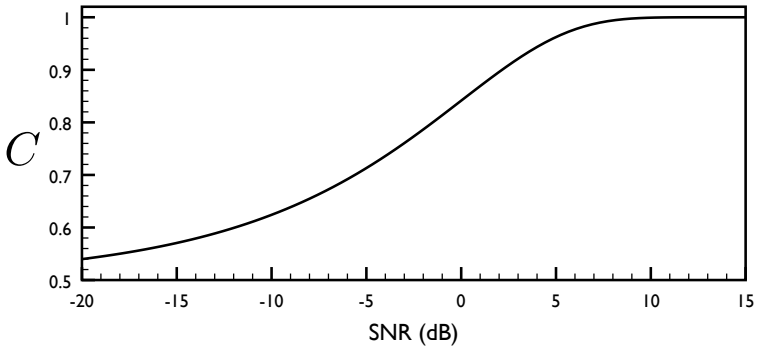


Figure 5.4: The fraction of '1's and '0's correctly received as a function of the signal-to-noise ratio *in decibels* for a simple binary scheme with Guassian noise and a decision level mid-way between V_0 and V_1.

5.2 Surprises and redundancy

When dealing with the effects of errors on *messages*, as opposed to single bits, we need to take into account how effectively we are using our encoding system to send useful information and how important the messages are.

Some messages are quite surprising, whereas others are so predictable that they tell us almost nothing. Some errors in some messages may be such that they actually give a clue as to what the error is. If we consider spelling mistakes, for example, sometimes we can tell that a statement is clearly wrong, but be able to guess pretty reliably from the context what the message should have been. We can also have statements that contain words that are spelled correctly, but by changing a few key letters, would still be spelled correctly, but mean something completely different. We can even have correctly spelled statements that seem unambiguous, but that are simply incorrect: then it would be hard to tell that there was an error at all.

For example, consider the sentence: 'The cheqe to pay for my gas bill is is the post'. There should be a 'u' after the 'q'. Here is it easy to work out that there is a missing letter, so this error (in the context of the English language) is easy to spot and subsequently correct.

Sticking with this example, we could remove the 'u' after q'q' in most English words, and leave the meaning intact (i.e. we'd get lots of spelling mistakes, but are unlikely to end up with valid words that confuse the intent of a message). In this particular case, the 'u' is **redundant**: if we encounter a 'q' in English, we know that there is (almost certainly) going to be a 'u' following it, without having to read the next letter.

The above sentence has another error in it: the word "is" is duplicated, and the second instance of it should read "in". The context here probably means that you know what it should mean. However, the rule that is broken here is now not one of simple spelling, but of grammar: this means that the second error requires a touch more sophistication to detect.

Clearly, it is valuable to choose a system of coding which makes errors as obvious as possible and allows us to correct received messages. To see how it is possible to produce systems which do this, we need to analyse **redundancy** and its effect on the probability of a message being correctly understood.

Earlier, we saw that the amount of information in a message can be expected to increase with $\log_2$ of the number of code symbols available. This makes the assumption that *all the available symbols are used* (a symbol which isn't used might as well not exist). It also assumes that *all the symbols are used with similar frequency*. Hence, the probability, p of a particular symbol appearing if we have M different symbols available would be

$$p = \frac{1}{M} \tag{5.14}$$

We can therefore say that the amount of information would vary with

$$\log_2(M) = \log_2\left(\frac{1}{p}\right) = -\log_2(p) \tag{5.15}$$

Consider the situation where we use a set of M symbols, $x_1, x_2, x_3 \ldots x_M$ for sending messages. By collecting a very large number of messages and examining them, we can discover how often each symbol tends to occur in a typical message. We can then define a set of probability values

$$p_i \equiv \frac{n_i}{N} \tag{5.16}$$

if n_i is the number of times that symbol x_i occurs in a typical message N symbols long. *Note that N needs to be a very large number if we want the p_i that we calculate to accurately reflect the underlying probability distribution for the different symbols.* If all the symbols tend to appear *equally* often, then we can expect that all p_i will be equal to $1/M$. More generally, however, the symbols will appear with *different* frequencies and each p_i will indicate how often each symbol appears. (This is certainly the case with the English language.)

When all the symbols are equally probable, the amount of information provided by each individual occurrence in the message will be $\log_2(M)$. In a message N symbols long, therefore, the *total* amount of information in a typical message would be

$$\begin{aligned} H &= N\log_2(M) \\ &= -N\log_2(p) \end{aligned} \tag{5.17}$$

as in the case for equal frequencies for all symbols, $p_i = p = 1/M$. This expression giving the total amount of information in terms of symbol probabilities indicates how we can define the amounts of information involved when the symbols occur with *differing* frequencies. We can then say that the amount of information provided just by the occurrences of, say, x_i, will be

$$\begin{aligned} H_i &= -n_i\log_2(p_i) \\ &= -Np_i\log_2(p_i) \end{aligned} \tag{5.18}$$

From this expression, we can see that the *smaller* the probability of a particular symbol occurring, the more informative it will be when it appears. 'Surprising' (i.e. rare) messages convey more information than predictable ones! The total amount of information in the message will therefore be the sum of the amounts carried by all the symbols:

$$\begin{aligned} H &= \sum_{i=1}^{M} -n_i \log_2(pi) \\ &= -\sum_{i=1}^{M} N p_i \log_2(p_i) \end{aligned} \tag{5.19}$$

From equation 5.18 we can say that every time x_i appears in a typical message it provides a typical amount of information *per symbol occurrence* of

$$\begin{aligned} h_i &= \frac{H_i}{n_i} \\ &= -\log_2(p_i) \end{aligned} \tag{5.20}$$

Note that we're using H to denote total amounts of information and h to denote an amount per individual symbol occurrence. From equation 5.20 we can say that the amount of information per symbol occurrence, averaged over *all* the possible symbols

$$\begin{aligned} h &= \frac{H}{N} \\ &= -\sum_{i=1}^{M} p_i \log_2(p_i) \end{aligned} \tag{5.21}$$

In general, this averaged value will differ from the individual h_i *unless all the symbols are equally probable*. It's interesting to notice that the form of the above expressions is similar to those used for entropy in statistical mechanics. Many books therefore use the term *entropy* for the measure of information in a typical message or code.

The above argument gives us a statistical method for calculating the amount of information conveyed in a typical message. Some messages, of course, aren't typical. The information content of a specific message may be rather more (or less) than is usual. The above expressions only tell us the amounts of information we tend to get in a typical message.

Let's now consider a specific message N symbols long where each symbol, x_i actually occurs A_i times. The amount of information provided by each individual symbol in the message is still $-\log_2(p_i)$, but there are now A_i of these, not the Np_i we'd expect in a typical message. We can therefore substitute A_i into equation 5.19 and say that the total amount of information in *this particular message* is

$$H = -\sum_{i=1}^{M} A_i \log_2(p_i) \tag{5.22}$$

In order to convey information, each of the symbols we use must have a defined meaning, otherwise the receiver can't make sense of them. This is another way of saying that the

number of available symbols, M, must always be *finite*. Since any particular symbol in a message must be chosen from those available, then we can say that

$$\sum_{i=1}^{M} p_i = 1 \tag{5.23}$$

In most cases the chance of a particular symbol occurring will depend to some extent upon the previous symbol. (For example, in English a 'u' is much more likely to follow a 'q' than any other letter. Or if we are sending a signal in the form of an analogue voltage along a wire, there is a limit to how quickly the voltage can change.) Some combinations of symbols occur more often than others (again using English as an example, we're more likely to see 'th' than 'xq'). The term **intersymbol influence** is used to describe the effect whereby the presence (or absence) of some symbols in some places affects the chance of other symbols appearing elsewhere.

To represent this effect, we can define a *conditional probability*, $p_{i\to j}$, to be the probability that the jth symbol will follow once the ith symbol has appeared. The chance that the symbol combination $x_i x_j$ will appear can then be assigned the *joint probability*

$$p_{ij} = p_i p_{i\to j} \tag{5.24}$$

Just as the amount of information provided by a symbol taken by itself depends upon its probability, so the *extra* information provided by a following symbol depends on how likely it is once the previous symbol has already arrived. In English, for example, when we encounter a 'q', we can be almost certain that the next symbol will be a 'u', so after receiving the 'q', the 'u' doesn't provide much extra information. The 'u' can be said to be **redundant** once the 'q' has arrived. It is important to notice, however, that although it doesn't provide any real extra information, it is a useful way of checking the correctness of the received message.

The term *conditional entropy* is often used to refer to $h_{i\to j}$, the average amount of information which is communicated by the jth symbol after the ith has already been received. Since h is proportional to $\log_2(p)$, the *joint entropy*, h_{ij}, (which is the amount of information provided by this pair of symbols taken together) must simply be

$$h_{ij} = h_i + h_{i\to j} \tag{5.25}$$

If we wish to maximise the amount of information in a typical message, then we'd like every symbol and combination of symbols to be as *improbable* as possible, i.e. to minimise all the p_i. Equation 5.23, however, tells us immediately that when we make one symbol(or combination) less likely, some others must become *more* likely. The symbols can't *all* be made less likely without adding new ones. The general effect of intersymbol influence is to reduce the amount of information per symbol. We can therefore expect that the information content of a message is maximised when the intersymbol influence is *zero*. When this is the case,

$$h_{ij} = h_i + h_j \tag{5.26}$$

i.e. the amount of information communicated by the two symbols is simply the sum of the amounts due to each symbol alone. In this situation, none of the symbols are redundant. Since this is the *best* that we can hope for, it follows that in general

$$h_{i\to j} \leq h_j \tag{5.27}$$

which simply means that the extra information produced by the following symbol can never exceed that which it would have as an individual if there were no intersymbol influence.

5.3 Chapter 5: take-home messages

We've seen how noise can lead to random **errors** when we communicate a signal. These random errors mean we can never be absolutely certain that the information we've received is absolutely correct. We can make a statement as strong as 'never' because noise is *always* present in real systems. The amount of 'information' contained in a signal pattern depends on *how confident* we can be that we have correctly received it, and we've looked at a simple example of how we can quantify that confidence.

You should now understand that the amount of information contained in a message depends upon the probabilities (or typical frequencies of occurrence) of the different available symbols. It should also be clear that the chance of errors being received depends upon the signal to noise ratio, and *how* it depends on the signal to noise ratio.

The amount of information in a **specific** message can differ from a typical or average message, and we've used the term 'surprising' to refer to a message in which symbol frequencies differ from their typical values. Inter-symbol influence reduces the information content, but can help us check whether a received message is correct.

Chapter 6

Detecting and correcting mistakes

6.1 Errors and legality

We have seen earlier that random noise will tend to reduce the amount of information collected by making us uncertain as to how correct the resulting message pattern is. We've also seen, in the context of written English at least, how *redundancy* can provide a way to check for mistakes and, in some cases, correct them. One of the advantages of digital signal processing systems is that they are relatively (note: not totally) immune to the effects of noise: A signal to noise ratio of just 10:1 is enough to ensure that over 99.9% of digital bits will be correct.

For short, unimportant, messages this level of immunity from errors is fine, but it isn't good enough for other situations. For example, consider a computer loading a 1 MB (8-ish million bits) program: the error rate mentioned above would mean that the loaded program would contain over 6000 mistakes, which would almost certainly cause the program to fail, and possibly bork the computer. It's also important to realise here that the term 'error rate' doesn't mean that the errors appear at regular intervals. (If it did, we'd know which bits were incorrect and correct them easily enough!) The errors will be randomly placed, so the rate just indicates what fraction of the bits are likely to be wrong and doesn't tell us where they are.

The rate at which errors occur can be reduced by improving the SNR, but there is another way based on the deliberate use of redundancy. By introducing some inter-symbol influence, we can make some patterns of bits **illegal**, i.e. we arrange that they can *only occur as the result of a mistake*. This means that we can detect that the signal pattern contains an error. The main disadvantage is that we have to reduce the amount of information we trying to get into a given message since some of the symbols are now being used to 'check' others, rather than sending any extra information of their own.[1] One of the simplest ways to deal with errors using deliberate redundancy is simply to repeat the message: the two versions can

[1] It can be argued that this doesn't really matter since, if we didn't do anything about it, random noise would destroy some of the information anyway, and we wouldn't know about it.

then be compared to see if they are the same.

If the probability that any particular bit or symbol in a message is correct is C, then the probability that it's actually an error must be

$$E = 1 - C \tag{6.1}$$

as it must be either right or wrong. So, if we send the message twice and compare the two copies, the chance that *both* copies have a symbol or bit error in the same place will be E^2.

For example, if a system has an SNR such that $C = 0.999$, then E is 0.001. The chance that both copies of a specific bit are wrong is therefore $E^2 = 0.000001$, so in a typical pair of repeated messages, there is only a 1 in 10^6 chance that both copies of any particular bit will be wrong. The chance that a particular bit will be wrong in one copy and correct in the other will be $CE = 0.000999$, and that chance that both are correct will be $C^2 = 0.998001$.

When we compare two versions of a long message we therefore typically find that a fraction $(C^2 + E^2)$, $\equiv 99.8002\%$ of the bits agree with their copies, and a fraction $2CE$, $\equiv 0.1988\%$ differ. Where they differ we have detected the presence of the error which caused the disagreements and know where in the message they appear. Once we know about the errors, we could take appropriate action, for example asking the transmitter to repeat the parts of the message in question. The sending of the 2nd copy of the message is mostly redundant because they should both contain the same information. If we'd only sent one copy, however, it would contain about 0.1% mistakes, but we wouldn't know about them, so we need redundancy so that we can spot mistakes and takes steps to recover the lost information.

Using a pair of messages is still not a perfect system as there is still a fraction E^2 of errors that we won't spot as both copies will have been changed in the same place.

Although spotting differences between the copies tells us where the errors are, it doesn't tell us which of the two versions is correct. If we moved to sending the message three times, the chance that all copies of a specific bit being correct are $C^3 \equiv 99.7\%$. There are three chances for any one version to be wrong, so the chance is $3C^2E \equiv 0.2994\%$. Similarly, the chance that two versions both have an error is $3CE^2 \equiv 0.0002997\%$, and that chance that all three are wrong is $E^3 \equiv\ 10^{-7}\%$.

One effect of the above method is to reduce the undetected error rate (E^3) even further,. However, the main benefit is that nearly all the errors can now be *corrected* as in most of the error cases, only one of the copies is wrong, so the receiver can go with the majority vote to decide what the signal should be. This will still yield occasional mistakes (when two out of the three copies are wrong) but from the figures shown this would only happen for about one correction in a thousand.

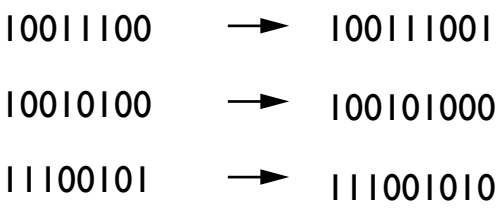


Figure 6.1: Adding parity bits to three 8-bit words to get three 9-bit words.

6.2 Parity and blocks

The disadvantage of sending the message three times is that we are using extra symbols to repeat it rather than to carry new information. This makes the method relatively inefficient and slow. Fortunately, there are other methods for detecting and correcting errors which don't reduce the overall information carrying capacity so much. One of the most common digital techniques for this is the use of **parity** bits, but before we go on to look at these we'll consider the concept of binary *words*.

From our previous discussions, you should already be familiar with the idea of using a set of binary digits (*bits*) to represent information. It's usual to refer to groups of 8 bits as a **byte** of information (which stems from early computers which handled 8 bits at a time). More generally, the term **word** has come to mean a group of bits which carry a convenient amount of information. Each binary word can be regarded as a digital *symbol*. Unlike the term 'byte', 'word' can mean any convenient number of associated bits. To show how *parity checking* can be used to detect (and correct) errors, imagine a system where the information is initially held as a series of 8-bit words. The system may want to transmit (or process in some other way) a series of words. The *parity* of each word can be defined to be *odd* or *even* depending on how many '1's it contains. On this basis, the word 10110100_2 has even parity, and the '$_2$' at the end of the number just indicates that it's in binary notation. We can now add an extra bit onto each word to represent its parity. For example, if the word was even we could add a '1' and if it was odd we could add a '0'. This converts our 8 bit words into 9-bit words, as shown in figure 6.1. We then transmit or process these new *9-bit* words instead of the original 8-bit ones.

The extra *parity bit* that we've added on to every word doesn't carry any fresh information, it simply confirms the parity of the first 8 bits in the new 9-bit word. This means that the patterns we transmit are now partially redundant and this redundancy can be used by the receiver to check for errors. Note that with the system we've chosen, every *legal* 9-bit word has an odd number of '1's. The receiver can now read each 9-bit word as it arrives and check that its parity is odd, as expected.

If random noise changes one of the bits in in a word during transmission, then the received word will have an even number of bits, so the receiver can recognise that the word is *illegal*, i.e. not a pattern that the transmitter would send and that it must contain an error. Note

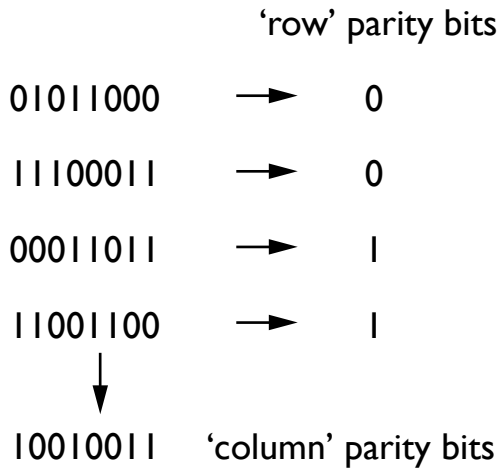


Figure 6.2: A block of information and the associated parity bits calculated from it.

that parity bits could be put anywhere in the word, and don't necessarily need to adhere to the scheme we've shown here: this is just one possible example.

Using this example, we can define some ways to quantify the degree of redundancy in the coding system that we use to transmit information. In the above case, each 9 transmitted bits contains 8 bits worth of real information. We can say that the *efficiency* of the coding system is

$$\text{efficiency} = \frac{\text{no. of information bits}}{\text{total no. of bits}} \tag{6.2}$$

In this case, the ratio is 8/9, so the transmission system has an efficiency of 0.888. The *redundancy* can be defined as one minus the efficiency, i.e. 0.111 in this case. This parity checking system can detect occasional 1-bit errors, but can't *correct* them. To do that we can use a slightly more complex approach based on what are called **block codes**. A simple example is shown in figure 6.2.

We now generate a set of 'row' parity bits for checking each word, and a set of 'column' parity bits. In the example shown, the original block of $(4 \times 8) = 32$ bits is used to produce 12 extra parity bits, and all 44 bits are transmitted to our receiver. To illustrate what happens when a random error occurs during transmission, we'll assume that the received version of the data in figure 6.2 is that shown in figure 6.3.

The receiver collects the data bits and parity bits sent by7 the transmitter, then calculates its own version of what the parity bits *should* be from the data bits it has actually received. In our example, one of the bits has been changed from a '0' to a '1' during transmission. As a result, the received and computed parity bits will not agree and the receiver thus knows that there is a mistake in the received block of data. One parity disagreement identifies the row that the incorrect bit is in and the other identifies the column: the error can thus be both detected and corrected. The ability of block coding to allow the detection and correction of errors is an important feature of modern information processing.

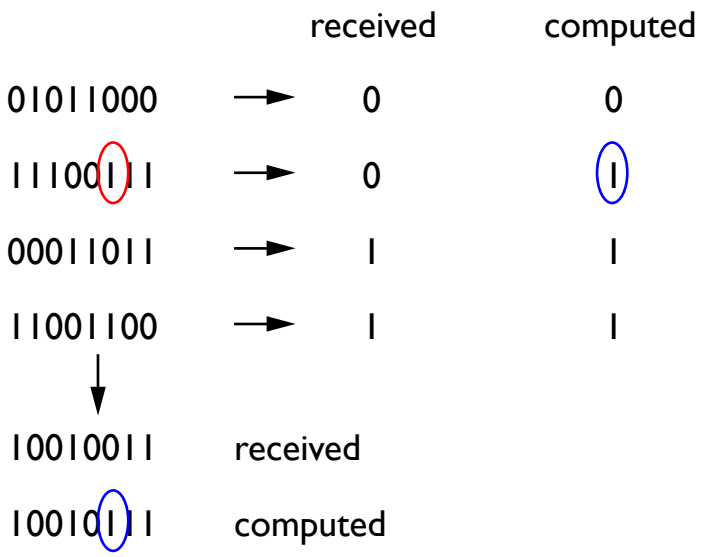


Figure 6.3: Differences between received and calculated parity bits when errors as received.

Note that our choice of block dimensions is arbitrary, and blocks can have more than two dimensions. You should be able to see (and work out) how different block sizes will lead to different levels of redundancy.

The choice of block arrangement thus depends on how concerned we are about the effects of noise: the example discussed above runs into trouble if a block contains *more than one error*. 2-bit errors can be detected, but only 1-bit errors can be corrected.

In general, the error detecting and correcting ability arising from block coding can be defined in terms of a measure called **the minimum hamming distance**. Block codes work because some transmitted patterns of the binary digits are illegal. The *hamming distance* between any pair of legal words is defined as the number of bits which have to be changed to convert one (legal) word into another (legal) word. (Remember, conversion by mistakes into *illegal* words can always be detected.) The *minimum hamming distance* is defined as the lowest hamming distance we can find between *any* pair of legal words in the chosen code system. This provides us with a number which tells us how well a coding system can cope with errors.

The properties of well designed coding systems with various minimum hamming distances are defined in table 6.1.

In a given situation, we'd start by deciding how many errors at a time we need to be able to spot or correct, then use the MHD to tell us how many illegal patterns have to 'surround' each legal pattern. This then tells us how much redundancy and how many parity bits we need.

MHD = 1	No error immunity
MHD = 2	Detects one error, no correction
MHD = 3	Detects and corrects one error
MHD = 4	Detects up to two errors, can correct one error
MHD = 5	Detects two errors, can correct two errors

Table 6.1: The error detection/correction capabilities for coding systems with different minimum hamming distances.

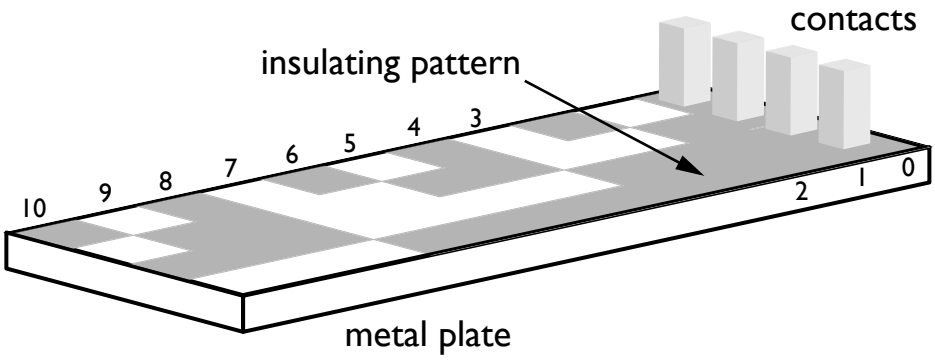


Figure 6.4: A simple linear position encoder based on a conducting plate with an insulating pattern on top.

6.3 Choosing a coding system

There is a huge variety of coding systems, and many are designed to have specific features in specific situations. Most are designed to combat random errors as we saw in the previous section on block codes. We'll now look an example where we can use a coding system to cope with *systematic* errors.

Figure 6.4 shows a simple linear position encoder based on a conducting plate with an insulating pattern on top. A row of contacts make or don't make electrical contact with the plate depending on their position as the plate is moved: if a contact is above bare metal, then current can flow through it to the plate, and by measuring whether or not each contact is connected to the plate we can generate a binary number related to the contacts' position along the plate.

There is a key disadvantage to the straightforward binary encoding employed here, namely that it may require more than one bit to change simultaneously. Each time a bit changes, the system may momentarily read that bit to fluctuate between one value and the next (imagine what may happen as a contact moves from an insulating patch to a conducting patch.)

Too make any errors associated with a transition between a conducting patch and an insulating

Number	Gray	Walking		Number	Gray	Walking
0	0000	00000000		8	1100	11111111
1	0001	00000001		9	1101	11111110
2	0011	00000011		10	1111	11111100
3	0010	00001111		11	1110	11111000
4	0110	00011111		12	1010	11110000
5	0111	00111111		13	1011	11100000
6	0101	00111111		14	1001	11000000
7	0100	01111111		15	1000	10000000

Table 6.2: Two coding systems for a linear position encoder that involve only one bit changing with each change in position.

patch easily detectable, we can instead choose a system of encoding each position such that in moving from one position to the next, *only one bit will change at a time*. Thus, the system can know that if it detects a change in two or more bits, it is due to a momentary fluctuation and, for example, it can re-measure the pattern. Table 6.2 shows two methods of encoding the position using binary data such that only one bit changes at a time, called *Gray* and *walking* codes.

The walking code obviously uses redundancy to achieve its effect, whereas the Gray code is simply a rearrangement of the binary numbers for 0 to 15. The reason that the Gray code can avoid the errors of the straightforward binary encoding *without* introducing any redundancy is that the errors thrown up by the encoding system are *systematic*, i.e. due to the way in which it works. Because any errors *are not random*, they can be overcome without a loss in efficiency. Note that is is not always possible to correct systematic errors, and whether or not it is will depend on the nature of the systematic error.

The leads to an important distinction between systematic and random errors: the former may (depending on the system) be overcome without any redundancy necessary, whereas the latter will always require redundancy in order to detect and or correct errors.

Another method we can use to protect ourselves from mistaking received errors for reliable information is a technique know as **soft decision making**. This method depends upon being able to spot 'suspect' bits, and when combined with a block-checking code is a powerful way of reducing the effects of random noise.

This technique requires thought about the *transmission* system rather than the *encoding* system. A simple example is electronic digital transmission along metal wires. Here bits can be lost due to momentary loss of contact as well as random noise. This can produce **bursts** of errors where a *series* of bits is missed The most direct method of transmitting digital data is to send, say, TTL voltage levels[2], where between 0 and 1 V denotes '0' and between 3

[2]TTL stands for transistor-transistor logic,just one of many families of digital logic. Not terribly widely used

and 5 V denotes '1'. A momentary loss of signal may produce either 0 V (received as '0') or allow a receiving TTL gate to float high (giving a received '1'). Thus, depending on the receiver circuit used, a temporary loss of data will look like a string of '0's or '1's.

Various systems exist to avoid this. One well established transmission system is known as **RS 232**. Here a positive voltage in a particular range (3 to 15 V) signals '0' and a negative voltage in a particular range (-3 to -15 V) signals '1'. The clever part here is that a momentary signal loss will give *zero* voltage, which the receiver can label as 'don't know'. This system effectively uses *three* logic levels to send binary data, although the transmitter only ever attempts to use two of the levels.

In practice, this technique does have one potentially significant disadvantage which can be illustrated using the example of the RS232 logic levels. In the absence of any attempt to detect the 'don't know' condition the receiver could decide whether a '1' or '0' was being communicated by checking whether the received voltage was negative or positive. Random noise would therefore have to change the voltage level by at least 3 V in order to produce an error.

In order to be able to sense message interruptions the receiver must be designed so as to respond to some range of voltages, $\pm X$, centred on 0 by deciding that the signal level is 'undefined'. Random noise now only has to alter the received voltage by an amount $(3 - |X|)$ V, to make a bit appear unreliable. Similarly, the random noise only needs to produce a momentary voltage fluctuation of more than $|X|$ to make a momentary loss of signal as an apparently reliable '1' or '0'. This means that we can't avoid this problem by making X very small without giving up the ability to spot when data is failing to arrive. As a result, assigning an intermediate range of levels to mean 'undefined' leads to an increase in the frequency of errors produced by random noise. However, provided that the SNR is high, this increase can be small enough to be an acceptable price for being able to sense momentary data losses.

6.4 Chapter 6: take-home messages

We have seen that there are mechanisms that we can use to detect and correct random errors. Coding schemes can be designed to minimise the generation of **systematic** (i.e. non-random) errors *without* any redundancy.

The addition of parity information, as well as techniques that simply rely on repetition provide protection against errors at the expense of transmission **efficiency**, i.e. they add **redundancy**. You should understand that the amount of protection a *particular* scheme offers against random errors is related to the redundancy: the more protection, the more redundancy. (For example, surrounding legal symbols with larger numbers of illegal symbols.) The ability of a coding system to detect and correct errors can be measured in terms of its **minimum**

these days as a technology.

hamming distance.

Chapter 7

The information carrying capacity of a channel

7.1 Signals look like noise

One of the most important practical questions which arises when we're designing or using an information transmission/processing system is "what is the **capacity** of this system?", i.e. how much information can it transmit or process in a given time. We formed a rough idea of how to answer this earlier on, but can now proceed to a better defined answer by deriving **Shannon's equation**, which allows the precise determination of the information carrying capacity of *any* signal channel.

Consider a signal which is being *efficiently* communicated (i.e. no redundancy) in the form of a time-dependant analogue voltage, $v(t)$. The pattern of voltage variations during a specific time interval T allows a receiver to identify which one of a possible set of messages has actually been sent. Ant any two instants, t_1 and t_2, during a message the voltage will be $v(t_1)$ and $v(t_2)$.

Using the idea of *inter-symbol influence*, we can say that because there is no redundancy then $v(t_1)$ and $v(t_2)$ will appear to be *independent* of one another provided that they're far enough apart in time to be worth sampling separately. This means that we can't tell what one value will be just from knowing the other. (Remember that for any *specific* message both $v(t_1)$ and $v(t_2)$ are of course determined in advance by the message contents, but the point here is that *the receiver* can't determine which of the possible messages has arrived until it *has* arrived. (It is important to make sure you understand this argument, which may seem merely tautological at first glance.)

This leads to the remarkable conclusion that a signal which is communicating information efficiently will vary from moment to moment in an *apparently random* manner, i.e. an efficiently encoded signal looks *very much like random noise!*

This is, of course, why random noise can produce errors in a received message. **The statistical properties of an efficiently signalled message are similar to those of random noise.**

In order to detect and correct errors therefore, we need to make the signal less 'noise-like', and this is of course where redundancy comes in. (If we have patterns in a message we know to be illegal, we know not to expect them, so the receiver does have some 'prior knowledge' about the likelihood of what a message will contain as there will be a set of patterns that should not be present.) Although redundancy (as added by parity bits, for example) reduces the system's information carrying capacity, it helps us distinguish signal details from random noise.

The main purpose of the present discussion, however, is to investigate the *maximum* possible capacity of a channel, so we have to avoid redundancy, allowing the signal to have the 'unpredictability' associated with random noise.

The amount of noise present in a system can be represented by its mean noise power:

$$N = \frac{V_N^2}{R} \tag{7.1}$$

where V_N is the rms noise voltage and R is the characteristic impedance of the channel/system. We can represent a typical message similarly in terms of its average signal power:

$$S = \frac{V_S^2}{R} \tag{7.2}$$

where V_S is the rms signal voltage.

As a real signal must have a *finite* power, for a given set of possible messages there must be some maximum possible power level, limiting V_S to some range. The *instantaneous* signal voltage must also be limited to some specific range V_S'. A similar argument must also apply to the noise. As we are assuming an efficient signal system (no redundancy), the statistical properties of the signal and the noise must be similar, which implies that if we measured either for long enough we'd find that their fluctuations had the same peak/rms voltage ratio. (This is valid if the *bandwidths* of the signal and the noise are the same.) We can therefore say that during a typical message, the noise voltage fluctuations will be confined to some range

$$\pm\, V_N' = \pm\eta V_N \tag{7.3}$$

where η is the **form factor** and is defined from the *signal's* properties as

$$\eta \equiv \frac{V_S'}{V_S} \tag{7.4}$$

The details of what η turns out to be will depend on the statistical properties of the signal, but the above argument is a general one. We should ensure that S is as large as possible to minimise the effects of the noise. It's reasonable, therefore, to expect that an efficient system will ensure that for every typical message, S is almost equal to some maximum value, $P_{\max}$, implying that in such a system most messages will have a similar power value. Another way to express this is to say that *only* messages with mean powers similar to this maximum value are 'typical'. Those with much lower powers are unusual, i.e. rare.

7.2 Shannon's equation

The signal and the noise are **uncorrelated**, which means that they're not related in a way that lets us predict one from the other. The total power from combining the signal and the noise is therefore

$$P_T = S + N \tag{7.5}$$

which is equivalent to saying that the typical combined rms voltage will satisfy

$$V_T^2 = V_S^2 + V_N^2 \tag{7.6}$$

As the signal and noise are statistically similar (both look random), their combination will have the same form factor as either taken itself. We can thus expect that the combined signal and noise will generally be confined to a voltage range $\pm\eta V_T$.

How much information can we expect a single sampled value to contain? Well, if we now divide this range into 2^b bands of equal size (requiring 2^b symbols or numbers to label each band differently), each band would cover a voltage range of

$$\Delta V = \frac{2\eta V_T}{2^b} \tag{7.7}$$

The voltage band occupied by the signal at any moment can therefore always be indicated by a b-bit binary number. This is another way of describing what happens when we take digital samples with a b-bit analogue-to-digital converter (ADC) working over a total range $2V_T$. (This is just a re-tread of the arguments about quantisation in section 2.3.)

There is no point having ΔV smaller than $2\eta V_N$ as the noise will tend to randomise the actual voltage by this amount, so more resolution (i.e. bits) will be meaningless. The maximum number of bits of information we can obtain regarding the signal level at any moment will therefore be given by

$$2^b = \frac{V_T}{V_N} \tag{7.8}$$

which you should be able to show is equivalent to

$$2^b = \sqrt{1 + \frac{S}{N}} \tag{7.9}$$

so

$$\begin{aligned} b &= \log_2 \left[\left(1 + \frac{S}{N} \right)^{1/2} \right] \\ &= \frac{1}{2} \log_2 \left(1 + \frac{S}{N} \right) \end{aligned} \tag{7.10}$$

is the information content (in bits) of a *single* sample.[1]

[1] There's obviously a bit of wiggle-room in here in terms of nailing down *exactly* what the criteria for setting the voltage resolution should be. This one has the advantage of giving an end result consistent with what we want to show!

If the bandwidth of the channel is B and the frequencies that it can support extend down to dc, then the maximum frequency of a signal that can be transmitted along the channel is also B. The sampling theorem tells us that in order to capture *everything* about this signal, we need to sample at a rate of at least $f_s = 2B$, and it also tells us (although perhaps not immediately obviously) that we don't gain any more *information* by sampling any faster than this. (If we did sample at a higher rate, then successive samples wouldn't be independent of each other.)[2]

The *maximum* rate at which we can hope to squirt information along our channel is thus the product of the amount of information contained in a single sample (which is a function of the noise on the channel relative to the signal power it can accommodate) and the rate at which we can take independent samples at the other end, i.e., $f_{s,\max}b$.

This maximum rate is known as the **capacity** of the channel, and tying together the arguments above will be

$$C = B\log_2(1 + \mathrm{SNR}) \tag{7.11}$$

Equation 7.11 is **Shannon's equation** (after a chap called Claude E. Shannon) and represents the maximum possible information transmission rate through a given channel or system, and is a mathematical statement (although our derivation has been more intuitive than rigorous) of the ideas that we deduced in the first few chapters. It tells us that the maximum rate at which we can transmit information along a channel is set by the bandwidth, the signal level and the noise level.[3]

7.3 Choosing an efficient transmission system

In many situations we are given a physical channel for information transmission (a set of wires and amplifiers, microwave beams, optical fibres, two wee guys waving flags, or whatever) and have to decide how we can use it most efficiently. This means we have to assess how well various information transmission systems would make use of the available channel. To see how this is done we can compare transmitting information in two possible forms: as an analog voltage and a serial binary data stream. We can then decide which would make the best use of a given channel.

When doing this it should be remembered that there are a large variety of ways in which information can be represented. This comparison only tells us which out of the two we're considering is better. If we really did want to find the 'best possible' we might have to compare quite a few other methods. For the sake of comparison we will assume that the signal power at our disposal is the same regardless of whether we choose a digital or an analog

[2] We'd certainly get a prettier plot, i.e., the data would be a lot easier to interpret. But some of the data would be redundant.

[3] You may also encounter definitions of channel capacity in terms of bits-per-channel-use, rather than bits per second. This would be equation 7.10

form for the signal. It should be noted, however, that this isn't always the case and that any variations in available signal power with signal form will naturally affect the relative merits of the choices.

Noise may be caused by various physical processes, some of which are under our control to some extent. Here, for simplicity, we will assume that the only significant noise in the channel is due to unavoidable thermal noise. Under these conditions the noise power will be

$$N = k_B T B \tag{7.12}$$

Thermal noise has a 'white' spectrum, i.e. the noise power spectral density is the same at all frequencies. Many of the other physical processes which generate noise also exhibit white spectra. As a consequence we can often describe the overall noise level of a real system in terms of a **noise temperature**, T_N, which is linked to the observed total noise by re-arranging equation 7.12 to yield a temperature which would lead to the amount of noise actually seen. The concept of a noise temperature is a convenient one and is used in many practical situations, and we'll meet it later. It's important to remember, however, that a noisy system will likely have a noise temperature far in excess of its *physical* temperature: a very noisy amplifier doesn't have to glow in the dark...

Many real signals begin in an analogue form so we can start by considering an analogue signal which we wish to transmit. The highest frequency component in this signal is at a frequency of W Hz. The sampling theorem tells us that we would therefore have to take at least $2W$ samples per second to capture all the signal information.[4] If we choose to transmit the signal in analog form we can place a low-pass filter in front of the receiver which rejects any frequencies above W. This filter will not stop any of the wanted signal from being received, but rejects any noise power at frequencies above W. Under these conditions the effective channel bandwidth will be equal to W and the received noise power N will be equal to $k_B T W$. Using Shannon's equation we can say that the effective capacity of this analog channel will be

$$C_{\text{analogue}} = W \log_2 \left(1 + \frac{S}{k_B T W}\right) \tag{7.13}$$

In order to communicate the same information using a serial string of digital values we have to be able to transmit two samples of m bits each during the time required for one cycle at the frequency, W, i.e. we have to transmit $2mW$ bits per second. The frequencies present in a digitised version of a signal will depend upon the details of the pattern of 1s and 0s. The highest frequency will, however, be required when we alternate 1's and 0's. When this happens each pair of '1' and '0' will look like the high and low halves of a signal whose frequency is mW (*not* $2mW$). Hence the digital signal will require a channel bandwidth of mW to carry information at the same rate as the analogue version.

Various misconceptions are common on the question of the bandwidth required to send a serial digital signal. The most common of these are:

[4] Note that this is the requirement based on the *frequency content*: it doesn't tell us anything about the required vertical resolution.

1. "Since you are sending $2mW$ bits per sample, the required digital bandwidth is $2mW$."
2. "Since digital signals are like square waves, you have to provide enough bandwidth to keep the square edges so you can tell they're square, not sine waves."

Neither of the above statements are true. The required signal bandwidth is determined by how quickly we have to be able to switch level from '1' to '0' and vice versa. The digital receiver doesn't have to see 'square' signals, all it has to do is decide which of the two possible levels is being presented during the time allotted for any specific bit.

In order to allow all the digital signal into the receiver whilst rejecting 'out of band' noise we must now employ a noise-rejecting filter in front of the receiver which only rejects frequencies above mW. The effective capacity of this digital channel will then be

$$C_{\text{digital}} = mW \log_2 \left(1 + \frac{S}{k_B T mW}\right) \tag{7.14}$$

This shows the capacity of the channel at our disposal if we can set the bandwidth to the value required to send the data in digital serial form. Note that this is *not* the actual rate at which we need to send data: the digital data rate is

$$I = 2mW \tag{7.15}$$

It will only be possible to transmit the data in digital form if we can satisfy two conditions:

1. The channel must be able to transmit frequencies up to mW.
2. The capacity of the channel must be $\geq I$.

The digital form of signal will only communicate information at a higher rate than the analog form if

$$I > C_{\text{analogue}} \tag{7.16}$$

so there is no point in digitising the signal for transmission unless this inequality is true. The number of bits per sample, m, must therefore be such that

$$m > \frac{1}{2} \log_2 \left(1 + \frac{S}{k_B T W}\right) \tag{7.17}$$

otherwise the precision of the digital samples will be worse than the uncertainty introduced into an analogue version of the signal by the channel noise. (This is just saying that the number of bits we use per symbol must be at least equal to the number of bits of information that a single sample is likely to contain: compare with equation 7.10.) As a result, if the digital system is to be better than the analog one, the number of bits per sample must satisfy equation 7.17. (Note that this also means the initial signal has to have a SNR good enough to make it worthwhile taking m bits per sample.)

Another constraint on the bit-depth is that the rate at which we need to send data can't exceed the digital channel capacity C_{digital}. This is because I cannot exceed the digital channel capacity, C_{digital}, i.e.,

$$2mW \leq mW \log_2 \left(1 + \frac{S}{k_B T m W}\right) \tag{7.18}$$

i.e

$$m \leq \frac{S}{3k_B T W} \tag{7.19}$$

A digitised form of signal sent as a binary data stream will convey more information than an analog form over the available channel *if* we can choose a value for m which simultaneously satisfies 7.17 and 7.19, and the available channel can carry a bandwidth mW. If we can't satisfy these requirements the digital signalling system will be poorer than the analogue one.

7.4 Chapter 7: take-home messages

A maximally efficient signal (i.e., one with no redundancy) provides the amount of information it does because its form is unpredictable in advance.[5] This means that the statistical properties of such a signal are similar to those of random noise. We can use intuitive physical ideas to derive Shannon's equation, and use it to determine the information carrying **capacity** of a channel, which allows us to decide whether a digital or analogue representation makes the best use of a given channel.

[5] Of course, even signals with some redundancy aren't completely predictable, or they'd convey no information at all...

Chapter 8

Example: the CD player

This chapter uses the example of the CD audio system to illustrate with specific examples some of the basic principles of information gathering and processing. It provides an excellent example of many digital data processing methods and allows us to explore the relationship between signals held in equivalent analog and digital forms. Both the source information gathering (i.e. the recording studio, etc) and the information replay system (the CD player) can be used to illustrate a variety of highly effective measurement and information processing techniques. The CD system can be simultaneously regarded as:

1. a measurement system, collecting audio information;
2. a signal processing system;
3. an information communication channel/storage system.

Information theory texts often tend to concentrate on systems where an information source and a receiver are directly connected by some channel. Information is then communicated through the channel in real time. Arrangements which store information for recovery at a later time can also be considered as communication systems. In general, the ideas and techniques of information theory can be applied equally well to both real time and stored or 'delayed' messages. The disc recording process then becomes an information transmitter or source. The CD player is a form of information 'receiver', and the disc itself is an information 'channel'.

When designing or choosing any information transmission system we must start by defining the properties of the signals we wish to communicate. For the CD system we must communicate two channels of audio information, recorded in a form which can be used to reconstruct a stereo sound-field. As with most human forms of communication the actual requirements would vary from one case to another. There is not an obviously correct choice for the required signal bandwidth, so we'll consider the system as it is, rather than discuss whether or not a different specification would be better.

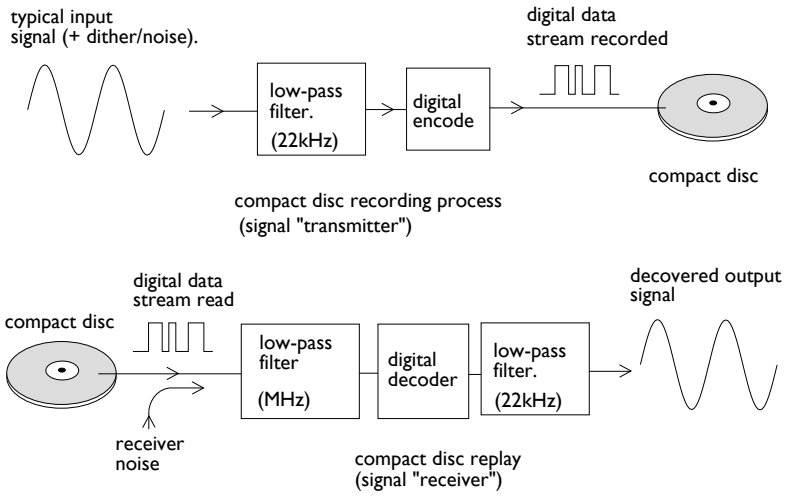


Figure 8.1: The compact disc as an example of a communications channel.

The CD system has been based upon the assumption that high fidelity sound reproduction requires a uniform frequency response from below 10 Hz to above 20 kHz and a dynamic range of more than 90 dB. This led to the decision to sample each of the stereo channels (left and right) 44100 times per second, and to take 16-bit digital samples. Using 16-bit words, the ratio of the largest possible signal (which does not go out of range) to the quantisation interval is $1{:}2^{16} = 1{:}65{,}536$. This voltage ratio is equivalent to a power ratio of 96.3 dB, so we can expect the dynamic range of the CD system to be of this order.[1] The input to a CD digital recording system is normally dithered in order to suppress quantisation distortion. This raises the noise floor a bit (a few dB), but the added noise is less annoying to the ear than the distortion that would arise without dithering. From the sampling theorem we can expect that the chosen sampling rate will allow signal frequencies up to 44.1/2=22.05 kHz to be recorded and replayed.

The sampling rate of 44.1 kSa/s means that the ADCs need to be preceded by low-pass anti-aliasing filters such that frequency components above 22.05 kHz are reduced to negligible amplitude.

8.1 The CD encoding process

On the basis of the figures given above we can expect that a CD lasting 60 minutes will, as a bare minimum, have to store

$$44100 \times 16 \times 2 \times 60 \times 60 = 5.08 \ \times 10^9 \text{ bits}$$

[1] The actual dynamic range is usually expressed for (say) sinusoidal signals, when the rms values of signal and noise need to be considered. Hence, somewhere around the quoted value!

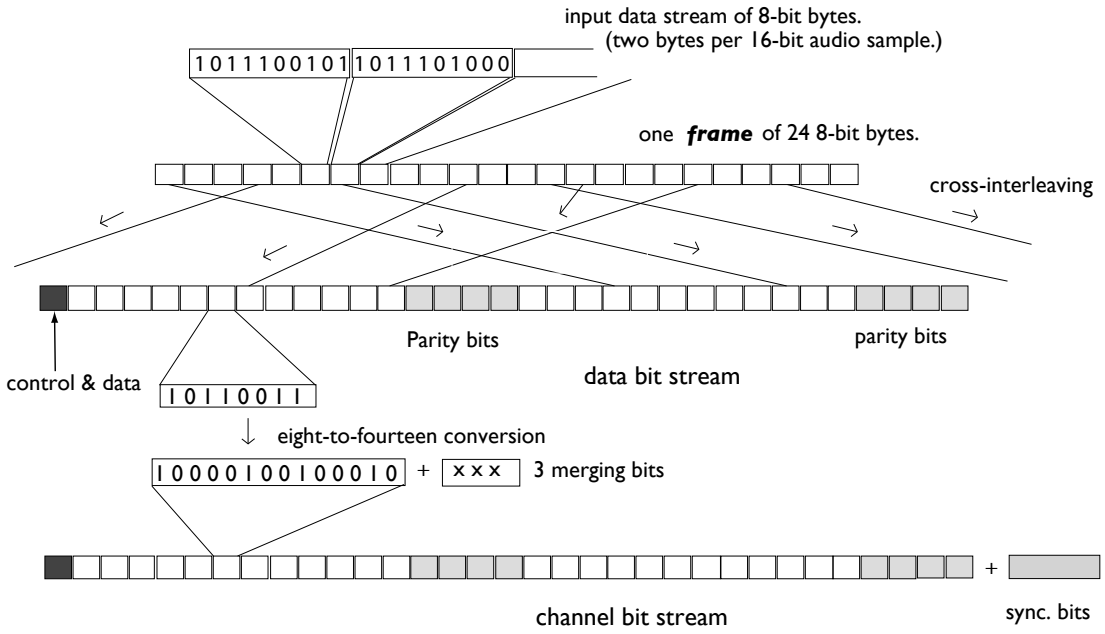


Figure 8.2: The CD data encoding system, showing how extra bits are added and how they are re-arranged.

of information. It won't be a big surprise that *redundancy* is employed for error detection and correction when the disc is re-played, so although the amount of *information* on a 60 minute CD remains around 5 Gbits, the number of bits of data is much greater. The bare information rate will be 1.41×10^6 bits per second, but the actual data rate will be considerably higher.

The encoding scheme employed for CD is quite complex. We'll only consider only some key elements to appreciate how the basic concepts of information theory have been applied. The explanation given here if from Jim Lesurf's book *Information and Measurement* (2nd ed.), which itself based it upon information provided by Philips (who developed the CD system along with Sony) in a special issue of the Philips Technical Review (Vol. 40(6) 1982).

Figure 8.2 represents the CD encoding/recording system. The input data is initially sampled in the form of a stream of 16-bit digital words. These words are collected into frames of 6 consecutive left/right pairs of digital samples. One frame therefore contains 192 audio bits which are then treated as a set of 24, 8-bit, audio symbols.

These audio symbols are re-arranged and some extra parity symbols are generated using an encoding scheme called a Cross Interleaved Reed-Solomon Code or CIRC. For our purposes it is sufficient to recognise that CIRC is a type of block code which generates a specific pattern of parity bits. CIRC also interleaves or 'rearranges' the sequence of the data bits. The interleaving process is designed to minimise the effects of momentary data losses. Some extra control and data bits are also added at this stage. These contain extra information (for example track numbers and running time) which are of use to the CD player. The result of the CIRC encoding stage is to convert each frame from 24 audio symbols into 33 data

symbols (each 8-bit, as before, giving a new total of 264 data bits). The parity bits provide some of the required ability to detect and correct random errors.

In practice, much of the data loss when replaying a CD occurs in brief bursts when the player encounters a hole, or a piece of dirt, or when vibration causes the laser to momentarily miss tracking the data. This causes a series of successive data bits to be lost, sometimes lasting for a number of symbols. Interleaving the symbols before recording (and de-interleaving them on replay) helps prevent successive audio symbols from being lost. It also 'spreads out' the data and parity bits to reduce the chance that both a given symbol *and* its associated parity bits will be lost. This interleaving process covers up to 28 frames and as a result, information from any pair of adjacent audio samples will usually be spaced some considerable distance apart on the actual CD.

A useful analogy is a piece of paper upon which a message has been typed. In the process of being passed to the person who wants to read it, the paper is attacked by a dog which tears it and eats a piece. As a result, when the message is read about 5% of the text is missing perhaps because the last few lines have been torn off. It is likely that any information which was contained by the missing lines is lost (inside the dog!).

However if the letters of the text had been typed onto the paper in a 'scrambled' order it would be possible to re-arrange the received text back into a message where occasional words would have a missing letter. (Of course, in order to do this the scrambling process must be known to the person receiving the message.) The result would probably be a readable message despite the loss of letters from some words. This is because of the natural redundancy of the English language which lets us make sense of text even when there are mistakes. The CIRC encoding process works in a similar way in that parity bits are used to add some redundancy, and the message is interleaved (scrambled) so that any brief breaks in the data stream should only cause single-bit losses is some samples. These can then usually be corrected because of the redundancy.

Following CIRC encoding each of the 8-bit data symbols is translated into a 14-bit channel symbol and an extra three merging bits are tacked onto the end of these 14. For obvious reasons Philips refer to this process as Eight to Fourteen Modulation (EFM). At the end of the frame of channel symbols another 27 synchronisation bits are added to make a total of 588 channel bits per frame. The sync bits are a unique pattern which the CD player uses to locate the beginning of each data frame.

The bits are then recorded as a sequence of pits cut into the disc. Those parts of the disc surface where no pit has been formed are referred to as the land. The recording is not made on the simple basis that '1'='pit' and '0'='land' (or vice versa). Instead a '1' represents the transition or edge between pit and land and '0' means "continue as before" and '1' means"change from pit to land, or land to pit".

The specific choice of which 14-bit channel symbol should represent each 8-bit data symbol has been made so as to try and satisfy a number of requirements. Firstly, a set of 14-bit codes has been selected whose patterns provide the largest possible Minimum Hamming Distance

between adjacent codes. This helps the CD player recognise and correct occasional random bit-errors in the recovered data stream. There are $2^{14} = 16,384$ possible choices of 14-bit channel symbols of which only $2^8 = 256$ are required. We can therefore surround (in symbol space) each legal pattern with 64 illegal ones. This allows the correction of single-bit (and most two-bit) errors.

The second factor which influenced the choice of 14-bit symbols was the decision to limit the maximum and minimum number of '0's which can appear between successive '1's of the recorded bit stream. This sets a maximum and minimum distance between successive pit-land edges on the disc. The codes chosen for CD recording ensure that there are always at least two '0's, and not more than ten, in between successive '1's. However, one symbol which finishes with a '1' may still need to be followed with another which begins with a '1'. The pair of symbols would then 'clash', violating the requirement for more than two zeros between any pair of '1's. This problem is overcome by the inclusion of three extra merging bits in between successive symbols. Now we can simply place three zeros in between symbols whenever we need to avoid a clash.

The symbols and merging bit patterns are also chosen to ensure that, on average, the encoded disc appears to the player as consisting of 50% land and 50% pit. This helps the servo control system in the CD player to correctly focus the laser spot it uses to read the recorded data. Finally, the symbols and merging bits are chosen so as to produce a strong component in the recorded signal spectrum at a predetermined frequency. This provides a *clock reference* signal for the CD player. The player can compare a filtered version of the recovered signal with a crystal oscillator and use this to adjust the disc rotation velocity.

The encoding process converts an initial 192 bit frame into a recorded frame of 588 bits. The number of channel bits recorded on a 60 minute CD is, therefore, around 15.5 Gb and the channel bit rate will be 4.32 Mb/s. This means that the actual channel bandwidth required must be over 2.16 MHz, not 0.7 MHz as it would be (at very bare minimum) if *just* the raw audio data was to be encoded and transmitted.

The ability of the CD system to withstand errors and imperfections in the disc or the replay process is remarkable: two key measures (there are more, but we'll ignore them) are:

1. **Maximum completely correctable burst length** (MCL) = 4000 data bits (2.5 mm of track length on disc.) This means that gaps in a track of up to 2.5 mm in an otherwise perfect disc should not lead to any loss of audio information. This indicates the power of the combination of the parity bits plus eight-to-fourteen modulation to correct the loss of a large number of successive channel bits.

2. **Maximum interpolatable burst length** (MIL) = 12000 data bits (7.7 mm track length.) Once the MCL has been exceeded some data will be lost. The interleaving process is, however, designed to ensure that no two adjacent audio sample values will be lost until over 12000 successive channel bits have become unreadable. The player can 'interpolate' the lost data samples.

8.2 The CD player as a measurement system

The CD player has to recover information from a spiral track of small pits which have been formed at a nominal information layer inside a compact disc. Unlike the vinyl analogue recordings, the CD does not have a 'continuous' groove and the optical sensor should never touch the disc. Hence the CD player must locate the required information without any mechanical guidance about where the data is to be found. Figure 8.3 illustrates the form of a typical CD information surface and the laser beam used to read data from the disc.

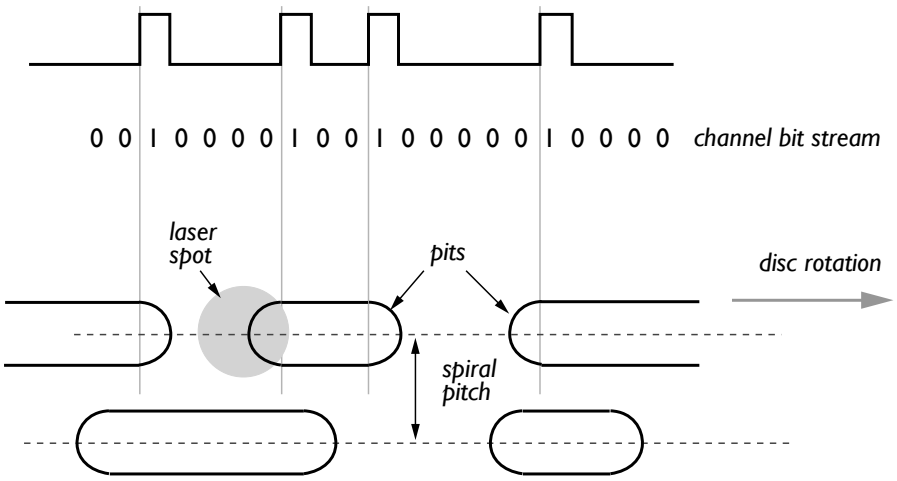


Figure 8.3: Data being replayed from the pattern of pits and lands on a CD.

Information is recorded on a reflecting layer a little below the the surface of a CD in the form of a spiral track of pits, and is read using a laser whose wavelength is around 0.7 μm. The spiral pitch (distance between adjacent turns) is 1.6 μm and the disc is rotated so that the position illuminated by the laser spot moves at a constant linear velocity of 1.25 metres/second.

The patterned plastic surface is coated with a thin layer of metal (usually aluminium, but some expensive CDs use gold instead) to make it highly reflective. This is then covered with a protective top coating of transparent plastic.

When using electromagnetic radiation to observe small-scale features, we wouldn't normally expect to be able to measure anything whose size is significantly smaller than the chosen wavelength. In the case of a CD the required bit recovery rate is 4.32 Mb/s and the the disc velocity is 1.25 m/s. This implies that each bit occupies a track length of just 0.29 μm, i.e. less than half the laser's wavelength! A number of factors help CD players to recover information from such a closely packed surface pattern.

- The laser beam is tightly focussed to produce a spot whose nominal diameter is typically around 1 μm. This requires an optical system of very high quality.
- The encoding system is designed to help the laser sense the surface features. Every

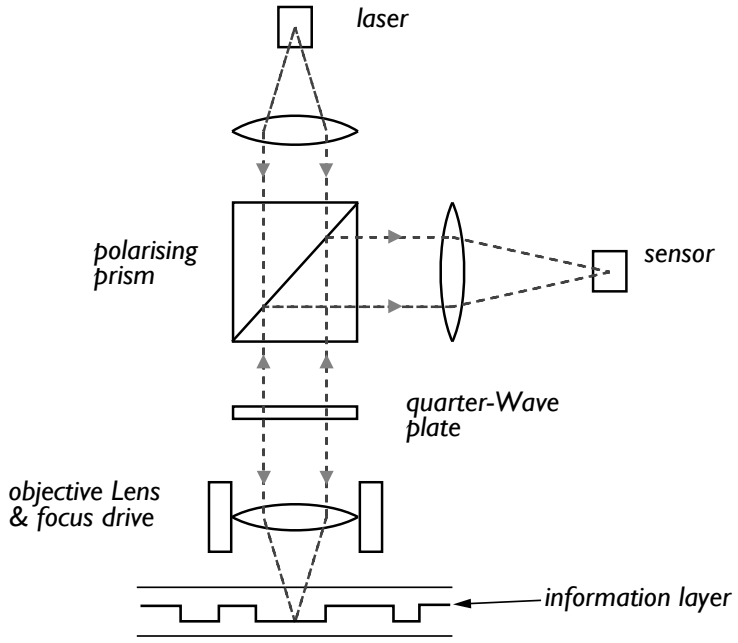


Figure 8.4: Simplified optical system for replay from a CD.

stretch of pit or land will be at least 3 bits long. This is a result of the coding requirements that; i) there must always be at least 2 zeros between adjacent ones; ii) pit-land edges represent encoded 1's. This means that pit-land edges will always be at least 0.87 μm apart , i.e. the length of each pit or land feature will always be comparable with the laser wavelength. This means it is possible to ensure that the laser spot will never illuminate more than a single edge at a time.

When the optical spot traverses a pit-land edge the magnitude of the beam reflected back into the sensor optics will momentarily dip almost to zero. The reason for this can be understood by considering what happens if half the spot energy falls upon land, and half into a pit. The reflected beam then consists of two portions, equal in magnitude but opposite in phase. As a consequence the total energy coupled back into the sensor beam would be zero.

CD players use a reading mechanism of the general form shown in figure 8.4, although the details vary. The optical system is invariably more complex than shown in figure 8.4, which doesn't show any of the additional requirements for tracking the data or maintaining the focus.

The system illustrated relies upon detecting the momentary dips in the observed reflected light level which occur at the pit-land edges. Laser light is focussed onto the disc information layer via a polarisation prism and a quarter-wave plate, an arrangement which means that the reflected light ends up on a sensor rather than back at the laser.

In practice we may find that the reflected energy is not divided exactly 50:50 at the pit edges.

The pit depths may also not be exactly a quarter wavelength. This means that the magnitude of the sensed reflection may not dip right down to zero. Despite this practical problem, the power of the replay laser is normally so large that we can obtain a high enough SNR ratio to measure the locations of pit-land edges with an uncertainty considerably smaller than a wavelength.

8.3 The CD player as a digital signal processing system

The stream of bits recovered from the disc is processed through a series of stages which reverse the encoding process. Minor errors can be completely corrected using the eight-to-fourteen redundancy and parity checking built into the system. Major errors may result in the unavoidable loss of information, but the player can then interpolate in order to 'recover' occasional lost samples. The recovered stream of digital values can then be passed to two digital-to-analog convertors (DACs) for conversion into an output pair of analog audio signals.

In principle, we could simply use the CD player to recover 44,100 pairs of digital samples per second and use a pair of 16-bit DACs to obtain analog signals. Whilst this approach would have the advantage of simplicity it may produce an output which exhibits the staircase distortions mentioned earlier.

Provided the input signal was dithered before sampling, any staircase distortions can in theory be removed by passing the output from the DACs through low-pass filters which reject frequencies above half the sampling frequency as, in an ideal system, all the unwanted frequencies produced by the staircase effect will be above 22.05 kHz. This is unsatisfactory, as simply using analog filters to 'clean up' the output waveforms is difficult to do cheaply due to the high-performance filters required, so CD players process the digital data in some way before presenting it to the convertors. The main objects of this processing are to computationally low-pass filter the audio data and to help prevent any imperfections in the digital systems (especially the DACs) from producing other distortions. The details will vary considerably, but it's worth a discussion of the main ideas of one method.

8.3.1 Over-sampling

Oversampling means that a set of sampled values is used to calculate the values we "would have obtained" at intermediate moments if the original input had actually been sampled more frequently. Provided the sample values we start with satisfy the sampling theorem these extra values contain *no new information*. Some early CD players employed $\times 4$ oversampling, converting an input data stream of 44,100 samples/sec (per channel) into 176,400 samples/sec. We can regard staircase distortion as being an unwanted high-frequency variation which has been added onto the signal. By x4 oversampling we produce the effect shown in figure 8.5.

Consider an input signal in the form of a 5.5 kHz sine wave which is sampled at 44,100 Sa/s.

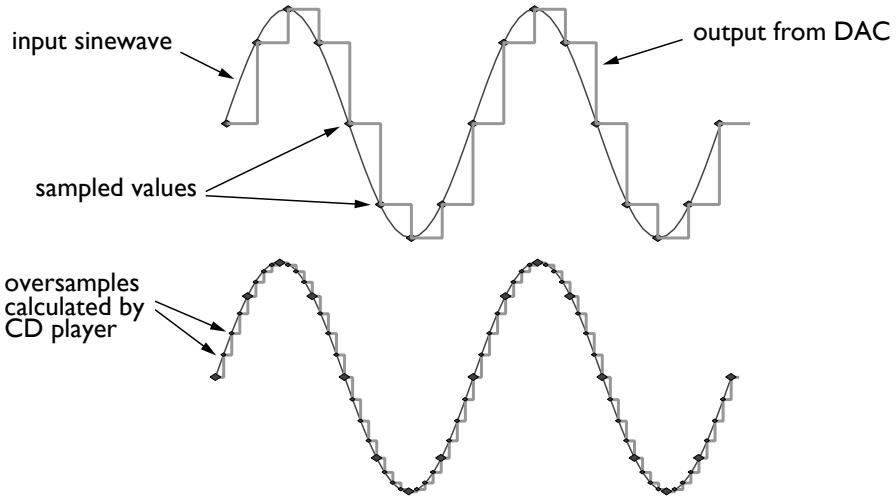


Figure 8.5: An example of *oversampling*. The upper waveform shows a 5.5 kHz sine wave sampled at 44.1 kSa/s. The lower waveform shows the reconstructed waveform with four times oversampling. Note that the size of the 'steps' is reduced at the frequency at which they occur is increased.

The simplest way to get back to an analog waveform would be to use a DAC which produces an output level appropriate for each sample and then holds this level until it is time to output the next sampled level. This kind of output is called *sample and hold* and produces the kind of staircase distortion shown.

If the samples form a complete record, we can use the measured sample values to calculate the actual signal level at any moment in between the sampled instants. These calculated values can then be given to the DAC in between the 'genuine' samples to produce the improvement showed in figure 8.5. The term "oversamples" indicates that these extra values don't contain any fresh information.

The use of $\times 4$ oversampled digital values reduces the staircase effect in two ways. The basic frequency of the unwanted staircase distortion is *increased* and its amplitude is *decreased*. As a consequence it becomes much easier to produce analogue filters which, placed after the DACs, will suppress this distortion without significantly affecting the wanted signal.

Figure 8.6 shows the in-band (the signal we want) and the out-of band (the bit we don't want) performance of the original Philips CD player chipset, excluding the effects of any low pass filter that would of course follow the DAC. More modern chipsets can give even more improvement (around 80 dB reduction in the unwanted distortions).

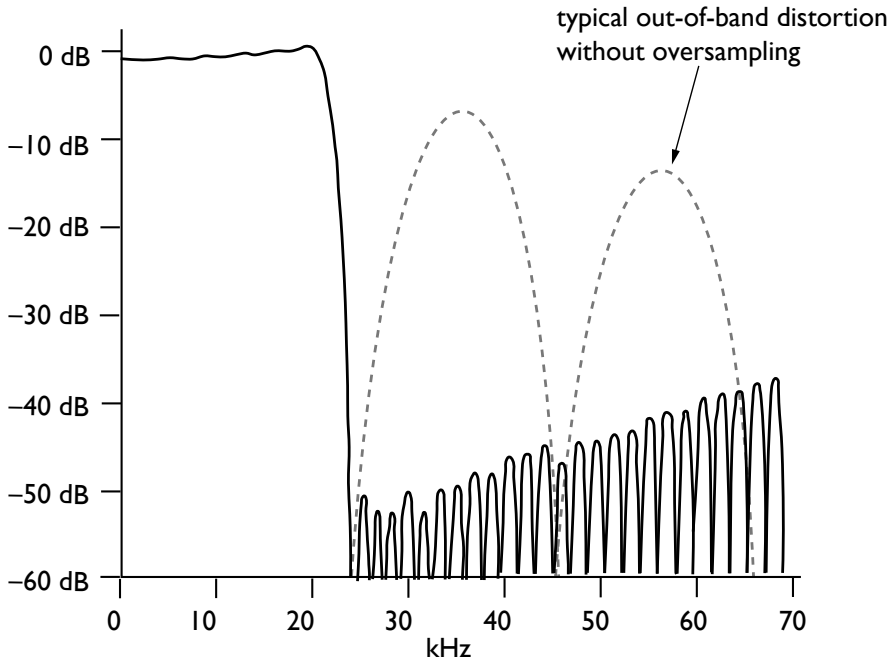


Figure 8.6: In-band and out-of-band component powers for one of the original Philips ×4 oversampling chipsets (from Philips Tech. Rev. vol. 40, no. 6).

8.4 One bit more

We've now had a look at how CD systems work. One of the subjects this included was the use of multi-bit DACs to turn the digital values back into an analog waveform. For various reasons it's difficult to make accurate multi-bit DACs. Figure 8.7 illustrates a system which takes an input voltage, converts it to an n-bit digital value and then turns that value back into an output voltage. This represents a sort of minimal digital information communication system.

The ability of the system to transfer a signal from input to output can be represented by a graph showing how the output voltage, v_{out}, from the DAC varies with the input voltage, v_{in}, presented to the ADC. This is called the *signal transfer curve* of the system. For a perfect matching ADC/DAC pair we would expect the system to have a transfer curve similar to figure 8.7 (b). The steps of this staircase are produced by the quantisation produced by the ADC. For an ideal system all the steps should have the same heights and widths and resulting staircase shape is said to be **monotonic**.

Multi-bit DACs use a variety of techniques to convert digital values back into an analog voltage. Unless they're perfectly made, these produce the two general sorts of errors shown in figure 8.7 (c) and (d). *Monotonicity errors* arise from imperfections in the DAC, meaning that some input digital values produce incorrect output voltages. The effect of this is to lift or lower some of the steps in the transfer staircase. An overall or *systematic* nonlinearity

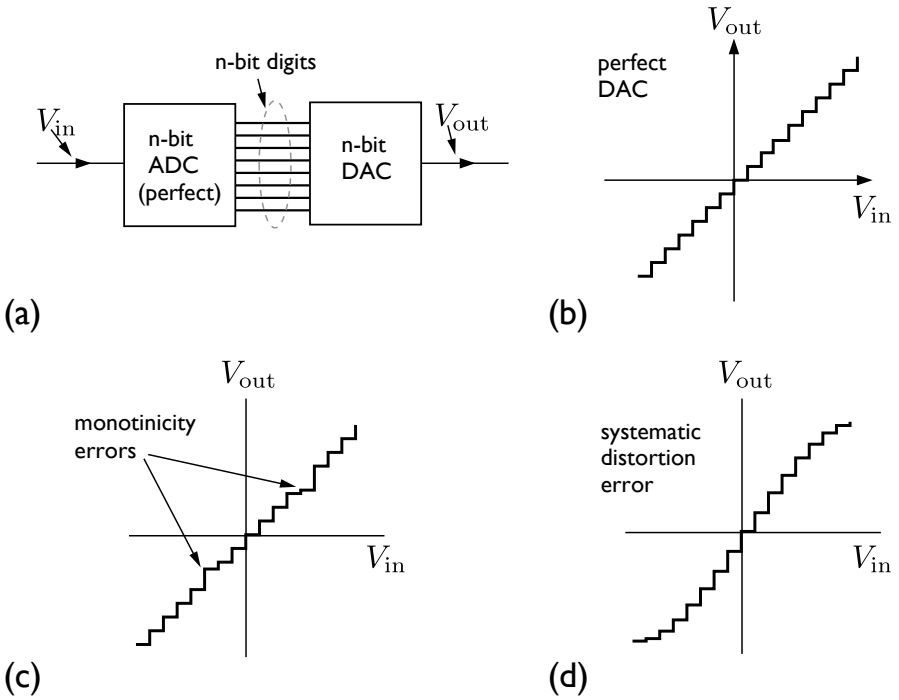


Figure 8.7: Signal transfer using an ADC and a DAC 'back-to-back'. (a) A systems takes an input voltage, converts it to an n-bit digital value then converts it back to an analog voltage. (b) The result obtained if the ADC and DAC are perfectly matched. (c) The effects of *monotonicity* errors. (d) The effects of a particular *systematic* distortion error.

results from imperfections in the DAC causing the output voltage to be wrong by an amount which varies relatively smoothly with the input. Both types of imperfection will cause the output signal to become distorted.

One way to avoid this distortion is, of course, to make and use very good ADC/DAC chips, but this leads to higher costs. Many modern systems instead use **one-bit** DAC systems, which we will proceed to look at. As well as a cheap solution for use in CD players, it's also a nice example of the conversion of information from one form to another without loss.

A conventional 16-bit DAC can output 2^{16} different output voltages, each corresponding to a different input digital value. A one-bit DAC can only produce 2 possible output voltages: 'high' or 'low'. The system uses one digital line, Z_i, which will either be 'high' ('1') or 'low' ('0') at any instant. This digital level is used to operate an *analogue signal driver*, an amplifier whose output is $+V_{\text{ref}}$ when $Z_i = 1$ and $-V_{\text{ref}}$ when $Z_i = 0$. The choice of V_{ref} is not important provided it is constant.

The digital output consists of a stream of pulse 'cycles' with a period T. The duration or width of each pulse is t. The output from the driver is passed through a time constant (or some other low-pass filter) chosen so that $RC \gg T$ (where RC is the time constant of the filter). We can think of the input waveform, v_z, as being a combination of a dc level plus some ac components which give the pulse shape. We can therefore expect the output

voltage, v_{out}, to approximately equal the value of v_z averaged over the last few pulse cycles.

Now the average value of v_z during one pulse cycle will be

$$\overline{v_z} = \frac{tV_{\text{ref}} - (T-t)V_{\text{ref}}}{T} \tag{8.1}$$

We can therefore expect that v_{out} will be approximately equal to the above expression, and we can simplify it to

$$v_{\text{out}} \approx \frac{V_{\text{ref}}(2t-T)}{T} \tag{8.2}$$

The above argument assumes that the time constant RC is large enough to make it act as an *integrator* and we can see that the filter will smooth out the pulses to give us a dc level the value of which depends on the ratio $(2t-T)/T$. This means we can obtain any output voltage we wish in the range $-V_{\text{ref}} \leq v_{\text{out}} \leq V_{\text{ref}}$ by choosing appropriate values for the pulse width, t, and the cycle period, T.

One very basic method which uses these ideas is **pulse width modulation** (PWM). Figure 8.8 (a) shows an input sinewave which is constrained to be within the range $\pm V_{\text{ref}}$ and figure 8.8 (b) illustrates the use of PWM to convey information about the sinewave which can be converted back into an analogue form.

In PWM, the cycle period, T, is kept constant and the pulse width is varied according to the signal voltage level, v_s, that we wish the system to output. Since we require the final output level to be v_s, we can rearrange equation 8.2 and say that

$$t = \frac{T(v_s + V_{\text{ref}})}{2V_{\text{ref}}} \tag{8.3}$$

Of course, in order for a CD player to make use of a one-bit DAC, it needs a method of converting the many-bit samples stored on the CD to a stream of values that can be used as described as above. One consequence of using a one-bit DAC that should hopefully make good sense is that if we're only using a string of one-bit samples to generate our analog output waveform, we will need to employ some form of *oversampling*: we can't use PWM (for example) to reproduce a waveform sampled at 10 kHz, say, if the one-bit values that we then filter are only generated at 10 kHz.

In general, we can describe a $p\times$ oversampling system as taking in m samples per second and generating $m' = pm$ output samples per second. Each input sample will have n bits and each output value will have n' bits. The rate at which bits of information enter the oversampler will therefore be mn. The rate at which they emerge will be $m'n'$. From the basic arguments of information theory we can expect that, provided the system works in a sensible way, no information need be lost provided that $m'n' \geq mn$. This is because the amount of information conveyed in a given time depends upon the rate of bit transfer, so the effect of having fewer bits per sample can be counteracted by having more samples. On the basis of this argument we can expect that a system which oversamples by at least $16\times$ should be capable of providing a stream of 1-bit output values which carry all the information from

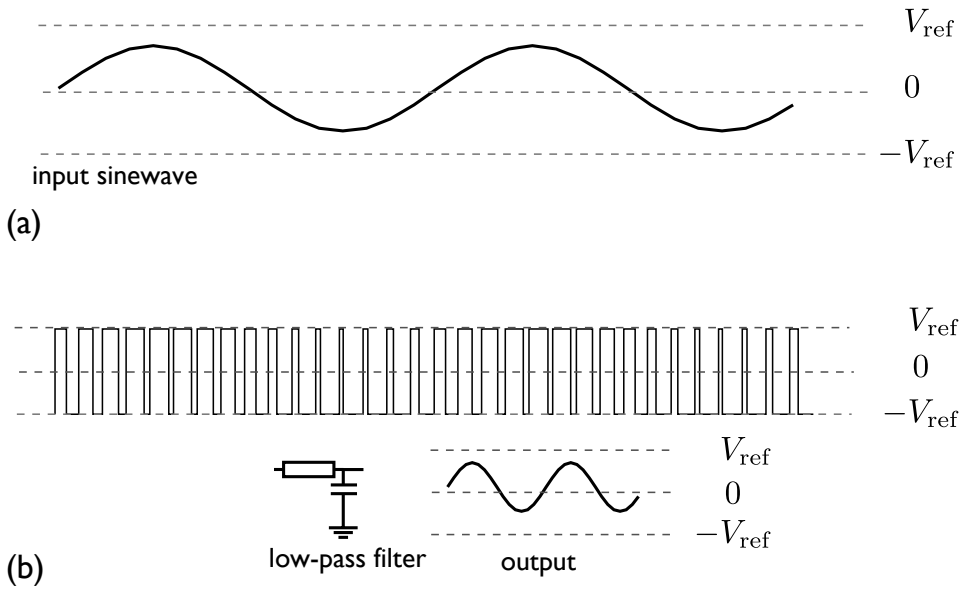


Figure 8.8: Pulse width modulation. (a) An input sinewave. (b) PWM produces a train of pulses with the same period but different widths: the *average* voltage (recovered by a low-pass filter) gives the desired analogue voltage.

an input stream of 16-bit samples. (In practice, we usually err on the side of caution and use an oversampling ratio that ensures that $m'n'$ is significantly greater than mn to avoid effects of any imperfections in the signal conversion process.)

If one were to work out the required clock rates to use simple PWM to implement a 1-bit DAC for a CD player, you'd come to the conclusion that simple PWM as illustrated above isn't up to the task, and more sophisticated digital processing systems are required (which aren't described here) to actually make use of 1-bit DACs in CD players. However, PWM is sufficiently common to merit inclusion in the course, and it makes sense to discuss it in the context of digital-to-analogue conversion.

8.5 Chapter 8: take-home messages

This chapter has looked at the CD player in particular, but only to illustrate a range of important notions in information and signal communication. The CD provides a specific example of the use of block coding and parity checking, as well as data interleaving, which can be applied to messages in general.

You should now understand how the pattern on the surface of a CD is formed You should now know what is meant by the term **oversampling** and why (in the CD example) it relaxes the requirements of post-DAC filtering.

You should now know that multi-bit DACs can suffer from practical problems of non-monotinicity and systematic errors which can distort the output waveform.These problems can be avoided by using a **one-bit** DAC system. You should now also understand that any digital system which takes an input of m samples/sec, each n bits long, and outputs m' samples/sec, each n' bits long can ensure that no information is lost provided that $m'n' \geq mn$. Pulse Width Modulation (PWM) and Pulse Density Modulation (PDM) can be used to generate a wave which can be averaged to obtain a required analog signal. This principle of averaging over a pattern of 1s and 0s can be used to recover an analog signal from other one-bit systems. Simple PWM/PDM systems aren't currently suitable for CD players.

Chapter 9

Data compression and reduction

So far we've taken it for granted that we will always try to preserve the *entire* information content of a signal. As we considered earlier, however, some messages aren't very 'surprising' and therefore (from an information theory point of view) don't contain much 'real' information. This raises two questions:

1. Can we re-code a signal into a form which requires fewer bits to send or store without losing any real information?
2. Do we need to carry *all* the details of a signal, or are there some which we can regard as 'trivial' or 'un-informative' and thus discard?

These are import questions because if we can reduced the number of bits required, we can send or store useful messages using a system with a lower capacity, i.e. a system which is cheaper! These techniques used to be a big deal when computer storage was expensive. Hard disks are now cheap, but we are more likely than ever to want to send cat videos to our aunties in Australia, so reducing the number of bits needed to send an item is hugely important. As anyone who regularly uses mobile broadband (or even fixed-line broadband) will appreciate, bandwidth is expensive!

Important note: We'll refer to techniques which use clever methods to reduce the amount of bits required to transmit/store a signal *without* information loss as data *compression*, and use the term data *reduction* or data *thinning* to denote techniques which discard information regarded as 'unimportant'.

9.1 Data compression

We'll start off looking at data *compression*. There are many ways of doing this, but we'll just look at a couple of examples.

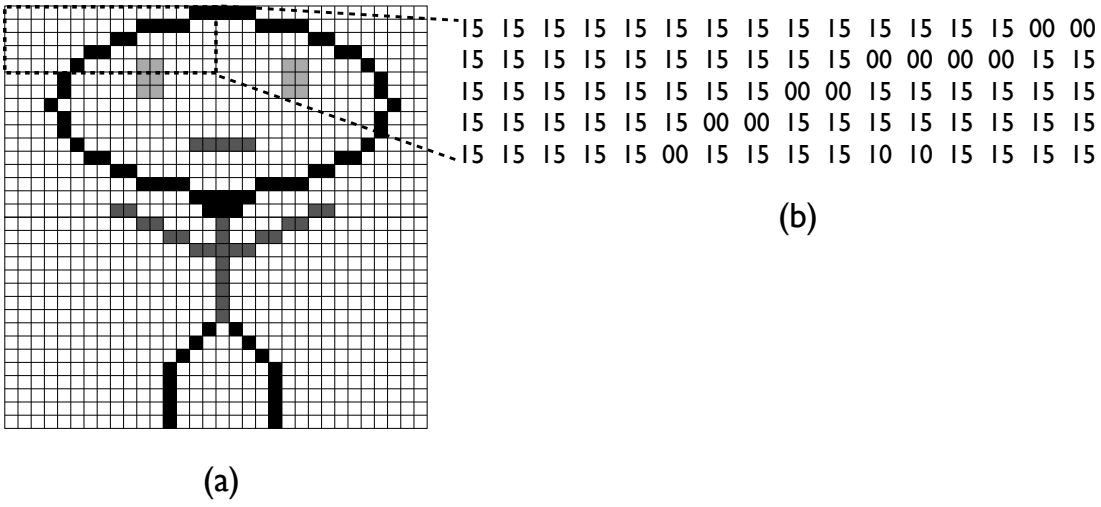


Figure 9.1: (a) An example of a 32×32 pixel 4-bit grey scale image. (b) The pixel values for the top left part of the image.

9.1.1 Run-length encoding

These days, static storage for information (hard drive space, memory cards, etc) is getting cheaper all the time, but the amount of information that one of us is likely to have is growing all the time, and transmitting information, although cheaper than it used to be, is still pretty expensive. Here we'll take a look at a simple example based on the ways that some images are encoded to see the features that all true data compression (as opposed to reduction) techniques share.

Very broadly speaking, image file formats can be divided into *vector* formats (where the image is made up of objects) and *bitmap* formats, where the colour of each pixel is specified. We'll look at bitmap images as they provide a clearer example of how compression techniques can work. A bitmap image divides the image up into an array of *pixels* (a corruption of 'picture elements'), and the colour of each pixel is stored as a number. The amount of information (details, range of colours) the picture can contain is then determined by the number of pixels in the image and the range of numbers we can store to indicate the colour of each pixel.

Figure 9.1 (a) shows an example of a bit-mapped image. For the sake of simplicity, we've limited the range of possible colours to just 16 and chosen an image which is only 32×32 pixels in size. Here we're just using four bits per pixel and only using the values to indicate the 'greyscale level' (i.e. how bright the pixel is). This makes the explanation easier, but the following arguments also apply to full-colour pixel-map systems.

In the image shown, the darkness of each black/grey/white pixel is stored as a number in the range 0→15 or %0000 to %1111 in binary. 0 means black, and 15 means white. We only need *half* an 8-bit byte to store the information about each pixel. Since there are

32×32 pixels, each requiring 4 bits, we need a total of 32×32×4=4096 bits or 512 bytes to specify all the details of the image as a bit-map. There are various ways we could record this information on disc or transmit it over a digital signal link. For example, we can start in a corner and group the pixel values together in pairs to get a string of 8-bit bytes.

Figure 9.1 (b), shows the relevant pixel values for the top left corner of the image. Although we're only looking at a chunk of the image, the arguments extend to the whole thing.

Considering the zoomed part of the image, the first pair of pixel colour numbers are 15 and 15 (%1111 and %1111 in binary) so when we group these together we get %11111111=255. We can continue the process and go line-by-line down the whole image. We can therefore store (or transmit) the picture's pattern as the series of bytes: 255, 255, 255, 255, 255, 255, 255, 0, etc. The number of bits or bytes required is determined by the 'size' of the image (i.e. the number of pixels it contains). To represent *any* 32×32 pixel, 16 colour pattern we use 512 bytes. This sort of coding is called *fixed length*, as the number of bits required doesn't depend on the actual picture pattern.

So, can we store or send all the picture information using fewer bits/bytes? The answer is (you probably saw this coming!), yes, *sometimes* we can by using a different way to code or represent the information. The technique we'll use here is called **run-length encoding**. This is based upon only storing information about where in the picture the colour (brightness in this case) level changes. To illustrate this process let's assume that the small array of numbers in figure 9.1 (b) is the whole image we want to store or communicate. As shown, this pattern is just 16 pixels wide and 5 pixels high. So we'd require 16×5×4=192 bits (24 8-bit bytes) to store or send it using the fixed length method described above.

Considering figure 9.1, the 16×5 chunk we've highlighted corresponds to the series of byte numbers

- First line: 255, 255, 255, 255, 255, 255, 255, 0
- Second line: 255, 255, 255, 255, 255, 0, 0, 255
- Third line: 255, 255, 255, 255, 0, 255, 255, 255
- Fourth line: 255, 255, 255, 0, 255, 255, 255, 255
- Fifth line: 255, 255, 240, 255, 255, 170, 255, 255

To run-length encode this, we start with the 'first' value (the top-left byte) and note its value (255). We then note how many successive bytes have this same value: in this case it's 7. The next value is 0, which occurs one time. We continue this process and if we ignore the line locations then we can write the 16×5 pixel segment as:

255, 7, 0, 1, 255, 5, 0, 2, 255, 5, 0, 1, 255, 6, 0, 1, 255, 6, 240, 1, 255, 2, 170, 1, 255, 2

i.e. we have described the 16×5 segment using 26 8-bit bytes, so 208 bits in total. Providing we know the details of the encoding process, we can use the information in these 208 bits to

reconstruct *all* of the 320 bits of original picture information. At first sight this may seem suspiciously too good to be true, but we find that this type of encoding can often reduce the number of bits/bytes required to store all the details of a picture. Similar (but more complex) methods can be used to reduce the number of bits/bytes needed to store various sets of information.

This is a process which (for images) is great if there are large areas of the same colour. It's not just useful for imaging, however: we could apply it to *any* form of data where we have strings of repeating symbols.

9.1.2 Huffman coding

Although we won't attempt to prove it here, data compression methods *all* exhibit the feature that they successfully compress some types of patterns but expand others. On average, they don't (unless they're badly designed!) make randomly chosen 'typical' patterns either smaller or bigger. However, most pictures, text files, etc, aren't really 'random'. (For example, a truly random picture would just show noise!) There are patterns which aren't of any value. For example, the text character sequence, "qgsdxf ftfngt zdplsdesd xot" isn't very likely to occur in written English. An information storage system which devotes as much storage space to it as to, "grass is green", is being wasteful. Similarly, some characters or symbols occur more often than others or convey less information.

The usefulness of compression techniques comes from matching the technique to the types of pattern you actually want to compress. It essentially removes the redundancy required to encode 'daft' or uninformative patterns. (The daft patterns are then the ones that would come out longer than the original when encoded.) For this reason a variety of compression techniques have been developed, each having its own good points. Here we'll consider a system called **Huffman coding** after it's inventor.

For the sake of illustration, we'll consider a made-up language consisting of just eight characters in total, which we'll represent by X_1, X_2,...,X_8. Overall in this made up language, the different characters occur with the probabilities p_1, p_2,...,p_8. From chapter **??**, we can say that the *average* amount of information (in bits) that a *typical* message which is N characters long will be

$$\overline{H} = -N \sum_{i=1}^{8} p_i \log_2(p_i) \tag{9.1}$$

where the bar over the H is indicating that we're talking about an average value (i.e. the one we'd typically expect). Of course, the *actual* amount of information in a *specific* message which contains A_i incidences of the charater X_i will be

$$H = -\sum_{i=1}^{8} A_i \log_2(p_i) \tag{9.2}$$

(It may be worth a quick review of chapter **??** to confirm that you understand why the above is the case.)

Consider the case where we've analysed the language and know the all of p_i:

- $p_1 = 0.125$
- $p_2 = 0.5$
- $p_3 = 0.05$
- $p_4 = 0.06$
- $p_5 = 0.055$
- $p_6 = 0.01$
- $p_7 = 0.17$
- $p_8 = 0.03$

A *typical* 16 character message (i.e. $N = 16$) would therefore contain 38.48 bits worth of information.

Given that we have 8 characters (symbols), then if we represent each symbol by a binary number of the same length (i.e. a *fixed length* code), we'd have to allocate 3 binary bits per symbol to give us the required range of symbol possibilities. This would require 48 bits to communicate or store a 16 character message. As this means more bits than information contained in the message, then it shows that the fixed length coding scheme is *inefficient*. This means that it must contain some *redundancy* which could be used to help detect and correct errors, but slows down the communication process.

This arises because the symbols/characters used aren't all equally probable. From the above values we can work out the amount of information typically provided by each individual character's appearance. For example, $h_1 = -\log_2(p_1) = 3$ bits worth of information for each occurrence of X_1. (Remember, from the way that logs work, you can work out a log to the base 2 by using the expression

$$\log_b(x) = \frac{\log_a(x)}{\log_a(b)} \tag{9.3}$$

so we can say (for example) that $\log_2(x) = \ln(x)/\ln 2.$) Similarly, $h_2 = 1$, $h_3 = 4.32$, $h_4 = 4.06$, $h_5 = 4.18$, $h_6 = 6.64$, $h_7 = 2.56$ and $h_8 = 5.06$.

This is a very clear demonstration that characters which appear more often only convey a relatively small amount of actual information per occurrence. For example, X_2 which, typically, makes up half the symbols in a message only provides 1 bit of information per appearance despite using 3 bits to code. You can also see that rare symbols provide a relatively large amount of information. This result is interesting because it shows that the

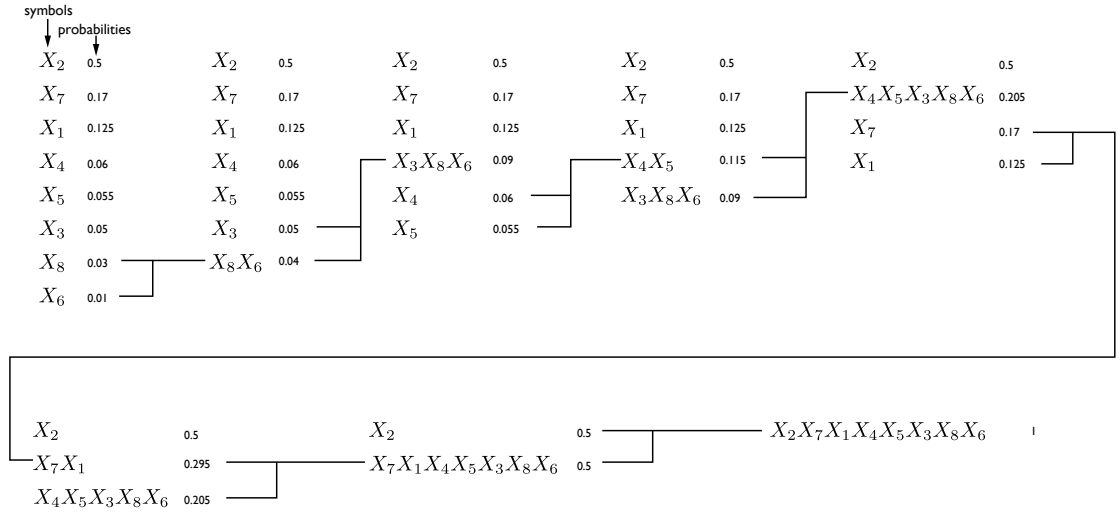


Figure 9.2: The construction of a *Huffman tree*.

fact that a symbol or character might be coded using three binary bits doesn't mean that it always carries just three bits worth of actual information. (However, if all the symbols were equally probable they'd each have a $p = 0.125$ and an h value of 3. Then the actual number of bits used to represent each symbol would equal the amount of actual information per appearance.)

It is this difference between the actual information content (in bits) and the number of bits required for fixed-length representation (where every symbol is represented by the same number of bits (3 in this example)) which allows us to compress data using the *Huffman* method. **Huffman coding** represents each character or symbol by a string of bits whose length varies from character to character. Highly probable characters (like X_2) are represented by short strings of bits and rare characters (like X_6) are represented by longer strings. The way Huffman codes are produced is shown in figure 9.2.

We start off by sorting the symbols according to their probability in descending order: this forms the first level of the tree: We then take the two symbols with the lowest probabilities and bring them together, adding their probabilities. We treat this gathered pair as a new symbol, and along with the other symbols, again list them in descending order of probability. We repeat this process until we end up having brought all the symbols together. (Obviously, the resulting 'symbol' must have a probability of 1.) The resulting *Huffman tree* will have as many levels as we have different characters/symbols.

In principle, we might find that three or more candidates at a given level share the lowest accumulated probability values: if this occurs we can pick two of them to combine and just carry on.

The recipe described above and figure 9.2 lead to the construction of the Huffman tree, but in order to be of any use, we now need to use it to construct the Huffman code for a given

character/symbol. For each symbol, we start at the *base* of the tree (where we have only one 'symbol' consisting of all the symbols combined, and work our way up the 'branches' until we find the 'lowest' branch (or furthest towards the right in the diagram) in which the character appears *on its own*. The points we arrive at in this way are the *terminal nodes* of the characters, and there is only one for each character. Each time a decision is made, we label the upper decision with a '1' and the lower decision with a '0', as shown in figure 9.3. If we note the values associated with each decision for each character, then the resulting string of bits forms the Huffman code for that character.

For example, consider the character X_1. Starting from the base, we are immediately confronted with a decision: we choose the path that includes X_1, and note the value associated with the decision (0). The next decision (which leads to the combination X_1X_7) yields 1, and the following decision gives us 0. At this point, we have arrived at X_1 by itself, i.e. the location of the *terminal node* for X_1 and thus we represent X_1 by the binary number 010. The codes for the other characters are worked out in the same way, and the complete set of Huffman codes is also shown in figure 9.3.

We can see from the above that the Huffman code for the most commonly occurring character (X_2) is the shortest, having just one bit. The least common codes (for X_6 and X_8) are the longest at five bits each. The codes for X_1 and X_7, which occur with a probability whose value almost equals the value we'd get for equally probable characters, have 3 bits each, the same number as we need for an equal-length coding system.

Consider a specific, but fairly typical 16 character long message consisting of:

$$X_4X_2X_2X_1X_2X_2X_3X_5X_2X_2X_8X_2X_7X_2X_1X_7$$

If we used a fixed-length coding system, we'd need 48 bits to store/communicate this message. However, using our Huffman codes, we'd only need 37 bits. Using equation 9.3 and the probabilities for each character, we can work out that the information content of the message is $H = 36.74$ bits, which is just a little less than what a 'typical' message would contain. Because the number of occurrences of each character is pretty close to what their probabilities would suggest (which it should be as we've chosen what is close to a typical message!), then the information content of the message is pretty close to the number of bits needed to encode it.

So far so good: we have a nice encoding process. However, it's generally quite nice to be able to decode messages as well as encode them! To do this, we start with the first bit in our bit-stream. We then check against our list of codes: if the first bit on its own is a legal code (i.e. it corresponds to one of our characters), then it represents one of our characters, and we note which character.

If the first bit on its own *isn't* a legal code, then we consider the first *two* bits: if they together are a legal code, then we note the associated character. If not, we consider the first *three* bits of the message, and so on until we *do* find a legal code. We are effectively

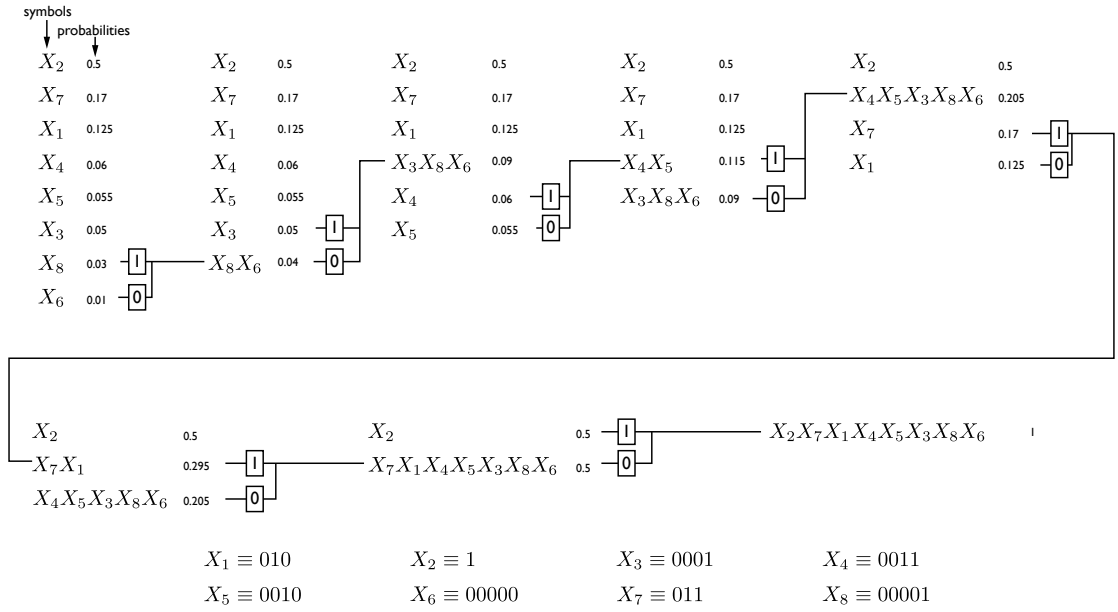


Figure 9.3: By working back from the end of the Huffman tree, and noting the bit values each time we need to make a decision to follow our chosen symbol higher up, we can construct the codes for each symbol X_i.

'working our way up' the Huffman tree until we reach the termination node of the relevant character that we are decoding. Once we've found a legal code, we repeat the process until we run out of bits by which time we've decoded the entire message.

The message example give above was deliberately chosen to be typical. Consider what we would be faced with if we used our Huffman codes to encode a string of 16 instances of X_6. This would require 80 bits, whereas if we'd stuck with a fixed-length coding system (3 bits per character), we'd only require 48 bits. In this particular case, using the Huffman codes has resulted in a *longer* bit stream. This makes a good deal of sense, as such a message is clearly *not* typical.

It can sometimes, therefore, be useful to base the Huffman codes on the *actual probabilities for each character in a specific message*. This is referred to as *adaptive* coding. However, when this is done, it requires the codes to be used for each character to be communicated as part of the message in some standard form. This would mean that for a 16 character message, the total length would be so long as to make the exercise pointless, but if the message was a very long one, then it'd be worth considering. (Do bear in mind, though, that longer messages would be more likely to be 'typical', so sticking with the overall statistics for the character set would probably work out okay.)

The big strength of adaptive coding really comes when we're compressing different types of data which have different symbol probabilities. For example, both a 256 level grey scale image and a plain-text file in ASCII format will require a total of 256 different symbols, which can

each be represented as an 8-bit byte. However, the frequencies with which the different byte values will occur will vary greatly between an image and a message in English text. A coding scheme devised for text would be a pretty poor way to encode an image, and vice versa. Adaptive encoding means that we can use the same program to compress or decompress different types of file and still generally get reasonable results.

9.2 Data reduction

The previous section shows that it's possible to compress data without any loss of information (unless, of course, our data describes something which looks like random noise). Sometimes, however, this true data compression still results in files which are larger than we'd like. In such cases, it may be necessary to discard some of the information in order to fit all the data into a file of the size that we can handle. When we do this, we discard *unimportant* information, so that when we reconstruct the data at the other end, the loss of information won't have had a significant effect. 'Important' and 'significant' in this case usually relate to how perceptible the changes to the file that the removal of the data has resulted in will be. For example, if we're removing data from an image, how much effect does it have on what the final image look like? If we discard some data from a sound file, can we tell when we listen to the result?

Such processes are often referred to as data compression, but they are actually discarding data, hence the distinction between data compression and data reduction/thinning that we made at the beginning of the chapter.

Removal of data while not compromising the end result can be a remarkably successful process, but relies very much on sensible decisions about constitutes 'unimportant' information when we're throwing it away. This depends a great deal on the application in hand, so we'll look at a couple of examples. We'll firstly consider JPEG encoding for images, then have a look at some of the key ideas associated with data reduction/thinning in digital audio files.

9.2.1 JPEG encoding

The .jpg file format is a widely use format for images, particularly photographs. (The name 'JPEG' is actually an acronym for the 'Joint Photographic Experts Group' that defined the standard, but it's become conventional to refer to the format by this name as well.) There is no single format for JPEG images, instead the JPEG (JFIF to be pedantic, but life is too short...) is more like a series of options that may be selected and used as preferred. Here we'll just outline a typical JPEG thinning process.

In our run-length encoding example we saw how an image may be represented by a bitmap that records a set of values to specify the brightness and colour of a pattern of pixels. In the

example we used there, we used a greyscale image, so only needed one number to describe each pixel. A colour picture obviously requires the recording of extra data.

The simplest way to record the colour and brightness information would be to have three values, R, G, and B, to indicate the levels of red, green, and blue for each pixel. This approach works but turns out not to be very efficient for a number of reasons. For example, the human eye is much more sensitive to changes in the intensity of green light than either red or blue. In addition the eye has a lower resolution for colour changes than for changes in brightness.

To take account of these characteristics, the JPEG (along with many other image related systems) define the brightness and colour of each pixel in terms of three specific values.

The first is the *luminance* which is defined as

$$Y = 0.3R + 0.59G + 0.11B \tag{9.4}$$

This is equivalent to the 'brightness' of each pixel. The levels of red, green and blue are weighted by differing factors, which makes the resulting value more sensitive to changes in the green level than to the other colours. In this way, the luminance actually takes the behaviour of the eye into account to give the optimum performance for a given range of data values.

The two other values for defining each pixel are the *colour difference* or *chrominance* signals which can be defined as

$$C_B = B - Y \tag{9.5}$$

and

$$C_R = R - Y \tag{9.6}$$

We now have three different values, but because of how they are related to each other and to R, G and B, we can still convey any trio of R, G and B values, so no information is lost. All we've done is turn three numbers for describing a pixel into a different three numbers. The new set of numbers, however, offer some advantages which become apparent during the JPEG creation process.

The JPEG format includes two requirements regarding the size and layout of an image:

- Images cannot be more than 65 536 pixels in width or height.
- The chrominance values must have *half the horizontal resolution* of the luminance values.

We can use figure 9.4 to see what this means. Figure 9.4 (a) represents a small part of the colour pattern of an image in terms of an array of red green and blue values. These are converted to into a set of luminance and chrominance values as shown inn figure 9.4 (b). The luminance values are obtained from the R, G and B values using equation 9.4, and

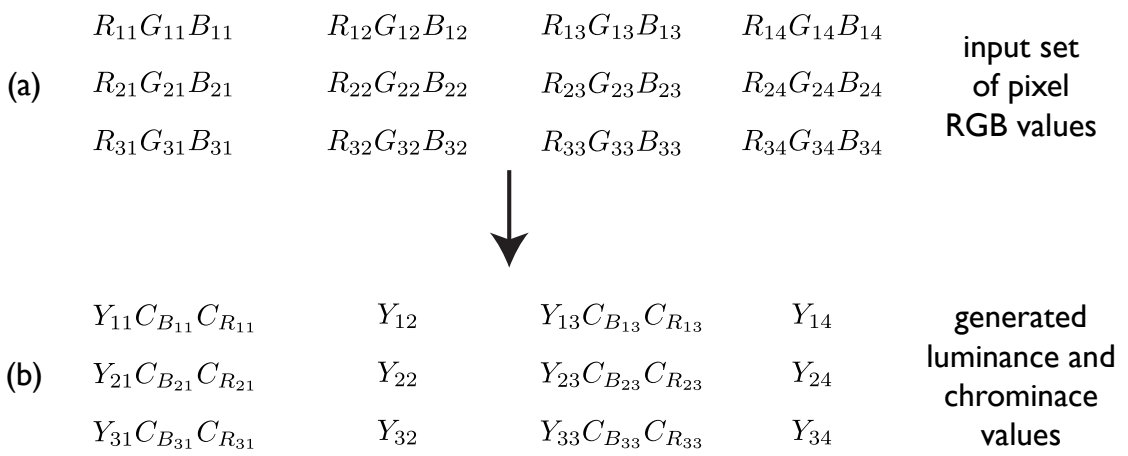


Figure 9.4: (a) An input set of pixel colour data as red, green and blue values. (b) The luminance and chrominance values generated from these. (Note that although the chrominance values are labelled by their location in the lower grid, they are generated from horizontally adjacent *pairs* of pixels.)

we have one for each pixel. The chrominance values are average the colour information for horizontally adjacent pairs of input pixels. This obviously must mean that some of the original image data is lost.

We mentioned earlier that the key to successful 'lossy compression' was to be sensible about the way in which we choose to throw away data. The human eye tends to recognise shapes in terms of brightness changes, so the loss of colour information usually has surprisingly little effect on the appearance of the image. It does, however, mean that we've already discarded on the order of a third of the data without significantly degrading the *perceived* image. (The 'perceived' is important here!) The arrangement where we half the number of colour values by only producing results for alternate horizontal positions is said to be in '4:2:2' format, and this arrangement is required for the input to standard JPEG compression.

The luminance and chrominance values are now divided into three separate sets for Y_{mn}, $C_{B_{mn}}$ and $C_{R_{mn}}$, and processed separately.

Each set of values is then split into chunks of 8×8 values, and each chunk is then processed by a method known as the *discrete cosine transform* or DCT. This is an operation similar to the (discrete) Fourier transform, but only works out the cosine values associated with each frequency. (The'discrete' comes from the fact that as we are applying the transform to discretely sampled points, then we can only expect components at particular frequencies.) The transform is done in *two* dimensions, as an image is obviously two dimensional.

The DCT values for the luminance (for example) are computed using the expression

$$Z_{ij} = \frac{1}{4} c_i c_j \sum_{m=0}^{7} \sum_{n=0}^{7} Y_{mn} \cos\left(\frac{\pi i(m+1/2)}{8}\right) \cos\left(\frac{\pi j(n+1/2)}{8}\right) \qquad (9.7)$$

where i and j both run from 0 to 7. This results in a 2D array or matrix (Z), also containing 8×8 values which describe the *spatial frequencies* that describe the contents of the 8×8 chunk that we've just transformed. The notion of a spatial frequency may appear to be an odd one at first, but is in fact fairly straightforward: just as frequency describes how quickly something changes as a function of time, *spatial* frequency describes how quickly something changes as a function of position. The indices i and j denote a particular spatial frequency, and there are two indices as we're talking about frequencies in two different dimensions (horizontal and vertical directions in this case).

We thus now have a set of spectral values containing all the information in our input set. In practical terms, i and j represent the number of half-cycles of fluctuation in luminance across the block in the horizontal and vertical directions for each component in the spectrum. For example, $i = j = 0$ represents the amount of uniform brightness across the block, i.e. the 'dc' level of the data.

The coefficients c_i and c_j are chosen so as to normalise the results correctly. $c_i = 1/\sqrt{2}$ if $i = 0$ and $c_i = 1$ if $i \neq 0$, similarly for c_j.

It's a fair at this stage to question the point of all this: why bother transforming from a spatial representation to an inverse spatial representation (i.e. spatial frequency) at all? The answer comes back to the idea of being sensible about what information we choose to throw away, as will become clearer in the next stage of the process.

Having divided the image into blocks and performed a DCT on each block to convert the spatial information from each block into spectral information for each block. We now proceed to discard more of the data. We do this by selecting a *threshold* value for the elements Z_{ij}, and any values which fall below this, we set to zero. What we're doing here is deciding that if a given frequency component is small enough, then it won't make too much difference to the perception of the image, so we may as well do without it. This is shown in figure 9.5. Figure 9.5 (a) shows a 'typical' array of Z_{ij} values which we wish to process. The values closer to the top left corner of the array represent the lower spatial frequencies, and those towards the bottom right corner represent the higher spatial frequencies. The values are chosen purely for the sake of example, but do show that in typical images, there do tend to be larger components at lower spatial frequencies (slow changes in luminance across the image) than at higher spatial frequencies.

As can be seen from figure 9.5 (b), the thresholded (is that a word?) data contains a lot of zero values. This shows that we must have lost some information, but the idea is that we choose the threshold value (in this case 20) so that it should not make a significant difference to the reconstructed image. The next step is to *run-length encode* the thresholded spectral data. This is done by taking a 'zig-zag' path through the array as shown in figure 9.6. In most photographic images of natural scenes, there is a tendency in most areas of the image for the low spatial-frequency components to have significantly larger amplitudes than the high spatial frequency components. The thresholding process therefore tends to remove the high frequency details of the image. It will only do this, however, in blocks of the image where the high-frequency data is small enough to fall below the chosen threshold level. In areas

121	-81	58	22	-22	10	-3	0
70	-31	-59	4	-2	0	-5	-1
-65	41	-20	36	22	10	0	-1
39	-6	-39	17	7	5	1	0
-6	-12	-20	-5	-8	0	0	0
-15	11	0	2	1	0	0	0
4	0	-1	0	-1	0	0	0
-1	0	1	0	0	0	0	0

(a)

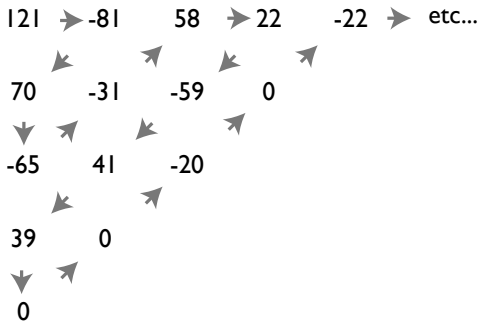
121	-81	58	22	-22	0	0	0
70	-31	-59	0	0	0	0	0
-65	41	-20	36	22	0	0	0
39	0	-39	0	0	0	0	0
0	0	-20	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(b)

Figure 9.5: The effect of *thresholding* on the result of a DCT on an 8×8 block of data. (A) The DCT data and (b) the result of thresholding with a value of 20.

121	-81	58	22	-22	0	0	0
70	-31	-59	0	0	0	0	0
-65	41	-20	36	22	0	0	0
39	0	-39	0	0	0	0	0
0	0	-20	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(a)



(b)

Figure 9.6: Run-length encoding of a thresholded block of spectral data. (a) The data that we wish to encode and (b) the run-length encoding takes a 'zig-zag' path through the array.

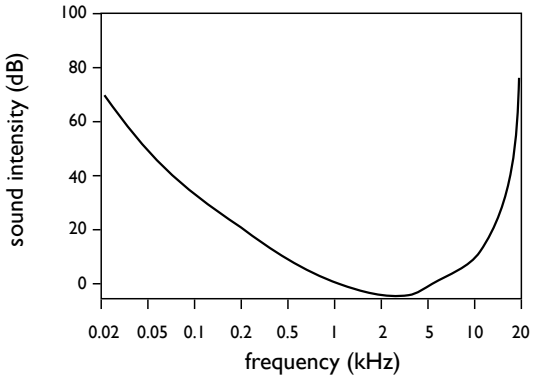


Figure 9.7: The hearing threshold for a typical healthy young adult male.

containing lots of high spatial frequencies (e.g at a well defined edge), then these will be above the threshold, and thus will be preserved by the process.

The path taken through the array when doing the run length encoding means that we tend to cluster together the low frequency components (top left corner) and cluster together the high frequency components. This maximises the length of runs of successive zeroes, and increases the amount of compression that we get from the run-length encoding. Note that the JPEG process thus includes both data reduction/thinning AND true data compression.

9.2.2 Audio 'compression' ideas

Another huge application of data thinning/reduction is in the field of digital audio. There are a huge number of digital audio formats out there. Some, such as the FLAC format are lossless formats, using just true data compression. We'll consider some of the general principles behind 'lossy compression' formats (which include MP3, AAC, ATRAC, etc) without getting too specific about one format. The ideas are more important than the details!

As with images, when we thin or reduce audio data, we're obviously throwing away some of the original information, so the key to a successful format is to make intelligent choices about *what* we discard. The criteria are clearly going to be different than for the JPEG example, but the general principle still holds that if we can identify information that doesn't make a large *perceived* difference to the final result, then we can get rid of it.

It turns out that because of the way our hearing system works, that we often *can* identify audio information that we can discard without greatly affecting the perceived quality. We exploit two characteristics of our hearing system: *masking* and the *threshold* level. *masking* is the effect where a loud sound affects our ability to perceive a quieter tone at a similar frequency, and the *threshold* represents the lowest sound level which can be heard.

Figure 9.7 shows the lowest sound intensity that which can be heard by a typical healthy

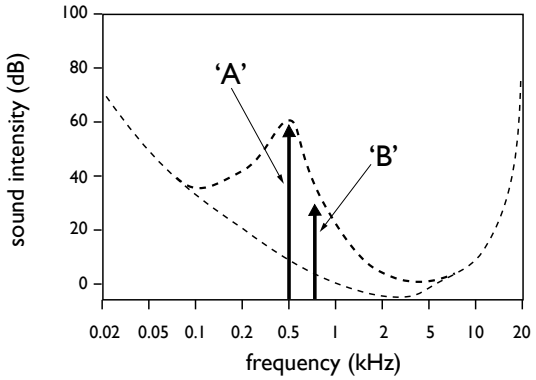


Figure 9.8: The effect of *masking*: the louder tone A renders the quieter tone, B, inaudible, even though they are not at exactly the same frequency.

young male, aged 15 to 25. (Females tend to have slightly better hearing, and hearing often gets worse with age.) The line indicates the quietest sounds at various frequencies which are *just* on the limit of being audible. The sound intensities are quoted in dB referenced to a level of $10^{-12}\ \mathrm{Wm}^{-2}$, which is about the limit of hearing at around a couple of kHz.

Figure 9.8 illustrates the phenomenon of *masking*. In this example, a tone of 60 dB at 500 Hz makes another tone of 30 dB at 700 Hz 'inaudible'. (This arises from the properties of a mechanical resonator in the ear that forms part of the sensing.) This effect along with the threshold effect can be exploited to discard some frequency components in an audio signal without the perceived sound of the reconstructed waveforms being drastically affected. In reality, hearing varies from person to person, so it is appropriate to question how effective the process may be in a given case, but in general modern audio data thinning/reduction systems can produce convincing results and very large degrees of 'compression'.

A number of different audio encoding systems exploit these effects to identify data that may be discarded. The time domain audio signal is split into overlapping chunks, which are then transformed into frequency spectra, often using a form of discrete cosine transform (but in 1D, rather than 2D). These spectra are analysed for the components they contain, and any that are below threshold, or masked by other components are discarded.

The details of how this is done vary from system to system, but one thing that lossy audio compression techniques have in common is that they rely on a good *psycho-acoustic model* of how the hearing system works in order to make sensible judgements about what data can be discarded, and this parallels some of the ideas in the JPEG process. The results can be impressive: lossy compression methods (i.e. data reduction/thinning) can result in outputs containing from around 5% to 20% of the input data, compared to about 50% to 60% for lossless audio compression methods (i.e. 'true' data compression.) A five-minute audio stream sampled in stereo at 44.1 kHz with 16 bits/sample would contain 50.4 MB of data, whereas a song lasting this length on your digital audio player may only be a handful of MB.

9.3 Chapter 9: take-home messages

You should now understand how we can use data **compression** to remove redundancy from messages and store/transmit them using fewer bits. You should also understand how **run-length** and **Huffman** codes are generated and used.

You should now the basic principles behind **data thinning** or **reduction**, and understand how the presented examples work. Each example exploits features of the situation where it is applied (i.e. the properties of human vision and hearing) to identify and discard relatively 'unimportant' information. Each technique optimises its output in an adaptive way to the signal details. You should now also understand that data reduction techniques can be useful provided that the process of deciding which parts of the signal to discard is performed in a manner appropriate to the context, i.e. it depends upon the nature of the signal and the use it will be put to. It should also be clear that the success of data thinning depends critically on how well the system avoids removing information about signal details which do, in fact, matter!

Chapter 10

Sensors, amplifiers and noise

10.1 Basic properties of sensors

Sensors take a variety of forms, and perform a vast range of functions. When a scientist or engineer thinks of a sensor they usually imagine some device like a microphone, designed to respond to variations in air pressure and produce a corresponding electrical signal. Many other types of sensor exist. For example, I am typing this text into a computer using an array of 'keys'. These are a set of pressure or movement sensors which respond to my touch with signals which trigger a computer into action.

Every sensor is a form of *transducer*. The microphone is a good example, it converts some of the input acoustical power falling upon it into electrical power. In principle, we can measure anything for which we can devise a suitable sensor. In this lecture we will concentrate on simple electronic sensors whose output can be detected and boosted using an amplifier. The basic properties of a sensor and amplifier are illustrated in figure 10.1. Note that, so far as the amplifier is concerned, the sensor is a signal 'source' irrespective of where the signal may initially come from.

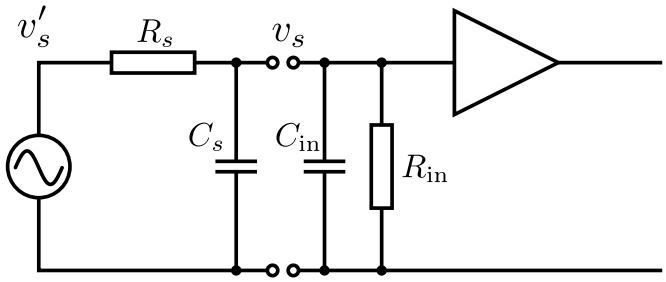


Figure 10.1: A source-amplifier combination.

The source is assumed to be producing a varying signal voltage, v_s'. The amplifier has an input resistance, R_{in}. (Both the source/sensor and the amplifier also have some capacitance, but for now we'll ignore that.) The instantaneous signal power level entering the amplifier's input will be

$$P_{\text{in}} = \frac{v_s^2}{R_{\text{in}}} \tag{10.1}$$

Now P_{in} *must* be finite and limited by whatever physical process is driving the sensor. Yet equation 10.1 seems to imply that we could always get a higher power level from the source by changing to an amplifier with a lower input resistance, R_{in}. This apparent contradiction can be resolved by accepting that the voltage, v_s , seen coming from the source must, itself, depend upon the choice of R_{in}. The way in which this occurs should be clear from figure 10.1. The sensor must have a non-zero source resistance, R_s, which its output passes through. As a result the signal voltage at the amplifier's input will be

$$v_s = \frac{v_s' R_{\text{in}}}{R_s + R_{\text{in}}} \tag{10.2}$$

(This makes sense as we can consider R_s and R_{in} as forming a potential divider to which is applied the voltage v_s', and is somewhat of a re-tread about the noise power matching arguments made earlier on.) The signal power entering the amplifier will therefore be

$$P_{\text{in}} = \frac{{v_s'}^2 R_{\text{in}}}{(R_s + R_{\text{in}})^2} \tag{10.3}$$

The maximum signal power which the amplifier can draw will be available when $R_{\text{in}} = R_s$. A lower input resistance **loads** the source too much, v_s falls, and reduces the available power.

This result is a general one which arises because the amount of power generated can never be infinite. *All* signal sources will have a non-zero source resistance (or output resistance). In a similar way we can expect all real amplifiers and signal sources to exhibit a non-zero capacitance. This is called the *source capacitance* for a source/sensor and the *input capacitance* for an amplifier.

From figure 10.1 we can see that these two capacitances, C_s and C_{in} , are in parallel. For the voltage seen at the amplifier's input to be able to change we have to alter the amounts of charge stored in these capacitances. The current required to do this must come through R_s and R_{in}. From the point of view of the capacitors these offer two parallel routes for charge to move from one end of the capacitors to the other, so they appear *in parallel*. This combination of capacitance and resistance means that the voltage seen by the amplifier, v_s, *cannot respond instantly* to a swift change in the source voltage, v_s'. Changes in v_s are smoothed out with a time constant, $\tau = RC$, where R and C are the parallel combinations of the input and amplifier values.

These resistances and capacitances may be actual components in the system, or may be a result of some other physical mechanisms. In each case their effects can be modelled using the kind of circuit shown in figure 10.1, irrespective of whether they're deliberate additions or stray effects. These capacitances and resistances are *always* non-zero. Hence it is *impossible*

to change a measured signal level infinitely quickly. This is another way of stating the basic principle of information processing that no signal can have an infinite bandwidth (i.e. reach infinite frequencies). If it did, it would be able to convey an infinite amount of information in a limited time. In the real world this is impossible.

10.2 Amplifier noise

When designing or choosing a measurement system we need to be able to compare the performance of various amplifiers and select the ones most appropriate for the job in hand. Various criteria will affect the choice, ranging from price to gain. When making accurate measurements it is usually preferable to choose amplifiers which generate the lowest possible level of noise.

A wide range of devices have been used to amplify signals. Although their details differ we can expect that they all operate at a temperature above absolute zero and, as a result, will produce some thermal noise. Similarly, for their input and output signals have non-zero powers, they must pass some current, hence producing some shot noise.

It seems to be one of the basic laws of nature that *all* gain devices, generate *excess* noise, i.e. they all produce more noise than we would predict from adding together the thermal noise and shot noise. For the sake of example we can consider the behaviour of an amplifier based on a Field Effect Transistor (FET) of the sort illustrated in 10.2. The device shown is called an n-channel junction FET. This is made by forming a channel of n-type semiconductor in a substrate of p-type semiconductor. The channel-substrate boundary forms a pn junction which behaves like a normal diode. As a result, provided we avoid forward biassing the gate-channel boundary, then almost no current flows between the gate and the channel. The charge in the gate (and the substrate) repels the free electrons in the channel and prevents them from coming too close to the walls of the channel. This produces *depletion zones* near the walls whose size depends upon the applied gate potential.

When we apply a voltage between the source and drain contacts, electrons flow through that part of the channel which has not been depleted. We can think of the channel as a slab of resistive material of length, L, and cross sectional area, A. For a material of resistivity, ρ, such a slab would have an end-to-end resistance, $R = \rho L/A$. Varying the gate voltage alters the depletion zones and hence changes the effective cross sectional area, A, of the channel. As a consequence, when we vary the gate potential the effective resistance between source and drain changes. The FET therefore acts as a source-drain resistor whose value depends upon the gate potential. This description of the operation of an FET is too simple to explain all the detailed behaviour of a real device but it suits our purposes. In practice the drain-source voltage is usually sufficiently large that the potential difference between the drain and gate is much greater than that between source and gate. As a result the depletion region inside the channel is much smaller at the source end than at the drain, so the cross-sectional area of the effective channel is quite small at one end.

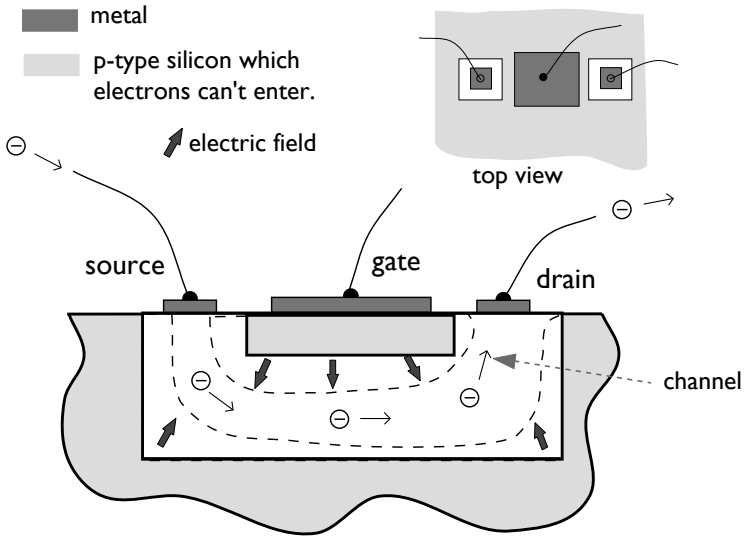


Figure 10.2: n-channel junction field-effect-transistor (JFET).

From the simple description given above we would expect the channel current to increase in proportion with the applied drain-source voltage. However there is a tendency for any increase in drain voltage to enlarge the depleted region near the drain. This reduces the channel area, limiting any current increase. As a result we find that, for reasonably large drain-source voltages, the FET behaves more like a device which passes a drain-source current controlled by the gate potential. Because of this the gain of an FET is usually given in terms of a *transconductance*. This can be defined as the change in drain-source current divided by the change in gate potential which causes it.

The gate-channel is normally reverse biassed, so almost no gate current is required to maintain a given gate potential. As a consequence the input resistance of an FET is very high, typically 10 MΩ or more. However, to alter the gate potential we must vary the charge density within the gate, so we have to move some charge into or out of the gate. As a consequence the gate-channel junction has a small capacitance. For a typical FET the gate-channel capacitance is a few tens of pF or more.

Noise is generated within the FET by various physical processes, for example:

- Shot noise fluctuations in the current flowing through the channel;
- Thermal noise in the channel resistance;
- Thermal motions of the gate charge carriers, producing random fluctuations in the size and shape of the depletion region, and hence in the channel resistance.

All of these effects (and others which have been ignored) will vary according to the bias voltages and currents, and the exact details of the device. Rather than undertake a detailed

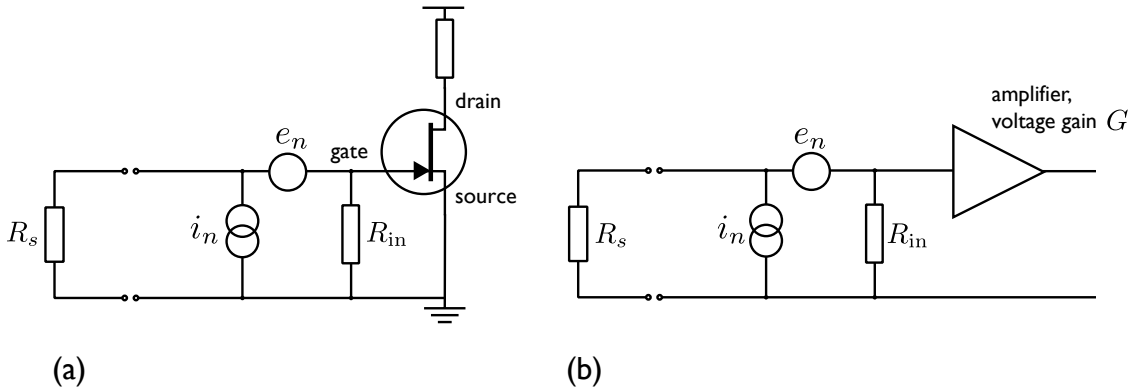


Figure 10.3: Noise models for amplifiers. (a) A simple JFET amplifier. (b) A general amplifier noise model.

analysis of these effects (which may be futile as some of the underlying processes are poorly understood!) we can model the behaviour of the FET (or *any other gain device*) in terms of a fictitious pair of noise generators. This approach is very useful when we are mainly concerned with comparing one amplifier with another and don't want to bother with the details of where the noise is actually coming from.

Figure 10.3 (a) represents a simple amplifier using an FET. The noise produced by the real FET and the other components which make up the amplifier are assumed to come from a mythical noise voltage generator, e_n, and noise current generator, i_n, connected to the amplifier input. Figure 10.3 (b) represents the way in which this idea can be generalised to apply to any amplifier, irrespective of its details. The noise performance of any amplifier can now be described by the appropriate values of e_n and i_n . These are normally specified as rms voltage and current *spectral densities*, the units of e_n usually being $\mathrm{nV}/\sqrt{\mathrm{Hz}}$, and those of i_n being $\mathrm{pA}/\sqrt{\mathrm{Hz}}$. Figure 10.4 shows the typical manner in which they vary with fluctuation frequency.

A noise producing process which we've not mentioned so far is *generation-recombination noise* (GR-noise). A large number of electrons do not normally take part in conduction as they do not have enough energy to escape their orbit around a particular atom. Every now and then, however, one of these bound electrons may interact with a passing electron or a lattice vibration (a *phonon*) and gain enough energy to escape. This process can be regarded as 'lifting' an electron up into the conduction band and leaving behind it a 'hole' in a lower band.

Sometimes the newly freed electron does not move away swiftly enough to avoid dropping back into the hole. But if it manages to get away we find that a pair of extra charge carriers have joined those able to provide current flow through the material. Eventually, an electron will pass close enough to the hole to fall into it and the total number of available charge carriers will return to its original value. This process means that the current flowing through the channel as a result of an applied voltage will tend to fluctuate. (Note that this process is different from shot noise.)

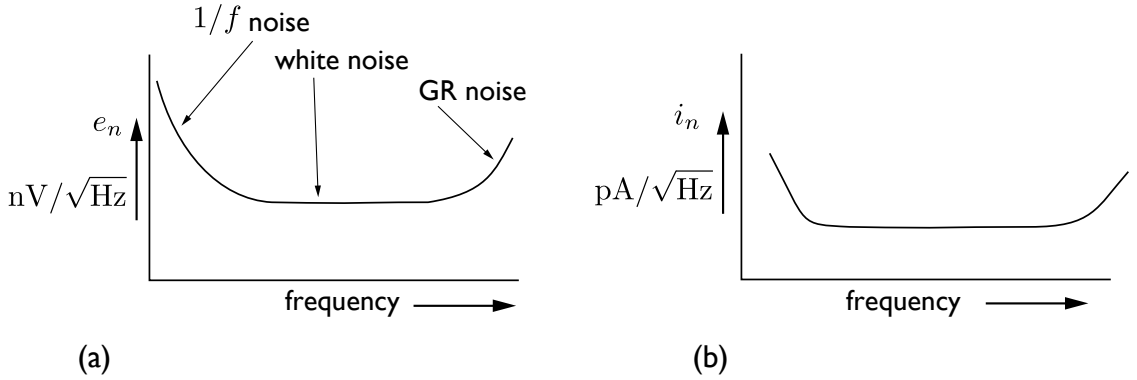


Figure 10.4: Typical shapes of noise power spectral density curves: (a) voltage noise and (b) current noise.

There is a difference in potential between the channel and the gate/substrate. Any new electron-hole pairs generated near the channel walls will tend to be pulled apart. For an n-channel FET the field will sweep the 'new' electron into the channel and pull the hole back into the substrate. As a consequence, the random creation of carrier pairs in the region near the gate-channel junction produces a small, randomly varying, current flowing across the boundary. This in turn causes random variations in the size and shape of the depletion region which produces an extra noise current into the channel.

From figure 10.4 we can see that, at high frequencies, the noise power spectral density tends to increase with frequency. This is due to GR-noise produced by quantum mechanical effects. Although energy must be conserved overall, quantum mechanics permits the energy of a system to fluctuate by an amount ΔE provided the fluctuation only lasts a time $\Delta T \approx h/\Delta E$, (where h is Plank's constant.) In a semiconductor whose energy gap is ΔE this means that electron-hole pairs may be created without the required specific energy input, ΔE, provided they vanish again in a time ΔT. As a result, when we consider periods of time which are less than this time the density of carriers in the material appears to fluctuate randomly.

These short-lived random variations in the number of free charges mean that the current which flows in response to an electric field also varies. If we consider shorter periods we are allowed to consider larger energy fluctuations and an increasing number of electrons, tied more strongly to their atoms, can briefly join in this process. Hence this effect produces a noise level which increases with frequency (i.e. with decreasing fluctuation period). This effect does not create noise power out of nothing. The initial ΔE is a sort of 'loan' which must be repaid since, if we want to observe a change in the current, we must apply an electric field to drag the electron-hole pair apart. This field hence does some work in producing the extra current.

All gain devices exhibit some amounts of voltage noise, e_n, and current noise, i_n. The precise levels and their associated frequency spectra depend upon the type of device, how it is made and operated. When comparing bipolar transistors with FETs we generally find that bipolar devices have higher current noise levels and FETs have higher voltage noise.

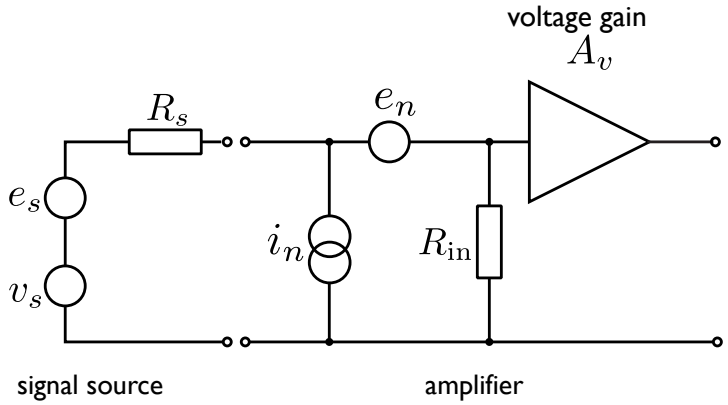


Figure 10.5: A model to illustrate source-amplifier coupling, where the source resistance is modelled as a 'noise-less' resistor in series with a noise voltage e_s.

10.3 Specifiying amplifier noise

In practice we are often not told how e_n and i_n vary with frequency for a particular amplifier. Instead we are presented with a single value which indicates the overall amount of noise the amplifier produces. This value may be specified in various ways. The most common measures are the *noise resistance*, R_n, the *noise temperature*, T_n, the *noise factor*, F, and the *noise figure*, NF. Whilst any one of these values can be useful for encapsulating the behaviour of an amplifier it should be clear that a single number cannot contain all the information offered by a detailed knowledge of the e_n and i_n spectra. They should therefore be used with care.

Figure 10.5 illustrates a system which amplifies the signal voltage, v_s, generated by a source whose output resistance is R_s . The amplifier is assumed to have a voltage gain of A_v , an input impedance of R_{in}, and produces a noise level equivalent to a combination of a noise voltage generator, e_n, and noise current generator, i_n, located as shown at the amplifier input. A signal source at a temperature, T, will itself produce thermal noise equivalent to a voltage generator whose rms magnitude is

$$e_s = \sqrt{4k_B T B R_s} \tag{10.4}$$

placed in series with the source.

For the sake of simplicity we'll assume a unit bandwidth ($B = 1$ Hz) and that the source does not produce any other form of noise. This means that the source is as 'noise free' as we can expect in practice. Taking into account all of the noise generators shown in figure 10.5, the total rms noise voltage, e_0 , which is output by the amplifier will be such that

$$e_0^2 = |A_v|^2 \left[\left(\frac{e_n R_{\text{in}}}{R_s + R_{\text{in}}} \right)^2 + \left(\frac{e_s R_{\text{in}}}{R_s + R_{\text{in}}} \right)^2 + \left(\frac{i_n R_s R_{\text{in}}}{R_s + R_{\text{in}}} \right)^2 \right] \tag{10.5}$$

The source signal, v_s will produce a voltage v_o at the amplifier output:

$$v_o = \frac{A_v R_{\text{in}} v_s}{R_s + R_{\text{in}}} \tag{10.6}$$

We can now define the *system* gain (as distinct from the *amplifier* gain) as

$$H \equiv \frac{v_o}{v_s} = \frac{A_v R_{\text{in}}}{R_s + R_{\text{in}}} \tag{10.7}$$

Note that this value takes into account both the amplifier's voltage gain and the voltage attenuation produced by R_s and R_{in} acting as a potential divider. Hence $H < A_v$.

We can now regard the *total* noise at the output of the system as being due to a single input noise voltage generator, e_t. From the above definition of the system gain we can expect that

$$e_t = \frac{e_o}{H} \tag{10.8}$$

which, when combined with the above expressions, leads to

$$e_t^2 = e_s^2 + e_n^2 + i_n^2 R_s^2 \tag{10.9}$$

The noise in the system has now been gathered into a single number, e_t whose value indicates the total noise present in the system. From this we can define each of the noise measures mentioned earlier.

The **noise factor**, F is defined as

$$F \equiv \frac{\text{total noise power}}{\text{source resistance noise power}} \tag{10.10}$$

The **noise figure** is the same quantity in dB:

$$NF = 10 \log_{10}(F) \tag{10.11}$$

For a perfectly noise-free amplifier e_n and i_n would both be zero. Such an amplifier would have a noise factor of unity and a noise figure of 0 dB.

The **noise resistance**, R_n, can be defined by equating the amplifier's contribution to the total noise to a thermal noise level

$$4k_B T R_n \equiv e_n^2 + i_n^2 R_s^2 \tag{10.12}$$

where T is the physical temperature of the amplifier (normally assumed to be around 300 K). It is prudent to avoid noise resistance values to avoid any confusion with input impedance.

The **noise temperature**, is a concept which avoids this confusion. The noise temperature, T_n, is defined by

$$4k_B T_n R_s \equiv e_n^2 + i_n^2 R_s^2 \tag{10.13}$$

Remember, T_n is *not* the physical temperature of the amplifier (although for very high performance systems, it may be influenced by it).

When comparing amplifiers and gain devices listed in manufacturer's catalogues we're frequently only given one of the above measures as an indication of the noise level. When examining these figures it is important to compare like with like. All of the above measures explicitly depend upon the chosen source resistance, R_s . Furthermore, the frequency dependence of e_n and i_n will vary from one gain device to another. As a result, two values of a noise measure are not directly comparable if they are given for different frequencies.

To measure the voltage and current noise levels of a particular amplifier we can observe the effects of short-circuiting and open-circuiting the amplifier input terminals (i.e. setting R_s to zero and to infinity). When $R_s = 0$, the current noise present cannot produce any observable voltage. The output noise from an amplifier whose input is shorted is therefore due only to its input voltage noise generator, e_n.

When we open-circuit the amplifier input we produce an effective source resistance of $R_s = \infty$. The noise current generator now produces an rms voltage $i_n R_{\text{in}}$ across the amplifier's input resistance. The noise fluctuations this produces are uncorrelated with those produced by the noise voltage generator. Hence they combine to produce a total rms noise voltage at the amplifier's input of $\sqrt{e_n^2 + i_n^2 R_{\text{in}}^2}$ when the amplifier input is open-circuit. By measuring the amplifier's output noise level in both situations we can therefore determine values for both e_n and i_n.

10.4 Optimising signal-to-noise ratio

Sometimes we can alter the physical details of a signal source or use a transformer to change the source's apparent resistance whilst maintaining the available signal power. Earlier we found that the amount of noise and signal power we see coming from a system depends upon the source resistance. Is there a value for the source resistance which produces an optimum (i.e. maximum) signal to noise ratio? If so, what is this value?

Figure 10.6 illustrates the use of an idealised transformer which has a turns ratio of $1 : \beta$. This steps up/down the output signal and noise voltages produced by the source to βv_s and βe_s, respectively. The transformer cannot output any more power than it receives. For loss-less transformer the input and output powers will be the same. As the output voltage is a factor β times that generated by the source it follows that the output current must be $1/\beta$ that flowing through the source. Consequently, the combination of the source and transformer appears to any following circuit to have an effective source resistance of $R'_s = \beta^2 R_s$.

Using similar arguments to those we've used previously, we can say that the total noise level for the system is equivalent to an rms voltage e_t such that

$$e_t^2 = (\beta e_s)^2 + e_n^2 + (i_n \beta^2 R_s)^2 \tag{10.14}$$

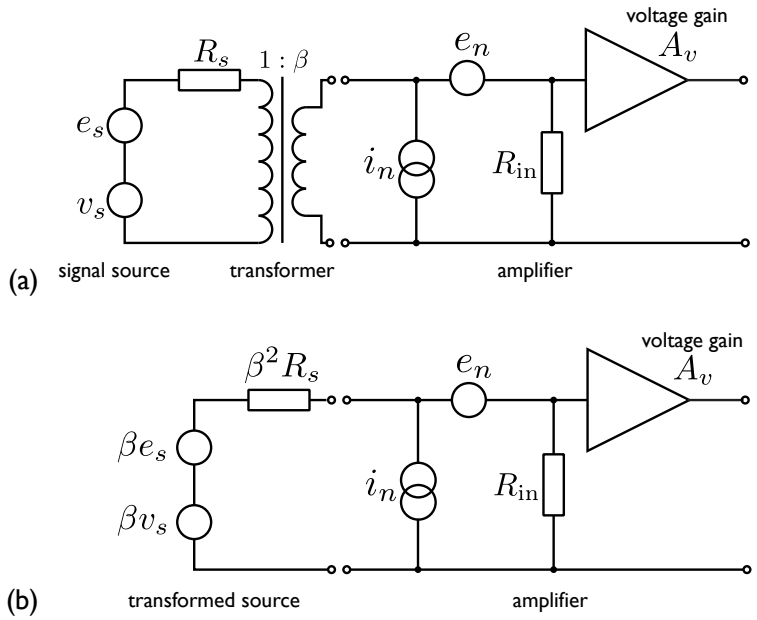


Figure 10.6: (a) A signal source connected to an amplifier via a transformer with a $1:\beta$ turns ratio. (b) What the configuration looks like when we model it as a source connected directly to the amplifier, showing what effect the transformer has on the apparent source signal and noise voltages, and the apparent source resistance.

A signal voltage, v_s, generated by the source will produce a signal/noise

$$\text{SNR} = \frac{(\beta v_s)^2}{e_t^2} \tag{10.15}$$

which obviously depends on β. Whenever possible it would be preferable to select the value of β which maximises SNR. This is equivalent to the value which minimises $e_t^2/(\beta v_s)^2$. The optimum choice of β can therefore be found from

$$\frac{\partial}{\partial \beta}\left[\frac{e_t^2}{(\beta v_s)^2}\right] = 0 \tag{10.16}$$

so

$$\frac{2\beta i_n^2 R_s^2}{v_s^2} - \frac{2e_n^2}{\beta^3 v_s^2} = 0 \tag{10.17}$$

which is satisfied when

$$\beta^2 = \frac{e_n}{i_n R_s} \tag{10.18}$$

Since the transformed source resistance, R_s', presented to the amplifier is $\beta^2 R_s$ it follows that the optimum value for this resistance will be

$$R_s' = \frac{e_n}{i_n} \tag{10.19}$$

For the reasoning above, we've assumed that the source resistance presented to the amplifier could be altered by a transformer. In other situations we can modify the signal source or

replace it with another and alter the source resistance without altering the available signal power. Irrespective of how this is done the above result tells us that, for an amplifier whose noise is represented by a voltage generator, e_n, and current generator, i_n, the maximum possible SNR will be obtained when the source resistance equals e_n/i_n.

We saw earlier that the optimum signal power transfer will occur when we choose a source resistance which equals the amplifier's input resistance. In general, e_n/i_n does not equal the input resistance of the amplifier. As a result, the source resistance which provides the best signal power transfer usually isn't usually the value which gives the best possible SNR.

Books on high frequency electronics tend to recommend that, whenever possible, we arrange that the source's output resistance and the amplifier's input resistance should be matched, i.e. have the same value. (The same approach is recommended when the signal is carried using a transmission line.) This gives the most efficient transfer of signal power, but may result in a SNR below the highest possible value. As a result there is often a conflict and we have to choose either a source resistance which provides the highest possible signal/noise ratio or a source resistance which maximises the signal power transferred.

In practice, we often have with a source whose properties are fixed, but we can select which amplifier to use. Each amplifier has a particular input resistance, R_{in}, and a noise level equivalent to specific e_n and i_n values. Because of the conflict between optimum signal transfer and SNR there won't usually be a perfect choice of amplifier so we must usually make a choice based upon an assessment of the relative importance of these factors for the job in hand.

10.5 Behaviour of cascaded amplifiers and transmission lines

Figure 10.7 illustrates the use of a pair of amplifiers to increase the signal from a source. Each amplifier has an input resistance, Z_{in}, and an output resistance, Z_{out}.

The signal power input into the first amplifier will be

$$P_{\text{in}} = \frac{v_s^2 Z_{\text{in}}}{(R_s + Z_{\text{in}})^2} \tag{10.20}$$

For a given signal voltage this power will be maximised if $R_s = Z_{\text{in}}$. As a result we should usually try to equate, or **match**, these impedances whenever it is convenient to do so. A similar result arises when we connect amplifiers together. Each amplifier views the preceding one as a source having a particular resistance. Whenever possible we should arrange that the input impedance of an amplifier should equal the output impedance of the preceding one. This ensures that signal power is not wasted.

In the arrangement shown in figure 10.7 the amplifiers are connected by a length of transmission line of *characteristic impedance*, Z_c. Co-axial cables, pairs of wires, microwave waveg-

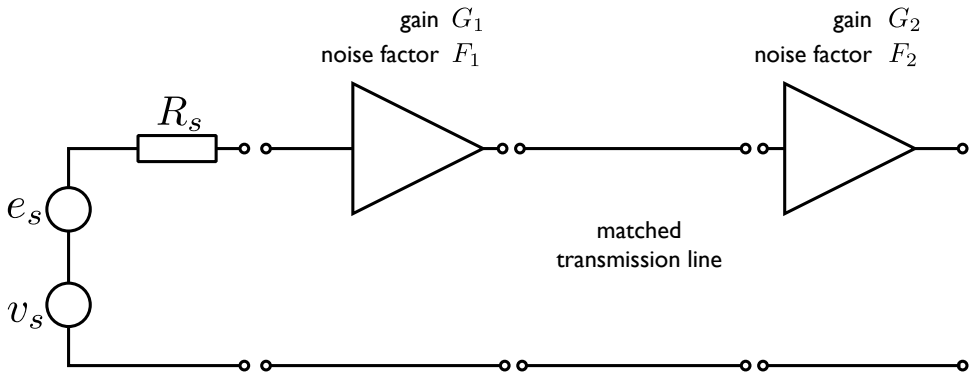


Figure 10.7: A source followed by two *cascaded* amplifiers which are connected together via a matched transmission line.

uides, light fibres, etc, are all examples of transmission lines. Each can be used to carry signals over long distances. To understand the concept of characteristic impedance, imagine a signal source transmitting a signal into an *infinitely long* transmission line. To transmit power along a line it has to send both a non-zero voltage (or electric field) and a non-zero current (or magnetic field) out along the line. This power then moves away from the source, along the infinitely long cable, never to return.

The amount of current the source has to put into the cable to 'drive' a given voltage will depend upon the type of transmission line. However, so far as the source is concerned, the power transmitted into the cable is 'lost' just as if the cable were a resistor. The value of the resistor which would require the same voltage/current ratio is said to be the characteristic impedance of the line. If we end a finite length of line with a load whose resistance equals the line's characteristic impedance the current/ voltage ratio of the signal perfectly matches that required by the load. Hence all the signal power flows into the load.

In figure 10.7 the load at the output end of the line is the input impedance, $Z_{\rm in}$, of the second amplifier. By arranging that $Z_{\rm in} = Z_c$ we ensure that all the signal power passing along the line is coupled into the second amplifier (assuming that the transmission line doesn't lose any of the power on the way). As far as the first amplifier is concerned, it then sees an output load resistance, Z_c, since none of the power it transmits comes back to it. As a result, to efficiently transmit signal power along the transmission line we should, whenever convenient, arrange that $Z_{\rm in} = Z_c = Z_{\rm out}$. This is called **impedance matching** the system.

We'll assume the impedances throughout the system have been matched. The first amplifier has, when matched, a noise factor F_1, and a power gain, G_1. The second has, when matched, a noise factor, F_2, and power gain, G_2.

Thermal noise in the source will produce a noise power spectral density at the input of the first amplifier of

$$N_1 = \frac{e_s^2}{4R_s} = k_B T \tag{10.21}$$

From the definition of noise factor, it follows that this amplifier supplies the following one with a noise power spectral density of

$$N_{o1} = F_1 G_1 k_B T \tag{10.22}$$

If we were to connect the second amplifier's input directly to a matched source resistance instead of linking it to the output from the first amplifier it would supply an output noise power spectral density

$$N_{o2} = F_2 G_2 k_B T \tag{10.23}$$

Since the source resistance would itself be generating a noise power spectral density of $k_B T$, an amount $G_2 k_B T$ of what we see coming out of the amplifier would originate in the *source*, not the amplifier. The noise spectral density which is generated inside the second amplifier is therefore $F_2 G_2 k_B T - G_2 k_B T$.

The total output noise power spectral density for the arrangement in figure 10.7 will therefore be

$$N_T = G_2(F_1 G_1 k_B T) + G_2(F_2 - 1)k_B T \tag{10.24}$$

If we consider the combination of amplifiers as a single 'multi-stage' amplifier with power gain $G_1 G_2$, we can define an overall noise factor, F_T, such that

$$N_T = F_T G_1 G_2 k_B T \tag{10.25}$$

Combining the above expressions, we find that the noise factor of the two cascaded amplifiers will be

$$F_T = F_1 + \frac{F_2 - 1}{G_1} \tag{10.26}$$

This is an extremely important result: if we choose a first amplifier with a moderately high gain, then the noise factor of the pair will be dominated by the first amplifier (unless the second is *much* noisier). This result arises because the signal level presented to the second amplifier is much larger than that presented to the first. Hence the second amplifier would have to generate a considerable amount of noise to significantly degrade the overall SNR. For this reason we usually only need to ensure that the first amplifier in a chain has a low noise factor. (Assuming the transmission line is perfect.) This is good news for system designers: you can spend most of your cash a a really quiet high gain first amp, and worry less about what follows.

Any real transmission line will lose some of the signal power it is given to convey. For example, a co-axial cable will dissipate some power due to the resistance of its metal conductors. The transmission line will change (attenuate) the signal power by a factor, α i.e. an output power, P, supplied by the first amplifier will provide a power, αP, (where $\alpha \leq 1$) to the second.

In many cases α will be close to unity. Under these circumstances the combination of the first amplifier and transmission line have an overall power gain, αG_1, and we need only worry about the first noise factor, F_1. However, if the transmission line is long enough and α is low enough for αG_1 to become comparable with (or less than!) unity, we find that the signal

power reaching the second amplifier is not significantly larger than that reaching the first. Under these circumstances the noise factors of both amplifiers become important.

The above argument for two cascaded amplifiers can be extended to situations where three or more are chained together. For example, for three amplifiers in a chain the overall noise factor (neglecting transmission line losses) would be

$$F_T = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} \tag{10.27}$$

10.6 Chapter 10: take-home messages

You should now know that all signal sources must have a non-zero source (or output) resistance and a non-zero source capacitance. That all the noise mechanisms in a system can be simplified into a an equivalent pair of mythical noise generators at the input to the system. A ?new? noise mechanism, Generation-Recombination has been introduced and it power spectral density has been seen to increase with fluctuation frequency. You should also now know that the total system noise can be simplified into a single generator value and the result may be specified in terms of various figures ? Noise temperature, resistance, figure, or factor. It should also be clear that a single figure of this kind can only be used to compare one amplifier to another when the source resistances are the same. You should also now know that the current and voltage noise levels of an amplifier can be measured by recording the output noise level when the amplifier's input is open- and short-circuited.

Chapter 11

Signal averaging

11.1 Measuring signals in the presence of noise

When measuring a small but steady signal in the presence of random noise we can often improve the accuracy of the result by making a number of measurements and taking their average. This approach has the great advantage that it is easy to do (given enough time) but it cannot overcome all of the practical problems which arise when making real measurements. In particular, there are two sorts of problem which simple averaging copes with rather poorly: $1/f$*-noise*, and the presence of *background effects*.

When considering the merits of various signal processing systems we're primarily interested in comparing the SNRs they can offer. It's this ratio which largely determines how precise a measurement can be (and measurement includes receiving information). A low signal level can always be enlarged if we can afford a suitable amplifier. However, this won't lead to a more accurate result if the measurement was already noise limited because we'll boost the noise level along with the signal, and will end up with extra noise due to the amplifier.

Note that the following arguments assume the power gain, G, of an amplifier (or filter) is simply equal to $|A|^2$ where A is the voltage gain. This is only really true when the amplifier's input resistance is equal to the output load resistance it drives. Similarly, it is assumed that the power, P, at any point is simply equal to V^2, where V is the rms signal voltage. This is only correct for a load resistance of unity (one Ohm). These assumptions make some of the mathematical expressions a bit simpler and don't change any of the conclusions. In practice, when working out the properties of a real system these factors have to be taken into account.

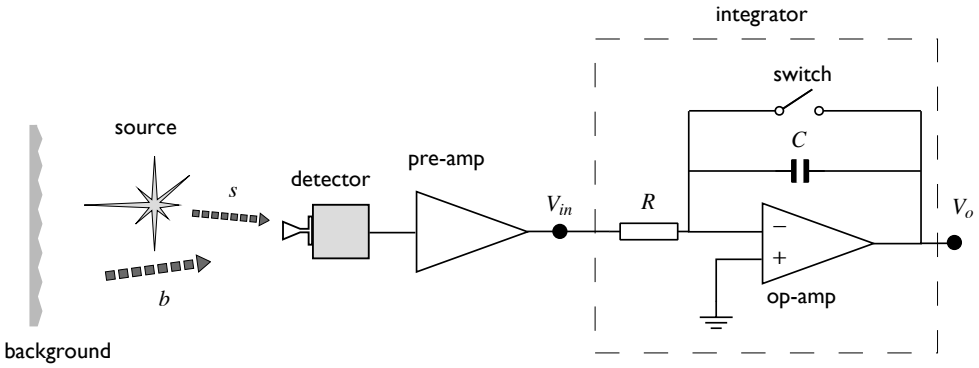


Figure 11.1: A simple analog integrator.

11.2 Limitations of simple averaging

Consider the system shown in figure 11.1. A source, S, produces a response from a detector which is then amplified, and passed through an analog **integrator** to a voltmeter. The integrator is made using an operational amplifier, resistor, and capacitor, although we care more about the fact that it preforms as an integrator, not how the circuit works, although you should be able to follow the argument.

A normal operational amplifier has two signal input terminals, generally called the *inverting* and *non-inverting* inputs (shown by the signs on the diagram). The output voltage the op-amp produces is proportional to the difference between these two input levels. This arrangement allows the op-amp to be used as the heart of a *feedback* arrangement. The voltage gain of a typical op-amp is very large (usually over 100,000) so a reasonable output voltage only arises when the voltages at the inverting (-) and non-inverting (+) inputs are almost identical. For example, if the output voltage is 1 V and the gain is 100,000 then the two inputs will only differ by 10 μV. In the circuit shown in figure 11.1 the non-inverting (+) input is connected directly to 0 Volts. The inverting input (-) is connected via a capacitor to the amplifier's output. The simplest possible state of this arrangement is when both input voltages, and the output voltage, are all at 0 V. We can therefore imagine the system starting off in this state.

When we apply an input voltage, $V_{\rm in}$, to the resistor a current, $I = V_{\rm in}/R$, will begin to flow through it as the other end of the resistor is initially at 0 V. This current starts flowing into the amplifier, stimulating a change in its output voltage. Because the signal is being presented to the inverting input the output voltage this produces will have the opposite sign to the input.

Any change in the output voltage will have to alter the amount of charge in the capacitor, C, i.e. a current will be drawn through the capacitor. As a result we find that most of the current flowing through the resistor passes on through the capacitor as the output voltage changes. Since the op-amp's gain is very large only a relatively tiny amount of the input current needs to actually enter the op-amp to generate the output voltage this process requires.

The small current, i, flowing into the op-amp's input will be the difference between the input and capacitor currents

$$i = \frac{V_{\text{in}}}{R} + C\frac{\mathrm{d}V_o}{\mathrm{d}t} \tag{11.1}$$

As the amplifier gain is large we can expect that $i \ll V_{\text{in}}/R$ so we can reasonably assume that it is virtually zero and re-arrange 11.1 as

$$\frac{\mathrm{d}V_0}{\mathrm{d}t} = -\frac{V_{\text{in}}}{RC} \tag{11.2}$$

Having begun with an output voltage $V_0 = 0$ at time $t = 0$, we can therefore say that the output voltage at some later time, $t = T$, will be

$$V_0(T) = \int_0^T \frac{-V_{\text{in}}}{\tau}\mathrm{dt} \tag{11.3}$$

where $\tau = RC$ has the units of time and is called the *time constant* of the integrator. In effect, the system behaves as if all of the input current, I, is collected into the capacitor and the arrangement functions as an integrator, the output voltage being proportional to the time-integral of the input.

In practice the capacitor can be initially shorted by closing the switch connected across it. This sets the output voltage to zero. When a measurement commences the switch is opened and integration begins. For a steady input signal voltage, v, the output voltage after a time, T, will simply be proportional to vT. Hence the integrator performs the useful function of 'adding up' the signal voltage, v, over a period of time. As a result we need not actually take a series of voltage readings and calculate their average. Instead we can use an integrator, read V_O after a time, T, and define the average input signal voltage, $\langle v \rangle$, during this period to be

$$\langle v \rangle = -\frac{V_O\tau}{T} \tag{11.4}$$

Any real integrator, regardless of how it works, will be powered from voltage rails which supply some specific fixed voltages. As a result, we cannot allow the circuit to go on integrating a signal voltage for an indefinite time as, eventually, V_O will reach the rail voltage and integration must then stop. To overcome this problem we may repeatedly read the output voltage, V_O , after a moderate time interval, t, and reset the integrator output to zero by briefly closing the shorting switch before allowing another integration over another period, t. The resulting set of readings for V_O can then be added together to obtain the voltage which would have been reached if the circuit had been able to integrate successfully over the whole period. Many practical systems combine the use of an analog integrator with this method of repeated reading and resetting.

The effect of noise on an integrated result can be understood in terms of the integrator's effective *power gain* at any frequency, f. At any frequency the noise can be represented by a 'typical' input of the form

$$v_n = A_c\cos(2\pi ft) + A_s\sin(2\pi ft) \tag{11.5}$$

where for real noise, A_c and A_s *will vary randomly from moment to moment*, as the *phase* of the signal is unpredictable (i.e. random). The values of A_c and A_s at any instant are therefore *independent*, i.e. we can't predict one from knowing the other. However, on average, we can expect their magnitudes to be the same. We can therefore say that the time averaged power of this 'noise like' input will be

$$P_{\text{in}} = \frac{\langle A_c \rangle^2}{2} + \frac{\langle A_s \rangle^2}{2} = A^2 \tag{11.6}$$

where expression 11.6 essentially defines A to be the mean amplitude of each individual component. The factors of $1/2$ appear because we are averaging $\sin^2$ quantities over a number of cycles.

Since the actual amplitudes of the sine and cosine components of the noise are statistically independent we can expect their contributions to the noise level at the integrator's output to also be independent. Their combined effect at the output will therefore equal the sum of the powers they individually produce. Integrating the effects of the two contributions over a period, T, we obtain two voltages. These must then be squared separately and then added to obtain the total output noise level

$$\begin{aligned} P_{\text{out}} &= \left[\frac{1}{\tau}\int_0^T A\cos(2\pi f t)\mathrm{d}t\right]^2 + \left[\frac{1}{\tau}\int_0^T A\sin(2\pi f t)\mathrm{d}t\right]^2 \\ &= \frac{A^2\sin^2(\pi f T)}{(\pi f \tau)^2} \end{aligned} \tag{11.7}$$

We may define the integrator's power gain to be the ratio, $G = P_{\text{out}}/P_{\text{in}}$. Comparing expressions 11.6 and 11.7 we can therefore say that

$$G(f) = \frac{\sin^2(\pi f T)}{(\pi f \tau)^2} \tag{11.8}$$

Having discovered the integrator's power gain we can now say that the total output power produced, after integration, by an input white noise power density, N, will be

$$\begin{aligned} P_n &= \int_0^\infty NG(f)\mathrm{d}f \\ &= \int_0^\infty \frac{N\sin^2(\pi f T)}{(\pi f \tau)^2}\mathrm{d}f \\ &= \frac{NT}{2\tau^2} \end{aligned} \tag{11.9}$$

The output signal power produced by integrating a steady input level, v, over a period, T, will be

$$P_s = V_O^2 = \frac{v^2 T^2}{\tau^2} \tag{11.10}$$

Combining this with the result for noise we can therefore say that, when accompanied by an input 'white' noise power spectral density, S, we obtain a final signal to noise ratio of

$$\frac{P_s}{P_n} = \frac{2v^2 T}{N} \tag{11.11}$$

This result is a very important one. It tells us that the signal to noise ratio of a measurement obtained using an integration method can increase linearly with the integration time, T. In practice this means we can often expect to improve the accuracy of a measurement by integrating for longer. The integration process is mathematically equivalent to making a series of measurements and adding them together. We can therefore generalise this result. If we make p measurements, each integrated over a period, t, and add them we obtain a result whose signal to noise ratio will be

$$\frac{P_s}{P_n} = \frac{2v^2 pt}{N} \tag{11.12}$$

What matters here is the *total measurement time*, pt, not the choice of each individual period, t. Note also that the choice of the integrator's time constant value, τ, *does not affect the signal to noise ratio*. In a real measurement situation we should simply choose a τ value which provides a convenient output level after each sample integration period, t. Provided that we avoid voltages which are too large or too small to measure reliably with our instruments, the value of τ has no effect on the signal to noise ratio, and hence the accuracy, of the final result.

In practice we're often interested in obtaining a value proportional to the signal *voltage* (or current) level rather than the power. The integrated output signal voltage increases linearly with pt. However it is the output noise *power* which increases linearly with time, so the typical output noise *voltage* increases as $\sqrt{pt}$. Hence the accuracy of a measured voltage will increase in proportion with the square root of the measurement time.

Figure 11.2 illustrates the effect of integrating an input which consists of a combination of a steady 'd.c.' level plus some white noise. In this case the magnitude of the input dc voltage is a quarter of the rms noise voltage. It can be seen that the integrated result allows the steady level to 'grow' linearly with time whilst the effects of noise only change relatively slowly.

The analog integrator is a convenient way to obtain a result averaged over a period of time. In principle we could use a simpler method. For example, we could regularly note down the reading on a voltmeter, then add up all the readings. The result would be a piecemeal value for the level summed or integrated over the period of the readings. Provided that the readings were taken often enough to form a complete record we'd get the same information as if we'd used an analog integrator. No matter what method we use for 'adding up' measurements over the time period the result would be the same. When measuring a signal in the presence

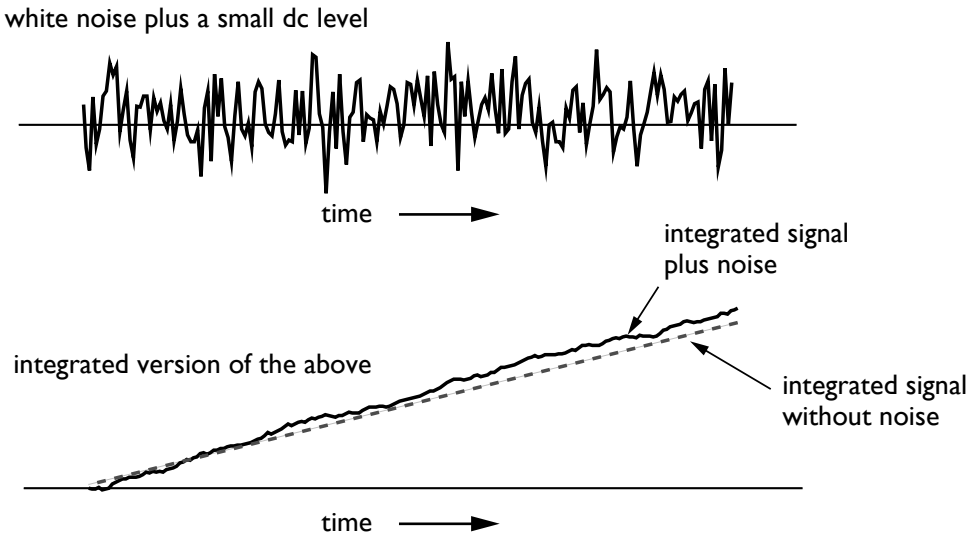


Figure 11.2: A small dc signal with some superimposed noise, and the integrated (signal+noise).

of white noise we get a final power SNR which improves linearly with the measurement time (i.e. the signal/noise voltage ratio increases with the square root of the time take for the measurement).

Although we won't attempt to prove it here, a similar result arises when we look for other signal patterns in the presence of white noise. From an information theory viewpoint a steady dc level is just one example of a specific signal pattern. Any other pattern can be searched for in the presence of noise. Although we would have to process the signal+noise patterns differently we will discover the same basic result. When the noise is white the final accuracy of a measurement improves with the time just as the above example.

The above conclusion applies for a white noise spectrum. A different result arises when $1/f$ noise is present. Consider, for example, a case similar to the above but where the noise has a NPSD

$$N(f) = \frac{e}{f} \tag{11.13}$$

The effective noise power observed at the output of the integrator will be

$$\begin{aligned} &= \int_0^\infty NG(f)\mathrm{d}f \\ &= \int_0^\infty \left(\frac{e}{f}\right) \frac{\sin^2(\pi f T)}{(\pi f \tau)^2} \mathrm{d}f \end{aligned} \tag{11.14}$$

It simplifies matters to make a change of variable to $z \equiv \pi f T$, then to write

$$P_n = \frac{eT^2}{\tau^2} \int_0^\infty \frac{\sin^2(z)}{z^3} \mathrm{d}z$$
$$= \frac{eT^2}{\tau^2} \mathbf{I} \tag{11.15}$$

where $\mathbf{I}$ is integral in z. The bad news is that this integral evaluates to infinity, which would give as an SNR of zero!

The integral 'blows up' in this way because we have assumed that the noise power spectral density $\to \infty$ as$f \to 0$. In reality we wouldn't notice the noise components at frequencies $f \ll 1/2T$ as fluctuations. They would look like a fairly steady level during the particular observation time we've used and become indistinguishable from the signal. This simply confirms that we can't get rid of the effects of $1/f$ noise by integrating or adding together lots of measurements.

In practice the noise present in a measurement system will have both white *and* $1/f$ components. The total noise spectrum can then be represented as a NPSD, N_t, equal to

$$N_t = N + \frac{e}{f} \tag{11.16}$$

Provided the total measurement time, $pt \ll N/e$, we won't observe any significant effect from the $1/f$ noise as the measurement would be dominated by the white noise. Under these circumstances we can expect to obtain an improvement in measurement accuracy by increasing pt. However, once $pt \geq N/e$, the effect of the $1/f$ noise becomes significant and any further increase in pt will produce little or no improvement of the measurement accuracy.

A similar analysis can be carried out for other forms of signal filter and signal summing or integration systems. Although the details will depend upon the choice of system the consequence is much the same. There is a practical limitation, set by the existence of $1/f$ noise, to the improvement in measurement accuracy we can obtain simply by averaging or summing over ever longer periods of time. In order to obtain any further increase in accuracy we must, instead, devise some measurement technique which avoids the effect of $1/f$ noise.

Another serious problem which can arise when making simple, direct measurements is due to the presence of any unwanted *background* signals. Consider as an example the case illustrated in figure 11.1 where we are using a sensor to detect the output from a faint source of light. If we place the source and detector in an ordinary room we find that some of the light striking the detector does not come from the source we wish to measure. Instead it comes from the room lights, or in through the windows of the room. Hence the output we observe from the detector is partly produced by an unwanted background.

One way to deal with this problem is to try and reduce the background level, ideally to nothing. We can, for example, switch off the room lights and place opaque covers over the windows to produce a dark room. Although this means we tend to fall over the furniture it

will reduce the unwanted background level. Unfortunately, some background light will remain. This is because the room will be at a temperature above absolute zero. To avoid freezing its inhabitants the room temperature will probably be somewhere around 280 K to 300 K. Hence all of the surfaces in the room will emit some thermal radiation. Unless we totally enclose the detector in a box cooled to absolute zero there will always be some background radiation falling upon it. (And, of course, if we *totally* enclose it in a box, we can't get the signal onto it!)

Since all sensors and detectors respond to energy or power in one form or another a similar result occurs in every measurement system. We may therefore expect that there will be always be an unwanted background level falling upon any detector. In many cases we can reduce this background until it's low enough to be ignored, however it is impossible to really reduce it to zero.

As an alternative to trying to get rid of the background we can set out to measure it in the absence of the actual signal and then subtract its effect from the final measurement of interest. This approach, called *background subtraction*, is widely used to deal with the problems of measuring very small quantities in the presence of an unwanted background.

11.3 Periodic signals

The analysis above deals with extracting a small, steady (i.e. dc) level from noise, via integration. We can also pull non-constant signals out of noise by increasing the measurement time, provided that they are **periodic**. We do this by **averaging** the signal. This is usually done these days using digital instruments, that sample a waveform. The most common example is the digital oscilloscope.

In fact, *any* periodic signal is going to have some noise on it. We can regard our signal of interest as consisting of the 'true' signal (i.e., what it'd be if there was no noise) plus some random noise.

If we have a series of time records of noise with an rms amplitude of v_n, then, as we expect noise to average to zero (eventually), we'd expect that averaging a number of these records together would give us noise with a lower rms amplitude.

This behaviour can be seen in figure 11.3. Part (a) of the figure shows the results of averaging together different numbers of traces of Gaussian noise. Part (b) shows the rms noise level of the averaged waveforms as a function of the number of averages. The dashed line in the graph goes as $1/\sqrt{p}$, where p is the number of averages. Theoretically, with p averages, the rms noise voltage goes as $1/\sqrt{p}$, which is equivalent to saying that the noise *power* goes as $1/p$ (which is what the discussion above told us).

If we have a periodic signal plus some random noise, and average that, then provided the amplitude of the periodic signal remains constant, averaging that alone will simply give us

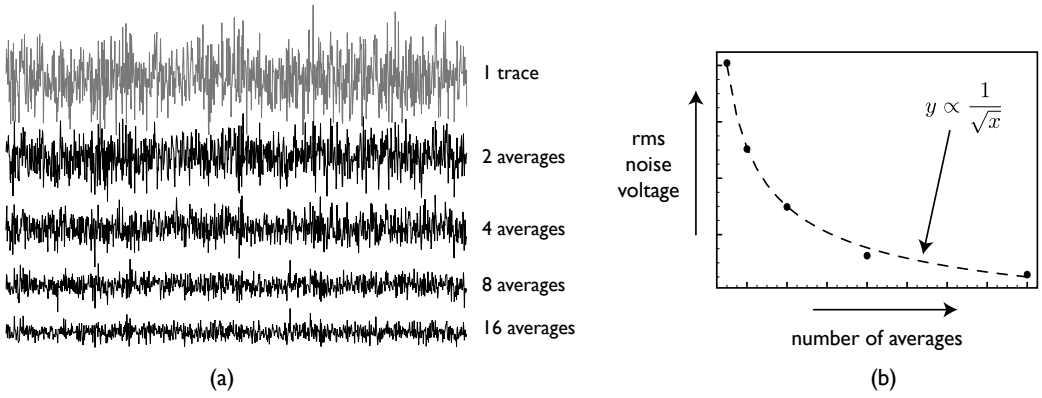


Figure 11.3: Averaged Gaussian noise. (a) The results of averaging different numbers of traces of Gaussian noise. (b) The resulting rms voltage of the averaged trace, as a function of the number of averages.

back the periodic signal. The noise, however, reduces with the number of averages as shown above.

The conclusion arrived at in equation 11.12 was reached when considering a steady dc signal, but the noise was spectrally white. There is nothing particularly special about a steady signal: it is just one possible form. Hence, we'd expect the SNR of a periodic signal with white noise added to improve by averaging in the same way: after all, it's another way of extending the total measurement time.

This is done by acquiring a large number of traces of our periodic signal, all of the same number of samples, at the same sampling rate. We then average by calculating the average value of all of the first samples in each trace, and the result of this gives us the first 'sample' in our averaged trace. We do the same for all the sample positions, to end up with a 'cleaner' signal.

Some example results of this are shown in figure 11.4, where the underlying signal is an exponentially decaying sine-wave.

In the figure, we can see the effect of increasingly large numbers of averages taken of the same periodic signal. The rms noise level is twice the peak signal amplitude. With no averaging at all, it is not really apparent that there is a signal there at all. 16 averages are enough to hazard a guess as to what it might be, and it looks not bad by 1024. Note that the vertical scale is the same for all the traces.

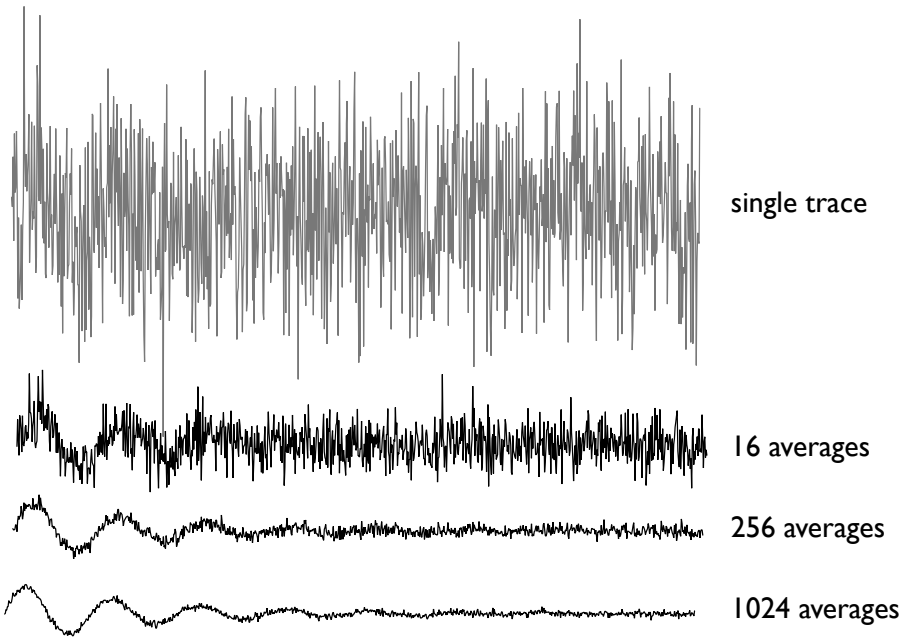


Figure 11.4: One period of a small periodic signal, consisting of an exponentially decaying sine-wave with peak amplitude of 0.5, with Gaussian noise of rms amplitude 1 added to it. The top trace shows the case with no averaging, and the lower three traces show improvement in signal visibility with increasing numbers of averages.

11.4 Chapter 11: take-home messages

You should now know that an **integrator** can be used to improve the SNR ratio of a measurement of a steady signal in the presence of noise. It should be clear that when the random noise is spectrally 'white', the SNR we can obtain is proportional to the total time devoted to the measurement. That the precise choice of the **time constant** of an analogue integrator doesn't normally affect the final result when we're using it to look at a steady level. Remember that, since the power SNR improves in proportion with the time, the accuracy of a voltage measurement increases with the square root of the measurement time. It is important to realise that integration or averaging doesn't always provide an improvement in the measurement's SNR: in particular, it doesn't help with background signals or $1/f$ noise.

We've also seen that we can use averaging on time-varying signals, as long as they are periodic. In order for this technique to be useful, we need to start the data-acquisition for each trace at the same time in the signal's period.

Chapter 12

Coherence, mixers and heterodyning

Coherence is a phenomenon which appears in many situations. In this course we've alluded to *phase* once or twice, but the phase of a signal will now become more important to our discussions. A signal or wave is said to be coherent if its behaviour at various times/places is linked in a deterministic way. Most textbooks and discussions tend to concentrate on simple examples like sinewaves. This is mathematically convenient, but it's useful to remember that many other kinds of field or wave pattern can show coherence.

Some of the most common requirements when dealing with coherent signals and waves are that we should measure the size (amplitude or power) and the frequency of the signal or wave. This requirement arises in many situations. For example, we may need to do this to be able to control the output from a laser or to recover information (e.g. music from a radio station). These measurements can be done in various ways. We begin by looking at how **mixers** can be used to make **heterodyne** measurements of the properties of a signal. Mixers appear in various forms throughout physics and engineering. Here we will consider two examples. The first is a conventional electronic diode of the kind used in radios, TVs, and other electronic systems like radar. The second is an optical photodetector.

12.1 Mixer diodes

Conventional electronic diodes are made by joining together two dissimilar materials. The two most common types are pn-junctions and Schottky diodes. The pn-junction is formed by joining two pieces of semiconductor, one piece doped to be n-type, the other p-type. The Schottky diode is made by joining together a suitable piece of metal and a piece of semiconductor. The general properties of these diodes are illustrated in figure 12.1. The first property to note is that the diode only passes electric current when the applied voltage between its terminals/wires has the polarity referred to as *forward biassed*. When reverse biassed, almost no current can get through the diode. The second property is that the size of the diode current when forward biassed is not simply proportional to the applied voltage

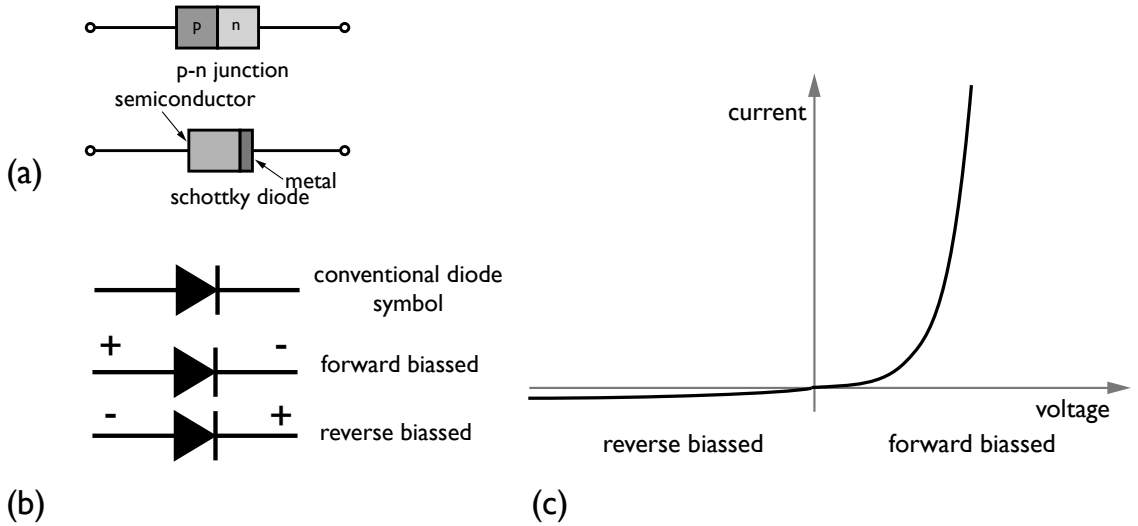


Figure 12.1: (a) Simple models of p-n and schottky diodes. (b) The usual symbol for a diode, and the applied voltage directions for forward and reverse bias. (c) The characteristic current/voltage (I/V) curve of a typical diode.

between the wires. The diode is therefore said to be a **non-linear** device.

Many textbooks describe the I-V behaviour of a diode using the ideal diode equation:

$$i_d = I_s \left(\exp \left[\frac{\eta v_d}{k_B T} \right] - 1 \right) \tag{12.1}$$

where I_s and η are constants whose values depend on the details of the diode. In practice, the shape of a diode's I-V curve depends on the details of how it was made, hence many real diodes only approximate to exponential behaviour. In general, the I-V behaviour of a real diode can be represented in terms of a suitable polynomial series,

$$i_d = \sum_{n=0}^{\infty} a_n v_d^n \tag{12.2}$$

where the a_n are chosen to suit the diode being considered. For our purposes it's convenient to simplify this and assume that we have a *square law* diode whose behaviour obeys

$$i_d = a v_d^2 \qquad v_d \geq 0 \tag{12.3}$$

$$i_d = 0 \qquad v_d < 0 \tag{12.4}$$

This square law assumption makes the following explanation much easier to follow. It also makes the diode work very well! As a result, many real diodes are deliberately manufactured to behave in a square law manner, so the assumption turns out to be a good description of reality. Consider the situation illustrated in figure 12.2. Here we are applying a combination of three input voltages to a square law diode; an input signal, v_s, a steady (dc) level, V_{DC}, and a sinewave, v_{LO}. The total voltage across the diode at any instant will therefore be

$$v_d = V_{DC} + v_s + v_{LO} \tag{12.5}$$

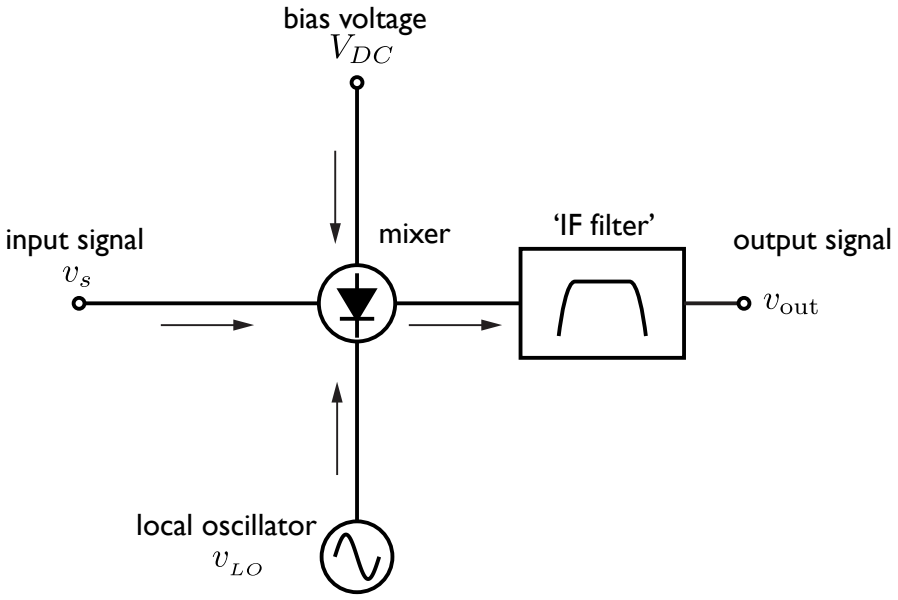


Figure 12.2: Diode heterodyne mixer arangement.

where

$$v_{LO} = B\cos(2\pi f_L t) \tag{12.6}$$

We'll begin by assuming that the input signal is also sinusiodal:

$$v_s = A\sin(2\pi f_S t + \phi) \tag{12.7}$$

We can also arrange things such that $V_{DC} \geq |A| + |B|$, which will mean that the diode is always forward biassed. For a square law device this means that the diode current at any instant will be

$$i(t) = a\left[V_{DC} + A\sin(2\pi f_S T + \phi) + B\cos(2\pi f_L t)\right]^2 \tag{12.8}$$

This can be rearranged into the form

$$\begin{aligned} i(t) = {} & \frac{1}{2}aB^2\cos[2\pi(2f_L)t] - \frac{1}{2}aA^2\cos(2\pi[2f_S]t + 2\phi) \\ & + aAB\sin[2\pi(f_S + f_L)t + \phi] + aAB\sin[2\pi(f_S - f_L)t + \phi] \\ & + 2aV_{DC}[A\sin(2\pi f_S t + \phi) + B\cos(2\pi f_L t)] + \frac{aA^2}{2} + \frac{aB^2}{2} + aV_{DC}^2 \end{aligned} \tag{12.9}$$

Looking at equation 12.8 we can see that, having applied a pair of input voltage fluctuations at the frequencies f_S and f_L, we set up current fluctuations at the frequencies $2f_S$, $2f_L$, $(f_S + f_L)$, and $(f_S - f_L)$, as well as at the frequencies f_S and f_L. There is also a steady current component (i.e. 0 Hz). This pattern of current fluctuations can be treated as the output from the diode when it is driven by the input voltages we've chosen. We can now

guide a portion of this output current out via a suitable filter which only passes frequencies similar to the **difference frequency**, $(f_S - f_L)$. As a result we can get an output voltage

$$v_{\text{out}} \propto aAB \sin[2\pi(f_S - f_L)t + \phi] \tag{12.10}$$

We'll be considering the uses of this effect later on. For now, it is enough to note three significant points:

- The amplitude of v_{out} is proportional to the amplitude, A, of the input signal.
- The frequency, $(f_S - f_L)$, of the output depends upon the signal frequency, f_S.
- The phase, ϕ, of the output is the same as that of the signal.

As a result, we can expect a measurement of the amplitude, frequency, and phase of the output to provide information about the amplitude, frequency, and phase of the input signal. In effect the heterodyne system lets us shift the frequency of a signal whilst preserving any information we might need to know about its amplitude, etc.

Consider an example where we have an input signal at a frequency of, say $f_S = 50$ GHz $(50 \times 10^9$ Hz. This comes from a 'distant' signal source, i.e. one which isn't a part of our measurement system. The signal is applied to the diode along with an output at, say, $f_L = 49.999$ GHz, from a local oscillator (LO), i.e. one which is part of our measurement system. The diode's non-linearity is said to **mix** these two signals to produce an output at the frequency, $(f_S - f_L) = 1$ MHz. This frequency is much lower than that of the original signal, hence it is much easier to amplify, filter, and measure its properties. The diode, local oscillator, and filter act together as a **heterodyne receiver**. Heterodyning is the process of 'beating together' or mixing two different frequencies to obtain an output at some other, related frequency. When used in this way the diode is called a *mixer diode*.

The heterodyne receiver is an example of a system which uses the technique called *frequency conversion*. An input at the frequency f_S is converted to an output at $(f_S - f_L)$. For obvious reasons this is also called *down conversion*. In this case the output filter selects an output frequency which is lower than either the signal or local oscillator frequency. If we'd wished we could have used a filter which only passed frequencies around the *sum* frequency, $(f_S + f_L)$. The heterodyne system would then perform *up conversion* on the input signal. In either case it is normal to refer to the filtered output from the mixer as the **intermediate frequency** (or IF) output. This is because the output from the mixer and filter is often processed by other circuits before being reconverted into the output finally required.

12.2 Photoconductive detectors as mixers

Photodetectors are different to diodes, but do have some similar properties. Generally speaking, we can divide photodetectors into two general types: i) photoconductors and other 'bulk

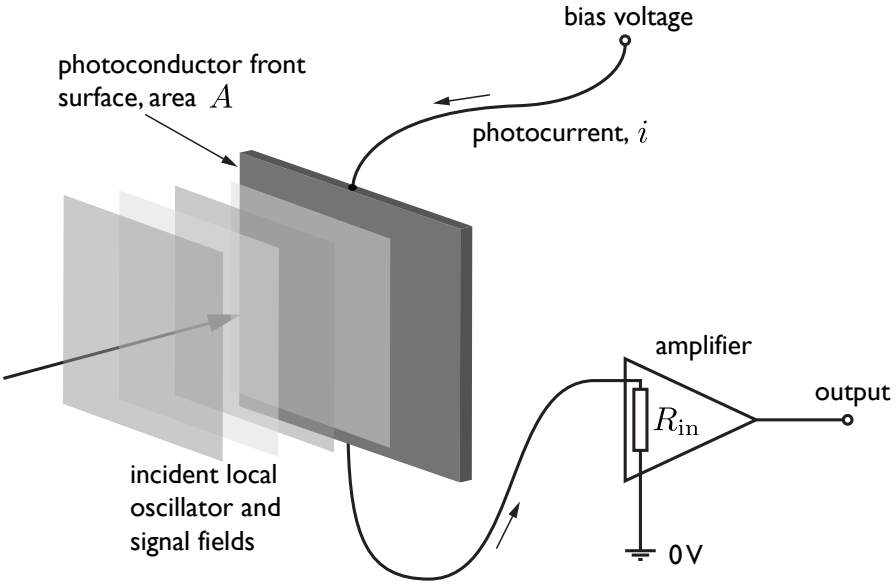


Figure 12.3: A photoconductive mixer.

effect' detectors; ii) photodiodes. Here we'll concentrate on photoconductors, but we would in fact get a similar result if we examined the behaviour of photodiodes. Figure 12.3 illustrates a photoconductive detector being illuminated with some electromagnetic radiation. The useful property of a photoconductive material is that, when we apply a suitable bias voltage and illuminate it with suitable radiation it conducts a current whose magnitude is proportional to the light *power* falling upon it.

The current, i, passing through the photoconductive detector when illuminated by light with power P can be said to be

$$i = \frac{q\eta P}{hf} \tag{12.11}$$

where f is the frequency of the light, h is Planck's constant and q is the charge on an electron. η is the *quantum efficiency* of the detector, a measure of the chance that a photon hitting the detector will create a free electron in the detector, allowing it to contribute to the current. Light of power P will consist of P/hf photons per second, which will produce $\eta P/hf$ free electrons per second, hence the form of equation 12.11.

In the situation illustrated in figure 12.3 the photoconductor has an illuminated surface area, A, and we are shining two electromagnetic waves onto it. These are a signal electric field, $E_S \cos(2\pi f_S t)$, and an LO produced field, $E_L \cos(2\pi f_L t)$. The total field at the detector surface at any instant, t, will therefore be the sum of the two. In free space the ratio of the sizes of magnetic and electric fields is the *free space impedance*, Z_0. Using this, we can say

that the power falling on the photoconductor at any instant will be

$$\begin{aligned} P(t) &= \frac{E^2(t)A}{Z_0} \\ &= \frac{1}{Z_0}A(E_S\cos[2\pi f_S t] + E_L\cos[2\pi f_L t])^2 \end{aligned} \tag{12.12}$$

In most cases of practical interest we find that the signal and LO frequencies are similar, i.e. $|f_S - f_L| \ll (f_S + f_L)$. When this is true we can say that the detector current at any moment will approximately be

$$i(t) = \frac{q\eta A}{Z_0 h f}(E_S\cos[2\pi f_S t] + E_L\cos[2\pi f_L t])^2 \tag{12.13}$$

where

$$f \equiv \frac{f_S + f_L}{2} \tag{12.14}$$

is the approximate value of the signal and LO frequencies. Looking at equation 12.13 we can see that the output current depends upon the *square* of the applied electric field, which tells us that the photoconductor is a square law device. Putting equation 12.12 into 12.13, using trig identities, and rearranging, we get

$$\begin{aligned} i(t) = &\frac{q\eta A}{2Z_0 h f}\left[E_L^2\left(1+\cos[2\pi(2f_L)t]\right) + E_S^2\left(1+\cos[2\pi(2f_S)t]\right)\right] \\ &+ \frac{q\eta A E_S E_L}{Z_0 h f}\left[\cos(2\pi[f_S+f_L]t) + \cos(2\pi[f_S-f_L]t)\right] \end{aligned} \tag{12.15}$$

If the signal and LO frequencies are very high (for example, visible light or high microwave frequencies) then the current fluctuations at $2f_S$, $2f_L$ and $(f_S + f_L)$ are so high that won't be able to propagate along the wires to the amplifier (and if they did the amplifier would probably ignore them!), so as a result, the amplifier will see a current

$$i(t) = i_S + i_L + 2\sqrt{i_S i_L}\cos\left[2\pi(f_S - f_L)t\right] \tag{12.16}$$

where

$$i_S \equiv \frac{q\eta A E_S^2}{2Z_0 h f} \tag{12.17}$$

is the steady current level the photoconductive detector would produce if it were illuminated by the signal alone, and

$$i_L \equiv \frac{q\eta A E_L^2}{2Z_0 h f} \tag{12.18}$$

is the current that would be produced by illuminating with the local oscillator alone. The final contribution,

$$2\sqrt{i_S i_L}\cos\left[2\pi(f_S - f_L)t\right] \tag{12.19}$$

is the current fluctuation produced at the LO and the signal move in and out of phase with each other.

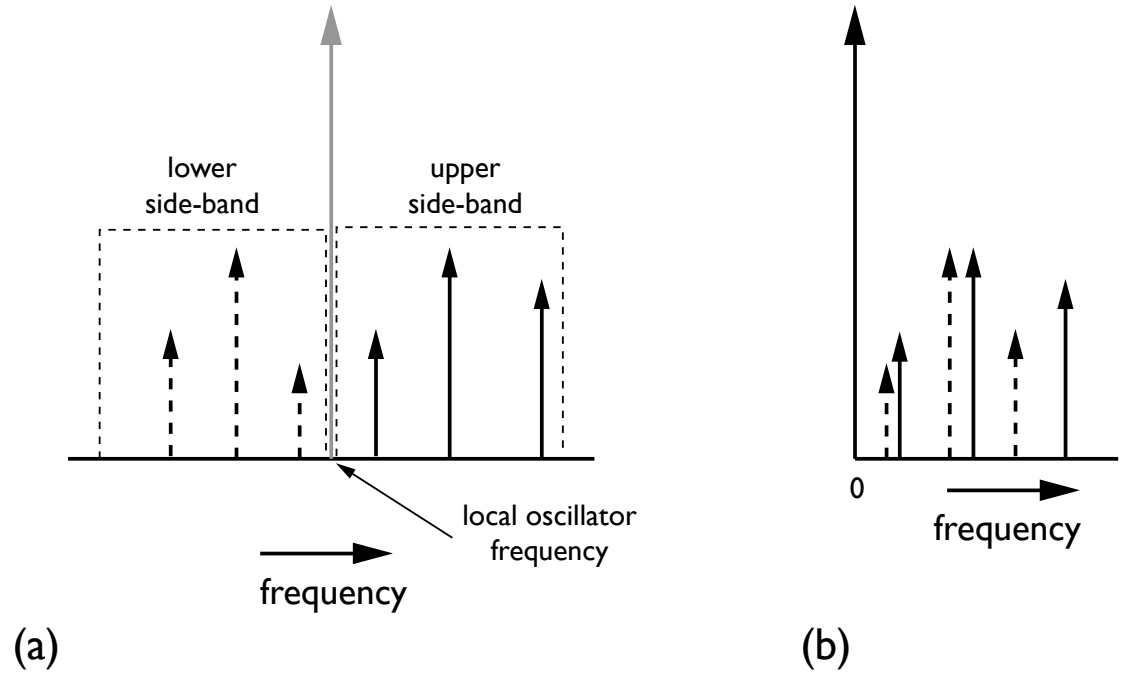


Figure 12.4: Down-conversion of a wide-band input signal by a heterodyne mixer. (a) The spectrum of the input signal. (b) The IF output spectrum. The upper side-band components are indicated by solid arrows and the lower sideband components by dashed arrows.

This oscillatory output current has similarities to the difference frequency output from a diode mixer. Its amplitude, frequency and phase are related to those of the signal in much the same way. As a result we can make a heterodyne system using a photoconductive detector as a mixer in place of a conventional diode. The importance of this result is that photodetectors can be made to operate at signal frequencies well above those possible with conventional diodes. Hence we can build and use heterodyne systems at visible frequencies and above. Using either form of mixer (or any other suitable nonlinear device) we can convert a high frequency input into a lower frequency output. This is convenient because we can often amplify and measure lower frequencies more easily. As a result, heterodyne systems are very useful when we want to make measurements upon high frequency signals or waves.

12.3 Use of heterodyne systems with complex signals

The explanations of heterodyning above assume that the input signal consists of just a single frequency component. In general, we can represent *any* input as some appropriate combination or *spectrum* of frequency components. The details of this spectrum can provide us with information about the signal source. It can also be a pattern deliberately used to convey messages, e.g. to send radio or TV signals. The action of the heterodyne system when presented with such an input is illustrated in figure 12.4.

We can represent the input field as the sum of a series of components

$$E_S(t) = \sum_{n=1}^{m} E_n \cos(2\pi f_n t + \phi_n) \tag{12.20}$$

The system illustrated in figure 12.3 would produce a down-converted output component fo each input component. The resulting signal voltage across the IF amplifier's input resistance, R_{in} would therefore be

$$v_{IF}(t) = \sum_{n=1}^{m} A_c E_n \cos\left[2\pi(f_n - f_L)t + \phi_n)\right] \tag{12.21}$$

where

$$A_c = \frac{q\eta A E_L R_{\text{in}}}{Z_0 h f} \tag{12.22}$$

There are two important feature of this result. Consider initially a signal which only contains frequencies in the range $f_i > f_L$. This region is called the heterodyne system's *upper sideband*. The output IF spectrum produced by such an input is shifted down in frequency by an amount f_L and its amplitude scaled by an amount proportional the the LO field amplitude, E_L. As a result, we can determine the details of the input spectrum by examining that of the IF output.

For components in the *lower sideband* frequency range, i.e. where $f_i < f_L$, we find that $(f_n - f_L) < 0$. Due to the symmetry of the cosine function (i.e. $\cos(x) = \cos(-x)$), an input which consists only of lower sideband components produces a down converted output spectrum which is a sort of mirror image of the input signal's spectrum. Like the upper sideband result, the frequencies and amplitudes of the IF components depend on the signal, hence we can use the IF spectrum to determine the details of the input signal.

A problem can arise when the input signal simultaneously presents the heterodyne mixer with frequency components both above and below the LO frequency. When this happens the output IF spectrum is a combination of the down converted upper sideband and the down converted mirror imaged lower sideband. This can lead to confusion, as we can't tell if an IF component corresponds to a signal frequency above or below f_L.

This problem is known as **sideband ambiguity**. This effect is inherent in the heterodyne mixing process. The most common solution to this problem is to employ a *single sideband filter*, placed between the signal source and the mixer. The filter is built to only allow through frequency components from one, chosen, sideband. When such a filter is in place we can know that a given IF output component can only have come from a signal component at one of the two possible signal frequencies because the other would not have been able to reach the mixer. The addition of such a filter turns the heterodyne system in a **superheterodyne** system.

The single sideband (SSB) filter is conventionally called an *RF* filter, where RF stands for 'radio frequency' (even in systems that work at frequencies nowhere near radio frequencies!)

Using a superheterodyne receiver we can collect information about a laser, or a distant star, or we can process the modulated signals produced by a distant TV station. In each case the information we require is obtained by measuring the details of the signal spectrum as conveyed by the IF output.

12.4 Conversion gain

Our photoconductive mixer example above told us that this heterodyne system could be expected to produce an output IF current (after the dc component is filtered out) of

$$i_{IF}(t) = \frac{q\eta A E_S E_L}{Z_0 h f} \cos\left[2\pi(f_S - f_L)t\right] \tag{12.23}$$

when provided with a single frequency input signal of amplitude E_S and pumped or driven with an LO input of amplitude E_L. The mean IF *power* the mixer supplies to the following amplifier (if it has resistance R_{in}) is therefore

$$P_{IF} = \left\langle i_{IF}^2(t) \right\rangle R_{\text{in}} \tag{12.24}$$

Since the average value of the $\cos^2$ function is $1/2$, then we can say

$$P_{IF} = \frac{R}{2}\left(\frac{q\eta A E_S E_L}{Z_0 h f}\right)^2 \tag{12.25}$$

The signal and LO powers falling on the mixer will be

$$P_S = \frac{E_S^2 A}{2Z_0} \tag{12.26}$$

and

$$P_L = \frac{E_L^2 A}{2Z_0} \tag{12.27}$$

By combining the above three expressions, we can write

$$P_{IF} = 2P_S P_L R_{\text{in}}\left(\frac{q\eta}{hf}\right)^2 \tag{12.28}$$

When using a specific LO power level, we can define a **conversion gain**:

$$G \equiv 2P_L R_{\text{in}}\left(\frac{q\eta}{hf}\right)^2 \tag{12.29}$$

which we can use to rewrite equation 12.28 as

$$P_{IF} = GP_S \tag{12.30}$$

Equation 12.29 shows that the IF power level is proportional to the input signal power level. The value of the conversion gain essentially determines how much IF power we get from a given input signal level.

The above argument indicates that the conversion gain value is proportional to the LO power level and the IF amplifier's input resistance. This implies that we could obtain as high an output IF level as we might like by increasing the LO power and/or amplifier resistance. In practice there are two reasons why this isn't always true:

Firstly, the above analysis assumes that each of the incoming stream of signal and LO photons have a chance, η, of creating a free electron in the detector. However, the number of electrons inside the mixer is finite. As a result, if we keep increasing the rate at which photons arrive we'll eventually begin to 'run out' of available electrons. As this happens, the chance of an extra photon producing a new freed electron will reduce. Hence the detector's quantum efficiency, η, tends to fall once the signal+LO level goes above some finite value. Secondly, we have to maintain an electric field across the detector in order to sweep out the liberated electrons and get a measurable current. This field is created by the applied bias voltage. Figure 12.3 shows that this bias is applied to the detector and the amplifier's input resistance in series. As a result, the actual voltage across the detector will be $V_{\mathrm{bias}} - iR$ where i is the current through the detector (including both the IF and the dc contributions). The above analysis actually assumes that $|iR| \ll V_{\mathrm{bias}}$. Making R or i too large has the effect of reducing the bias field across the detector, hence reducing the current level produced by a given signal+LO power.

For the reasons above (and some others we haven't mentioned) the expression 12.5 is only reliable when the the input signal+LO power level, amplifier resistance, and output current are 'small', i.e. low enough not to disturb the assumptions mentioned above. At higher levels the conversion gain value tends to *saturate*: it stops increasing with power or amplifier input resistance. Any further increase in the input power or amplifier resistance then causes the gain to *fall*. As a result, there is always an optimum LO power level and amplifier input resistance which gives the highest possible conversion gain. The precise details of these values will depend upon the actual detector.

The conversion gain, G, is one of the standard values which tend to be measured and quoted to indicate the performance of a heterodyne or superheterodyne receiver system. G is called a 'gain' because it is a ratio of an output power to an input power, although in most cases it will be less than one (i.e. less than 0dB). It is called a conversion gain because it indicates the relative level of an output which has been converted to a frequency which differs from that of the input.

12.5 Noise temperature

We've seen earlier in the course that all systems at finite temperature are subject to *thermal noise* and also that whenever something is quantised (current or light levels, for example) we are subject to *shot noise*. Receivers are, of course, no exception to this, and thus will have noise associated with them. Just like the amplifiers we looked at not long ago, a receiver has associated with it a **noise temperature** which may be used to describe its noise properties.

Consider again the photoconductive mixer shown in figure 12.3. Ignoring all the noise mechanisms *except* the shot noise we can say that it provides an output SNR of

$$\mathrm{SNR} = \frac{\left\langle i_{IF}^2 \right\rangle}{\left\langle i_n^2 \right\rangle} \tag{12.31}$$

where $\left\langle i_n^2 \right\rangle$ is the mean squared value of the noise current. If we refer back to section **??**, we can remind ourselves that we decided that the mean squared fluctuations of a current should be proportional to the *square root* of the current. By theoretical analysis or measurement, we can show that

$$\left\langle i_n^2 \right\rangle = 2qIB \tag{12.32}$$

where B is the bandwidth over which we observe the fluctuations. As we pointed out before, the key thing to note here is that the size of the noise current depends on the actual current.

If we consider again the photoconductive mixer in figure 12.3, then if we ignore all the noise mechanism except for shot noise, we can say that this provides an output SNR of

$$\mathrm{SNR} = \frac{\left\langle i_{IF}^2 \right\rangle}{\left\langle i_n^2 \right\rangle} \tag{12.33}$$

as both the signal current (the signal now being the IF from the mixer) and the noise current flow through the same amplifier input resistance R_{in}. From section 12.2 we can say that

$$\begin{aligned} \left\langle i_{IF}^2 \right\rangle &= 4 i_S i_L \left\langle \cos^2 \left[2\pi (f_S - f_L) t \right] \right\rangle \\ &= 2 i_S i_L \end{aligned} \tag{12.34}$$

In practice we can usually expect the LO power to be much higher than the signal power. This means that $i_L \gg i_S$, so for practical purposes we can assume that the current through the detector is essentially $i \approx i_L$, so we can rewrite equation 12.33 as

$$\begin{aligned} \mathrm{SNR} &= \frac{2 i_S i_L}{2 q i_L B} \\ &= \frac{i_S}{qB} \end{aligned} \tag{12.35}$$

and with reference to equation 12.11, we can say

$$\mathrm{SNR} = \frac{\eta P_S}{hfB} \tag{12.36}$$

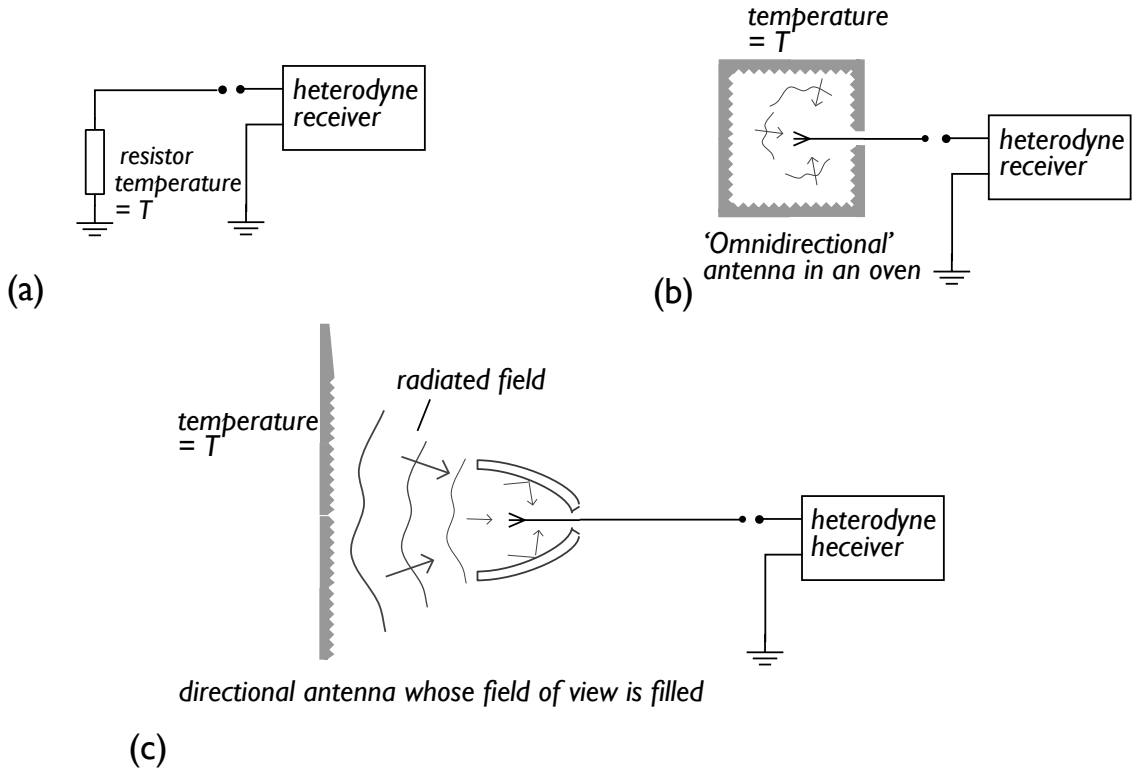


Figure 12.5: Three different thermal sources coupled to a heterodyne receiver. (a) The thermal source is a resistor at temperature T. (b) The thermal source is an omnidirectional antenna (one which receives radiation equally from all directions) in an oven of temperature T. (c) The thermal source is a directional antenna whose field of view is filled by an object at temperature T.

As we'd expect, the signal to noise ratio depends upon the signal power. This result shows that it also depends upon the signal frequency. This is because we're considering shot noise. Each of the electrons in the detector current corresponds to a detected photon. The higher the signal frequency, the higher the individual photon energy, and the smaller the number of photons per second which correspond to a given input power level. As a result, higher signal frequencies lead to higher shot noise.

Consider now situations illustrated in figure 12.5. In each case the signal entering the heterodyne system comes from a thermal source at temperature T. The receiver doesn't know what it is 'looking at', it only knows that a given power spectrum is being supplied as an input signal. We can assume that the receiver has an IF filter of bandwidth, B, and that it is a *single sideband* system, i.e. an rf filter only allows it to respond to signals on one side of the LO frequency. It therefore only sees a spectral bandwidth, B, from the source.

The spectrum of a thermal black body source peaks at a wavelength, λ_m, such that $\lambda_m T = 5.1$ mmK. For signals in the *Rayleigh-Jeans* part of the spectrum, which means for frequencies $f \ll c/\lambda_m$, the signal power reaching the mixer from a source of temperature, T, will be

$$P_S = k_B T B \tag{12.37}$$

and if we combine this with equation 12.36 we get

$$\mathrm{SNR} = \frac{\eta k_B T}{hf} \tag{12.38}$$

and we can use the concept of a noise temperature, T_n to define the performance of our heterodyne system by choosing T_N such that

$$\mathrm{SNR} = \frac{T}{T_n} \tag{12.39}$$

and as before, T_n is the temperature of a thermal source which would provide a power equal to the noise power. If shot noise was the *only* noise generating process, then our heterodyne receiver's noise temperature would be

$$T_n = \frac{hf}{k_B \eta} \qquad (\text{best case}) \tag{12.40}$$

This is absolutely the lowest noise temperature we could hope to have for our receiver as we've ignored every other possible noise generating mechanism! Equation 12.40 tells us that for any single side-band superheterodyne receiver, the best possible noise performance is a function of the frequency we're operating at and the quantum efficiency of the device we're using as a mixer.

In reality, the noise temperature of most systems will be much higher than the minimimum value derived above, and is often much higher than the physical temperature that the system is at.

12.5.1 Measurement of receiver noise temperature

Specifying the behaviour of a heterodyne system in terms of a noise temperature is convenient because noise temperature values are easy to measure. The fact that the value depends upon many factors (quantum efficiency, shot noise level, thermal noise level, amplifier noise, etc.) doesn't matter when we're just comparing systems looking for the best (i.e. lowest noise) one. We can just measure their noise temperatures and choose the one we prefer. To see how we can measure a system's noise temperature consider the figure 12.6. This shows a complete heterodyne system used for making measurements of the power radiated by a thermal source. The IF output is amplified and passed to a second 'square law' detector, the output of which is filtered to provide an output, v_{out}, which is proportional to the IF power averaged over some moderate time interval. Hence the system provides an output voltage proportional to the power falling on the mixer.

The noise generated in the receiver is treated as if it came from a thermal source of temperature T_n which adds its noise power to the thermal signal power coming from the actual source. Hence when the input comes from a thermal source of temperature, T_1, the smoothed output voltage from the receiver will be

$$V_1 = k_B B(T_1 + T_n) G_c G_a \tag{12.41}$$

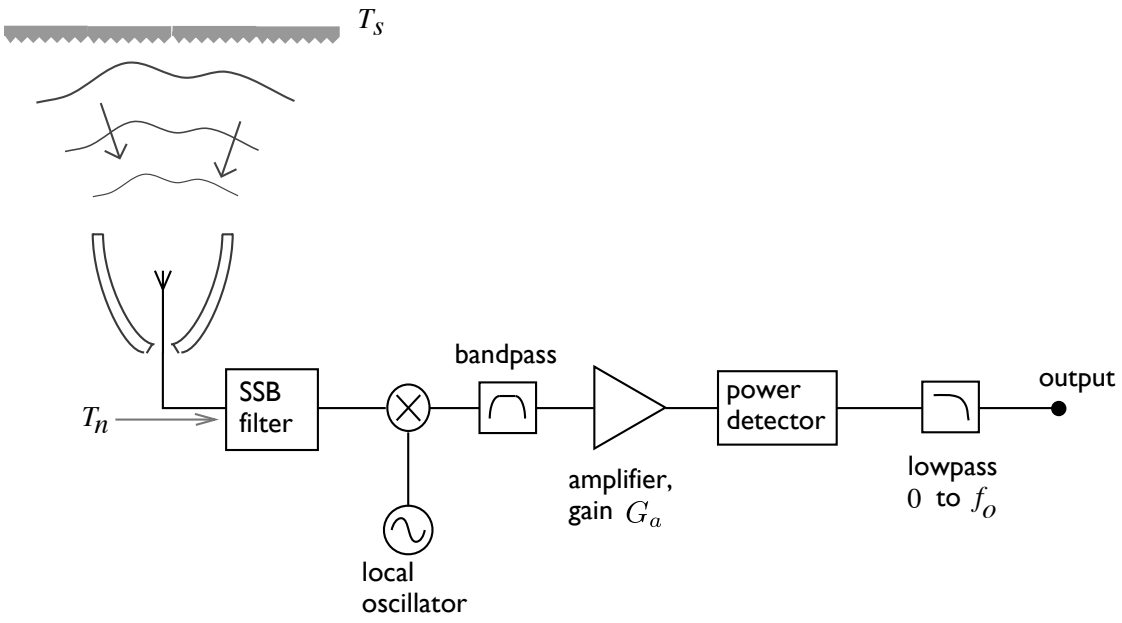


Figure 12.6: A heterodyne system used for temperature measurement (by measuring input thermal noise power).

where G_c is the mixer's conversion gain and G_a is the gain of the IF amplifier.

Similarly, when presented with a thermal source at temperature T_2, the system will give us an output of

$$V_2 = k_B B(T_2 + T_n)G_c G_a \tag{12.42}$$

The expressions for the two output voltages can be combined and manipulated to yield

$$T_n = \frac{T_1 V_2 - T_2 V_1}{V_1 - V_2} \tag{12.43}$$

So all we need to do to measure T_n for our receiver is point it at two thermal sources of (known!) different temperatures, and measure the output voltage we get for each one.

12.6 Minimum detectable temperature

When observing thermal (or other 'wideband') signal sources we often need to know how *small* a temperature change (or signal power level) a receiver can see over the effects of its internal noise and this is not simply equal to its noise temperature. The output low pass **post detection** filter will have some bandwidth, B_D, so will smooth out fluctuations over some *time constant*, $\tau_D = 1/2\pi B_D$. For thermal noise, a mean power level of P will fluctuate by an amount $\sqrt{P}$ from moment to moment, and it is these fluctuations in the power level which limit our ability to see low power levels or detect small changes.

As a thermal noise source has a 'white' spectrum, doubling the IF bandwidth would therefore double the signal power level. The observed fluctuations would increase by a factor of $\sqrt{2}$, so we'd expect the SNR of the system to improve by $\sqrt{2}$. It's thus reasonable to expect that the minimum detectable change in temperature, ΔT will be such that

$$\Delta T \propto \frac{T_n}{\sqrt{B_{\mathrm{IF}}}} \tag{12.44}$$

(Note that B_{IF} is the IF detection bandwidth, *not* the post-detection bandwidth!) The spectrum of fluctuations will be filtered by the post detection filter. Provided this has a bandwidth, $B_D < B_{IF}$ we can expect it to affect the size of the observed random fluctuations. The narrower its bandwidth (or larger its time constant value), the more of the fluctuation spectrum it will average away, so we can expect that

$$\Delta T \propto \frac{T_n\sqrt{B_D}}{\sqrt{B_{IF}}} \tag{12.45}$$

The key point here is that we can use a receiver whose noise temperature is fixed to detect a smaller input level or change by *increasing* the IF bandwidth and/or by *decreasing* the post detection bandwidth.

We can also express the idea above by

$$\Delta T \propto \frac{T_n}{\sqrt{B_{IF}\tau_D}} \tag{12.46}$$

which is a nice way of reminding ourselves that what really matters is the measurement or *integration* **time**. In effect, a time constant provides us with a result 'averaged' over a period equal to one time constant. More generally, we can make measurements over some longer period, take their average, and then substitute the total measurement time in place of τ in the above expressions. This is an example of a general result: the SNR of a measurement often increases if we record values over a longer period and average them together. This should not come as a massive surprise, as we looked at the benefits of signal averaging in chapter 11.

The value of this result can be illustrated by taking an example where we have an IF filter with a bandwidth of 100 MHz and a post detection time constant of 1 second. Using these filters, a receiver whose noise temperature is 1000 K would be able to detect a 0.1 K source with an SNR of unity in one second. Given 100 seconds this would improve to being able to detect an 0.01 K source (or a change in temperature of 0.01 K). This behaviour explains how radio astronomers are able to detect distant galaxies with very low effective brightness temperatures using receivers with noise temperatures of thousands of K. They 'simply' sit pointing the telescope and receiver at the same source for hours on end!

12.7 Chapter 12: take-home messages

It should now be clear that we can use either a conventional diode or a photodetector as part of a **heterodyne** arrangement to measure the amplitude, frequency, and phase of an input signal. In general, *any* nonlinear device can be used as the basis of a heterodyne system when combined with a suitable **local oscillator**. You should also now understand that the main advantage of a heterodyne system is that is allows us to ?shift down? a signal to a lower, more convenient, frequency range without losing information about the amplitude, frequency, and phase of the original signal. This allows us to take in very high frequency signals and transfer their spectral pattern to lower frequencies which are easier (often cheaper!) to amplify and examine.

We've also seen how the performance of a superheterodyne receiver system depends upon its basic properties. The minimum possible **noise temperature** tends to increase with the input signal frequency. The ability of a system to detect low level signals (or small changes in level) depend upon the noise temperature, the IF bandwidth, and the post detection time constant (or measurement time).

Chapter 13

Spectrum analysis

We've spent a while looking at waveforms in the time domain, and occasionally making reference to the frequency (or spectral) content of signals. As we saw earlier when discussing information theory, we can extract the frequency content of a signal from its time domain form by taking a Fourier transform of it. There are, however, other methods we can use to measure the frequency content (or spectrum) of a signal and in this short chapter we'll have a look at them.

13.1 Simple filter-based system

Figure 13.1 shows a very simple way of making a *spectrum analyser* which (unsurprisingly!) allows us to analyse the spectrum of a signal. A signal of interest is split an applied to the inputs of an array of *band-pass* filters. Each of the filters, B_n, passes a narrow frequency range, $f_{\min} + (n-1)B$ to $f_{\min} + nB$, where B is the frequency resolution that we require and $f_{\min}$ is the lowest frequency that we are interested in. By using N such filters in our filter bank, we can cover a total spectra range of NB.

The bank of filters feeds a bank of power detectors: each of these is simply a device (possibly based on a diode) that produces an output voltage that is proportional to the power that it 'sees'.

If our signal of interest was the output of a mixer system, then this system would give us information about the IF spectrum. In that case, we could regard the whole system as being equivalent to a 'parallel' set of separate heterodyne receivers, each with an IF bandwidth of B, and each with post-detection bandwidths governed by the low pass filter after each power detector. Assuming a single-sideband system that only looked at the upper sideband, then this would make us sensitive to frequencies from $f_{\mathrm{LO}} + f_{\min}$ to $f_{\mathrm{LO}} + f_{\min} + NB$, where f_{LO} is the local oscillator frequency used in the heterodyne system.

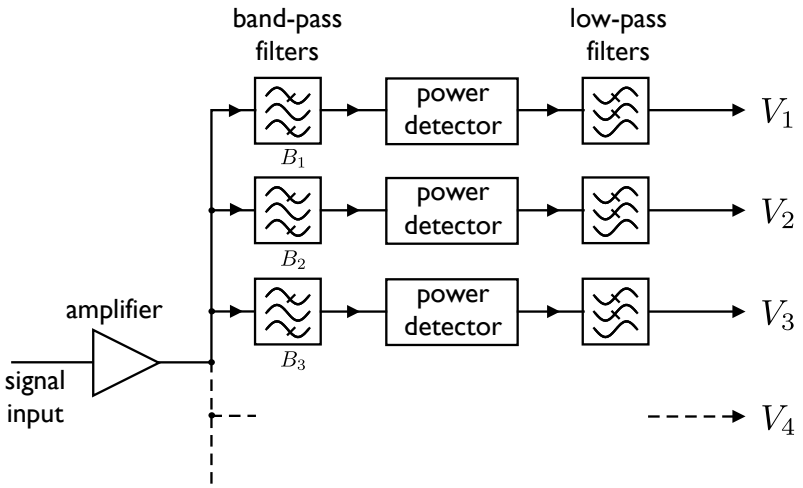


Figure 13.1: (A simple spectrum analyser based on a *filter bank* and a series of power detectors.

To measure the spectrum with increased frequency resolution, we can reduce the value of B, although this of course means that we would need to increase the number of filter-detector channels if we still want to cover the same total spectral range. We also need to bear in mind that the narrower our resulting frequency "bins" are, the less power will be in each one, so the increased spectral resolution would come at the expense of measurement sensitivity. If we were looking at narrowband signals with lots of power in just a few bands, or had loads of signal to play with then this wouldn't be such an issue, but if we were trying to measure the spectrum of say, a thermal body, then this would matter more.

13.2 Simple mixer-based system

A more sophisticated method of spectrum analysis makes use itself of frequency mixing. A simple system of this type is shown in figure 13.2. The signal whose spectrum that we're interested in is applied to the 'RF' port of the mixer (I've used inverted commas here as we're not necessarily talking about radio frequency signals!) and an local oscillator is applied to the LO port. The IF port of the mixer is connected to a very narrow band-pass filter at a fixed frequency, f_0, which means that we'll only see anything after the filter if the difference frequency is equal to f_0.

If the input signal is filtered so that it is always either lower or higher than the LO frequency, then if we know the LO frequency and anything makes it to the power detector through the filter, then we know what the input frequency component causing the output is. If we then *sweep* the LO frequency (in a known way), then the output will be non-zero when the input signal contains a component such that the difference frequency is f_0, and will be zero otherwise. Hence, we can measure the spectrum of an input signal simply by sweeping the LO frequency, and we only require one narrow filter, one power detector (which gives an output

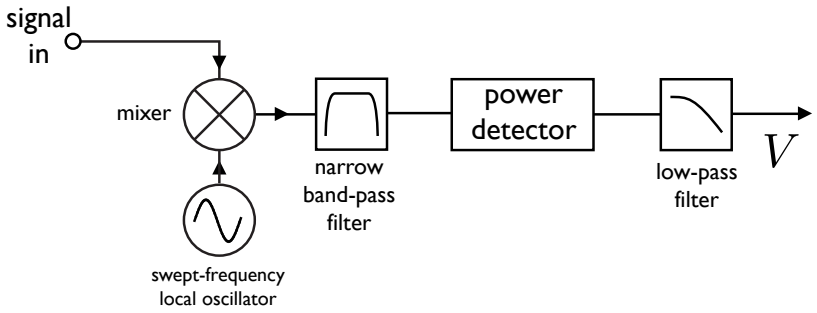


Figure 13.2: A simple spectrum analyser based on a mixer and a swept-frequency LO.

voltage proportional to the input power) and a low-pass filter after the power detector.

The two systems measured above have the advantage over Fourier methods that they do not require the signal to be sampled at a rate of twice the highest frequency it contains, which could prove a challenge at, say, 100 GHz. Modern systems can, however, perform Fourier transforms on signals up to a few GHz. There are systems which employ a mixer front end to *down-convert* a very high input signal frequency to a regime where digital sampling is easier.

13.3 Chapter 13: take-home messages

This short section has shown a number of ways to directly measure the **spectrum** of a signal. It should be clear that it can, in principle, be done with a huge stack of filters, but that a heterodyne system with a swept-frequency local oscillator is more practical.

Chapter 14

Amplitude modulation and demodulation

14.1 Basics of modulation

The earlier part of this course discussed what we mean by information, but now we'll look at how EM fields can be used to communicate information. In general, we can represent a coherent sinusoidal field or wave as

$$S(t) = A\cos(2\pi ft + \phi) \tag{14.1}$$

where A, f and ϕ are the amplitude, frequency and phase respectively. We're using S here as we could be talking about a voltage or a field, so we're keeping it general. Information can be sent by **modulating** one or more of these quantities, and for obvious reasons, signals modulated in these ways are called *amplitude modulated* (AM), *frequency modulated* (FM) or *phase modulated* (PM). (If we're sending a signal through free space, then of course equation **??** would describe a *field*, so we could also modulate the *polarisation*, but this form of modulation is only used for special purposes.)

We'll begin our discussion of modulation by looking at *amplitude modulation*.

14.2 An amplitude modulation circuit

We can define amplitude modulation by saying that a *modulating signal*, $m(t)$, should produce an AM wave of the form

$$S(t) = A_0[1 + m(t)]\cos(2\pi ft + \phi) \tag{14.2}$$

A_0 represents the 'unmodulated amplitude' of the wave (i.e. when $m(t) = 0$). A positive value of $m(t)$ makes the wave's amplitude larger, a negative value makes its amplitude smaller. Amplitude modulation can be produced in various ways, but we'll use the example of a *square*

law FET as shown in figure 14.1. The steady signal, $A_0 \cos(2\pi f t + \phi)$ is called the **carrier**. This gives us something to modulate and 'carry' the information pattern from place to place. The frequency f is called the *carrier frequency* and A_0 is the unmodulated carrier amplitude.

A square law n-channel FET (field effect transistor) will pass a drain-source current

$$i_{ds} = k(v_g + V_p)^2 \tag{14.3}$$

where V_p and k are characteristics of the FET we've chosen, and provided we keep v_g in the range $V_p \leq v_g \leq 0.5$ V. In the circuit shown in figure 14.1 we apply a gate voltage which is a combination of the output from a local oscillator, $A\cos(2\pi f t)$, and a modulation input, $m(t)$, which is the information pattern we want to send from place to place. As a result, the gate voltage (assuming the two gate resistors have the same value) will be

$$v_g = \frac{1}{2} m(t) + \frac{A}{2} \cos(2\pi f t) \tag{14.4}$$

Provided v_g is kept within the required range, this will result in a current

$$i_{ds}(t) = \frac{k}{4} \left[m(t) + A\cos(2\pi f t) + V_p \right]^2 \tag{14.5}$$

This expands to

$$\begin{aligned} i_{ds}(t) = & \frac{k}{4} \left[m^2(t) + V_p^2 + 2V_p m(t) \right] \\ & + \frac{k}{4} \left[2m(t) A\cos(2\pi f t) + 2V_p A \cos(2\pi f t) \right] \\ & + \frac{k}{4} \left[A^2 \cos^2(2\pi f t) \right] \end{aligned} \tag{14.6}$$

The resulting voltage v_d is

$$v_d = V_b - R i_{ds} \tag{14.7}$$

where R is the resistance between the FET's drain terminal and the bias voltage V_b. Now,

$$\cos^2(\theta) = \frac{1}{2} \left(1 + \cos(2\theta) \right) \tag{14.8}$$

Thus, the last term in equation 14.6 is a combination of a steady current and a fluctuation at $2f$. For simplicity we can arrange that the frequencies associated with $m(t)$ all $\ll f$. This means that the first part of the expression consists of a steady current plus some fluctuations at frequencies well below f. We can now use a bandpass filter, designed to only pass frequencies $\approx f$ to remove low and high frequencies and obtain an output

$$v_{\text{out}}(t) = -\frac{Rk}{4} \left[2m(t) A\cos(2\pi f t) + 2V_p A \cos(2\pi f t) \right] \tag{14.9}$$

which we can rewrite as

$$v_{\text{out}}(t) = A_0' \left[1 + m'(t) \right] \cos(2\pi f t) \tag{14.10}$$

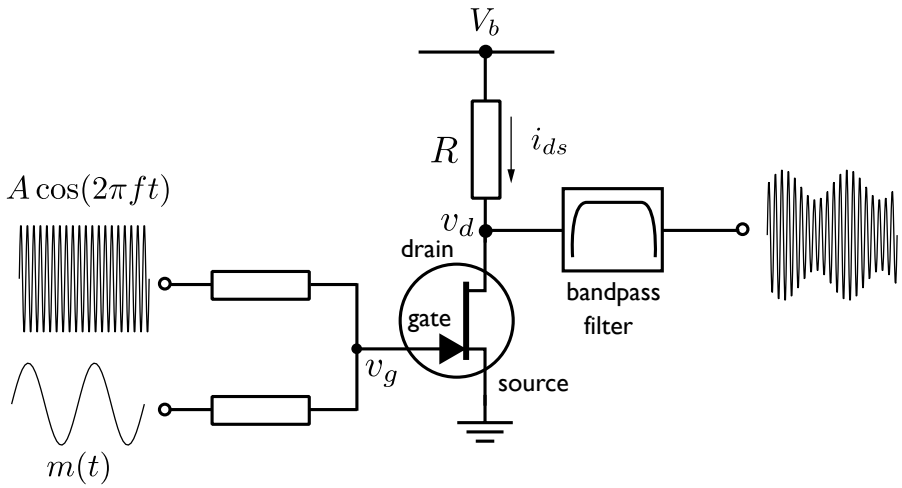


Figure 14.1: A simple amplitude modulation circuit using a square-law FET.

where

$$m'(t) = \frac{m(t)}{V_p} \tag{14.11}$$

and

$$A'_0 = -\frac{RkV_pA}{2} \tag{14.12}$$

i.e. the output is a wave whose unmodulated amplitude is A'_0 and is amplitude modulated by an amount, $m'(t)$, proportional to the input modulating signal, $m(t)$. The circuit therefore behaves as an amplitude modulator. Hurrah!

14.3 The envelope detector

There are various ways to measure or detect the amplitude (as opposed to the power) of a waveform. Here we'll consider one of the simplest, used by cheap portable radios, etc, the *envelope detector*.

The envelope detector is shown in figure 14.2. The diode only allows positive voltages through (in electronics parlance, it acts as a *half-wave rectifier*. (Rectification is the process of turning an ac signal into a single-polarity signal: this is called half-wave as it allows through the positive half, and blocks the negative half.) When the input wave's amplitude increases, the capacitor voltage is increased via the rectifying diode. When the input's amplitude falls, the capacitor voltage is reduced by being discharged by a 'bleed' resistor, R. The main advantage of this form of AM *demodulator* is that it is very simple and cheap. However, simple and cheap are not always best, and it does have some problems associated with it.

The circuit relies upon the behaviour of the diode: allowing current through when the input

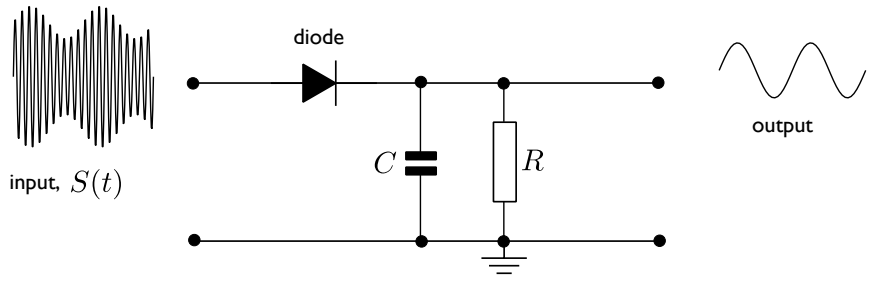


Figure 14.2: A simple amplitude de-modulation circuit: the *envelope detector*.

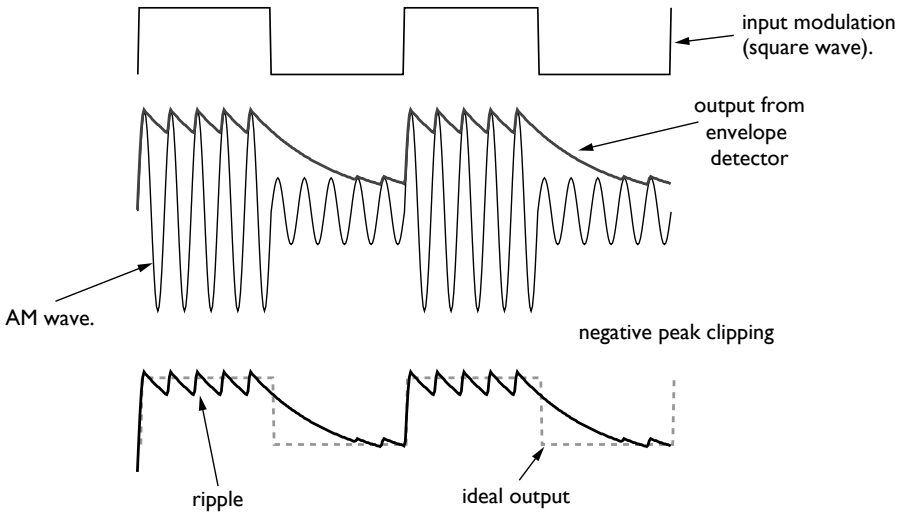


Figure 14.3: *Ripple* and *peak-clipping* effects from a simple envelope detector.

is positive with respect to the capacitor voltage, hence charging the capacitor to the peak voltage level, but blocking any current from flowing back out through the diode when the input voltage is below the capacitor voltage. Unfortunately, all real diodes are non-linear, in that the current they pass varies with the applied voltage. As a result, the demodulated output is slightly distorted in a way which depends upon the diode's current vs voltage characteristic. For example, most cheap AM transistor radios produce output signals with about 5-10% distortion, fine for casual listening, but nowhere good enough for hi-fidelity reproduction. As a result, this simple type of AM demodulator isn't any good if we want the recovered waveform to be an accurate representation of the original modulating waveform.

The circuit also suffers from the problems known as *ripple* and *negative peak clipping*. These effects are illustrated in figure 14.3. The ripple effect happens because the capacitor will be discharged a small amount in between successive peaks of the input AM wave.

Figure 14.3 shows what happens in the worst possible situation where the modulating signal

is a square wave whose frequency isn't much lower than the carrier frequency. Similar, but less severe, problems can arise with other modulating signals.

Consider what happens when we have a carrier frequency, f_c, and use an envelope detector whose time constant, $\tau = RC$. The time between successive peaks of the carrier will be

$$T = \frac{1}{f_c} \tag{14.13}$$

Each peak will charge the capacitor to some voltage, V_{pk}, which is proportional to the modulated amplitude of the AM wave. Between each peak and the next the capacitor voltage will therefore be discharged to

$$V'_{\mathrm{pk}} = V_{\mathrm{pk}} \exp(-T/\tau) \tag{14.14}$$

and if $T \ll \tau$, then

$$V'_{\mathrm{pk}} \approx V_{\mathrm{pk}}(1 - T/\tau) \tag{14.15}$$

so the peak-to-peak ripple voltage will be

$$\Delta V \approx \frac{V_{\mathrm{pk}}}{f_c \tau} \tag{14.16}$$

A sudden, large reduction in the amplitude of the input AM wave means that capacitor charge isn't being 'topped up' by each cycle peak. The capacitor voltage therefore falls exponentially until it reaches the new, smaller, peak value. To assess this effect, consider what happens when the AM wave's amplitude suddenly reduces from V_{pk} to a much smaller value. The capacitor voltage then declines according to

$$v = V_{\mathrm{pk}} \exp(-t/\tau) \tag{14.17}$$

This produces the negative peak clipping effect where any swift reductions in the AM wave's amplitude are not correctly represented in the envelope detector's output. Here we've chosen the worst possible case of square wave modulation. In practice the modulating signal is normally restricted to a specific frequency range. This limits the maximum rate of fall of the AM wave's amplitude. We can therefore hope to avoid negative peak clipping by arranging that the detector's time constant $\tau \ll 1/f_m$ where f_m is the highest modulation frequency used in a given situation.

Negative peak clipping will be reduced by a smaller value of τ, but the ripple will be increased by a smaller value of τ, so in practice we should pick R and C such that

$$\frac{1}{f_m} \gg \tau \gg \frac{1}{f_c} \tag{14.18}$$

to minimise the effects of these problems. Obviously, we can only do this if $f_m \ll f_c$.

Although an envelope detector will demodulate AM signals, a mixing-based technique is generally preferred.

14.4 Characteristics of AM waves

In general, we can imagine amplitude modulating a carrier with a modulation input

$$m(t) = \sum_{i=1}^{N} a_i \cos(2\pi f_i t + \phi_i) \tag{14.19}$$

where all f_i lie within some *modulation bandwidth*, B_M, which extends from about zero frequency up to a frequency $f_{\max}$. This produces an AM wave

$$S(t) = A_0[1 + m(t)]\cos(2\pi f_c t) \tag{14.20}$$

which can be shown to yield

$$\begin{aligned} S(t) = {} & A_0 \cos(2\pi f_c t) \\ & + \frac{A_0}{2} \sum_{i=1}^{N} a_i \cos\left[2\pi(f_c + f_i)t + \phi_i\right] \\ & + \frac{A_0}{2} \sum_{i=1}^{N} a_i \cos\left[2\pi(f_c - f_i)t - \phi_i\right] \end{aligned} \tag{14.21}$$

The modulated wave thus consists of three parts: the *carrier* (unaffected by the modulation), the *upper sideband* components $(f_c + f_i)$ (which are an up-converted copy of the original modulation spectrum) and the *lower sideband* components $(f_c - f_i)$ which are a mirror image (in frequency terms) of the upper sideband components (mirrored, that is, around the carrier frequency).

This is shown in figure 14.4.

The AM wave essentially carries two copies of the modulation pattern: one in each transmission sideband. As a result, it occupies a *transmission bandwidth*, $B_T = 2B_M = 2f_{\max}$ i.e. it takes up twice as much bandwidth as the original information. This represents one of the main disadvantages of AM modulation. The number of independent transmissions we can make will be limited by the range of frequencies available and how much bandwidth each transmission requires. The *double sideband* nature of AM halves the number of independent signals we can send using a given range of transmission frequencies.

Some transmission systems adopt a modified form of AM called *single sideband modulation* (SSB). This is essentially an AM wave with one sideband suppressed or filtered before transmission. SSB modulated transmissions are called either upper sideband or lower sideband depending on which sideband is used to carry the modulation information pattern. By using SSB we can double the number of transmissions which will 'fit' into a given transmission band. However, SSB transmitters and receivers are more complicated (and expensive!) than the simple AM circuits described earlier. Hence they tend only to be used for specialised purposes like aircraft or ship communications.

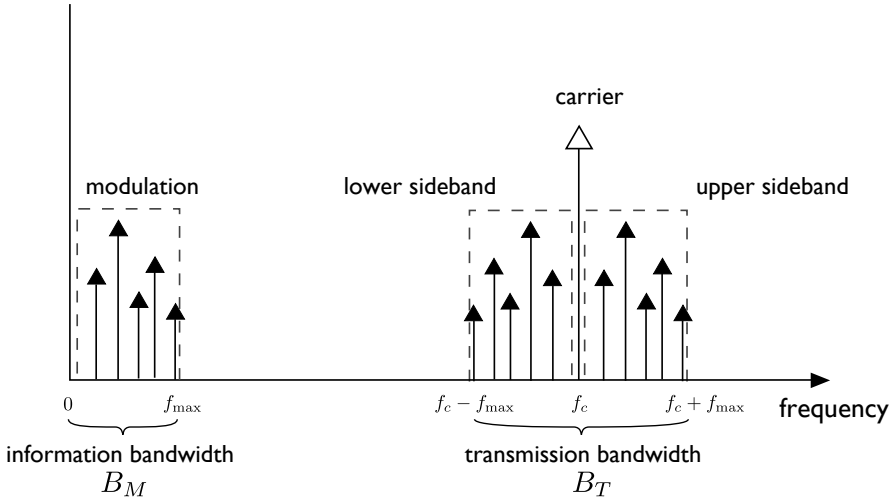


Figure 14.4: The spectra of the modulation waveform and the resulting AM wave.

From the above arguments we can see that normal AM is prone to being wasteful of transmission band-space. Another feature of AM is that it is also wasteful of *power*. To see why, consider what happens when the carrier is modulated by a single sinewave,

$$m(t) = a_m \cos(2\pi f_m t) \tag{14.22}$$

where $a_m > 1$. This means that, at times, $1 + m(t) < 0$ and the carrier becomes *phase inverted*. This process is illustrated in figure 14.5.

An envelope detector can't know that the carrier phase has been altered, hence it just responds to the +ve peaks of the inverted modulated carrier. This means it produces a distorted output whenever $1+m(t) < 0$. A number of more complex/expensive techniques can be used to avoid this problem. In effect, better AM demodulators sense the carrier phase and take this into account. However, 'cheap and cheerful' systems just use the envelope detection technique. For this reason it is usually necessary to ensure that the modulation is always limited so that $m(t) \geq -1$ to avoid this problem.

For simple sinewave modulation this means that we require $|a_m| \leq 1$. This means that a sinewave modulated AM wave will have the form

$$S(t) = A_0 \cos(2\pi f_c t) + \frac{A_0 a_m}{2} \cos(2\pi[f_c + f_m]t) + \frac{A_0 a_m}{2} \cos(2\pi[f_c - f_m]t) \tag{14.23}$$

Now, the mean carrier *power* transmitted will be

$$P_c = \frac{A_0^2}{2R} \tag{14.24}$$

where R is the impedance of the medium or system which the wave is being propagated

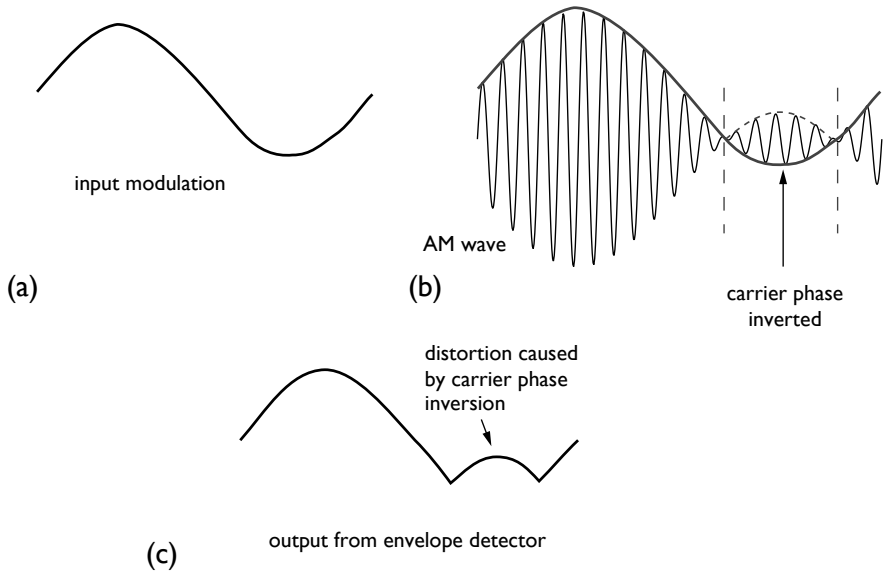


Figure 14.5: (a) A modulation signal for which $a_m > 1$. (b) The resulting AM wave showing *carrier phase inversion*. (c) The resulting output from an envelope detector which is distorted due to the carrier phase inversion.

along/through. Similarly, the combined mean power of the two sideband components will be

$$\begin{aligned} P_m &= \frac{1}{2R}\left[\left(\frac{A_0 a_m}{2}\right)^2 + \left(\frac{A_0 a_m}{2}\right)^2\right] \\ &= \frac{A_0^2 a_m^2}{4R} \end{aligned} \tag{14.25}$$

Since we can expect $|a_m| \leq 1$, then we can say that the ratio of the sideband power to the total transmitted power will be

$$\frac{P_m}{P_t} = \frac{a_m^2}{2 + a_m^2} \leq \frac{1}{3} \tag{14.26}$$

i.e., the largest acceptable amount of modulation means that only one third of the total transmitted power appears in the modulation sideband components of the AM wave. Put another way, this means that at least two thirds of the transmitted power is devoted to sending a steady carrier wave. This power is virtually wasted. It only serves two purposes: 1) to let envelope detectors work correctly; 2) to provide a steady signal which is maintained when there is no actual modulation, confirming the existence of the transmitter!

In situations like domestic broadcasting one transmitter may serve ten million receivers. Under these circumstances the cheapness of the ten million envelope detectors can justify the waste of transmitter power. For specialised applications it may however be more sensible to use a more power efficient modulation system and a more expensive receiver. Some transmission systems therefore use a *suppressed carrier* method. This is similar to conventional AM or SSB, but with the carrier filtered away. The resulting transmissions are very power efficient,

but require complex receivers which can be relatively difficult to tune. A particular problem of suppressed carrier systems arises when no modulation is being sent. Then there will be no sideband components. Nor will there be any carrier wave since it has been suppressed. Hence the suppressed carrier wave consists of nothing at all when there is no modulation! This certainly saves energy, but it can make it difficult to tune in a receiver correctly...

AM also suffers from being prone to *interference* effects. The AM demodulator essentially measures the power or amplitude of the signals presented to it. Any other power, e.g. pulses radiated by electric drills, sparking motors etc can also be detected. This is why AM radio tends to suffer from buzzes, crackles, whistles, etc. Other forms of modulation can avoid this sensitivity and are less prone to unwanted interference.

14.5 Chapter 14: take-home messages

We've seen that we can communicate information by **modulating** a **carrier** wave. *Amplitude* modulation uses variations in the carrier amplitude to convey information. One method of AM modulating a carrier wave is to use a square law device such as an FET.

The simplest method of de-modulating an AM signal is to use an envelope detector, which suffers from *ripple* and *negative peak clipping*. The magnitude of the modulation must be limited to avoid distortions arising due to carrier phase inversion. AM is wasteful of transmission bandwidth and transmitter power and is inherently sensitive to interference. Single sideband and suppressed carrier methods exist and are less wasteful of transmission bandwidth and power, but require more sophisticated forms of demodulator.

Chapter 15

Frequency modulation and demodulation

We've just seen how it is possible to convey information using amplitude modulation, and already mentioned that we can also use *frequency modulation* (FM) and *phase modulation* (PM). We'll now go on to look at the basic properties of FM/PM signals and the methods we use to produce them.

15.1 FM and PM

We should start by considering what we mean by the *frequency* of a waveform. For a simple sinewave the answer appears quite obvious, we can define the wave using an expression like

$$S(t) = A\cos(2\pi ft + \phi) \tag{15.1}$$

and then identify f as the frequency. An alternative way to express this is

$$S(t) = A\cos\left(\Theta(t)\right) \tag{15.2}$$

where $\Theta(t) = 2\pi ft + \phi$ is the wave's **phase** at an instant t. For a sinewave, f is constant and $\Theta(t)$ increases linearly with time at the rate

$$\frac{\mathrm{d}\Theta}{\mathrm{d}t} = 2\pi f \tag{15.3}$$

We can define the FM wave, produced when we modulate a carrier frequency, f_c, with a modulating signal, $m(t)$, to be

$$S(t) = A\cos(2\pi f_i(t)t + \phi) \tag{15.4}$$

where

$$f_i(t) = f_c + k_f m(t) \tag{15.5}$$

is the *instantaneous frequency* of the wave at a time t. k_f is a constant the value of which depends on the modulating system, and describes how large a frequency change we get for a given size of modulating signal.

The FM wave can now be used to convey information about the modulating pattern in a manner similar to the AM variations we examined in an earlier lecture. Note that, unlike an AM wave, the FM wave doesn't have a single frequency value. This makes an FM wave obviously different to an AM one. Instead, we can define two distinct quantities; its 'unmodulated' (i.e. carrier) frequency, and its modulated frequency, $f_i(t)$, which can change from instant to instant. The instantaneous phase of the modulated wave at any instant can be obtained by substituting equation 15.5 into the expression for $\Theta(t)$ and integrating to get

$$\Theta_i(t) = 2\pi f_c t + 2\pi k_f \int_0^t m(t)\mathrm{d}t \tag{15.6}$$

where, for convenience, we're assuming that the phase is zero at time $t = 0$.

In a similar way, we can define a *phase modulated* (PM) wave as having the form

$$S(t) = A\cos(\Theta_i(t)) \tag{15.7}$$

where

$$\Theta_i(t) = 2\pi f_c t + k_p m(t) \tag{15.8}$$

where k_p determines how much change of phase we get for a given size of modulation signal. The instantaneous *frequency* of the PM wave will therefore be

$$f_i(t) = f_c + k_p \frac{\mathrm{d}m(t)}{\mathrm{d}t} \tag{15.9}$$

Comparing the instantaneous frequency and phase expressions for the FM and PM cases we find that, unless we know something about the modulation in advance, it may not be obvious whether the signal is FM or PM modulated. In both cases the wave's frequency and phase vary from moment to moment. Mathematically speaking, FM and PM are almost identical twins. The only difference is that one corresponds to a modulation pattern which is the differential of that produced by the other. The good news is that this means we can mix FM and PM arguments and most of our conclusions about one apply to the other, although we do need to know in advance which type of modulation is being used if we want to recover the modulated information correctly. In general, however, both FM and PM waves are obviously different to an AM wave.

15.2 The spectrum of an FM signal

From equation 15.6 we can write an FM wave in the form

$$S(t) = A\cos\left[2\pi f_c t + 2\pi k_f \int_0^t m(t)\mathrm{d}t\right] \tag{15.10}$$

To get an insight into what the spectrum of a typical FM signal would be like, we'll consider a simple example where we're just modulating with a sinewave:

$$m(t) = A_m \cos(2\pi f_m t) \tag{15.11}$$

so the instantaneous frequency of the FM signal will be

$$f_i(t) = f_c + k_f A_m \cos(2\pi f_m t) \tag{15.12}$$

This tells us that f_i swings up and down either side of f_c over a range of $\pm k_f A_m$. This range is usually described in terms of the modulated signal's *peak frequency deviation* value, defined as

$$\Delta f = k_f A_m \tag{15.13}$$

as it indicates the largest swing or deviation in frequency either side of f_c. Note that Δf depends on the size of the modulation, A_m, but *not* on the modulating frequency, f_m. By combining equations 15.6, 15.11 and 15.13, we can say that the instantaneous phase of the sinwave modulated FM signal is

$$\Theta(t) = 2\pi f_c t + \frac{\Delta f}{f_m}\sin(2\pi f_m t) \tag{15.14}$$

It is conventional to define a quantity called the *modulation index*:

$$\beta \equiv \frac{\Delta f}{f_m} \tag{15.15}$$

and then write the FM wave as

$$S(t) = A\cos(2\pi f_c t + \beta\sin(2\pi f_m t)) \tag{15.16}$$

and this provides us with information on how the modulated signal varies with time. However we often need to know the frequency spectrum of the modulated wave, in order to be able to determine the bandwidths of any filters, amplifiers, etc that we require.

By consulting a good maths text we can find that the above equation can (with some effort) be rewritten as

$$\begin{aligned} S(t) = {} & AJ_0(\beta)\cos(2\pi f_c t) \\ & + A\sum_{k=1}^{\infty} J_{2k}(\beta)\left[\sin(2\pi[f_c + 2kf_m]t) + \sin(2\pi[f_c - 2kf_m]t)\right] \\ & + A\sum_{k=0}^{\infty} J_{2k+1}(\beta)\left[\cos(2\pi[f_c + (2k+1)f_m]t) - \cos(2\pi[f_c - (2k+1)f_m]t)\right] \end{aligned} \tag{15.17}$$

where $J_n(\beta)$ is the *Bessel function* (first kind, integer order n) for the value β. Like the error function encountered earlier in the course, we treat the Bessel function as something that we can look up. (Bessel functions pop up in a variety of areas in physics, from optics to QM.)

Clearly, this horrendous expression is much more complicated than for the case of a sinewave modulated AM wave. The sinewave modulated AM wave has only three spectral components, at frequencies f_c, $(f_c - f_m)$ and $(f_c + f_m)$. The FM wave has spectral components at all the frequencies $(f_c + nf_m)$, where n takes on all the integer values between $-\infty$ and $+\infty$. This rather startling result means that, strictly speaking, a system has to provide an *infinite* bandwidth to carry an accurate FM (or PM) signal!

However, things are not as bad as they seem: the behaviour of Bessel functions is such that there is a general tendency for

$$|J_n(\beta)| \to 0 \qquad \text{as} \qquad n \to \infty \tag{15.18}$$

and the higher order Bessel function values fall quickly with n when the modulation index, β, is small. (Phew!) In many practical situations we can arrange that $\beta \ll 1$ and when this is the case then we find that $J_0(\beta) \approx \beta$, $J_1(\beta) \approx \beta/2$ and $J_n(\beta) \approx 0$ for $n > 1$.

An FM signal produced with a low modulation index (so where $\Delta f \ll f_m$) is called a *narrowband* FM signal. For most purposes we can ignore the high-order Bessel function terms and represent the spectrum of a narrowband FM signal as

$$\begin{aligned} S(t) \approx\ & AJ_0(\beta)\cos(2\pi f_c t) \\ & + AJ_1(\beta)\left[\cos(2\pi[f_c + f_m]t) - \cos(2\pi[f_c - f_m]t)\right] \end{aligned} \tag{15.19}$$

This narrowband FM (or PM) signal is similar to AM in that it has sideband components at $(f_c \pm f_m)$, hence requires a transmission bandwidth of $2f_m$. Its spectrum, however, differs from that of an AM signal in two ways: firstly, the total amplitude of the modulated wave remains almost constant, and secondly, the two sideband components are 180 degrees out of phase with each other (i.e., their signs differ).

A 'high-β' FM wave can be thought of as a carrier whose frequency is varied over a relatively wide range. It will therefore require a transmission bandwidth of at least $2\Delta f$. Combining this result with that for a narrowband FM wave leads to *Carson's rule*, that the minimum practical bandwidth required to transmit an FM/PM signal will be

$$B = 2(f_m + \Delta f) \tag{15.20}$$

This rule is a useful guide when we have to choose a system to carry an FM signal. It should be remembered, however, that in theory FM signals require an infinite bandwidth if we want to avoid *any* signal distortion during transmission.

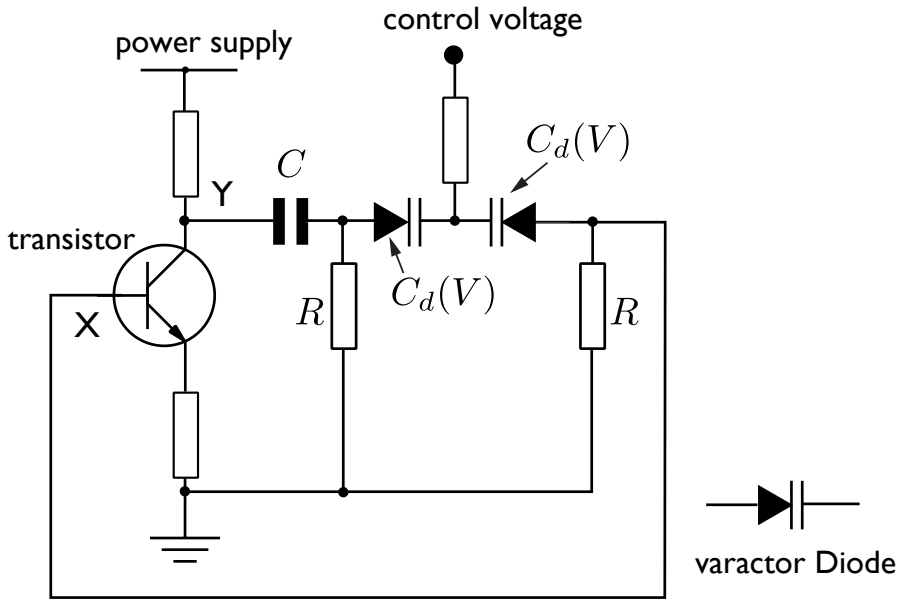


Figure 15.1: A simple voltage controlled oscillator.

15.3 FM sources

Having established the basic properties of FM/PM signals we can now look at two examples of ways to generate them. The first example is electronic, the second uses optical techniques. The electronic example is based upon a *voltage controlled oscillator*, a simple example of which is shown in figure 15.1.

There are a number of ways to make a voltage controlled oscillator, and figure 15.1 shows a simple example based on a single transistor. The transistor acts as a simple inverting amplifier, i.e the output at point Y will be an amplified and inverted (flipped by 180 degrees) version of the input at B. The two *varactor diodes* are a special type of diode with a large capacitance that can be varied by changing the applied control voltage, so act as variable capacitors. Thus, the transistor output (Y) is connected to the input (X) via a network of resistors and capacitors, including variable capacitors. If the phase shift produced by this network is 180 degrees, then the device will *oscillate*. The frequency this will happen at is a function of R, C and C_d and any small temporary fluctuation at this frequency will grow into a steady oscillation. By varying the control voltage, we can vary the frequency that the circuit will oscillate at, and can thus use the control voltage to allow us to modulate the oscillation frequency: i.e. we have an oscillator and a control point that allows us to create FM (or PM) signals.

VCOs (not necessarily of the above type) can be used up to frequencies of 100 GHz or so. In the THz region or above, we have to switch to *optical* techniques. A common example of this is to send our optical signal through a crystal whose refractive index may be modified by applying a a large electric field to it. This will change the refractive index of the material, and

thus the *phase length* of the crystal. By modulating the voltage across the crystal, we can modulate the phase of the output beam, which provides another method creating an FM/PM wave. Note that where the VCO actually creates the carrier signal and allows us to modulate it, the use of an electro-optic modulator relies on having a separate fixed-frequency coherent oscillator to generate the beam which is then modulated.

We've just presented two example here of how to produce FM/PM signals: many others exist although most have features in common with those described above.

15.4 FM/PM demodulation

We'll now go on to look at how we can *de-modulate* FM or PM signals and recover the information carried by them. We'll concentrate on the demodulation of FM signals, but similar methods are useful for general measurements of signal frequencies and for demodulation of PM signals.

The most popular electronic method for demodulation of FM signals is the *phase-locked-loop* or PLL. Before we explain what it is and how it can be used to demodulate FM signals, we'll take a look at one of the key elements in a PLL: the **double balanced mixer** (DBM), which is shown in figure 15.2.

Figure 15.2 (a) shows the circuit diagram for a DBM. It consists of four diodes linking two transformers. There are three ways or ports by which signals can get in/out of the DBM. It is conventional to call these the *RF*, *LO*, and *IF* ports since DBMs are often used as mixers in heterodyne systems. To see how the system works we can start by considering the LO port. This lets us apply a voltage between the points A and B in the circuit. When we apply a voltage which makes A positive with respect to B (which we'll assume occurs when $v_{\mathrm{LO}} > 0$) the diodes D_1 and D_2 will conduct and D_3 and D_4 will not. This means that D_1 and D_2 will present a very low resistance to any signals and D_3 and D_4 will present a very high resistance. As a result, the circuit will behave like figure 15.2 (b). The two transformers will be 'directly' connected together via the conducting diodes. Using 1:1 transformers and low-loss diodes we therefore find that, when v_{LO} is positive, $v_{\mathrm{out}} = v_{\mathrm{in}}$ (for a.c. signals, of course!). When we apply an LO voltage such that A is negative compared to B we can use a similar argument to show that, in this case, the circuit behaves like figure 15.2 (c). Hence when v_{LO} is negative, $v_{\mathrm{out}} = -v_{\mathrm{in}}$. We can therefore use the DBM as a sort of switch to control whether we pass on the the signal unaffected or invert it by applying the required polarity of v_{LO}.

In practice, a real DBM requires a given minimum amplitude of LO voltage to ensure the diodes have 'turned on/off' properly. For typical silicon or GaAs diodes this usually means we need about 1 $\mathrm{V_{pk-pk}}$ of LO at the diodes. DBMs of this type can be used for frequencies from below 1 MHz to tens of GHz. Equivalent arrangements (using other items in place of ordinary transformers) can be used at higher frequencies.

In a phase-locked-loop (PLL), the DBM is used in combination with a low pass filter (or

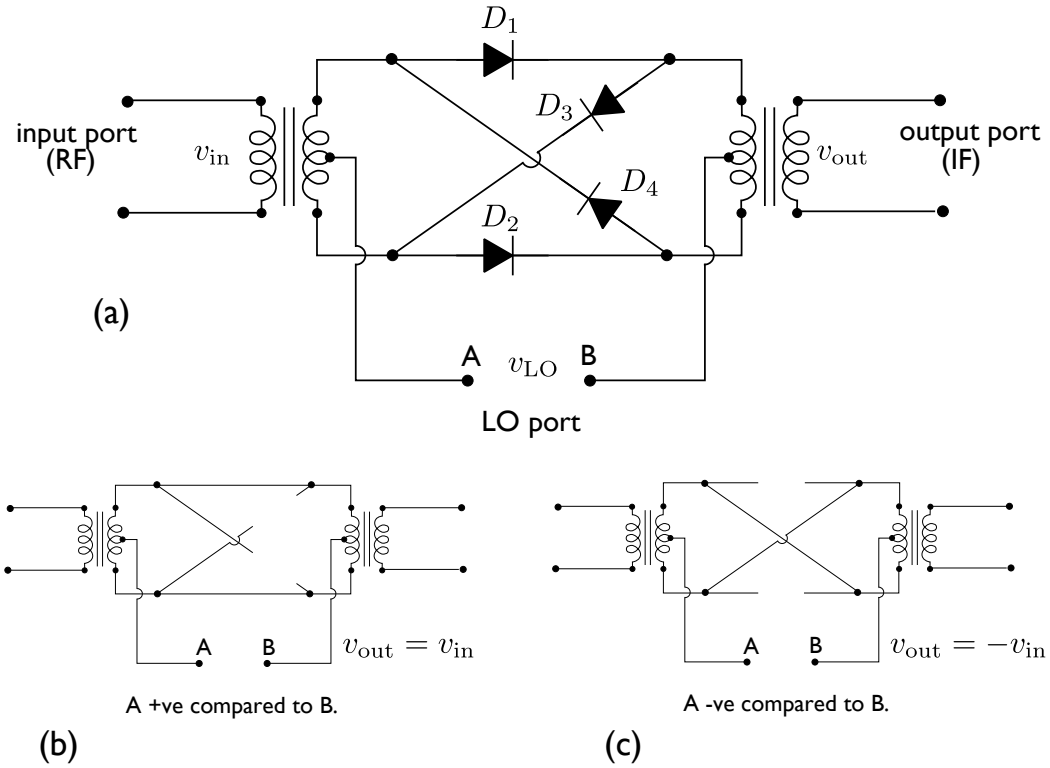


Figure 15.2: A double balanced mixer.

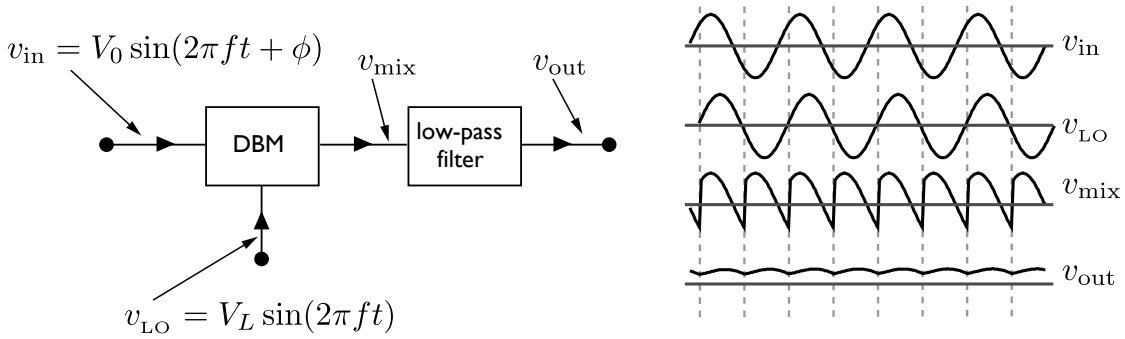


Figure 15.3: A DBM and low-pass filter combined to form a *phase detector*.

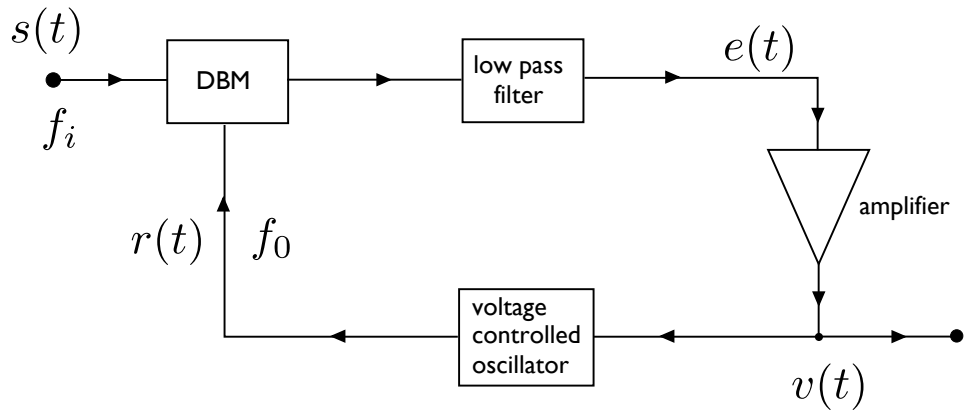


Figure 15.4: A *phase locked loop* system comprising of a DBM, low pass filter, amplifier and VCO.

time constant) as a *phase detector*. To understand how this works, consider the situation illustrated in figure 15.3. Here an input signal

$$v_{\text{in}} = V_0 \sin(2\pi f t + \phi) \tag{15.21}$$

is mixed with an LO of

$$v_{\text{LO}} = V_L \sin(2\pi f t) \tag{15.22}$$

These produce an output of

$$\begin{aligned} v_{\text{mix}} &= V_0 \sin(2\pi f t + \phi) && [v_{\text{LO}} > 0] \\ v_{\text{mix}} &= -V_0 \sin(2\pi f t + \phi) && [v_{\text{LO}} < 0] \end{aligned} \tag{15.23}$$

The output low-pass filter (time constant) smooths v_{mix} over a number of cycles to produce an output whose average level can be worked out by integrating v_{mix} over one cycle and dividing by the cycle period. (Since every cycle is just like the others this gives the same result as averaging over many cycles.) so we can say that

$$v_{\text{out}} = \frac{1}{T}\left[\int_0^{T/2} V_0 \sin(2\pi f t + \phi) - \int_{T/2}^{T} V_0 \sin(2\pi f t + \phi)\right] \tag{15.24}$$

which yields the pleasingly simple result

$$v_{\text{out}} = V_0 \cos(\phi) \tag{15.25}$$

The amount by which the output 'ripples' up and down this average value will depend on the details of the low pass filter. Here we can ignore this ripple and just assume that we get an output proportional to the input signal's amplitude multiplied by the cosine of the phase offset, ϕ, between the signal and LO inputs to the DBM. The system therefore provides an output which depends upon this phase.

Having established the use of the DBM we can now consider the PLL illustrated in figure 15.4. Here the input is an FM wave

$$s(t) = V_0 \sin(2\pi f_c t + \phi_s(t)) \tag{15.26}$$

whose carrier (unmodulated) frequency is f_c and whose modulation is represented by the time varying phase

$$\phi_s(t) = 2\pi k_f \int m(t)\mathrm{d}t \tag{15.27}$$

where $m(t)$ is the modulation pattern we wish to recover and k_f is the FM modulator's voltage to frequency conversion gain value. The LO input to the DBM comes from a VCO and can, in a similar way, be represented as

$$r(t) = \sin(2\pi f_0 t + \phi_L(t)) \tag{15.28}$$

where

$$\phi_L(t) = 2\pi k_L \int v(t)\mathrm{d}t \tag{15.29}$$

and $v(t)$ is the control voltage which sets the VCO's output frequency at any instant. k_L is the VCO's frequency to voltage conversion gain value. The output from the DBM, smoothed by the low pass filter, will therefore be

$$e(t) = V_0 \cos(\phi_s - \phi_L) \tag{15.30}$$

or

$$e(t) = V_0 \sin\phi \tag{15.31}$$

where

$$\phi \equiv \phi_s - \phi_L + \frac{\pi}{2} \tag{15.32}$$

where we've chosen to re-write it this way for reasons that will become apparent shortly. ϕ is called the *phase error* between the signal and the VCO output. (Note that this is actually the angle by which they depart from being $\pi/2$ out of phase, which is called *quadrature*). For similar reasons it is conventional to call $e(t)$ the *error voltage* produced by the phase detector since it is a voltage which indicated the magnitude and sign of the phase error.

Now the input voltage controlling the VCO will be

$$v(t) = A_v e(t) \tag{15.33}$$

where A_v is the voltage gain of the amplifier which follows the low pass filter. We can therefore say that

$$\phi_L = 2\pi k_L A_v \int e(t)\mathrm{d}t \tag{15.34}$$

Putting this into equation 15.32 and differentiating, we can obtain

$$\frac{\mathrm{d}\phi(t)}{\mathrm{d}t} = \frac{\mathrm{d}\phi_s(t)}{\mathrm{d}t} - 2\pi k_L A_v e(t) \tag{15.35}$$

which, using equation 15.31, is equivalent to

$$\frac{\mathrm{d}\phi(t)}{\mathrm{d}t} = \frac{\mathrm{d}\phi_s(t)}{\mathrm{d}t} - 2\pi k_L A_v V_0 \sin[\phi(t)] \tag{15.36}$$

When the system is 'phase locked' we should find that the signal and LO are almost in quadrature, i.e. $|\phi| \ll \pi/2$. We can therefore assume that $\sin(\phi) \approx \phi$ and simplify equation 15.36 into

$$\frac{\mathrm{d}\phi(t)}{\mathrm{d}t} = \frac{\mathrm{d}\phi_s(t)}{\mathrm{d}t} - K\phi(t) \tag{15.37}$$

where

$$K \equiv 2\pi k_L A_v V_0 \tag{15.38}$$

is called the phase lock loop's *loop gain*. The output from the PLL will be

$$v_{\mathrm{out}}(t) = v(t) = A_v e(t) = A_v V_0 \sin[\phi(t)] = A_v V_0 \phi(t) \tag{15.39}$$

i.e.

$$v_{\mathrm{out}}(t) = \frac{K}{2\pi k_L}\phi(t) \tag{15.40}$$

If we observe an output voltage

$$v_{\mathrm{out}}(t) = a\sin(2\pi f_m t) \tag{15.41}$$

the phase error must therefore be

$$\phi(t) = \frac{2\pi k_L}{K} a \sin(2\pi f_m t) \tag{15.42}$$

Putting this into equation 15.37 we can expect that

$$\frac{\mathrm{d}\phi_s}{\mathrm{d}t} = 2\pi k_f m(t) \tag{15.43}$$

Comparing these expressions we obtain

$$m(t) = \frac{2\pi k_L f_m}{k_f K} a\cos(2\pi f_m t) + \frac{k_l}{k_f} a \sin(2\pi f_m t) \tag{15.44}$$

By using a system which has a very high loop gain we can ensure that

$$\frac{2\pi k_L f_m}{k_f K} \to 0 \tag{15.45}$$

so we can ignore the cosine term and end up with

$$m(t) = \frac{k_L}{k_f} a \sin(2\pi f_m t) \tag{15.46}$$

so (eventually!!!) we finally discover that

$$v_{\mathrm{out}} = \frac{k_f}{k_L} m(t) \tag{15.47}$$

This result shows that the PLL's output voltage varies in proportional with the modulation, $m(t)$, we wish to demodulate *provided that two conditions are satisfied*:

- The loop gain is very large
- The loop maintains the VCO output and signal phases effectively in quadrature.

A more general explanation of the PLL's behaviour is that it continuously adjusts the VCO output to maintain phase quadrature. Any change in the input FM wave's frequency or phase tends to produce a DBM output which causes the VCO to 'track' the input. For the two to maintain a constant phase relationship they must keep 'in step', i.e. their frequencies must always be the same. Since the VCO's frequency depends upon the control voltage, $v(t)$, it follows that this voltage varies in proportion with the FM wave's frequency. Hence the variations of $v(t)$ provide us with the demodulated pattern we require. The PLL we've considered is called a simple *first order* loop. More complex types of loop also exist. These often work better but are harder to analyse. Their basic operation is the same as the system we've considered.

The PLL has some interesting features. The most useful of these is that it largely ignores fluctuations in the amplitude of the input FM wave. (You can see this is true because equation 15.47 doesn't include the input wave's amplitude, V_0.) This means that the PLL FM demodulator tends to ignore any unwanted interference which appears alongside the input signal. The loop is said to have the property of *AM rejection*.

Of course, the system can't work perfectly for *any* input signal size, no matter how small. The loop gain *does* depend upon V_0, so if the FM wave's amplitude is too small we can't assume that the gain is very large. Looking back at equation 15.44 we can find that a low input signal amplitude has four effects:

- It reduces the size of the output.
- It makes the size of the output depend upon the modulation frequency.
- It makes the demodulated output slightly out of phase with the modulation.
- It makes the system respond to any amplitude modulation.

For these reasons we usually try to ensure that the input wave's amplitude is large enough to avoid these problems.

The above analysis assumed that the VCO could always adjust its output to 'track' any changes in the FM wave's frequency. This is the same as assuming that the low pass filter isn't affecting the DBM's output. In reality, a low pass filter will tend to attenuate (and phase shift) any swiftly changing signals. As a result, the above arguments only strictly apply for a system which is *locked* (the signal and VCO maintain a steady phase relationship) when the modulation frequency is 'low' (i.e. passes through the filter without being affected).

Consider what happens when we start with the PLL not locked to an input signal. The input and VCO frequencies will differ by some amount $\Delta f = f_s - f_0$ so the output from the DBM

(averaged over a few cycles of carrier) will be

$$v_{\text{DBM}}(t) = V_0 \cos[2\pi(f_s - f_0)t] \tag{15.48}$$

i.e. the output from the DBM tends to oscillate at the *difference frequency*, Δf. Provided that Δf is low enough, this variation can pass through the low pass filter and the loop can use this to help it *lock onto* the input wave. This brings the two frequencies together, and removes this 'beating' effect. However, if the difference frequency is too high to get through the low pass filter the DBM's output won't be communicated to the VCO. The PLL then can't lock onto the input FM wave and essentially ignores it! As a result, the system can only lock onto an input and demodulate it if the signal frequency is such that $|\Delta f| \leq B$ where B is the bandwidth of the filter. The system is said to have a *lock in range* of $\pm B$ set by the choice of the loop filter. A loop 'free running' at a frequency, f_0, will therefore ignore signals outside the range $f_0 \pm B$.

Consider now what happens when we start with a situation where the system has locked onto a signal whose frequency equals the PLL's unlocked value, f_0 . By definition this is the frequency which the VCO produces when the control voltage, $v(t) = 0$. If we now slowly increase (or decrease) the frequency of the input signal we will produce a corresponding increase (or decrease) in $v(t)$ as the loop makes the VCO output track the signal. However, this process can't go on forever. If we keep on changing the signal frequency one of two things will eventually happen:

- The value of $v(t)$ will reach the 'power rail' voltage driving the amplifier or VCO.
- The phase error, $\phi(t)$ which produces $v(t)$ will reach $\pm 90°$.

Any further increase now *can't* produce a corresponding change in the VCO output. Instead, the loop will 'fall out of lock'. Which of the above possibilities occurs will depend upon the details of how the loop is built. In either case, the effect produces a finite *tracking range*. The loop, once locked, can follow an FM signal over a given tracking range, but will lose lock if the signal moves outside this range. An important feature of PLLs is that the lock-in range and the tracking range are set by different aspects of the system. Hence their values can be designed separately.

15.5 Chapter 15: take-home messages

You should now know that **frequency modulated** (FM) and **phase modulated** (PM) waves are very similar and have many features in common. You should also know what we mean by the *instantaneous* frequency and *instantaneous* phase of FM/PM signals and waves. The spectrum if an FM/PM signal is much more complicated than an AM wave, and that its sidebands can extend over a much wider transmission bandwidth. (In theory, infinite!)

You should also know how we can specify the modulation in terms of a **modulation index**, and a Peak Deviation. That the minimum bandwidth required for transmission can be obtained using **Carson's rule**. You should also understand how FM/PM signals can be created using a voltage controlled oscillator (VCO) to modulate an existing **carrier**.

Chapter 16

Phase sensitive detection

The *phase sensitive detection* (PSD) technique is widely used to deal with the problems caused by $1/f$ noise and unwanted background levels.It is of **immense** practical importance to experimental physics and (the reason we're looking at it at this point of the course) its method of operation is closely linked to demodulation and frequency mixing.

Figure 16.1 represents a typical PSD system, designed to provide a measurement of the signal level produced by a faint source, S, which in this case we've chosen to be a light beam. The technique works by arranging for the signal level to be 'switched on and off' in a controlled way. In the example shown, we're using a device called an optical *chopper* to periodically block and unblock the path of the beam to the detector. The frequency at which this happens will be a function of the number of blades and the wheel rotation rate, which we don't really care about: the main result here is that we've taken a nominally steady (or very slowly varying) signal, and converted it into one at some frequency f.

This means that the chopper is acting as a form of *frequency conversion system*: a steady (or

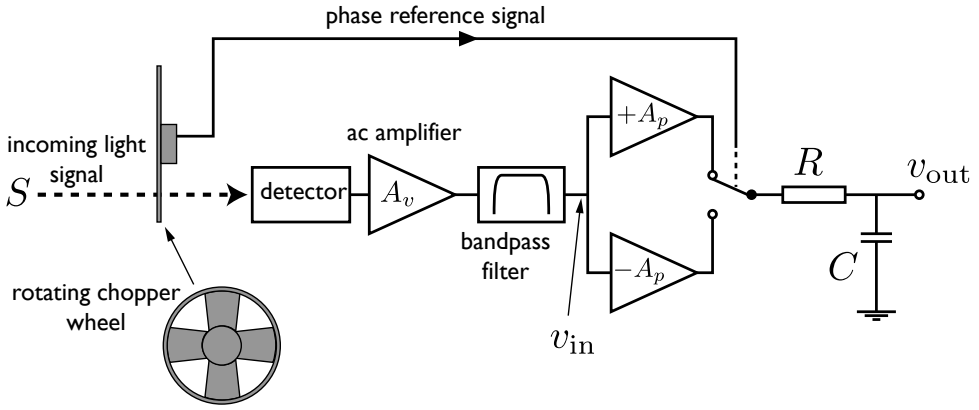


Figure 16.1: A *phase sensitive detection* system.

slowly varying) light power level arriving at the detector now produces (thanks to the chopper) a signal at a higher frequency. The advantage of this is that, thanks to the bandpass filter that follows the ac amplifier in figure 16.1, we can reject frequencies below the chopping frequency. This arrangement makes us a lot less susceptible to the effects of $1/f$ noise. Note that we are talking about frequencies of *fluctuations* in our (for example here, optical) signal, not the frequency of the light itself!

If the signal has a power of S, then when there is no blade in the way of the chopper, the detector output will be

$$V = \alpha S \tag{16.1}$$

where α is the *responsivity* of the detector, (in V/W) which tells us how large an output voltage we get for a given input power. When the input beam is blocked by a chopper blade, then the detector output will be (trivially) zero.

The alternating signal produced by the detector will therefore be a squarewave of magnitude V. We then amplify the signal using an amplifier with gain A_v and pass it through a band-pass filter which rejects and frequencies significantly above or below f. This filtered signal, $v_{\rm in}(t)$ then passes through an arrangement that switches its sign at the same frequency, f that the chopper imposes on the input: we call this the *reference frequency*. In the example of figure 16.1, we apply $v_{\rm in}$ to the input of two amplifiers, with voltage gains $+A_p$ and $-A_p$ respectively.

If the chopper's teeth and gaps cover equal areas larger than the field of view of the detector, then the detector output will look like a square wave. We can represent a square wave of frequency f as a sum of sinewaves at frequencies which are odd multiples of f. If the filter passes signals at f, but completely rejects signals at $3f$ and above, then the signal emerging from the filter will be

$$v_{\rm in} = \frac{2\alpha A_v}{\pi}\sin(2\pi f t) \tag{16.2}$$

The reference signal, produced by the chopper and used to drive the two amplifiers, ($+A_p$ and $-A_p$), is also at frequency f, as it comes from the same source: this means that $v_{\rm in}$ and the reference signal, $v_{\rm ref}$ are *coherently related*, i.e. they have a fixed phase relationship.

We'll begin out analysis of what we'd expect to see out of the system by assuming that $v_{\rm in}$ and $v_{\rm ref}$ are in phase with each other. As the sign of $v_{\rm in}$ is being switched by at the reference frequency, then the voltage at the output of the gain switching stage will be

$$v_p(t) = \left|\frac{2S\alpha A_v A_p}{\pi}\sin(2\pi f t)\right| \tag{16.3}$$

which looks like a *full-wave rectified* sine wave, as can be seen in figure 16.2. The voltage v_p is then passed through a *time constant* (in our example formed by a simple RC low-pass filter). Provided the time constant, $\tau = RC$, of this filter is substantially greater than the signal (reference) period, $T = 1/f$, then it will have the effect of smoothing out the half-cycle

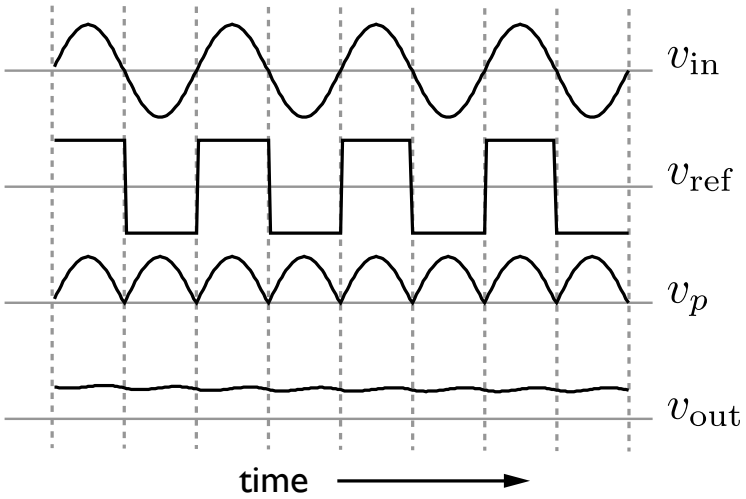


Figure 16.2: The waveforms in a phase sensitive detection system when the signal variation is *in phase* with the reference signal.

fluctuations in v_p to result in an output voltage, $v_{\rm out}$ which will settle at a mean voltage

$$v_{\rm out} = \frac{1}{T}\int_0^T v_p(t)\mathrm{d}t \tag{16.4}$$

It is only necessary to perform the integral over one cycle of v_p as they are all the same. Do note, however, that we only get a steady value of $v_{\rm out}$ if $\tau \gg 1/f$. Putting equation 16.3 into 16.4, we get

$$v_{\rm out} = \left(\frac{2}{\pi}\right)^2 \alpha A_v A_p S \tag{16.5}$$

The time constant performs the function of averaging v_p over a number of cycles to give a nominally steady output voltage, $v_{\rm out}$. Unless τ is infinite, however, then there will always be some *ripple* on $v_{\rm out}$, as shown in figure 16.2.

Provided that we know A_v, A_p and α, then we can determine the light power level, S simply by measuring $v_{\rm out}$ with a dc voltmeter!

We've assumed so far that the chopped signal and the reference signal are *in phase*, but this is not generally always the case (and in fact, when we generalise this discussion in a bit, it's generally *not the case*).

Consider the situation when, for some reason, the signal and reference waveforms differ in phase by some amount ϕ. If we define the time such that $t = 0$ corresponds to a moment when the chopper has just moved out of the detector's filed of view, then we can show that

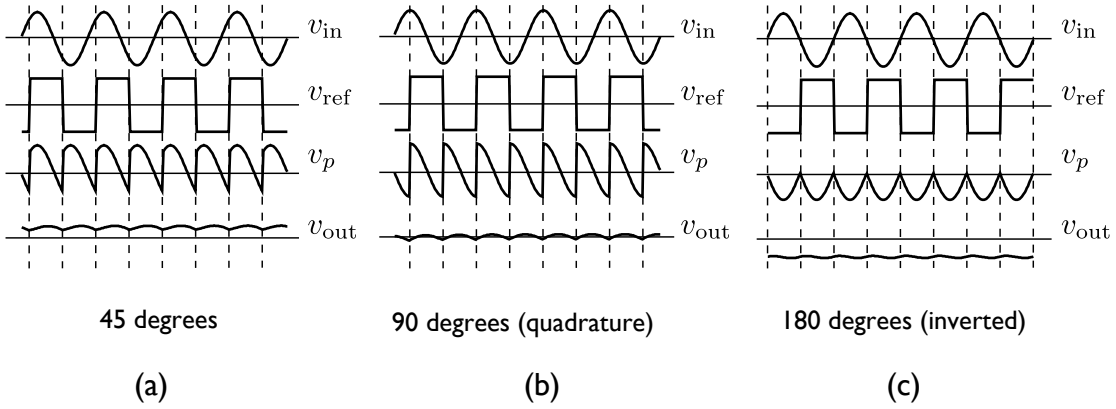


Figure 16.3: The waveforms in a phase sensitive detection system for different values of phase error, ϕ. (a) The input leads the reference by 45°. (b) The input leads the reference by 90°. (c) The input leads (or lags!) the reference by 180°.

the output from the switched-gain circuit, v_p, will be

$$\begin{aligned} v_p(t) =& \frac{2\alpha S A_v A_p}{\pi} \sin(2\pi f t - \phi) \qquad sin(2\pi f t) > 0 \\ =& -\frac{2\alpha S A_v A_p}{\pi} \sin(2\pi f t - \phi) \qquad sin(2\pi f t) \leq 0 \end{aligned} \tag{16.6}$$

and if we use equation 16.4 and slog through all the trig, then we find that our voltmeter at the output of the system will measure

$$v_{\text{out}} = \left(\frac{2}{\pi}\right)^2 \alpha A_v A_p S \cos(\phi) \tag{16.7}$$

i.e. the magnitude of the smoothed output is still proportional to S, but varies in proportion with the cosine of the *phase error*, ϕ. The signal behaviour for different values of phase error is shown in figure 16.3.

The noise produced in the detector, amplifiers etc can be regarded as a spectrum of components at various frequencies. If we observed the noise voltage over some *finite* period of time, then (in accordance to sampling theory arguments from earlier parts of the course) we could describe the noise using the general form

$$v_n = \sum_{i=1}^{N} \Delta V_i \cos(2\pi f_i t + \Phi_i) \tag{16.8}$$

where the values of ΔV_i and Φ_i vary *unpredictably* from one noise observation period to another. (Note: the subscript i in above is just used to denote the different frequency components that may make up the noise, so f_i is the i_{th} frequency component in the noise spectrum, and is not the same as the instantaneous frequency that we were talking about in our discussion of frequency modulation.)

From the statistical properties of noise, we can expect the average value of $(\Delta V)^2$ to depend on the mean noise power level. The phases can take any values. Due to the bandpass filter in the system, we're only concerned with noise components at frequencies similar to the signal chopping frequency f.

Initially, consider the noise component *at* f. We can re-express equation 16.8 as

$$v_n = \sum_{i=1}^{N} [A_i \cos(2\pi f_i t) + B_i \sin(2\pi f_i t)] \tag{16.9}$$

where

$$A_i = \Delta V_i \cos(\Phi_i) \quad \text{and} \quad B_i = \Delta V_i \sin(\Phi_i) \tag{16.10}$$

i.e.

$$v_n^2 = A_i^2 + B_i^2 \tag{16.11}$$

When we are considering the noise at the signal frequency (i.e. $f_i = f$), note that only the in-phase portion of the noise (represented by the cosine term of equation 16.9) will have any effect upon the output, as the quadrature (sine) portion will give no output. Since the noise phase varies at random, we'd expect to find that, on average, $A_i^2 \approx B_i^2$, which means that

$$A_i^2 \approx \frac{(\Delta V_i)^2}{2} \tag{16.12}$$

so only *half* the input noise power has an effect on the output: a phase-sensitive detection system rejects that half of the input noise which is *in quadrature* with the reference. This means that a correctly phase-up system will give us a better SNR than if we'd simply chopped the input signal and measured the chopped signal with an ac volmeter.

We now move on to consider a noise component whose frequency *differs* from the reference (chopping) frequency, i.e. at some frequency $f_i \neq f$. If f_i differs from f by some amount δf, then we can write the noise component as

$$\Delta V_i \cos[2\pi(f + \delta f)t + \Phi_i] \tag{16.13}$$

By using the identity for the cosine of a sum of two angles, we can find that this is equivalent to

$$\Delta V_i [\cos(2\pi f t)\cos(2\pi \delta f t + \Phi_i) + \sin(2\pi f t)\sin(2\pi \delta f t + \Phi_i)] \tag{16.14}$$

Again, the action of the PSD system means we only need to consider the cosine part of the noise component expressed by 16.14. In effect, it is equivalent to an input

$$\Delta V_i' \cos(2\pi f t) \tag{16.15}$$

where

$$\Delta V_i' = \Delta V_i \cos(2\pi \delta f t + \Phi_i) \tag{16.16}$$

i.e. the noise component produces an output which varies sinusoidally at the *difference frequency*, $\delta f = |f_i - f|$. An example is shown in figure 16.4.

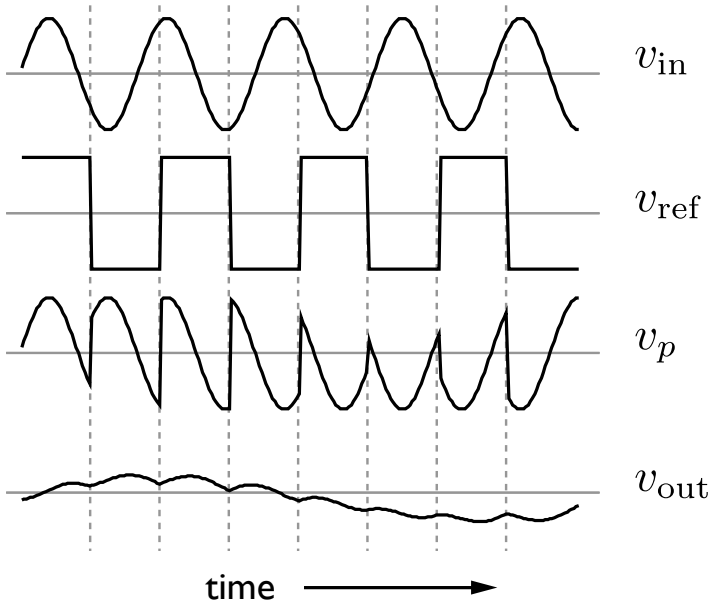


Figure 16.4: The waveforms in a phase sensitive detection system for different signal and reference frequencies. The output oscillates at the difference between the two. In the case shown, the times constant is long compared to both the signal and reference periods, so the difference frequency is passed at the output. If the time constant was large enough, the difference frequency would *also* be suppressed at the output.

(In figure 16.4, the time constant is longer than the periods of both the signal and reference, but much shorter than $1/(|f_{\text{sig}} - f_{\text{ref}}|)$, so the difference frequency is apparent. If the time constant was made long enough, then the difference frequency would also be suppressed.)

Now, the output resistor-capacitor *time constant* acts as a low-pass filter. It will only pass signal or noise fluctuations in the frequency range from dc (i.e. 0 Hz) up to $1/2\pi\tau$ Hz (where, as usual, $\tau = RC$ is the time constant value of the filter). For example, a 1 second time constant (e.g. with $R = 10\ \text{k}\Omega$ and $C = 100\ \mu\text{F}$) will only pass unattenuated frequencies below $\approx 0.16\ \text{Hz}$. This means that the output level will only be affected by signal noise in the frequency range 0.16 Hz either side of the chosen chopping frequency. In effect, the output time constant (in conjunction with the rest of the PSD system) acts just like a very narrow band-pass filter to block noise at frequencies which differ from the signal we're interested in. In theory the same result could be obtained using a very narrow filter. However, it is possible to build filters (or integrators which produce a similar result) with time constants of many seconds. To achieve the same results when using a 1 kHz chopping frequency we'd have to build a bandpass filter with a bandwidth of much less than 1 Hz at 1 kHz. Although not impossible, this would be much harder to make.

Perhaps more significantly, the PSD arrangement doesn't object if the chopping frequency alters, as the gain switching will still be carried out at the chopping frequency. So we can think of the PSD system as acting like a very narrow-band filter that tracks the frequency

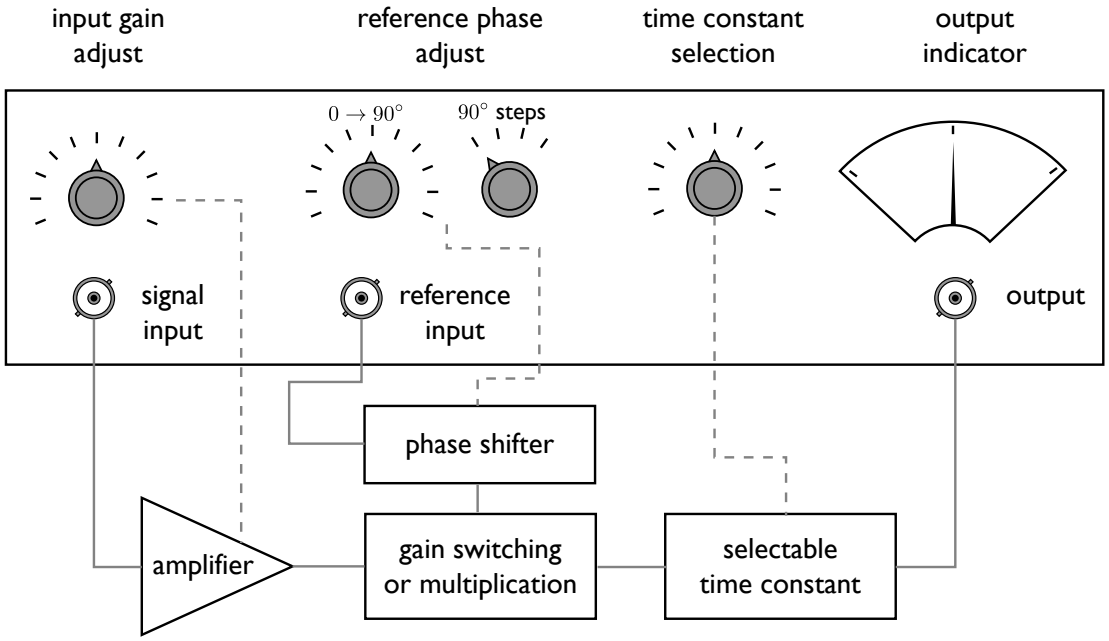


Figure 16.5: A *lock-in amplifier* is a widely used instrument to enable easy implementation of phase sensitive detection.

that we're chopping at. If we'd spent ages building the sharpest filter in the filter in the world, then we'd be pretty annoyed if the chopper motor slowed down (or speeded up) enough to end up outside the pass-band of our filter! Also, the characteristics of any real filter will tend to change with temperature, whether or not it is a Tuesday, etc...

We've shown quite a specific example of a PSD system, where we have a chopper 'hard-wired' into our system. A very widely used piece of laboratory equipment which allows us to use phase sensitive detection for a range of experiments is the *lock-in amplifier* (so called because, due to the PSD operation, it 'locks in' to a signal at a given frequency). A simple cartoon of such a system is shown in figure 16.5.

The lock-in amplifier takes signal and reference input and multiplies them together. The details of how this is done may vary: gain switching as shown in figure 16.1 is one possible technique, or the use of some arrangement of diodes or transistors. Some instruments digitise the input then numerically multiply it by a reference waveform, digitally filter the product and then communicate the result via a display or directly to a controlling computer.

When using a lock-in amplifier to perform phase-sensitive measurements, one must arrange for the signal and reference frequencies to be the same. Chopping a steady signal as in figure 16.1 would achieve this effect, or the signal could result from the modulation of some other quantity by a control signal derived from the reference.

The fact that v_{out} from a PSD system (including one based on a lock-in amplifier) depends on

the phase difference, ϕ between the signal and the reference means that if $v_{\text{out}} = 0$, it is not immediately obvious whether this is because the signal is *actually* equal to zero, or because it is in fact in quadrature with the reference. To remove this ambiguity, most modern lock-in amplifiers actually consist of two systems fed by the same input, but with a 90° difference between the two references. One can then measure the portion of the signal that is in phase with the reference and the portion that is in quadrature with the reference, enabling the magnitude and phase (with respect to the reference waveform) of the signal to be recovered. Even the cheapest of lock-in amplifiers will feature the ability to smoothly vary the reference phase from $0 \rightarrow 90^{\circ}$ as well to to add phase shifts to the reference signal in multiples of 90°.

16.1 Chapter 16: take-home messages

We have seen how **phase sensitive detection** (PSD) systems work. They can be used to avoid $1/f$ noise in detectors and amplifiers and can be used to subtract the effects of a steady background level. You should also understand that the PSD is an example of a heterodyne technique which uses frequency conversion, and appreciate the similarities between the operation of a PSD or lock-in system and multiplication.

Chapter 17

Digital modulation

We'll now take a look at how we can communicate information in *digital* form by modulation of a carrier waveform. Although this process is referred to as *digital modulation* (DM) it generally involves some form of AM and/or FM or PM. Hence digital modulation should be regarded as describing the ways that AM, FM or PM may be used to communicate information in terms of a pattern of 'bits' of information. In principle, the same forms of modulator or demodulator may be employed as when AM, FM or PM are used to send 'analogue' patterns. The use of digital representations of the information means that we now have to deal with a limited set of defined signal amplitudes or phases or frequencies, not the smoothly variable values which arise in an analogue representation.

17.1 Simple binary modulation

The simplest form of DM represents a series of logical 1s and 0s by simply switching on and off the carrier. For this reason it is often called *on-off keying* (OOK). An example of this is shown in figure 17.1. Here a '0' is represented by having the carrier off (i.e. reducing its amplitude to zero), and a '1' by having the carrier on (i.e. giving it a chosen amplitude, A).

In effect, we simply divide the transmitted signal into a series of timed 'chunks' and use these to indicate each bit in turn. The signal pattern during each 'chunk' (i.e. for each *bit*) is called a *symbol* (which shouldn't come as a huge surprise...)

Figure 17.2 shows a small part of the pattern above to make some details more obvious.

When a symbol is non-zero, it consists of an integer number of cycles of the chosen carrier period (we've used four here). If we'd picked a carrier frequency of, say, 20 MHz, then each symbol would have a duration of $0.2\ \mu\mathrm{s}$. The length of each symbol is called the *useful symbol duration*, T_s. In between successive symbols we may also have a short 'spacing' period called the *guard interval*, T_g, which more clearly separates one symbol from the following one.

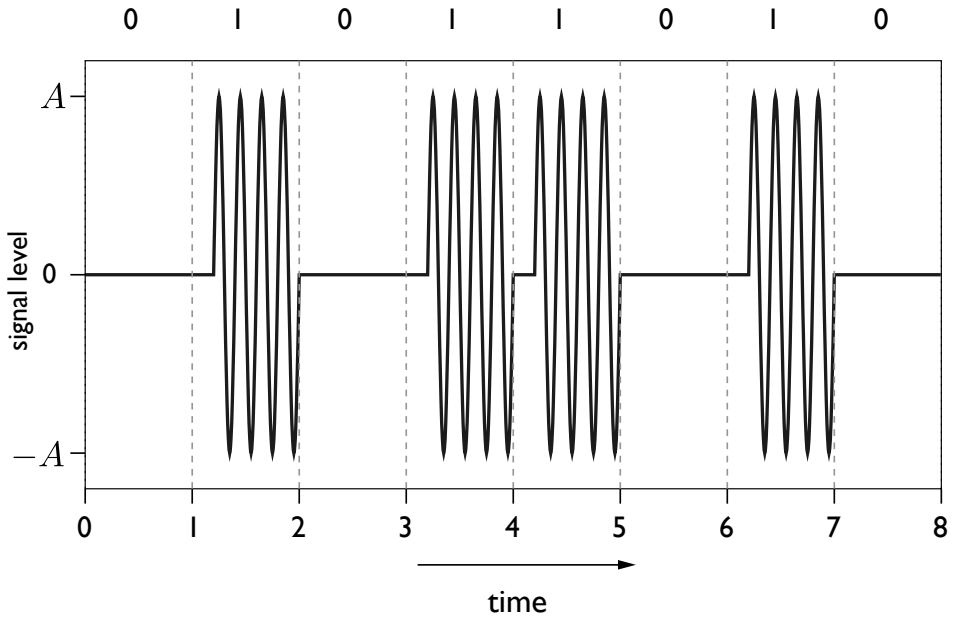


Figure 17.1: Simple *on-off keying* for sending a stream of digital bits.

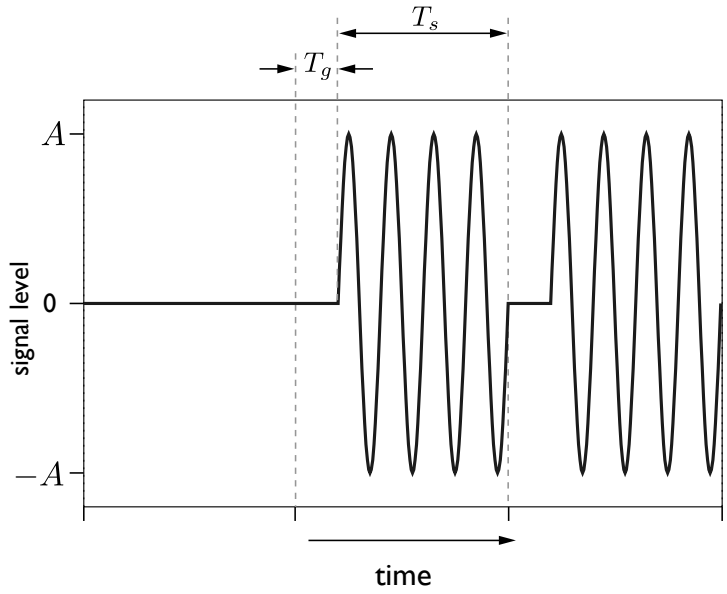


Figure 17.2: Symbol and guard durations for a digitally modulated signal.

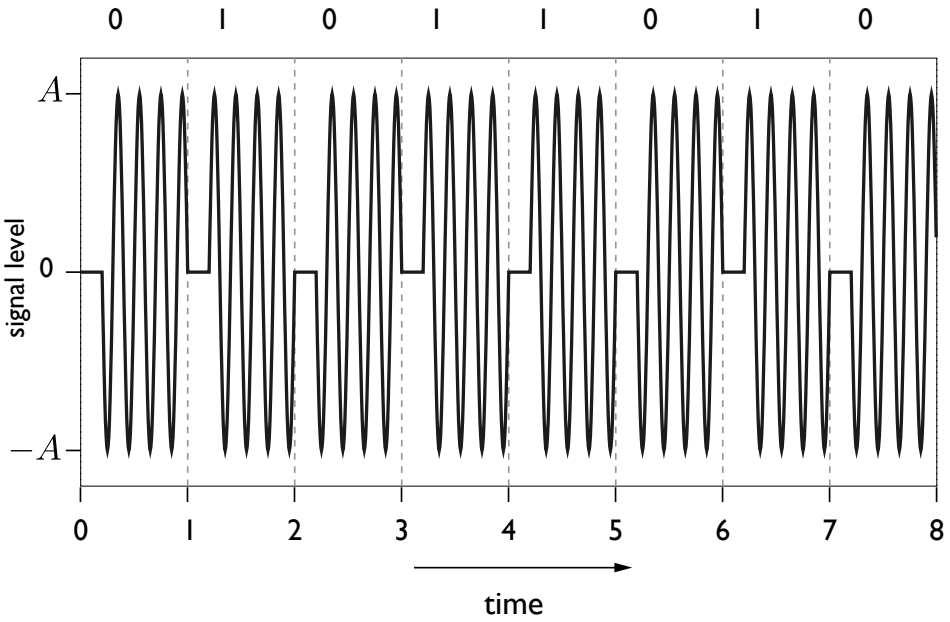


Figure 17.3: *Biphase modulation*, where the different bits (1 or 0) are represented by different carrier phases.

The on-off modulation can be represented by saying that the waveform may be described by

$$S(t) = a\sin(2\pi f_c t + \phi) \tag{17.1}$$

where when we are sending a '1', $a = A$ and $\phi = 0$ and when we are sending a '0', $a = 0$ and $\phi = 0$. (f_c is the carrier frequency.)

A simple alternative to OOK is to have a burst of carrier power during every symbol period, but to change the *phase* of the carrier to distinguish a '1' from a '0', as shown in figure 17.3. This method is called *binary phase shift keying* (BPSK) modulation, and is generally preferred to the simple on/off method. We can represent this in two different ways. We can *either* say that:

- for a '0' we now have $a = A$, $\phi = 0$, for a '1' we have $a = -A$, $\phi = 0$
- for a '0' we now have $a = A$, $\phi = 0$, for a '1' we have $a = A$, $\phi = \pi$

i.e. we can regard the modulation as either being an inversion of the carrier, or a change in its phase. For reasons which should become clear later, it is convenient to regard this as a change in the carrier phase.

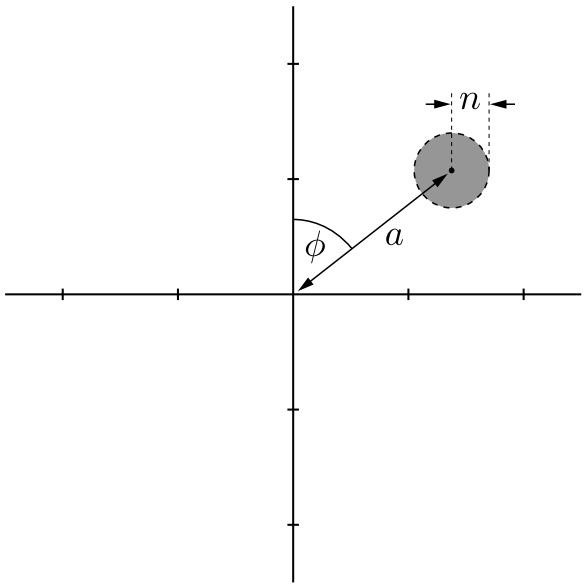


Figure 17.4: The black dot represents the symbol and the grey circle surrounding it represents the (greatly simplified) effects of random noise. The symbol amplitude is given by the radial distance from the origin and the phase by the angle from the vertical axis.

17.2 Power and noise

It is not perhaps immediately obvious why we should prefer BPSK to OOK modulation. Simple on/off modulation can easily be demodulated by a cheap circuit like an envelope detector, but to demodulate BPSK we require the receiver to detect changes in the phase of the signal, not its amplitude. The reasons for the preference can be understood by considering how much *power* we have to transmit to communicate the information patterns.

When the signal is received and presented to a demodulator, the desired signal will always be accompanied by some random noise. The difference between the signal for a 0 and that for a 1 needs to be large enough to be recognisable above this noise, and we can consider this with the aid of a plot of the type shown in figure 17.4. This shows the symbol amplitude, a, and phase, ϕ, as a point on a 2D plot (similar to an Argand diagram, but with the vertical axis defining 0°). If the typical noise level is represented by n, then we draw a circle of radius n around the intended/transmitted symbol position. This grey circle represents the area on the plot where a received symbol is likely to be appear as a result of being 'moved' by the noise. Note that the effects of real random noise are more complex than this very simple description!

We can now use this type of plot to compare the merits of OOK with BPSK. Figure 17.5 (a) shows the two symbols used in the simple on/off modulation example. The required separation of the symbols is indicated by A_r and in order to avoid the incorrect interpretation of a symbol, we need to ensure that $A_r > 2n$. For on/off modulation, this means that one type of symbol

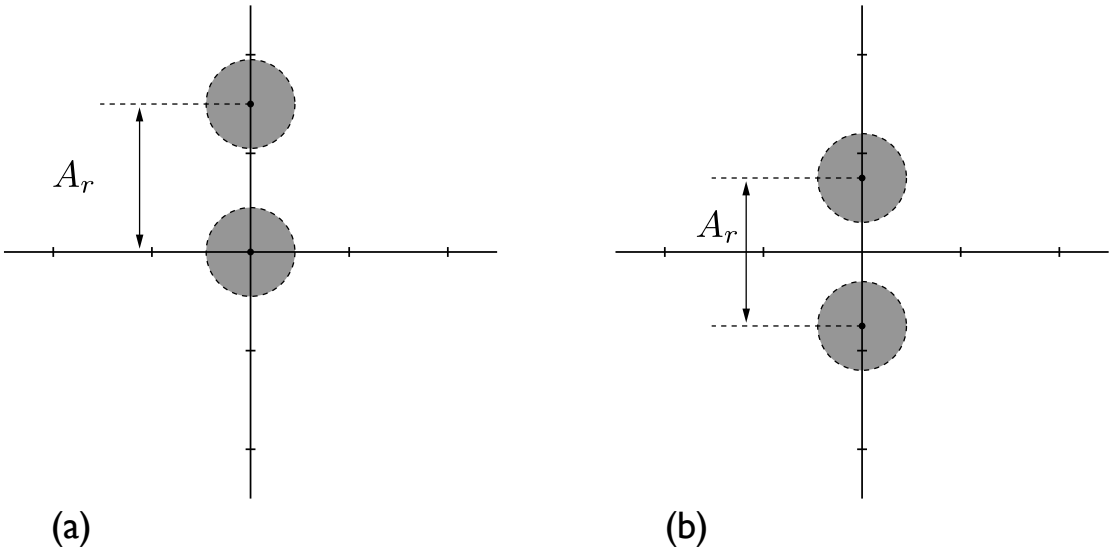


Figure 17.5: Comparison of simple on/off modulation with *biphase* modulation in terms of phase-amplitude plots.

has an amplitude of zero, and the other type of symbol has an amplitude of A_r.

For the bi-phase modulation case, however, we can use the same *magnitude* of amplitude for *both* types of symbol as shown in figure 17.5 (b). We still need to ensure that the symbols are separated by A_r, but we can do this by having one symbol type with amplitude $A_r/2$ and the other type with amplitude $-A_r/2$ (or the same amplitude, but inverted in phase.) We can, therefore, that BPSK requires only half of the signal amplitude that is required by OOK to avoid the results being upset by noise.

So far so good, but why does this matter? If we consider a sequence of bits containing roughly equal numbers of '1's and '0's. For the OOK case, if we ignore the guard duration, then this means that roughly half the time, the signal *power* will be $\propto A_r^2$, so the time averaged signal power will be $\propto A_r^2/2$.

For the bi-phase modulation case, however, the power for every symbol is $\propto (A_r/2)^2$, i.e. the mean required power level is $\propto A_r^2/4$. Both systems will carry signals with the same level of reliability, but the bi-phase system requires only half the power. If we use the same power level as for the OOK case, then BPSK will provide a higher level of protection against the effects of noise. The drawback is that we require more sophisticated receivers as they have to be able to recognise the phases of the received signals.

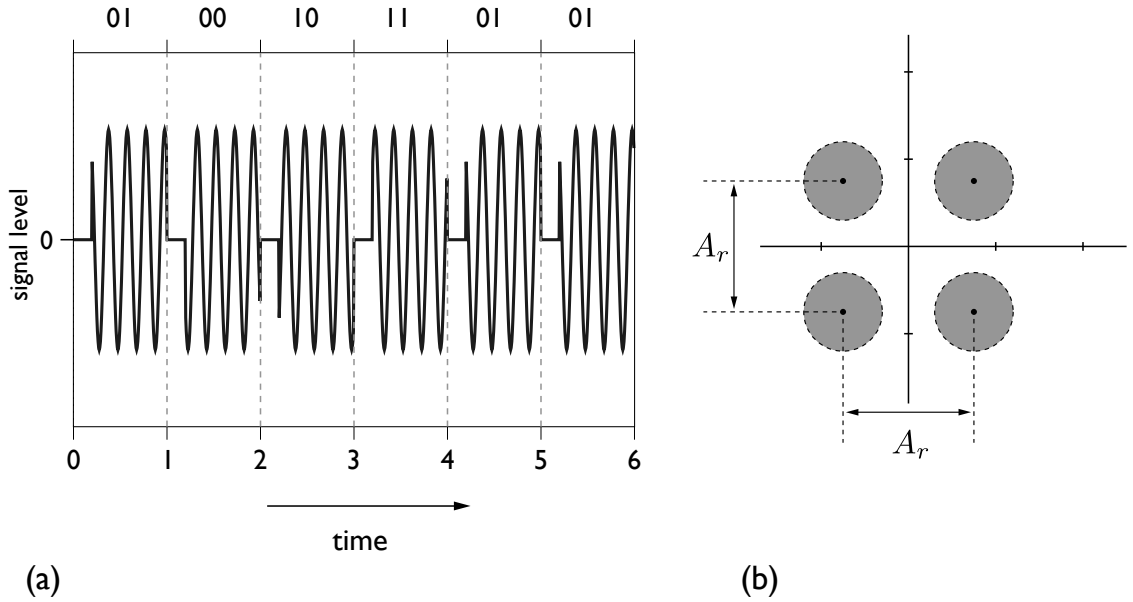


Figure 17.6: With *quadrature phase shift keying*, we have a choice of four symbols, so each symbol can represent *two* bits.

17.3 Multi-bit symbols

Given that we can modulate the carrier phase we can now consider using even more complicated forms of modulation by allowing the system to use more than two possible carrier phases for distinct symbols, and also more than one choice of carrier amplitude.

Figure 17.6 shows the use of *quadrature phase shift keying* (QPSK) modulation. This is similar to BPSK in that all the symbols have the same amplitude, but in this case we have *four* carrier phases to choose between. Since we now have four possible symbols we can use them to represent more than one bit per symbol. In effect, we can now communicate more than one bit at a time. Each possible QPSK symbol can be 'labelled' and used to represent *two* bits.

Looking at the example of QPSK illustrated in figure 17.6 we can see that the four distinct symbol patterns can be spaced an amplitude, A_r, apart, but since we are now doing this in two dimensions the actual amplitude of the carrier will be $A_r/\sqrt{2}$ in each case. The carrier phase can be any one of the four values, $\phi = \pi/4$, $3\pi/4$, $5\pi/4$ or $7\pi/4$.

An alternative way to describe this is to define the signal in the form

$$S(t) = a\sin(2\pi f_c t) + b\cos(2\pi f_c t) \tag{17.2}$$

where a and b can each take values of $\pm A_r/2$.

Note that QPSK provides twice the information per symbol that BPSK provides, but that it requires the signal amplitude, $\sqrt{a^2+b^2}$, to increase from $A_r/2$ to $Ar/\sqrt{2}$, so the signal

carrier power for QPSK must be twice that of BPSK for the same *symbol rate*. Thus we pay for the doubling in information capacity by having to provide double the power.

Having discovered that we can use a symbol to convey more than one bit at a time, we can extend this by allowing a choice of *amplitudes* as well as of phases. A common example of this is *quadrature amplitude modulation* (QAM). (Are these acronyms doing your head in yet? Because they're doing mine in...) This comes in various forms, and a typical example would be called something like '16QAM' or '64QAM' where the number indicates how many distinct symbols are available. Figure 17.7 shows an example. Note that it is common to describe the array of symbols displayed in this way as a *constellation* of symbol values.

For simplicity, we'll consider a square array of symbol locations, so the number of available symbols will be the square of an integer.

The example shown in figure 17.7 is 16QAM and hence has $N = 4^2 = 16$ symbols. For such an array the amplitude of the locations which are furthest from the centre (i.e. the symbols which require the maximum symbol amplitude) will be $A_r\left(\sqrt{N}-1\right)/2$. The carrier power required to transmit these highest-amplitude symbols will be

$$P_{\max} \propto A_r^2\left(\sqrt{(}N)-1\right)^2 \tag{17.3}$$

and when N is reasonably large this will approximate to

$$P_{\max} \propto N \tag{17.4}$$

However, the number of bits per symbol, m, varies with the number of symbols, N, according to

$$N = 2^m \tag{17.5}$$

so we can expect that the peak required carrier power will approximately tend to vary with the number of bits per symbol as

$$P \propto 2^m \tag{17.6}$$

The peak powers required for QAM tend to rise steeply as we wish to convey more bits per symbol. Hence in practice we may will to avoid choosing too high a value for m (and hence for N).

17.4 Orthogonal multiplexing

In order to be able to communicate many bits per symbol interval we can decide to use many carrier frequencies together: in effect, using many simultaneous transmissions, each providing a modest information rate, which combine to deliver a higher data rate. A group of modulated carriers used together in this way is called an *ensemble*. If we do this there is a neat trick which can make the approach particularly efficient in terms of choosing the carrier frequencies. To understand this, consider the frequency spectrum of a typical symbol.

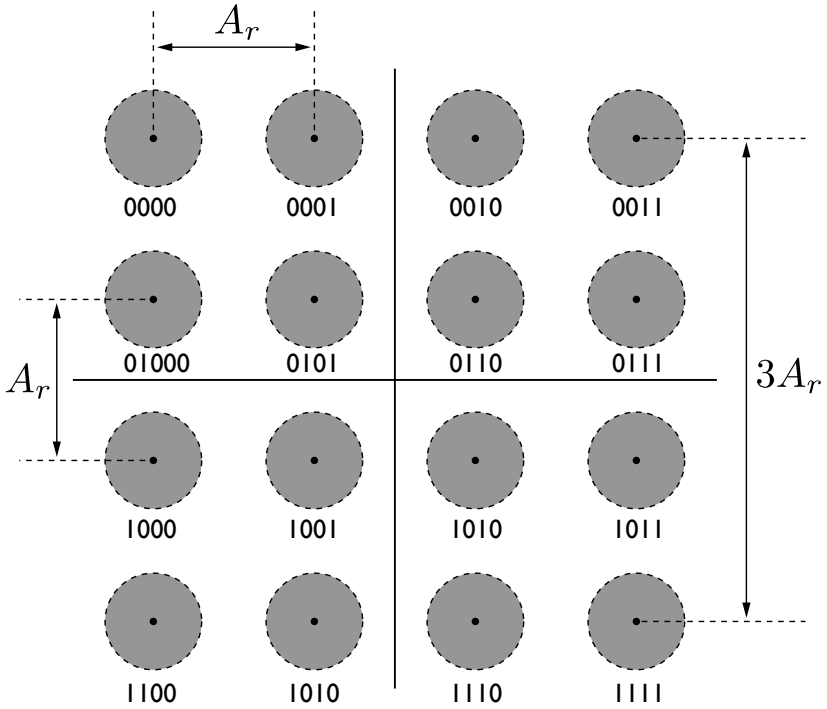


Figure 17.7: An example of *quadrature amplitude modulation*, in this case 16QAM. The labels which assign bit combinations to the different symbols are entirely down to us.

For the sake of a convenient example, we can choose one with 32 carrier cycles during the chosen symbol interval. If we record the signal over just the duration of the symbol interval and work out its power-frequency spectrum the results will be as shown in figure 17.8 by the solid line.

If we'd asked for the spectrum of a continuous sinusoid which extends over a very long (i.e. essentially infinite) time interval, then the spectrum would have only had one non-zero component, at the appropriate frequency. From our earlier exposure to information theory we can expect that limiting the observed waveform to a finite interval, T_S, causes the spectrum to broaden. The result is a sinc-squared pattern as shown in figure 17.8. The spectrum has a peak at the expected (carrier) frequency of 32 cycles in the interval. However note that it also has zeros or *nulls* at other frequencies corresponding to integer numbers of cycles in the interval which are $\neq 32$. This behaviour arises from the properties of *orthogonal functions*, that we looked at when discussing the Fourier methods that led to our acceptance of the sampling theorem.

The reason that we get a sinc function for the spectrum is that taking the Fourier transform of a rectangular (often called "top hat") function gives a sinc function.

The practical result is as illustrated in figure 17.8 which superimposes the power spectra for two different choices of possible carrier frequency, each choice having an integer number of cycles in the chosen symbol interval. The dashed line shows the spectrum for a symbol whose

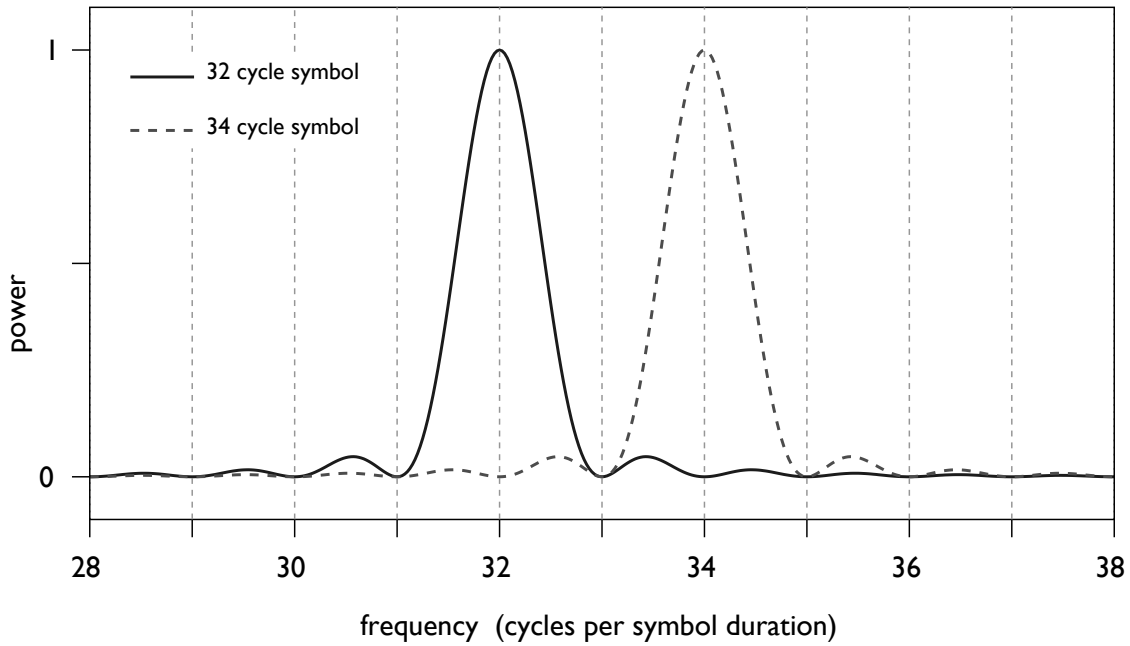


Figure 17.8: In *orthogonal frequency division multiplexing* (ODFM), the different integer numbers of carrier cycles per (fixed) symbol period for different channels ensures that the spectra for each channel do not interfere with each other when the signal is decoded.

carrier frequency is 34 cycles per symbol duration and the figure shows that it has zeroes at all the frequencies which have an integer number of cycles during the symbol period *except* 34.

If we now use both carriers simultaneously as parallel QAM streams we find in our receiver/demodulator that a circuit designed to measure the signal amplitude and phase at one frequency, f_k , will be unaffected by the presence of another carrier at f_q since this produces no contribution at f_k. This result is extremely important: it means that we can choose to simultaneously transmit a whole series of carriers and modulate each of them with their own information patterns. By choosing an *orthogonal* set of carrier frequencies we can prevent them from interfering with one another, and hence recover all the information they carry. The requirement we must obey for this to be possible is that we choose a finite symbol duration, T_S, and then select all the carrier frequencies such that they have different integer number of cycles in this time period.

This means we can transmit, say, 100 carriers, and modulate them to send 100 times more bits in a given time than we could using just one modulated carrier. Techniques of this kind are called *orthogonal frequency division multiplexing* (OFDM). More commonly you will see this referred to as COFDM or *coded orthogonal frequency division multiplexing* which refers to the way the system is modified and used in practice. COFDM is the basic modulation and communication method now used for digital TV and radio in the UK.

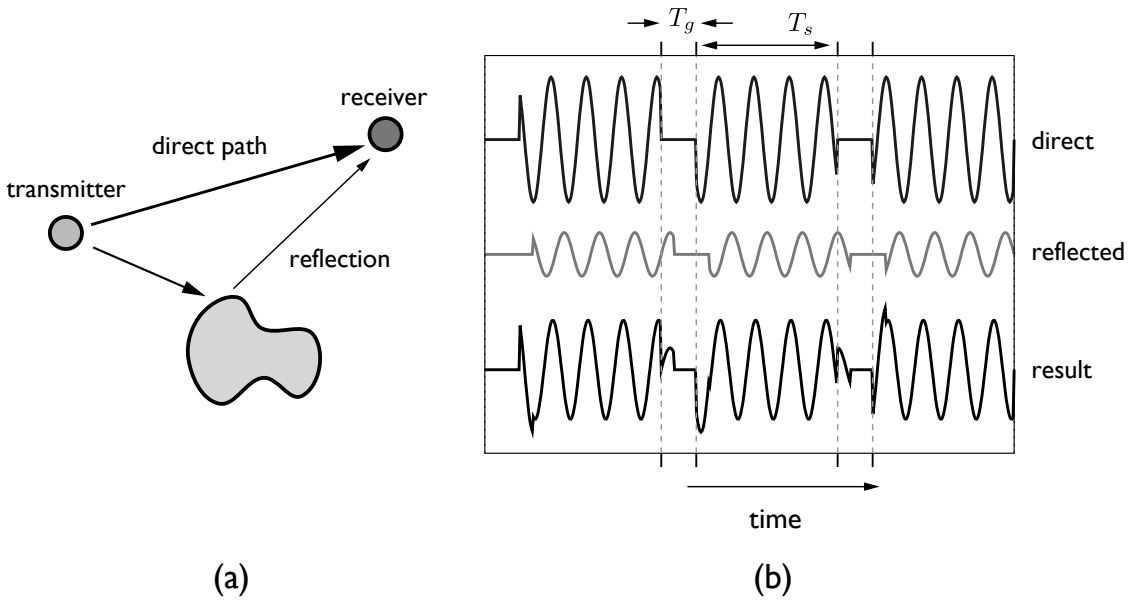


Figure 17.9: (a) The problem known as *multipath* arises when a transmitted signal reaches the receiver by more than one route. (b) The effects of multipath on a received digitally modulated signal for the case where there is only one extra path.

17.5 Multipath

A common problem in radio communications is **multipath**. As the name indicates, this is the situation where signals may travel from the transmitter to the receiver by more than one path. For example, we may get one signal path in a straight line from the transmitter to receiver, and another via a reflection from a hill or tall building. When these paths have different lengths their contributions to the received results arrive with different propagation delays. This is shown in figure 17.9. Note that in practice, there may be many such paths, giving contributions arriving with various time delays and amplitudes. Here we just consider one delayed contribution for the sake of simplicity.

From figure 17.9 we can see that the addition of the delayed contribution via the 'reflected' path has two consequences:

- The symbols have been extended and now have a 'tail' due to the reflected contribution continuing to arrive for a short period after the direct symbol has ended.
- The received result has its effective amplitude and phase altered by an amount that depends on the time difference between the paths.

The two paths are of lengths we can call Z (direct) and Z' (indirect). This allows us to

define the time delay between their relative arrivals to be

$$\delta t = \frac{Z' - Z}{c} \tag{17.7}$$

where c is the speed of light. The above shows why it is useful to have a guard Interval, T_G, in between the useful symbol periods, T_S. Provided that we have arranged for $T_G > \delta t$ then the delayed version of each symbol will have finished arriving before the next symbol in the sequence starts to arrive. This prevents the pattern transmitted for a given symbol from affecting what we see during the *next* symbol period. Where one symbol pattern interferes with, or unintentionally alters, the pattern we wish to send for another is called inter symbol interference (although note that it represents a different idea to the inter-symbol influence we encountered when considering symbol probabilities earlier in the course). In general we wish to avoid this as it may make it harder for the receiver to demodulate the symbols correctly.

The chosen value for the guard interval is thus important for avoiding data loss or errors due to multipath. We wish to choose a value for T_G that is large enough to ensure that no reflections will be arriving with delays greater than T_G, or, at least, we wish to ensure that any that do arrive later than this will have such low amplitudes as to have almost no effect on the amplitude and phase of the following symbol pattern.

A long guard interval can avoid one symbol significantly affecting the next, but the multipath may still alter the amplitudes and phases of the received symbols. To help deal with this we can insert a regular pattern of what are called *pilot* or *reference* symbols into the transmitted stream. These also allow the transmitter to inform the receiver what phase/amplitude corresponds to a given symbol. In effect, these pilot symbols make it possible to 'calibrate' the receiver and help correct for the effects of multipath.

17.6 Chapter 17: take-home messages

We have looked at how digital modulation can be used to convey information for a stream of bits in terms of a set of predefined symbols. The ideas here should fit very naturally with the discussion about data transmission in the earlier parts of the course. The number of bits which each symbol can convey depends on the number of different symbols available in the chosen modulation scheme, in terms of their distinct amplitudes and phases. It should also be clear that these have to differ by enough to be distinguishable over the noise which may be present, and that the power required increases as we want more bits per symbol.

You should also now understand that we can use **orthogonal** sets of carriers to provide parallel streams which can be modulated and detected without interfering with one another, thus increasing the total rate at which information can be conveyed.

We've also looked at the problem of **multipath**, which we can ameliorate by employing gurd times around our symbols.

Chapter 18

Antennae, link-gain and radar

So far, we've discussed what we mean by information and quantified the amount of information in a signal in a few particular cases. We've gone on to look at how we can get information onto a carrier signal, and how we can extract it from a carrier signal. Now it's worth a look at how we get these signals from A to B through free space. We'll take a look at antennae and some associated applications, such as radar.

18.1 Basic properties of coherent antennae

An *antenna* is a device which allows us to collect or radiate electromagnetic fields. Here we'll consider the general properties of antennas and later on will consider specific types of antenna and how they can be used to transmit patterns of electromagnetic energy from place to place.

In figure 18.1 the output from a coherent source (e.g. an oscillator) is directed out into free space using an antenna. The signal source is linked to the antenna by some kind of waveguide (microwave guide, light fibre, pair of wires or whatever). The antenna acts as a sort of transformer. It takes the electromagnetic field pattern, Ψ_g, moving along the guide and transforms it into some other pattern, Ψ_s , which is radiated out into free space.

With this simple picture we can establish two basic properties of any antenna:

- The antenna doesn't itself generate any power. So, unless the antenna is imperfect and dissipates some power, the total powers carried by the guide and free space fields must be the same. (In reality, all practical antennas tend to be slightly resistive so some power is normally lost during the process of turning Ψ_g into Ψ_s, but for now we can assume any loss is small enough to ignore.)
- The antenna is a *reciprocal device*: it behaves in the same way irrespective of which

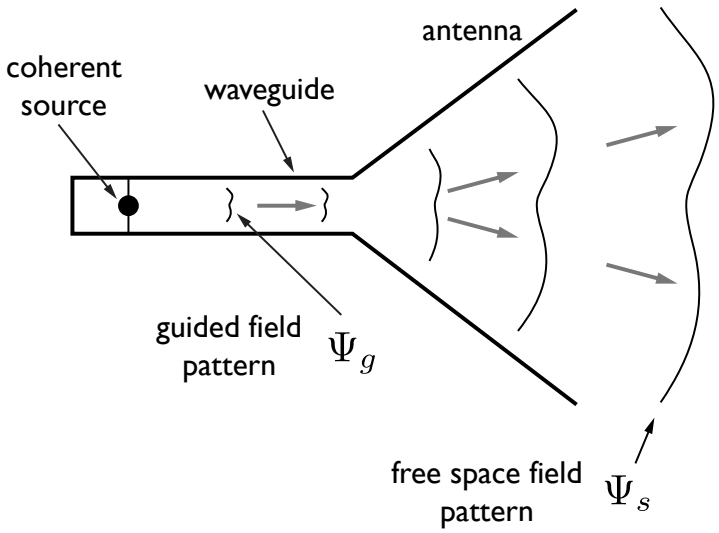


Figure 18.1: A simple schematic of an antenna system.

way we pass signal power through it. Imagine making a video showing how the field Ψ_g flows into the system and is transformed into Ψ_s . If we were to replay the video backwards the field Ψ_s would appear to enter from space, move through the antenna, and be transformed into a guide field Ψ_g heading towards the 'source'. For an antenna dealing with a coherent field we can't tell by looking at what happens which way we're playing the video unless we independently know something extra, e.g. that the device connected to the waveguide is a source, not a detector.

This reciprocal behaviour is a useful feature of a coherent antenna. It means that, in principle, the only real difference between a 'transmitting' ($\Psi_g \rightarrow \Psi_s$) and a 'receiving' ($\Psi_s \rightarrow \Psi_g$) antenna is the direction we've chosen to pass signals through it. A practical consequence is that we can often use identical antennas for transmitting and receiving signals through space. (Indeed, many systems use the same antenna for both transmitting and receiving signals). This reciprocal property also turns out to make analysing and understanding antennas a bit easier.

Theoretically one of the simplest types of antenna is a **Hertzian dipole** of the kind illustrated in figure 18.2. Although this sort of antenna isn't used much at high frequencies it is a good place to start because its properties are easy to analyse. The dipole consists of a straight piece of wire of length L with a small break at its centre. A pair of wires are then used to connect the break to a generator which can produce a current

$$i(t) = I_0 \sin(2\pi f t) \tag{18.1}$$

For the sake of simplicity we can assume that $L \ll c/f$, i.e. the dipole is very short compared with the wavelength of the radiation we're attempting to radiate. This means that we can

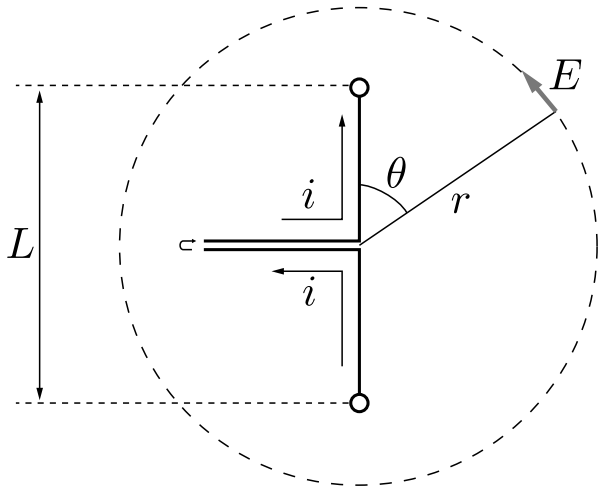


Figure 18.2: A *Hertzian dipole* antenna.

neglect the time it takes for any current to flow along the dipole. We can therefore assume that the current equals $i(t)$ everywhere along the dipole. In reality, of course, the current would have to "come from and go somewhere" at the wire ends since charge can't appear and vanish into/out of nothing. We can imagine two spheres or discs placed at the ends of the wires to act as charge reservoirs. (In a real antenna the currents on these would affect the antenna's behaviour, but we'll ignore that fact here to avoid complications!)

By reference to a suitable book on electromagnetics or antennas we can discover that a wire of length L carrying a uniform current oscillation of amplitude I_0 will radiate an oscillating electric field

$$E(t, r, \theta) = \frac{1}{\lambda r} 60\pi L I_0 \sin(\theta) \cos\left[2\pi\left(ft - \frac{r}{\lambda}\right)\right] \tag{18.2}$$

where r is the radial distance from the centre of the antenna to the position where we wish to determine the electric field, θ is the angle between the dipole wire and the line connecting the point and the centre of the dipole and $\lambda = c/f$ is the free space wavelength of the radiation. (This result is only true for the *far field* where $r \gg \lambda$). The behaviour of the field pattern nearer to the dipole is more complex.) The rms field at this position will therefore be

$$E(r, \theta) = \frac{60\pi L I_0 \sin(\theta)}{\sqrt{2}\lambda r} \tag{18.3}$$

If we imagine the dipole being located at the centre of a sphere of radius r then the *power density* (or 'flux'), $P(\theta)$, flowing through a unit area on the sphere's surface normal to the radial direction will be

$$P(\theta) = \frac{E^2(r, \theta)}{Z_0} \tag{18.4}$$

where Z_0 is the impedance of free space. Therefore we can say that for a Hertzian dipole,

$$P(\theta) = \frac{(60\pi)^2 L^2 I_0^2 \sin^2(\theta)}{2 Z_0 \lambda^2 r^2} \tag{18.5}$$

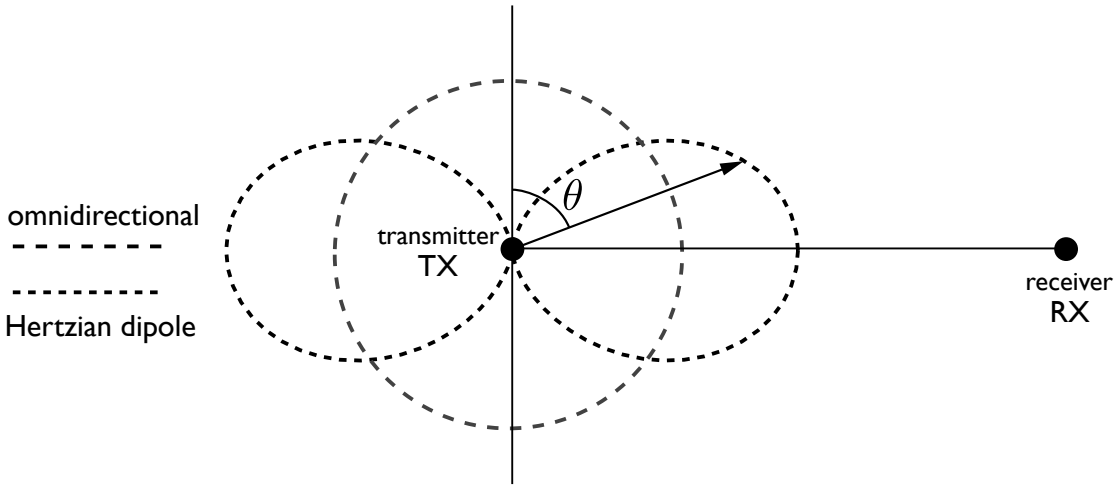


Figure 18.3: A comparision of the antenna power patterns for an *omnidirectional* antenna and a *Hertzian dipole*.

The way in which the radiated power density varies with direction θ is called the *antenna power pattern*. Some books also call this the antenna pattern, however this term is also used to indicate how $E(\theta)$ varies with direction, so we'll stick to the term power pattern to make things clearer. The antenna power pattern for the Hertzian dipole is shown in figure 18.3. Also shown is that for an *omnidirectional* antenna, i.e. one that radiates power uniformly in all directions.

Usually, an antenna will produce a direction dependent power pattern which can be specified in a spherical co-ordinate system as $P(\theta, \phi)$ in terms of two direction angles, θ and ϕ. However, a simple dipole has rotational symmetry about its dipole line, hence the power it radiates doesn't depend on the "round the dipole" angle, ϕ.

18.1.1 Gain and directionality

A Hertzian dipole produces a *directional* power pattern: it radiates more power in some directions than others. Consider a pair of antennas, each radiating the same total power from a transmitter or source, TX. Figure 18.3 compares the Hertzian dipole's power pattern with that produced by an isotropic radiator (sometimes also called an omnidirectional antenna) which radiates the same power in all directions. The dipole's power pattern means that it tends to radiate less than the isotropic radiator in directions around $\theta \approx 0$ and $\theta \approx \pi$. Since it radiates the same power overall, this means that it radiates more than the isotropic radiator in some other directions (those around $\theta \approx \pi/2$ and $\theta \approx 3\pi/2$).

A distant receiver or observer, "RX", in the $\theta \approx \pi/2$ direction will therefore pick up more radiated power from a transmitter dipole than an isotropic radiator if both radiate the same

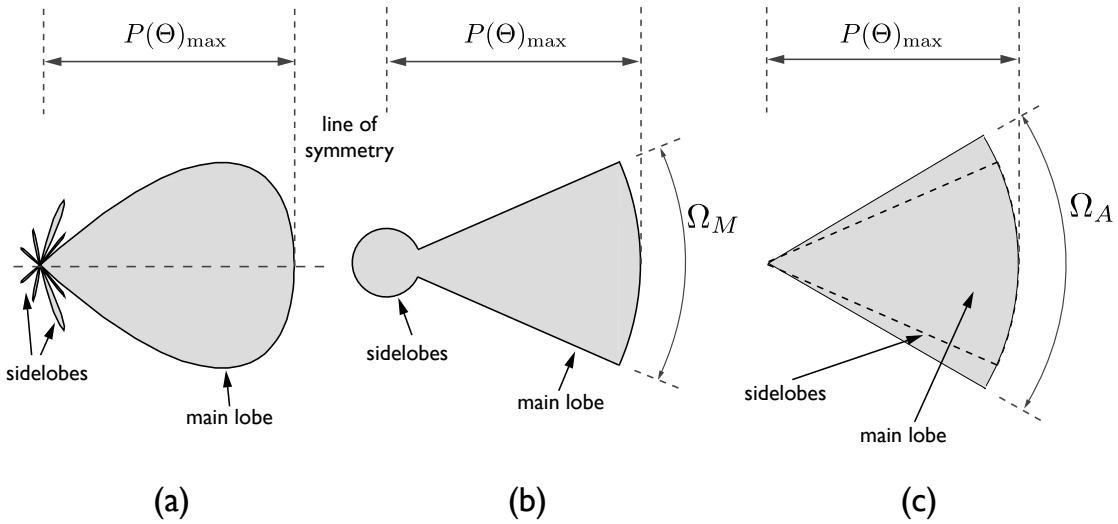


Figure 18.4: Representing the antenna power pattern. (a) A possible real antenna power pattern showing the main lobes and sidelobes. (b) and (c): Simplified representations of the power pattern.

total power. So far as the observer is concerned, using a dipole has the same effect as if the transmitter power had been boosted using an amplifier whose power gain was around $\times 1.5$. Antenna designers call this effect **antenna gain** and it is usually specified in decibels. The Hertzian dipole is said to have a gain of around 1.75 $\mathrm{dB_i}$ where the subscript "i" means "compared with an isotropic radiator". Note that, as with a transformer, this isn't a 'real' gain, we aren't getting more power out of the antenna that is being put into it. The dipole is simply redirecting the power it radiates, sending more in some directions at the expense of others. However, the observer "RX" doesn't know that.

In practice the gain over an omnidirectional antenna or isotropic radiator can't be measured since it is impossible to make a coherent isotropic radiator. Antenna gains are therefore measured by comparing the antenna's power pattern with that of a dipole $\lambda/2$ long: a **halfwave dipole**. This kind of antenna is quite easy to make, and it works quite well. Theoretical calculations show that the halfwave dipole has a shape similar to the Hertzian dipole, but with a gain over an isotropic radiator in the $\theta = \pi/2$ direction of 2.15 $\mathrm{dB_i}$. The measured gains of other antennas are therefore ofter specified in $\mathrm{dB_d}$where the subscript "d" means "compared with a halfwave dipole". This means that the figures quoted in these two ways can be related using the formula, $\mathrm{dB_i} = \mathrm{dB_d} + 2.15$.

We can easily make antennas which are more directional than a simple dipole. This 'squeezes' the radiated power into a smaller range of angles, hence increasing the gain and power density along the directions where $P(\theta, \phi)$ is highest. The details of a real antenna's power pattern can be quite complex and will depend on how we've made the antenna. Figure 18.4 shows how we can represent a typical antenna pattern. Here the real pattern is a 'solid shape' which illustrates the relative power densities radiated in each direction. The total power radiated in all directions will be proportional to the volume of this shape.

We can locate the direction of highest power density and call it the *axis* of the antenna. (It is also sometimes called the antenna's *boresight*). For the sake of simplicity we'll assume that the patterns we're considering have rotational symmetry about this axis. Hence we can ignore one angle, ϕ, and specify the power pattern, $P(\Theta)$, in terms of one angle. The power density radiated along the axis will be, from the above definition of the axis, be the peak level, $P(\Theta)_{\max}$.

The real pattern of most practical antennas can be divided into one *main lobe* which includes the axis, and a set of *sidelobes* pointing in other directions. The on-axis *gain* of the antenna can now be defined to be

$$G = \frac{P(\Theta)_{\max}}{P_{\text{isotropic}}} \tag{18.6}$$

where $P_{\text{isotropic}}$ is the power density which would be radiated uniformly in all directions by an isotropic radiator fed with the same total power.

The total power radiated into a given solid angle is proportional to the part of the shape's volume which fits inside that angle. We can simplify the main lobe shape by readjusting its volume into an 'idealised' main lobe which has a uniform power density equal to $P(\Theta)_{\max}$ over some solid angle, Ω_M. The angle of this 'smoothed out' main lobe is called the antenna's *main lobe angle*. In figure 18.4 (b) the remaining sidelobe power is smoothed out to a uniform power density in all directions *except* the idealised main lobe.

This simplified picture actually turns out to be good enough to let us describe most of the main properties of a real antenna, despite having discarded almost all the details of the real pattern. We can, in fact, take this process one stage further and redistribute the sidelobe volume/power 'around the sides' of the main lobe. This produces the pattern shown in figure 18.4 (c). Here the antenna power pattern is assumed to have a uniform power density equal to $P(\Theta)_{\max}$ over a solid angle, Ω_A , and that none of the power is radiated outside this solid angle. The angle, Ω_A, is called the **antenna angle** of the antenna. If we compare this idealised pattern to a sphere containing the same volume (i.e. the power pattern of an isotropic radiator emitting the same total power) we can show that

$$G = \frac{P(\Theta)_{\max}}{P_{\text{isotropic}}} = \frac{4\pi}{\Omega_A} \tag{18.7}$$

This result is key as it tells us that the larger the gain of the antenna, then the more directional it's behaviour will be. If we want to send a signal from one place to another, well defined location, then a higher gain antenna will give us a narrower beam, which allows us to use less power to transmit the signal (and may make it harder to eavesdrop on). On the other hand, radio and TV transmitters want to be able to cover a very large area, so a very directional beam may not be so useful.

Up to now we've concentrated on considering the antenna's behaviour when transmitting power. However, from the reciprocal property mentioned earlier we can expect the same antenna, when used to receive signals, to collect G times as much power as we'd get using an antenna whose sensitivity is omnidirectional. The gain value is therefore appropriate

whether we're using the antenna to transmit *or* receive. An alternative way to describe the behaviour of a receiving antenna is in terms of the power, W, it picks up when placed in a plane uniform field of power density, E^2/Z_0, where E is the rms electric field and Z_0 is the impedance of free space. We can link the power density falling on the antenna and the power it collects by an expression of the form

$$W = A_e \frac{E^2}{Z_0} \tag{18.8}$$

This expression can be used to define the antenna's *effective area*, A_e. For some high frequency and 'optical' antennas A_e is more or less equal to the actual area of the antenna which the incident field is striking. Other antennae, for example a dipole, don't have a physical area we can identify in this way. Although we won't prove it here, it can be shown that the effective area and beam angle of an antenna can be linked by the expression

$$A_e \Omega_A = \lambda^2 \tag{18.9}$$

where λ is the radiation's free space wavelength.

Combining equations 18.7 and 18.9, we can therefore say that

$$G = \frac{4\pi A_e}{\lambda^2} \tag{18.10}$$

The above expressions mean that, once we know the radiation wavelength, λ, we only need to know one of the three values, G, Ω_A or A_e, and we can work out the others. In effect, these three quantities are three ways to tell us the same thing about an antenna.

18.1.2 Radiation resistance

Up until now we've treated the the antenna as a sort of lossless power transformer. To complete our understanding of antenna basics we need to consider the electrical properties of an antenna. Figure 18.5 (a) shows a simple transmitter system. Power from a source is sent along some form of *transmission line* (i.e. a waveguide, pair of wires, light fibre, or whatever is a convenient way to carry the signal from place to place) to the antenna. The antenna then radiates power in a directional manner we can describe using its power pattern.

So far as the signal source or generator is concerned, the system looks like the circuit shown in figure 18.5 (b). The generator doesn't know anything about the details of the antenna or its power pattern. So far as it is concerned, the antenna/transmission line combination just looks like some sort of load impedance.

An ideal antenna would simply accept all the power sent to it from a source and radiate it away into space. So far as the *generator* is concerned, this behaviour is indistinguishable from what happens if it's output is connected to a load impedance which *matches* its own output impedance, Z_g. A real antenna won't radiate all the power it receives. Some will be

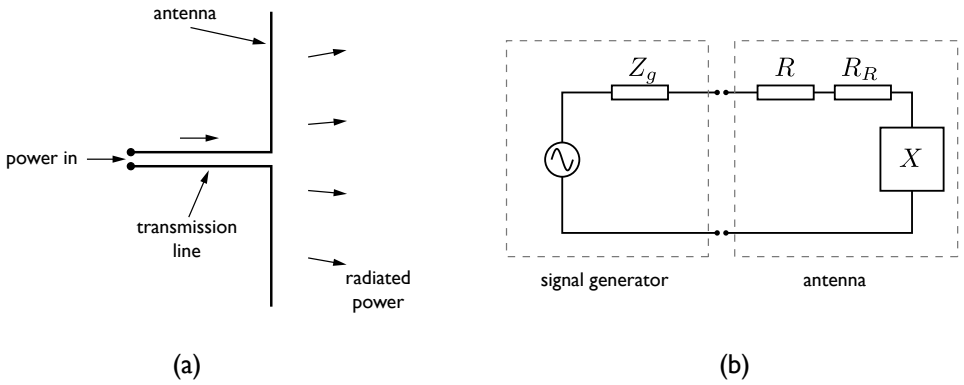


Figure 18.5: Circuit picture of a transmitting system. (a) A simple antenna. (b) An electronic model of the system.

dissipated in antenna losses and simply warm it up a bit. Some power may be reflected from the antenna back to the generator.

These three effects, radiation, loss, and reflection, can be represented by three impedances as illustrated in figure 18.5 (b). R_R is the antenna's *radiation resistance* value; it represents the antenna properties which allow power to radiate away. R is the *loss resistance*; it represents the ways power is dissipated, warming up the antenna. X is the antenna *reactance*; this represents any ways the antenna can store energy, returning it to the generator after a delay. This reactive behaviour is a bit like the way a capacitor or inductor can store electrical energy and release it later. For an ideal antenna we would therefore arrange that $Z_g = R_r$, $R = 0$ and $X = 0$. This would ensure that all the power the generator can produce is radiated into space. A value of $R > 0$ means that some power is 'wasted' warming the antenna. Since we can't expect to completely avoid dissipation losses, in practice we have to settle for attempting to ensure that $R < R_R$. All being well, we can usually get a zero reactance, although this may be difficult to arrange sometimes.

18.1.3 The small magnetic loop antenna

As we will see in a bit more detail later on, a serious disadvantage of the Hertzian dipole is that it suffers from a combination of two problems which can severely limit is efficiency as an antenna:

- For a Hertzian dipole, we usually find that $R_R \ll R$
- For a Hertzian dipole, we tend to find that $|X| \gg R_R + R$

This means that we find that the Hertzian dipole is not usually a very efficient antenna at low frequencies. As a result, when we have to use antennas whose sizes are very small relative to

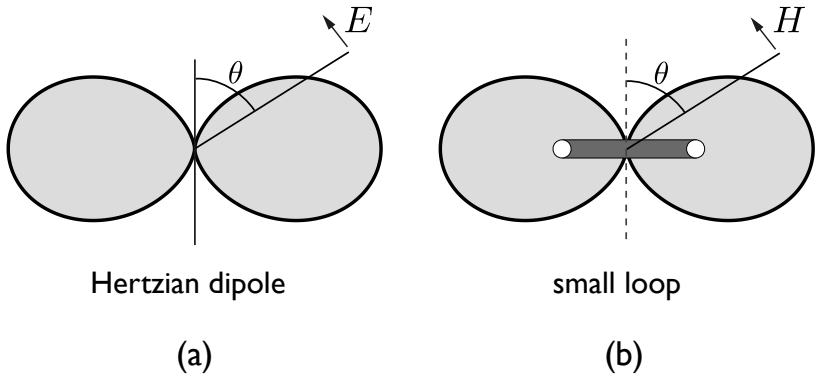


Figure 18.6: Comparison of antenna power patterns for (a) a Hertzian dipole and (b) a small magnetic loop.

the signal wavelength it can often make sense to use *magnetic loop* or *ferrite rod* antennas. Texts on antennas often consider a variety of forms of magnetic loop: circular, square, length much much shorter than diameter, length longer than diameter, etc. Although the details of these systems vary (for example, their inductance depends on the ratio of their length to diameter) they are *all* 'small' compared with the wavelength. As a result we can explain their general behaviour in a way that ignores many of these details.

When we set up an alternating current in a loop we essentially create a fluctuating magnetic dipole field about the centre of the loop. The far-field patterns of the various shapes of loop are all essentially identical as the dimensions of the loop are 'too small to be noticed' from a long way away since they are all tiny compared with the signal wavelength.

The behaviour of a small loop has some similarities to a Hertzian dipole. For example, as can be seen in figure 18.6 it has the same antenna power pattern, but oriented such that the nulls (which for a dipole are along the direction of the dipole wire) are aligned perpendicular to the plane of the loop. The main difference we'd observe in the far field patterns is that the electric and magnetic fields have been 'swapped over'.

If we define the direction angle, θ, to be with respect to the normal to the loop plane, we can say that for a loop, the electric field radiated into the far field will be

$$E_\phi = \frac{120\pi^2 n I_0 A \sin(\theta)}{\sqrt{2} r \lambda^2} \tag{18.11}$$

If we compare this to equation 18.3 we see that it has the same angular form, but is scaled differently and also depends on the ratio of the loop's area to the square of the wavelength (whereas the field for a dipole depends of the ratio of dipole length to wavelength). The other distinction is that the electric field component is parallel to the plane of the loop rather than parallel to the dipole wire.

Now if this were all there was to the story, there would be no real reason to prefer loops to dipoles at low frequencies. However, the loop turns out to have some significant advantages

due to the way it operates. In each case we have defined the radiated field in terms of a peak current, I_0, on the antenna. As we have already established, it can be difficult to induce a significant current level onto a dipole. Setting up a current in a loop is, however, relatively easy. We no longer have the 'open end' problem as we can push charge into one end of a looped wire and draw it out again at the other. Hence we can expect to be able to obtain relatively high currents (compared to a Hertzian dipole) quite easily. In addition, we can use a high value for the number of turns, n, around the loop to multiply up the effect of a given input current. For these reasons we tend to find that a small loop is easier to use than a Herztian dipole.

In reality, we still can't use a loop to radiate all the power we deliver to it: the efficiency of the loop will depend on the relative levels of the loop's real (dissipation) resistance and its radiation resistance. By going through the appropriate calculations we can find that, for a small loop, the radiation resistance will be

$$R_R \approx 31200 \times \left(\frac{nA}{\lambda^2}\right)^2 \tag{18.12}$$

In the absence of any resistive losses, the antenna gain of a small loop (and that of a Hertzian dipole) would be 1.75 dB, but in reality the effective gain is reduced by power dissipation due to the resistive losses. We can describe these losses in terms of a *radiation efficiency factor*

$$k = \frac{R_R}{R_R + R} \tag{18.13}$$

The antenna's gain (and thus effective area) are degraded by this factor compared to what we'd expect from a perfect antenna with the same directional behaviour. In practice, we tend to find that $R \gg R_R$.

18.2 More types of antenna

There are many different types of coherent antenna. All sorts of sizes, shapes, designs, etc. Trying to learn about them all would be painful and slightly miss the point of what we're about. Fortunately, we can divide antennas into three rough classes based on how their dimensions compare with the wavelength of the fields they're designed to work with. The Hertzian dipole has dimensions much smaller than the free-space wavelength, so we'll move on to look at the other two main classes.

18.2.1 Antennae with dimensions on the order of wavelength

The simplest type of practical antenna in this group is the **halfwave dipole**. The length of this is chosen so that the dipole resonates at the operating wavelength. The resulting current and voltage pattern this produces is illustrated in figure 18.7.

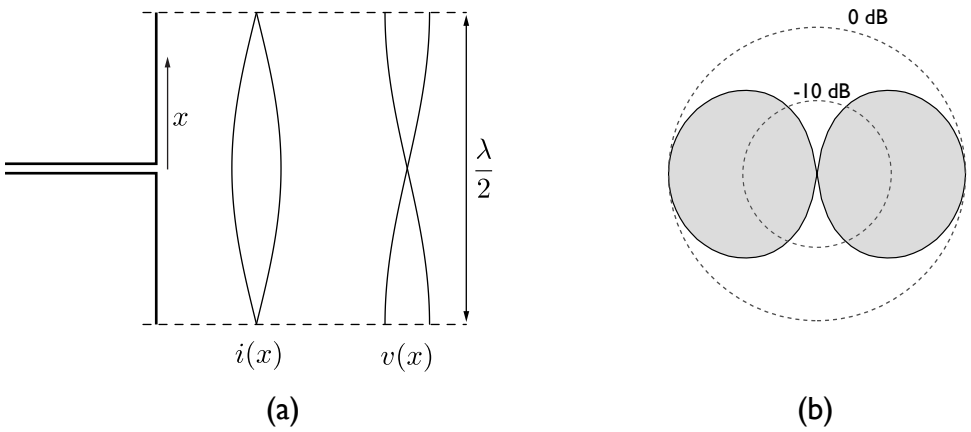


Figure 18.7: (a) The current and voltage distributions on a halfwave dipole. (b) The power pattern of a halfwave dipole. 0 dB refers to the maximum power level.

The current distribution along the dipole is

$$i(x,t) = I_0 \cos(2\pi f t) \cos\left(\frac{2\pi x}{\lambda}\right) \tag{18.14}$$

and the voltage distribution is

$$v(x,t) = V_0 \cos(2\pi f t) \sin\left(\frac{2\pi x}{\lambda}\right) \tag{18.15}$$

provided that f is approximately equal to the dipole's resonant frequency, $\approx c/\lambda$. Power is coupled into or out of a dipole via a pair of connections placed on either side of a small central gap. The ratio of the magnitudes, I_0 to V_0, determines the antenna's effective radiation resistance which is the impedance this gap presents to anything connected to it. For a halfwave dipole $V_0/I_0 = R_R \approx 75\Omega$. This is sometimes called the *driving impedance* of the antenna since it is the impedance power has to be 'driven into' when the dipole is used as a signal transmitter. As was mentioned earlier, the peak gain of the halfwave dipole is $2.15\ \mathrm{dB_i}$.

Halfwave dipoles are used a great deal at frequencies from a few to hundreds of MHz because they are so easy to make. They're so common that they're usually just called a dipole and it's taken for granted that they are about a half wavelength long unless their length is specifically mentioned. Although convenient, the dipole has a relatively low gain, however we can obtain much higher antenna gain values by using an *array* of dipoles. The behaviour of arrays can be analysed using the principle of *field superposition*: This means that if we have an array of n sources each producing a field, E_i at some point, R, then the *total* field produced when all the sources are radiating is simply the sum of their individual fields. Hence the total field produced by the array of sources will be

$$E_R = \sum_{i=1}^{n} E_i \tag{18.16}$$

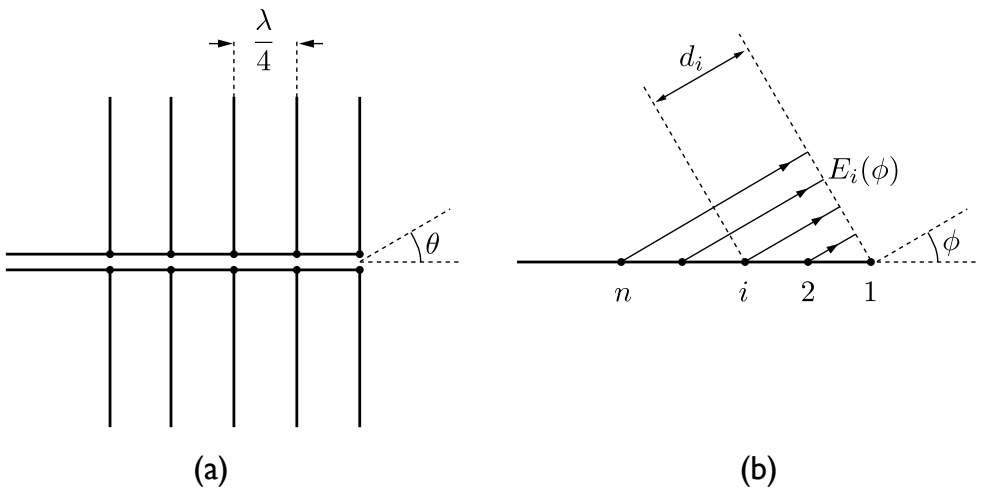


Figure 18.8: Views of an end-fire array from (a) the top and (b) the side.

Figure 18.8 shows an *end fire array* of dipoles. All of the dipoles are driven by power flowing along a common 'feeder' or waveguiding arrangement. (A pair of wires in this case.) The dipoles are spaced $\lambda/4$ apart. This means that the ith dipole receives its driving voltage $(i-1)(\pi/2)$ 'earlier' *in phase* than the front, $i = 1$, dipole.

Consider the field radiated in the direction at an angle, ϕ, to the $\theta = 0$ plane normal to the dipole lines. The front dipole will radiate a field

$$E_1 = E_0 \cos(2\pi f t) \tag{18.17}$$

in this direction. The ith dipole will be earlier by a phase $(i-1)\pi/2$ but the field it radiates must cover an extra distance

$$d_i = \frac{1}{4}(i-1)\lambda\cos(\phi) \tag{18.18}$$

to 'catch up' with the field from the front dipole moving off in that direction. This means that the total field radiated in the ϕ direction will be

$$E(\phi) = \sum_{i=1}^{n} E_0 \cos\left(2\pi f t + \frac{(i-1)\pi}{2} - \frac{2\pi d_i}{\lambda}\right) \tag{18.19}$$

i.e.

$$E(\phi) = \sum_{i=1}^{n} E_0 \cos\left(2\pi f t + \frac{(i-1)\pi}{2}[1-\cos(\phi)]\right) \tag{18.20}$$

(This assumes that all the dipole *elements* in the array are driven with the same amplitude. In practice this may not be correct and we'd have to alter the above expression. However, we're just using this example to illustrate a general method, so we can ignore this as an embarrassing detail...)

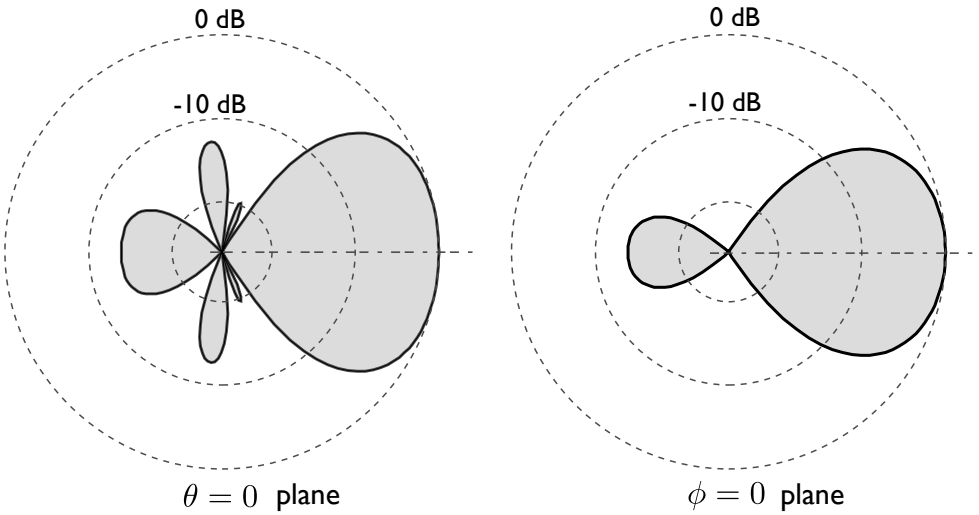


Figure 18.9: The principle plane power patterns of a 5-element driven array.

An analysis of the halfwave dipole shows that the field from a single dipole radiated at an angle, θ, to the plane normal to the dipole axes will have a pattern of the form

$$E(\theta) = \frac{\cos((\pi/2)\sin(\theta))}{\cos(\theta)} \tag{18.21}$$

Using the same arguments as above, we can therefore show that, in the $\phi = 0$ plane, the antenna *field* pattern will be

$$E(\theta) = \sum_{i=1}^{n} E_0 \frac{\cos((\pi/2)\sin(\theta))}{\cos(\theta)} \cos\left(2\pi f t + \frac{(i-1)\pi}{2}[1-\cos(\theta)]\right) \tag{18.22}$$

The $\phi = 0$ and $\theta = 0$ planes are said to be the *principle* planes of the antenna. Figure 18.9 illustrates the *power* pattern in the principle planes of an array of five *driven elements* of the kind considered above.

Most dipole arrays use just one driven element and all the others are *passive* or *parasitic* elements. (Note that the term, 'driven' is often still used when considering receiving antennas to mean the one(s) connected to the wires or waveguide.) The passive elements are dipoles which are simple strips of metal with no central break. Despite this, they behave as if they were linked to the wires/guide connected to the driven element. In effect, they are linked together by the fields radiated backwards and forwards between elements. Figure 18.10 shows two examples from the wide range of possible designs.

The *Yagi-Uder* array is named after its inventors (in practice it's usually just called a "Yagi"). It consists of one driven element plus one or more *directors* on one side and a *reflector* on the other. Figure 18.10 (a) shows a 5-element Yagi. D_1, D_2, and D_3 are the directors, R is the reflector. An incoming field sets up resonant currents on all the dipole elements. This causes

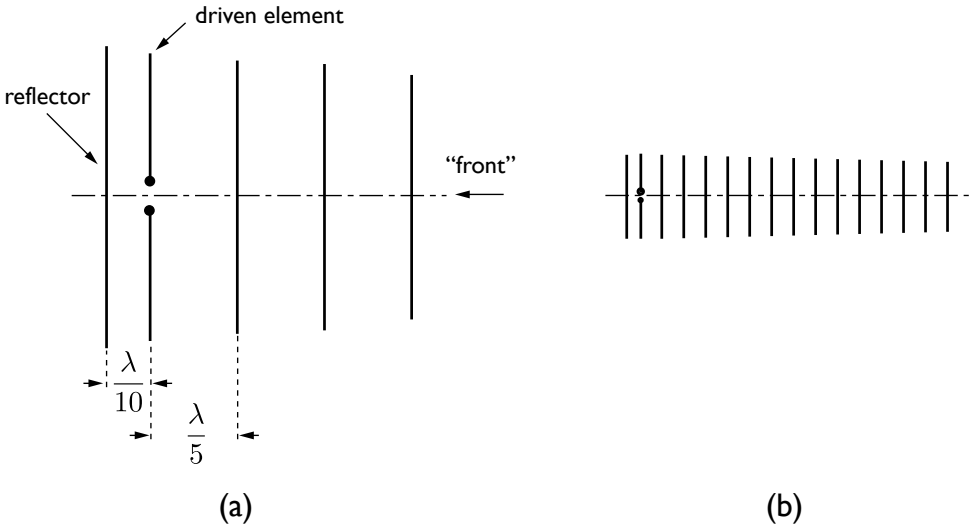


Figure 18.10: Two types of dipole array antenna. (a) The *Yagi-Uder* antenna. (b) A *log-period* array.

the passive elements to re-radiate signals. These re-radiated fields are then picked up by the driven element. Hence the total current induced in the driven dipole is a combination of the direct field striking it and the re-radiated contributions from the directors and reflector.

The relative phases of the contributions reaching the driven element depend upon three factors:

- The direction the incident signal is coming from
- The relative positions of the passive elements
- The lengths of the passive elements

This last effect is because the dipoles are resonators. The phase delay between an incident and a reflected wave depends on how long the dipole is in wavelengths. A shorter dipole tends to reflect 'more quickly'.

The resonances tend to 'tune' the antenna to a specific frequency band. The more elements there are of the same length, the narrower the band of frequencies over which the antenna will work and the higher the peak gain. The elements of a Yagi all have different lengths. The directors are all slightly shorter that the driven elements and the reflector is longer. This spread in sizes widens the antenna's working bandwidth at the expense of reducing the peak gain slightly. The spacing between elements can also be altered a little to improve the performance of a practical antenna. For the 5-element example the optimum gain-bandwidth product is when $R = 0.525\lambda$, $D_1 = 0.475\lambda$, $D_2 = 0.463\lambda$, and $D_3 = 0.425\lambda$. The director spacing is around $\lambda/5$ and the reflector is about $\lambda/10$ from the driven element. This produces an antenna with a peak 'front' gain value of around 15 $\mathrm{dB_i}$ and a bandwidth of $\pm 10\%$.

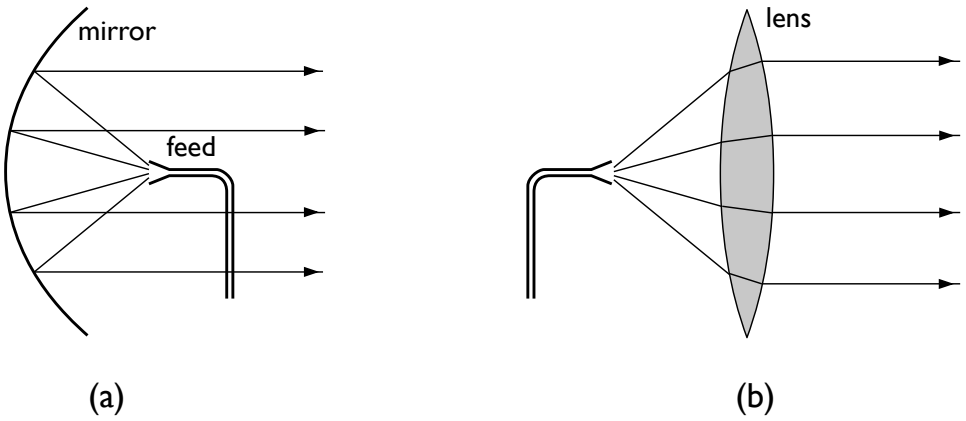


Figure 18.11: An antenna consisting of a feed and (a) a mirror and (b) a lens.

In principle, we can make Yagi antennas with as many elements as we like, but in practice *log period* antennas tend to be better when using more than about 8 elements. In these, the element spacing and size varies logarithmically along the antenna. Most VHF (100MHz) antennas are Yagi arrays. UHF (600 MHz–1 GHz) TV antennas are normally log periods with 12-24 elements. They give peak gains of 20–25 $\mathrm{dB_i}$ but have wider bandwidths than a Yagi, typically around $\pm$20%.

18.2.2 Antennae with dimensions much greater than wavelength

At microwave and near-optical frequencies it is usually possible to build antennas whose sizes are many wavelengths. These tend to take the general forms illustrated in figure 18.11. The main element of the antenna is either a mirror or a lens whose diameter is many wavelengths. The lens/mirror couples free space fields into or out of a much smaller feed element. This can be a simple dipole, or the 'cut end' of a standard waveguide or fibre. Here we will concentrate on the use of microwave guide, but similar arguments can be applied to other systems at other frequencies.

The *standard rectangular waveguide* usually used to get microwave signals from A to B consists of a rectangular metal pipe whose cross dimensions are approximately $\lambda/2 \times \lambda/4$. (This form of guide is widely used because it will only carry a single mode.) This means that we can precisely define and control the field pattern running along the guide. In principle we can use an open guide end to feed signals to a lens/mirror, or place it at the focus of a lens/mirror to pick up gathered power. However, the effective area of an open or cut single mode guide is around $\lambda^2/8$. This means that the open end will tend to radiate power into a solid angle of around 8 steradians, i.e. about two thirds of a sphere! (In fact, the actual effective area isn't exactly equal to the physical opening area, but it is of a similar size so this estimate is good enough to reveal the problem.) This would require a very 'deep' parabolic reflector or an impossibly large lens to collect. To avoid this problem it is usual to feed the

large mirror/ lens antenna with a smaller feed antenna or feed horn whose dimensions are between those of the guide and the main antenna. This feed is a sort of 'mini antenna' which helps us couple the guide to the main part of the antenna.

The simplest way to do this is to flare the end of the rectangular guide into a conical or pyramidal shape. This looks a bit like a rectangular version of the 'bell' on a brass band instrument and it does a similar job. The open end of the resulting pyramidal feed is the base of the pyramid and we can, in principle, give this as large an area as we like. In practice, things go may wrong if we flare up to an open end much bigger than about $5\lambda \times 5\lambda$. However, a horn with this aperture size will have an effective area of $\approx 25\lambda^2$, leading to a value for Ω_A of ≈0.04 steradians. This is directional enough to feed a practical mirror or lens system. In practice, most antennas of this type use more than one lens or mirror, and may use more complex feed arrangements. This leads to improved performance, but the basic method of operation is the same as described above. The gain and directivity of the total antenna system is largely determined by the area of the main lens/mirror *illuminated* by the feed system.

The larger the mirror (or lens), the higher the gain and the narrower the antenna angle at a given wavelength. To avoid edge diffraction problems it is usual to illuminate only the central portion of the lens. Hence the effective area of antenna is generally between 90% and 50% of the physical mirror area.

18.2.3 The link gain equation

We've discussed antennas as devices that can be used to transmit or receive in isolation, but of course in most communications applications where we send information from A to B, we would be using a pair of antennas, one at each end. We'll now look at what happens when we use a pair of antennas to send electromagnetic power from place to place.

Consider a transmitter, TX, which is radiating a total power, P_T. All of this power will pass through a sphere of radius r, centred on the transmitter. The total surface area of such a sphere will be $4\pi r^2$. As a result, if the TX antenna is isotropic (omnidirectional), then an area, A_R, laid on the sphere will intercept a power

$$P_R(\mathrm{omni}) = \frac{\mathrm{P_T A_R}}{4\pi \mathrm{r}^2} \tag{18.23}$$

This means that if we use a transmitter antenna with gain G_T aimed at the collecting area, then the collecting area will receive a power

$$P_R = \frac{P_T G_T A_R}{4\pi r^2} \tag{18.24}$$

Form out earlier discussion of the basic properties of antennas, we can say that the gain of a receiving antenna with an effective area of A_R will be

$$G_R = \frac{4\pi A_R}{\lambda^2} \tag{18.25}$$

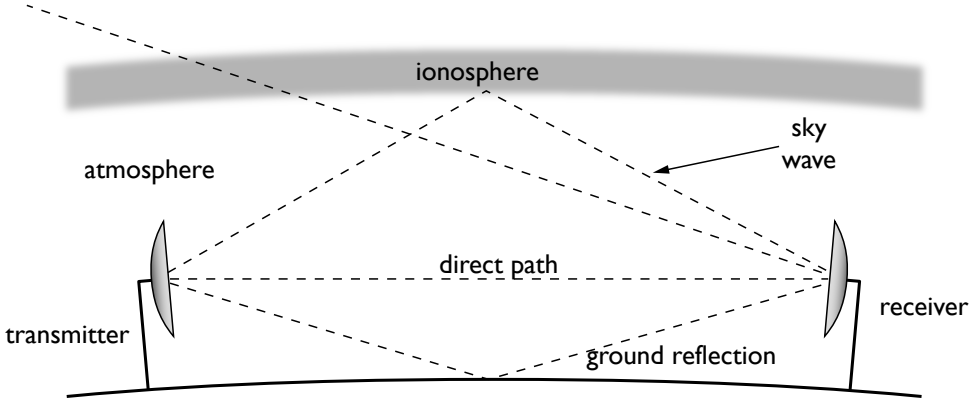


Figure 18.12: More realistic link behaviour.

If we combine this with equation 18.24, then we can get another expression for the received power:

$$P_R = \frac{G_T G_R \lambda^2}{16\pi^2 r^2} P_T \tag{18.26}$$

This is called the **link gain equation**. It allows us to work out how much power we will receive using a pair of antennas in a given situation. From it we can define the **link gain**:

$$G_L = \frac{G_T G_R \lambda^2}{16\pi^2 r^2} \tag{18.27}$$

Note that referring to this a 'gain' is somewhat optimistic: the received power is usually only a fraction of the transmitted power.

Equation 18.27 only truly applies in empty space. A terrestrial communications or signal link may in reality look more like the situation shown in figure 18.12.

The main factors affecting the real link's performance are:

- The radiation must pass through the atmosphere. This may absorb or scatter some of the field, attenuating the level reaching the receiver.
- The Earth is curved and tends to be covered with irregular features. This may cause the direct line of sight to be blocked. It may also provide surfaces which can reflect power into the receiver antenna (which can give rise to the *multipath* effects that we discussed when talking about digital modulation).
- Some of the transmitted power may initially move in a direction towards space, but be reflected by the ionosphere back towards the receiver.
- Some external sources may radiate unwanted noise power onto the receiver.

The effects of the ground and ionosphere vary a lot from place to place and time to time. For that reason they can only be analysed on a case-by-case basis, so we can indicate their effect by turning the above equation for the link gain into an approximation.

The effects of atmospheric attenuation can be described in terms of an atmospheric *attenuation coefficient*, $\alpha(f)$. We can use this to write a modified form of the link gain equation:

$$P_R \approx P_T \frac{G_T G_R \lambda^2}{16\pi^2 r^2} \exp(-\alpha(f) r) \tag{18.28}$$

α is a function of frequency (wavelength) and, at high frequencies, on the weather. We can often ignore it for frequencies below about 1 GHz. It rises at higher frequencies and may produce a loss on the order of 10 dB/km.

18.3 Radar

Radar (from *RA*dio *D*irection-finding *A*nd *R*anging) is a method commonly used to detect the bearing and distance to objects, which really came into its own during the Second World War and has seen massive use since. Radar is a nice system to look at for a number of reasons: it employs the ideas of coherence and mixing we've seen earlier, relates well to ideas of link gain, obviously makes use of antennas and is a technology that is extremely important in modern life in ensuring the safety of travel.

We'll look at two different types of radar: **pulsed radar** and **frequency-modulated continuous-wave** (FMCW) radar.

18.3.1 Pulsed radar

The simplest form of radar is illustrated in figure 18.13. A radar system includes a transmitter which emits a series of EM pulses. These pulses are radiated in a directional pattern using a suitable high-gain antenna. The radar then uses a receiver, usually connected to the same antenna, to detect any returned pulses being reflected from objects appearing in the antenna's field of view. The transmitter, antenna, and receiver are linked using a device called a *circulator*. This ensures that all (ideally!) the transmitter power is sent to the antenna whilst all the reflected power returned to the antenna is sent to the receiver, without the very sensitive receiver being damaged by the large powers that are transmitted.

The form of the circulator depends on the frequency range in which we're working, but they often make use of the interaction of EM radiation with magnetic materials. We're not concerned with the details of how systems at different frequencies work: all we care about is that in a *three-port circulator* which is the type shown in the figure, a signal entering at port 1 emerges from port 2, and a signal entering at port 2 emerges at port 3.

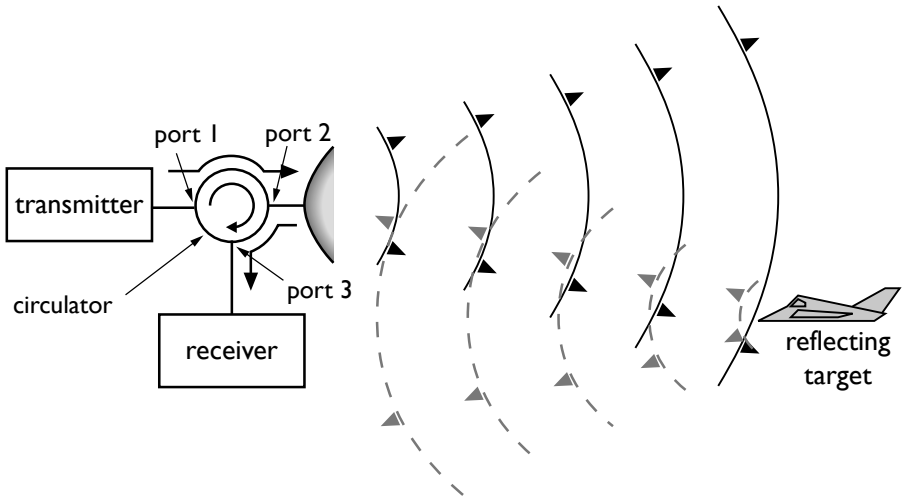


Figure 18.13: A simple pulsed radar system and target.

Radar systems determine the range of a reflecting target by measuring the time delay between the emission of a signal and the return of a reflection. Bearing can be determined in the simplest case just by using a reasonably directional beam. (Obviously, a target outside the field of view illuminated by the radar won't produce a reflection.)

Knowing the time delay, τ, between pulse emission and the arrival of the reflected return we can say that the target range, R, should be

$$R = \frac{\tau c}{2} \tag{18.29}$$

(where c is the speed of light).

One of the main problems which arises with simple pulsed radars is *range cell ambiguity*. This occurs because the radar emits a regular stream of pulses at a rate of, say, p per second. A target will therefore reflect a stream of pulses which return at the same rate. This presents the radar system with a problem when comparing the emitted and returned pulse streams. Is each returned pulse a reflection of the most recently transmitted pulse, or is it a reflection of an earlier pulse from a more distant target? In principle, the target can be at any one of a series of ranges

$$R_n = \frac{\tau c}{2} + \frac{nc}{2p} \tag{18.30}$$

where n is an integer (including zero). In effect, the radar's field of view is divided up into a series of *cells*, each having a 'depth', $D = c/2p$. To try an overcome this problem we can do one of two things:

- Reduce the pulse rate so that the first cell $(n = 0)$ is so deep that a target beyond it will be too far away to produce a detectable return
- Vary the pulse rate and attempt to work out which pulse produces which reflection.

In practical systems both approaches can be used together. This can often let us overcome the problem, but at the expense of degrading the system's performance in other ways. For example, the sensitivity of the radar is usually increased by integrating the return patterns from successive pulses. Reducing the rate and varying it means this will take longer, hence lowering the radar's sensitivity.

Another basic problem of radar stems from the fact that power must travel both to and from the target. Consider a target placed at a range, R, from an antenna of gain, G, operating at a signal wavelength, λ. The peak power level of each pulse is P_T . Using the same arguments from our recent antenna and link-gain discussions, we can say that the *power density* (or *flux*) at the target will be

$$F = \frac{GP_T}{4\pi R^2} \tag{18.31}$$

It is conventional to describe the target's overall reflectivity in terms of a *radar cross section* (RCS), σ. This is defined such that the total power reflected/scattered by the target, P' may be expressed as

$$P' = \sigma F \tag{18.32}$$

and it is assumed that P' is re-emitted isotropically. As far as the radar is concerned, therefore, the target behaves like an omnidirectional source radiating a total power P'. Hence the peak pulse flux returned to the radar antenna will be

$$F_R = \frac{\sigma F}{4\pi R^2} = \frac{\sigma G P_t}{16\pi^2 R^4} \tag{18.33}$$

We know that an antenna of gain G will have an effective area of

$$A = \frac{G\lambda^2}{4\pi} \tag{18.34}$$

hence the peak pulse power reaching the receiver will be

$$P_R = P_T \frac{\sigma G^2 \lambda^2}{(4\pi)^3 R^4} \tag{18.35}$$

This expression is a form of the **radar equation**. It has the same importance in radar as the link gain equation in signal transmission. The most significant feature of this expression is that, unlike the link gain, the power returned to the radar falls as the *fourth power* of the range. This means that to double the maximum range at which a given target could be just be detected we would have to increase the transmitted power by a factor of 16. (Assuming we don't alter anything else.) As a result, long range radar systems often have to emit extremely high pulse power levels. This can be a problem in many situations. For example, microwave power levels in excess of kilowatts have a tendency to fry anyone who walks into the beam! They can also have other undesirable environmental effects. Another obvious drawback is the waste of energy radiating high power pulses out into space.

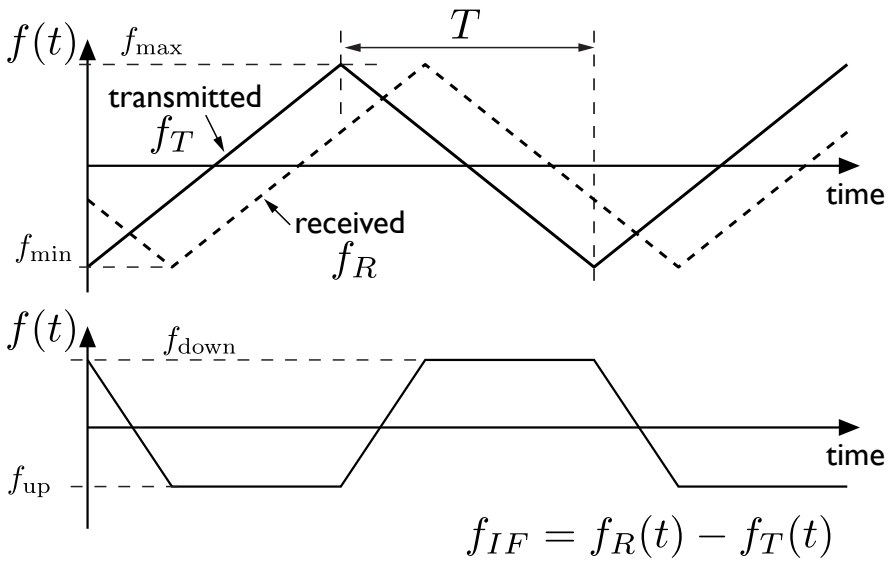


Figure 18.14: Transmitted, received and IF frequencies for an FMCW radar system as a function of time.

18.3.2 FMCW radar

We'll now move on to look at another form of radar which doesn't solve all the problems mentioned above, but which has some advantages over traditional pulsed radar. In frequency-modulated continuous-wave (FMCW) radar, the transmitted power is held constant, and instead the transmitted *frequency* is altered in a deliberate way. (The "continuous" in the name comes from the fact that we are continually transmitting.) Milimetre-wave systems using this method have been developed for car collision avoidance schemes and surveying applications. FMCW techniques provide very high precision at low ranges.

The sensitivity of a radar depends on how much reflected *energy* it can gather in a given time. In pulsed systems the peak power must therefore be high since, most of the time, no power at all is being transmitted or received. CW radars can therefore often use steady power levels two or three orders of magnitude lower than the peak levels required for comparable performance from a pulsed system.

The pulses of a conventional radar provide convenient transmitted signal markers which are used to determine the time taken for a wave to travel to the target and back. When using a CW transmission we have to modulate the signal in some other way. The simplest way to do this is to sweep the wave's frequency up and down regularly in a triangle-wave. This process is illustrated in figure 18.14.

In figure 18.14, $f_T(t)$ represents the the transmitted frequency as a function of time, and $f_R(t)$ represents the received frequency as a function of time. If the transmitted frequency remained constant, then provided that the target was stationary, the received frequency would also be constant, irrespective of the range to the target.

However, because the transmitted frequency is being changed as a function of time, the received signal (which has had to leave the radar earlier in order to make it to the target and back) will have a different frequency.

During an 'up' sweep, the signal frequency is increased linearly with time from $f_{\min}$ to $f_{\max}$ over some period T. (Note, T, is *not* the period of the triangular modulation: the triangle wave has a period of $2T$.) This means that the rate at which the frequency is *ramped* is

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \frac{f_{\max} - f_{\min}}{T} \tag{18.36}$$

We can use a heterodyne mixer to mix together the transmitted signal with the received signal. (A portion of the transmitted signal is usually used for the LO signal to the mixer, and the received signal provides the RF input to the mixer.) If we filter the mixer output to eliminate the sum frequency (which in many cases would be too high to propagate down a cable in any rate!), then the frequency of the mixer's IF output will be

$$\begin{aligned} f_{IF}(t) &= f_R(t) - f_T(t) \\ &= f_T(t-\tau) - f(t) \end{aligned} \tag{18.37}$$

Figure 18.14 also shows the IF output frequency as a function of time. During the parts of the modulating cycle in which f_R and f_T are both increasing, or f_R and F_T are both decreasing, the IF frequency is *constant*, and given by

$$\begin{aligned} f_{\mathrm{up}} &= -\tau\frac{\mathrm{d}f}{\mathrm{d}t} \\ &= -\frac{\tau(f_{\max} - f_{\min})}{T} \end{aligned} \tag{18.38}$$

and

$$\begin{aligned} f_{\mathrm{down}} &= -\tau\frac{\mathrm{d}f}{\mathrm{d}t} \\ &= \frac{\tau(f_{\max} - f_{\min})}{T} \end{aligned} \tag{18.39}$$

Due to the symmetry of the cosine function, a simple heterodyne system won't discriminate between the cases when the LO frequency is higher or lower than the RF frequencies, so the output most of the time would just be a cosine wave of $|f_{\mathrm{up}}|$. (It is possible to build systems which can discriminate between $+f$ and $-f$, but we won't bother with that here...)

For a single target, by measuring f_{IF} during the parts of the cycle in which it is constant, we can determine the range to the target, as we will know T, $f_{\max}$ and $f_{\min}$ as they are design parameters of a given system. (τ can also be determined from the length of the 'crossover' regions, where the IF frequency changes from f_{up} to f_{down} and vice versa.)

In reality, there may well be more than one target in the beam. Often, there is one 'actual' target, plus some scatter from the ground, etc. In this case, we can work out the *spectrum* of the IF signal by sampling the IF and taking the Fourier transform computationally. We will

see spikes in the spectrum corresponding to reflections from each target, and the frequency in the spectrum at which they occur will be proportional to the range.

If the target is moving, then the received signal will be doppler shifted, so $f_{\rm up}$ will differ from $f_{\rm down}$. From the difference between $f_{\rm up}$ and $f_{\rm down}$ it is possible to calculate the velocity of the target, as well as its range.

18.3.3 Radar on beam-filling targets

The analysis in section 18.3.1 assumes that we are looking for 'small' reflecting objects , i.e. targets whose cross-sectional size is small compared to the width of the radar beam. In some cases, however, we view an 'extended' target which essentially fills the field of view. The most common examples are situations where the radar is directed towards the ground (e.g. for a radar altitude measurements from an aircraft).

The significant thing about this type of situation is that (ignoring any losses due to air attenuation or scattering) **all** of the radiated power strikes the 'extended target' in an area (or volume) which is within the field of view of the radar receiver. As a result, in these cases the power level striking the target does not fall with the square of the range. Nor does the power reaching the target depend upon the antenna's gain or antenna angle, Ω, provided that the target is large enough to fill the beam. In such situations we can treat the extended target as if it were re-radiating an amount, $\eta P_T \exp(-2\alpha R)$, where P_T is the power transmitted by the radar, η is an effective reflectivity value for the target surface, R is the range, and α is the air's attenuation coefficient.

The manner in which power is re-radiated or scattered by the target object clearly depends a great deal on its nature and how it may be oriented with respect to the radar. For simplicity we can imagine it reradiating into a hemisphere which includes the direction of the radar and take any directional reflection/scattering properties into the appropriate η value. On this basis we can say that the power level returned to the radar will be

$$P_R = \frac{P_T \eta A_e}{2\pi R^2} \exp(-2\alpha R) \tag{18.40}$$

where A_e is the effective area of the radar's antenna. At low frequencies, we can often neglect the atmospheric attenuation, so the power level that the radar receives will be

$$P_R = \frac{P_T \eta G \lambda^2}{8\pi^2 R^2} \tag{18.41}$$

This expression differs from equation 18.35 in two significant ways. Most importantly, the received power only falls as the *square* of the range, unlike the standard radar equation. Secondly, the received power only depends upon G rather than G^2. This is because the transmit gain does not alter the power reaching the target, only how small a portion of the target area is illuminated and in view.

18.4 Chapter 18: take-home messages

You should now understand that a coherent antenna is a **reciprocal** device, i.e. that it's behaviour as a transmitter and receiver is the same, but with the 'video playing the other way'. An antenna is a form of transformer, which turns one kind of electromagnetic signal into another. You should see how an antenna produces a directional power pattern. That this behaviour can be described in terms of either a **solid angle**, or a **gain**, or an **effective area**.

You should also know that the way an antenna radiates power can be modelled using a **radiation resistance**. Any resistive losses or reflection can be modelled in terms of a *loss resistance* and *antenna reactance*, and we have to match the generator/antenna impedances and minimise the loss resistance to get optimum behaviour.

We've seen (qualitatively) how a variety of different sorts of antenna work. It is possible to choose a specific gain, operating frequency, etc, by assembling suitable **arrays** of dipoles.

Arrays can contain both driven and passive (or parasitic) elements. The behaviour of a complex antenna system can be worked out using the principle of field superposition to add together the contributions of all its parts. The methods used vary with the size of the antenna compared with the free space wavelength.

You should now understand how we can use antenna gain values to work out the **link gain** we can expect when sending power from place to place.

You should now understand how a **radar** system is able to measure the range to a reflective target, and understand the operation and key equations behind pulsed and CWFM radar systems. A radar may use a **circulator**to simultaneously connect a transmitter and receiver to a common antenna. The main limitations of radar are **range cell ambiguity**, caused by the repetition rate, and the tendency for the reflected level returned to the radar to fall as the fourth power of the range, as indicated by the **radar equation**. You should also understand how and why the radar equation is modified for beam-filling targets.