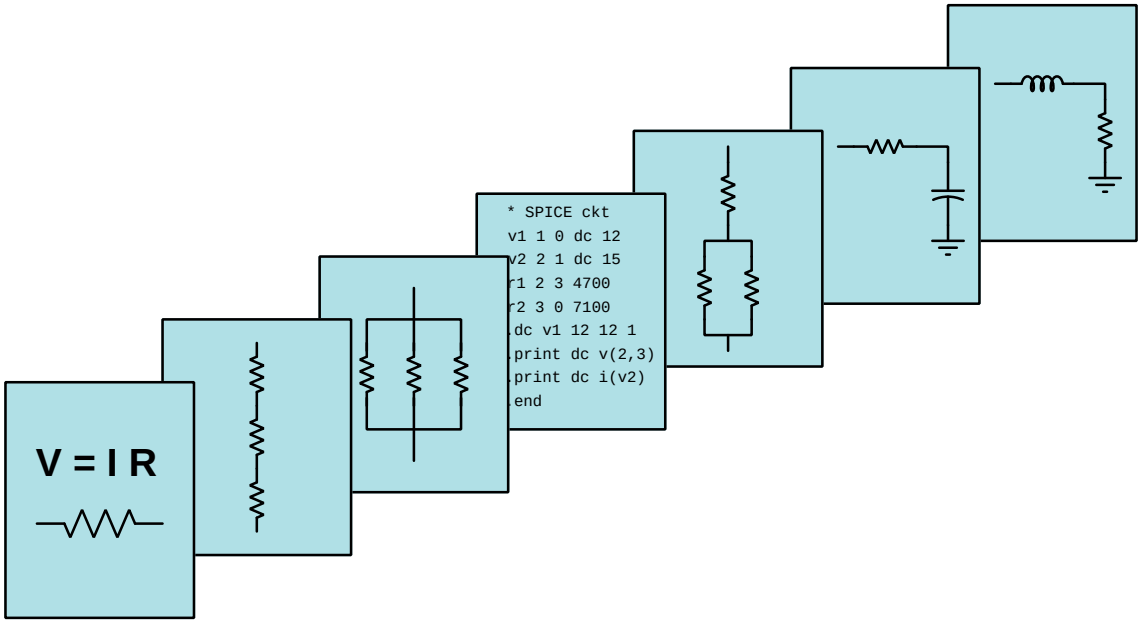
Modular Electronics Learning (ModEL) project



Introduction to Modulation

Last update = 16 April 2025

Contents

1 Introduction **3**
- 1.1 Recommendations for students . . . 3
- 1.2 Challenging concepts related to signal modulation . . . 5
- 1.3 Recommendations for instructors . . . 6

2 Case Tutorial **7**
- 2.1 Example: balanced mixer frequencies . . . 8
- 2.2 Example: simple diode mixer circuit . . . 10
- 2.3 Example: simple diode demodulator circuit . . . 13
- 2.4 Example: AD633 as a balanced mixer . . . 15
- 2.5 Example: downconverting AM receiver . . . 19
- 2.6 Example: GNU Radio Companion SDR demodulation flowgraphs . . . 23
- 2.7 Example: frequency-mixing in gear systems . . . 25

3 Tutorial **27**
- 3.1 Introduction to modulation . . . 28
- 3.2 Amplitude modulation . . . 31
- 3.3 Frequency modulation . . . 34
- 3.4 Phase modulation . . . 36
- 3.5 Pulse modulation . . . 37
- 3.6 Sampling modulation . . . 39
- 3.7 Frequency-shifting . . . 40
- 3.8 I-Q modulators . . . 45
- 3.9 Software-defined radio (SDR) . . . 49
- 3.10 Lock-in analyzers . . . 54

4 Historical References **55**
- 4.1 Arc converter transmitters . . . 56
- 4.2 Heterodyne radio reception . . . 59
- 4.3 Trunked telephony system . . . 67

5 Derivations and Technical References **69**
- 5.1 Mathematics of signal mixing . . . 70
- 5.2 Square-law mixing . . . 75

5.3 Gilbert cell circuit . . . 78
5.4 Trigonometry Reference . . . 84
5.4.1 The Unit Circle . . . 84
5.4.2 Pythagorean identity for sine and cosine . . . 84
5.4.3 Odd and even functions . . . 85
5.4.4 Sums and differences of angles . . . 85
5.4.5 Products and sums of sine and cosine functions . . . 85
5.4.6 Squares of sine and cosine functions . . . 85
5.4.7 Doubled angles . . . 85
5.4.8 Halved angles . . . 85

6 Programming References **87**
6.1 Programming in C++ . . . 88
6.2 Programming in Python . . . 92
6.3 Modeling signal modulation using C++ . . . 97
6.3.1 Digital amplitude-modulation . . . 98
6.3.2 Analog amplitude-modulation . . . 100
6.3.3 Digital frequency-modulation . . . 102
6.3.4 Analog frequency-modulation . . . 104
6.3.5 Digital phase-modulation . . . 106
6.3.6 Analog phase-modulation . . . 108
6.3.7 Pulse-width modulation . . . 110
6.3.8 Pulse-density modulation . . . 112

7 Questions **115**
7.1 Conceptual reasoning . . . 119
7.1.1 Reading outline and reflections . . . 120
7.1.2 Foundational concepts . . . 121
7.1.3 Carrier-less radio . . . 123
7.1.4 Reginald Fessenden's invention . . . 124
7.1.5 Bat sonar detector . . . 125
7.2 Quantitative reasoning . . . 126
7.2.1 Miscellaneous physical constants . . . 127
7.2.2 Introduction to spreadsheets . . . 128
7.2.3 Oscillographs comparing sinusoids . . . 131
7.2.4 Telephony up/down converter circuits . . . 133
7.2.5 High-side versus low-side injection . . . 135
7.3 Diagnostic reasoning . . . 136
7.3.1 Identifying modulation types . . . 137
7.3.2 Gear noise diagnosis . . . 140

A Problem-Solving Strategies **141**

B Instructional philosophy **143**

C Tools used **149**

D Creative Commons License **153**

E References **161**

F Version history **163**

Index **165**

Chapter 1

Introduction

1.1 Recommendations for students

Modulation is any process by which information becomes impressed upon a relatively high-frequency signal, which is useful for multiple reasons: high-frequency signals are relatively easy to filter apart from each other when blended along a common communications channel, and for some types of communication channels it is impossible to send data through them without being in the form of a high-frequency signal.

The three parameters of any sinusoidal waveshape alterable by modulation are amplitude, frequency, and phase; therefore we have amplitude modulation (AM), frequency modulation (FM), and phase modulation (PM). Both digital (high/low) and analog versions of each exist. Additionally, the "carrier" signal being modulated may be a digital *pulse* rather than a sinusoid. Once again multiple modulation techniques are common here, pulse-width modulation (PWM) and pulse-density modulation (PDM) being some of the more common.

Important concepts related to modulation include **radio communication**, **baseband** and **carrier** signals, **Morse code**, **harmonic** frequencies, **sideband** frequencies, waveform **envelopes**, **bandwidth**, **frequency** and **phase**, **heterodyning**, signal **mixing**, frequency **conversion**, **gain-bandwidth product**, **quadrature** signals, and **complex numbers**.

Here are some good questions to ask of yourself while studying this subject:

- How might an experiment be designed and conducted to explore the phenomenon of amplitude modulation? What hypotheses (i.e. predictions) might you pose for that experiment, and what result(s) would either support or disprove those hypotheses?
- How might an experiment be designed and conducted to explore the phenomenon of frequency modulation? What hypotheses (i.e. predictions) might you pose for that experiment, and what result(s) would either support or disprove those hypotheses?
- Why is modulation necessary for practical radio communication?
- What is meant by the "depth" of modulation?

- How do amplitude, frequency, phase, and pulse modulation schemes differ from one another?
- What will happen to the waveform of a carrier signal that is amplitude-modulated by a low-frequency sine wave?
- What will happen to the waveform of a carrier signal that is frequency-modulated by a low-frequency sine wave?
- What will happen to the waveform of a carrier signal that is phase-modulated by a low-frequency sine wave?
- What will happen to the waveform of a pulse signal that is pulse-width-modulated by a low-frequency sine wave?
- What will happen to the waveform of a pulse signal that is pulse-density-modulated by a low-frequency sine wave?
- What is *single sideband* modulation and why is it used?
- Why are FM radio signals largely immune to interference from lightning, whereas AM radio signals are susceptible?
- Why does phase modulation create a wider band of frequencies in the spectrum than frequency modulation, if they are so similar in other respects?
- How does true signal *mixing* differ from simple summation of multiple AC signals?
- In what way does practical signal mixing differ from signal multiplication?
- What is an artifact, in the electronic sense of the word?
- What does it mean for an amplifier to have a "gain-bandwidth product" parameter?
- How may signal frequencies be shifted either up or down at will?
- What is the purpose of a local oscillator?
- What allows the use of a square-wave rather than sine-wave local oscillator, if sine waves have a "purer" harmonic spectrum?
- Why is signal mixing typically used in radio receivers, especially digital receivers?
- Why is the I-Q modulation technique so popular?
- How does phasor mathematics relate to the I-Q modulation technique?
- What is Software-Defined Radio and how does this differ from more traditional radio technologies?

1.2 Challenging concepts related to signal modulation

The following list cites concepts related to this module's topic that are easily misunderstood, along with suggestions for properly understanding them:

- **Fourier's Theorem** – while this is no doubt a non-intuitive concept, it is incredibly useful. Knowing that any waveshape whatsoever may be reproduced by summing together the right combination of sinusoidal waves may defy our intuition, but at least seeing it is possible to synthesize common non-sinusoidal waveshapes like square and triangle using nothing but sine and/or cosine waves helps prove one can get non-round waves from lots of round waves added together. The practical upshot of this is that is it possible to consider very complex waveshapes as being nothing more than a set of sine waves added together. Since sine waves are easy to analyze in the context of electric circuits, this means we have a way of simplifying what would otherwise be a dauntingly complex problem: analyzing how circuits respond to non-sinusoidal waveforms. Time spent with a simple oscilloscope and spectrum analyzer viewing signal generator waveforms is also helpful in grasping how time-domain and frequency-domain representations relate.

- **Frequency-shifting or Heterodyning** – the fact that multiplying two sinusoidal signals in the time domain results in adding/subtracting those signals in the frequency domain is not intuitive at all, and is based on trigonometric identities. A section contained in the *Derivations and Technical References* chapter explains how the mathematics of this works.

1.3 Recommendations for instructors

This section lists realistic student learning outcomes supported by the content of the module as well as suggested means of assessing (measuring) student learning. The outcomes state what learners should be able to do, and the assessments are specific challenges to prove students have learned.

- **Outcome** – Demonstrate effective technical reading and writing

  Assessment – Students present their outlines of this module's instructional chapters (e.g. Case Tutorial, Tutorial, Historical References, etc.) ideally as an entry to a larger Journal document chronicling their learning. These outlines should exhibit good-faith effort at summarizing major concepts explained in the text.

  Assessment – Students show how quantitative results were obtained by the author in the Tutorial chapter's examples.

- **Outcome** – Apply the concept of heterodyning to quantitative determinations of signal frequencies

  Assessment – Calculate the frequencies output by mixer circuits given input frequencies; e.g. pose problems in the form of the "Telephony up/down converter circuits" and "High-side versus low-side injection" Quantitative Reasoning questions.

- **Outcome** – Identify modulation types from oscillographs

  Assessment – Identify the type of signal modulation applied to a carrier signal based on oscillograph displays of that modulated carrier; e.g. pose problems in the form of the "Identifying modulation types" Diagnostic Reasoning question.

- **Outcome** – Independent research

  Assessment – Locate mixer or modulator datasheets and properly interpret some of the information contained in those documents including bandwidth, DC power requirements, RF power limits, etc.

  Assessment – Read and summarize in your own words reliable historical documents on the development of radio modulation and heterodyne receiver techniques. Recommendations include any writing on the work of Reginald Fessenden who invented heterodyne reception.

Chapter 2

Case Tutorial

The idea behind a *Case Tutorial* is to explore new concepts by way of example. In this chapter you will read less presentation of theory compared to other Tutorial chapters, but by close observation and comparison of the given examples be able to discern patterns and principles much the same way as a scientific experimenter. Hopefully you will find these cases illuminating, and a good supplement to text-based tutorials.

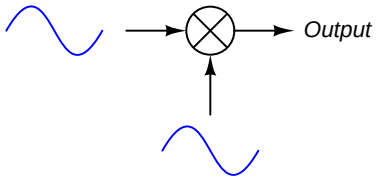
These examples also serve well as challenges following your reading of the other Tutorial(s) in this module – can you explain *why* the circuits behave as they do?

2.1 Example: balanced mixer frequencies

A *balanced mixer* is a circuit designed to multiply the instantaneous values of two input signals together, resulting in an output signal containing the *sum* and *difference* frequencies of those input signals. In an ideal world, such a mixer's output would *only* contain those sum and difference frequencies. In the real world there will always be some *leakage* of the original input frequencies into the output, as well as leakage of one input signal into the other input port, and even artifacts such as harmonics of those input frequencies owing to internal signal distortion within an imperfect mixer.

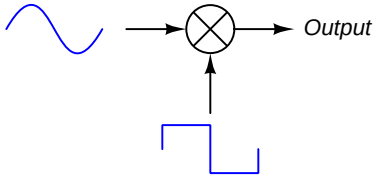
For the following mixer examples we will assume perfectly balanced mixing with total isolation between channels, meaning no leakage of signals between ports on the mixer whatsoever. Such a perfect mixer should only produce sum and difference frequencies at its output port and nothing else.

Consider the following cases of two sinusoidal signals input to a balanced mixer:



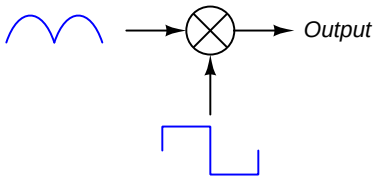
- **Input sinusoids:** 40 kHz and 55 kHz – **Output signal contains:** 15 kHz and 95 kHz
- **Input sinusoids:** 3 kHz and 70 kHz – **Output signal contains:** 67 kHz and 73 kHz
- **Input sinusoids:** 2 MHz and 30 kHz – **Output signal contains:** 1.97 MHz and 2.03 MHz

Consider the following cases of a sinusoidal signal mixed with a square wave:



- **Input sinusoid:** 10 kHz ; **Input square:** 25 kHz – **Output signal contains:** 15 kHz, 35 kHz, 65 kHz, 85 kHz, 115 kHz, 135 kHz, etc.
- **Input sinusoid:** 2 kHz ; **Input square:** 10 kHz – **Output signal contains:** 8 kHz, 12 kHz, 28 kHz, 32 kHz, 48 kHz, 52 kHz, etc.
- **Input sinusoid:** 1 MHz ; **Input square:** 5 kHz – **Output signal contains:** 0.995 MHz, 1.005 MHz, 0.985 MHz, 1.015 MHz, 0.975 MHz, 1.025 MHz, etc.

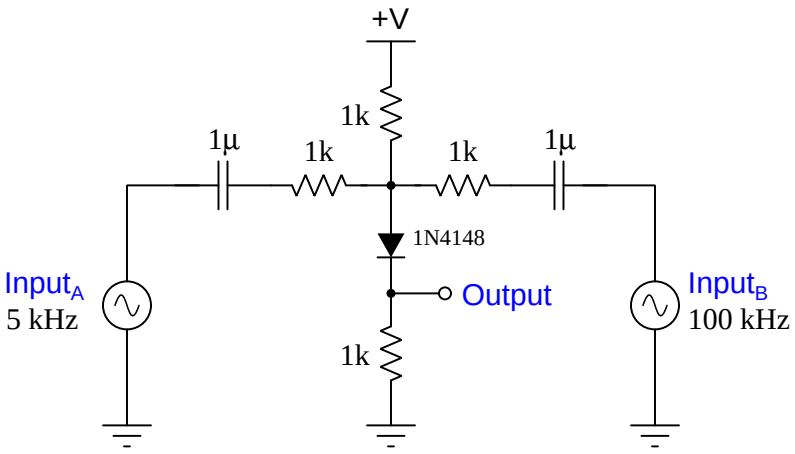
Consider the following cases of two non-sinusoidal signals mixed together – a non-symmetrical waveform containing both odd and even harmonics, mixed with a square wave which of course only contains odd harmonics:



- **Input non-symm:** 8 kHz ; **Input square:** 1 kHz – **Output signal contains:** 7 kHz, 9 kHz, 5 kHz, 11 kHz, 3 kHz and 13 kHz, etc. ; 15 kHz, 17 kHz, 13 kHz, 19 kHz, 11 kHz, 21 kHz, etc. ; 23 kHz, 25 kHz, 21 kHz, 27 kHz, 19 kHz, 29 kHz
- **Input non-symm:** 40 kHz ; **Input square:** 2 kHz – **Output signal contains:** 38 kHz, 42 kHz, 34 kHz, 46 kHz, 30 kHz, 50 kHz, etc. ; 78 kHz, 82 kHz, 74 kHz, 86 kHz, 70 kHz, 90 kHz, etc. ; 118 kHz, 122 kHz, 114 kHz, 126 kHz, 110 kHz, 130 kHz, etc.

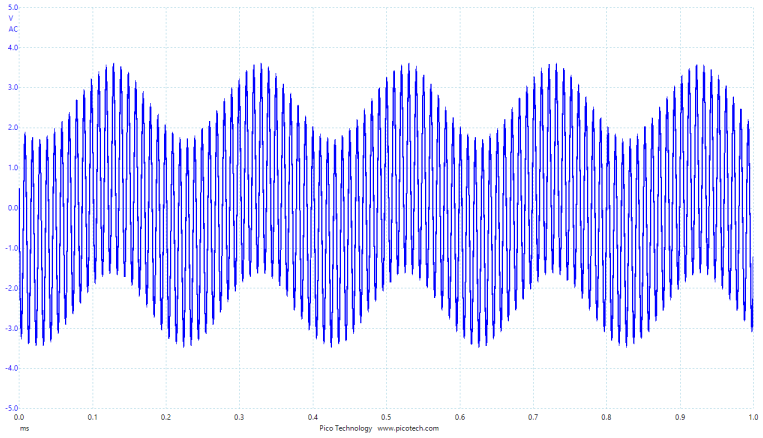
2.2 Example: simple diode mixer circuit

In this circuit two AC voltage signals (5 kHz and 100 kHz) are mixed using a single diode as the nonlinear element. An adjustable DC voltage source (+V) provides biasing for the diode, allowing us to explore the mixing behavior for different DC operating points:

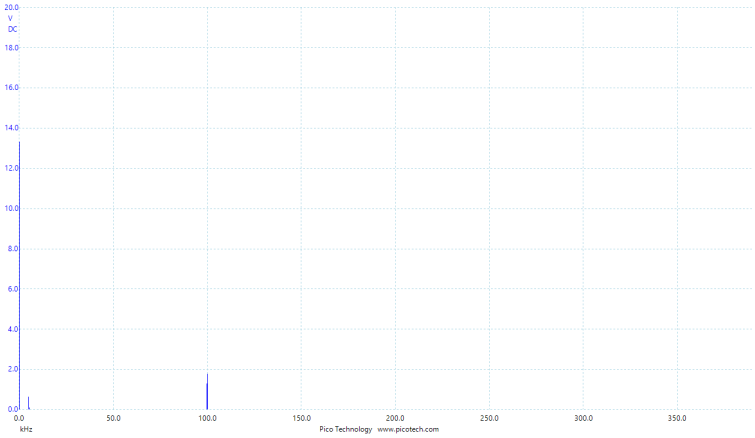


With a strong DC bias – strong enough to keep the diode continuously conducting – the signal we see at the output terminal is not "mixed" properly, but is merely the sum of the two waveforms:

Time-domain view



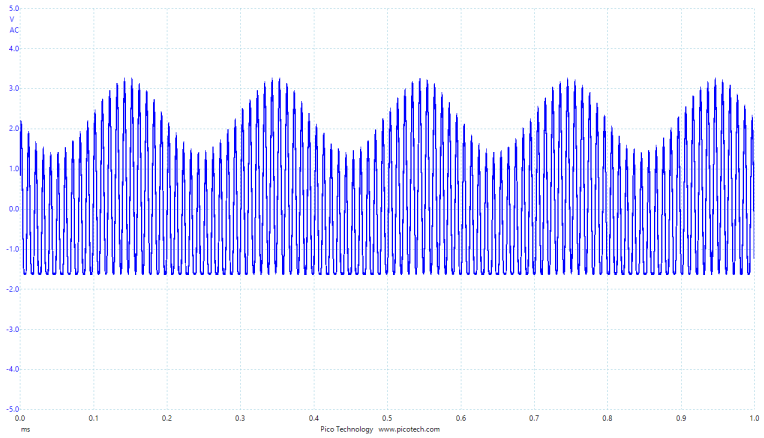
Frequency-domain view



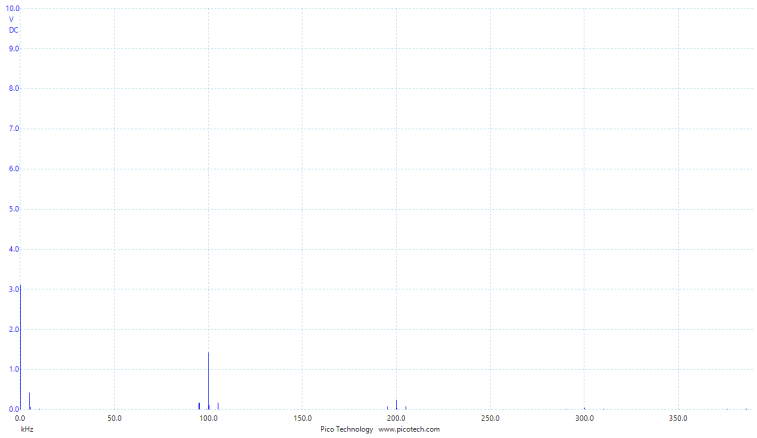
The spectrum display shows a single peak at 100 kHz (signal "B") and a single peak at 5 kHz (signal "A"), with no real mixing. A DC peak also appears at the far-left end of the spectrum, representing the DC bias voltage dropped across the output resistor.

With a weaker DC bias the diode stops conducting during portions of the cycle, the result being a "clipped" time-domain waveform that actually exhibits variations in amplitude (i.e. amplitude-modulation). The resulting spectrum shows sidebands at 95 kHz and 105 kHz in addition to the original signal peaks at 5 kHz and 100 kHz:

Time-domain view



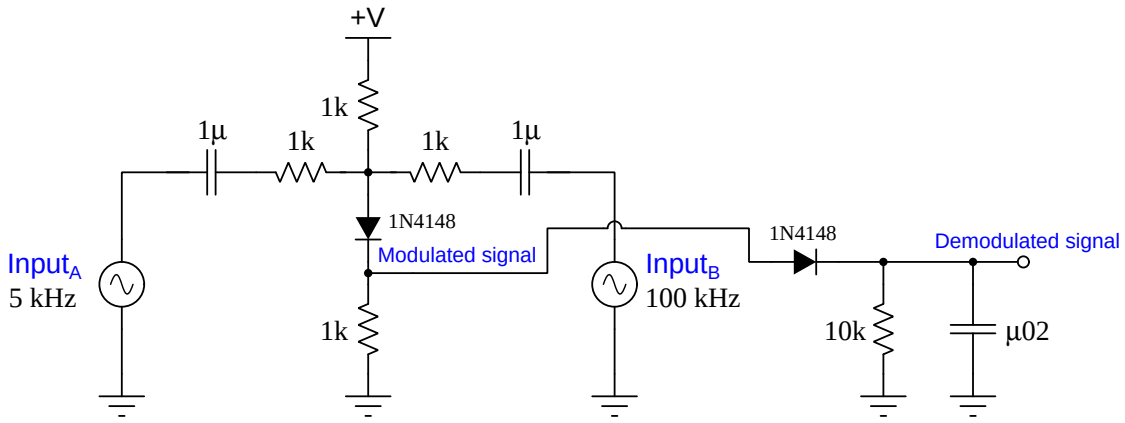
Frequency-domain view



Now we see the original DC, 5 kHz, and 100 kHz peaks as before, but with the addition of sidebands surrounding the 100 kHz carrier peak: one at 95 kHz and another at 105 kHz. Interestingly, we also see smaller peaks at 195 kHz, 200 kHz, and 205 kHz. The 200 kHz peak represents the second harmonic of the 100 kHz carrier signal, generated by the diode's clipping of that sinusoidal signal (making it asymmetrical, therefore containing even-numbered harmonics). This second harmonic frequency of 200 kHz also gets mixed with the 5 kHz signal to produce 195 kHz and 205 kHz sidebands.

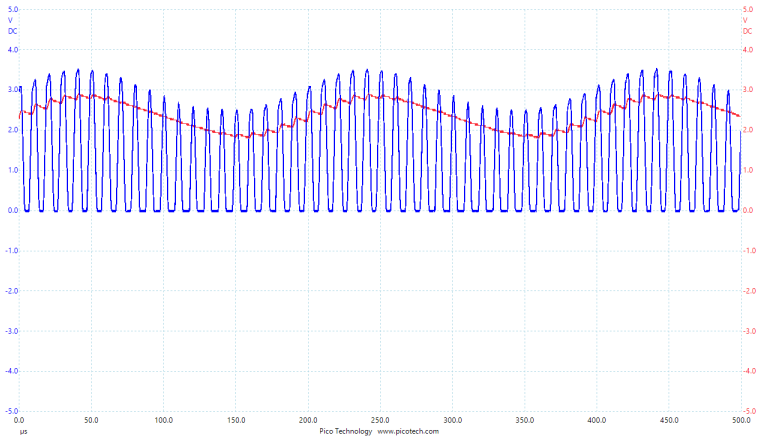
2.3 Example: simple diode demodulator circuit

Taking the same simple diode mixer circuit shown in the previous section, we may build another diode network to demodulate the AM-encoded information:



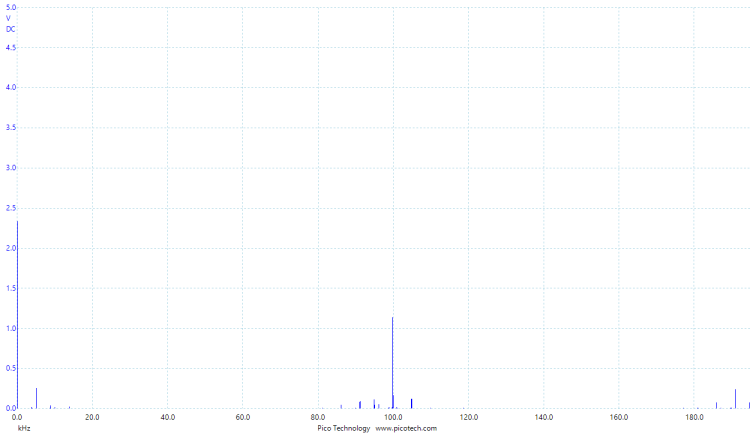
Examining the modulated (blue) and demodulated (red) signals in the time domain, we see the rectifying-filtering action of the diode and RC network as it samples the envelope of the modulated signal, yielding the original 5 kHz waveform (with some distortion) and largely stripped of the 100 kHz carrier:

Time-domain view

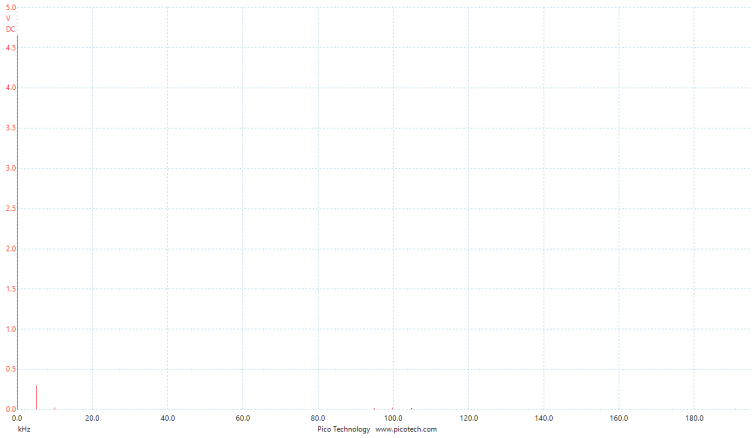


Next we will view the frequency-domain spectra of these two signals, one at a time so as to clearly differentiate between the modulated signal (blue) and the demodulated signal (red):

Frequency-domain view (modulated signal)



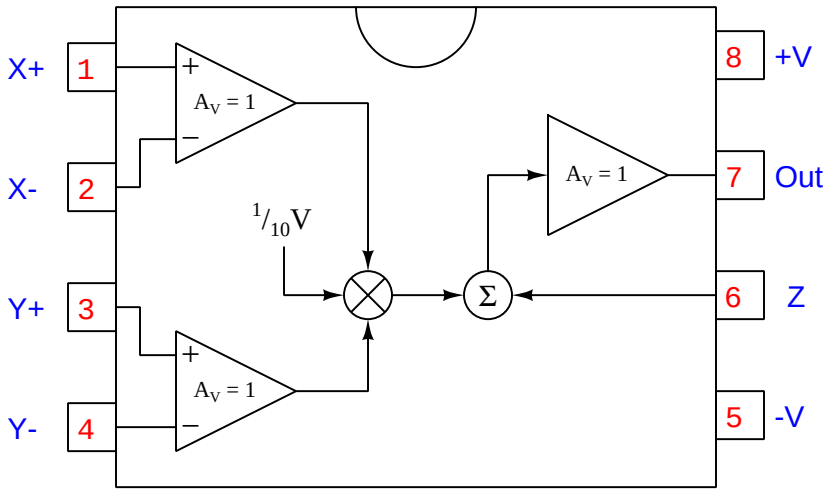
Frequency-domain view (demodulated signal)



As you can see, the 5 kHz peak remains in the demodulated spectrum, while the 100 kHz carrier and its sidebands are all but vanished. Interestingly, the DC peak is higher in the demodulated signal than in the modulated signal as a result of the filtering action of the capacitor in the diode demodulator network.

2.4 Example: AD633 as a balanced mixer

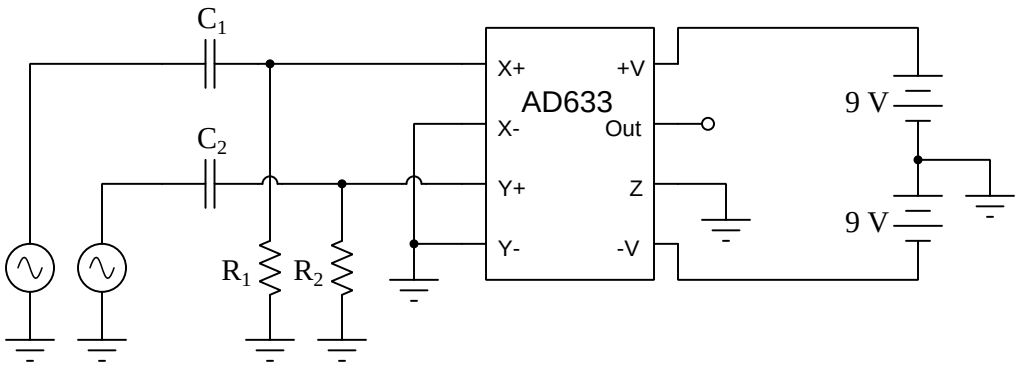
The Analog Devices AD633 analog multiplier IC works well as a balanced mixer for audio frequencies. An internal block diagram is as follows:



Two unity-gain amplifiers receive the analog signals to be multiplied (in our case, the baseband signal X and carrier signal Y we intend to mix), pass it to the multiplying cell, and then an optional offset is added to that product signal and buffered for the output. Its transfer function is:

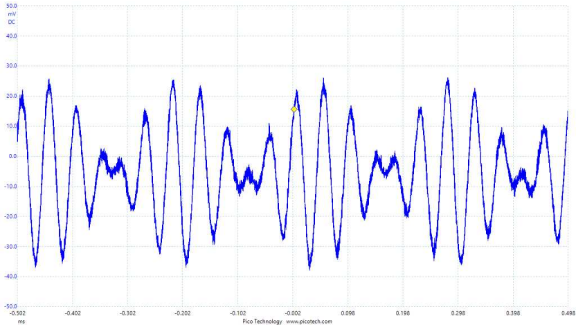
$$Out = \frac{(X_+ - X_-)(Y_+ - Y_-)}{10 \text{ Volts}} + Z$$

Here we will use the AD633 to amplitude-modulate (AM) a 20 kHz carrier signal with a 2 kHz baseband signal, using a pair of 9-Volt batteries to form a split DC power supply, and coupling capacitors to ensure no DC offset (bias) voltage from either signal source reaches the multiplier:

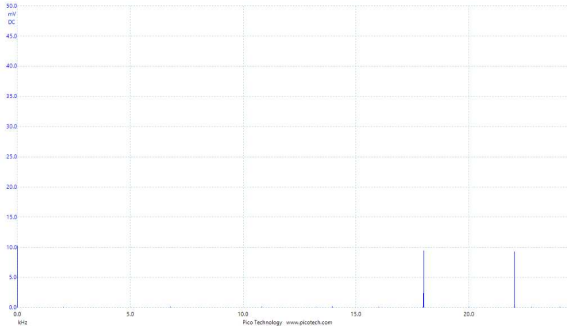


The output signal is then measured in both the time and frequency domains, with the 2 kHz baseband signal and the 20 kHz carrier:

Time-domain (oscilloscope) plot:

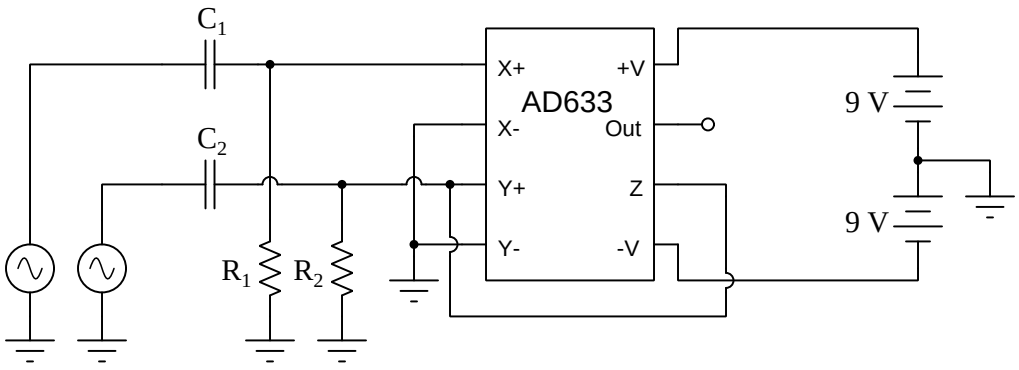


Frequency-domain (spectrum analyzer) plot:

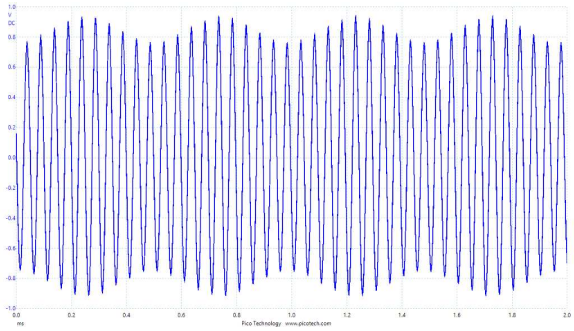


What we see here is what we'd expect for a balanced mixer circuit: just two sidebands with no trace of the original baseband or carrier signals in the output spectrum.

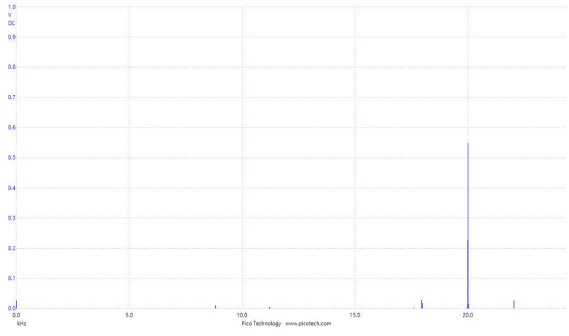
If we make a small modification to the circuit, connecting the carrier signal (Y) to the offset input (Z) so that the carrier will intentionally "leak" into the output spectrum, we see a more traditional AM result:



Time-domain (oscilloscope) plot:

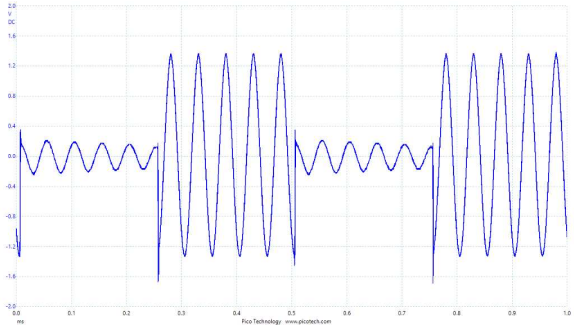


Frequency-domain (spectrum analyzer) plot:

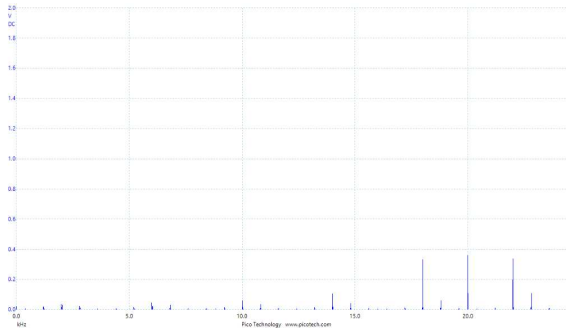


Lastly, we will substitute a 2 kHz square-wave as the baseband signal, while retaining the 20 kHz sine-wave as the carrier:

Time-domain (oscilloscope) plot:



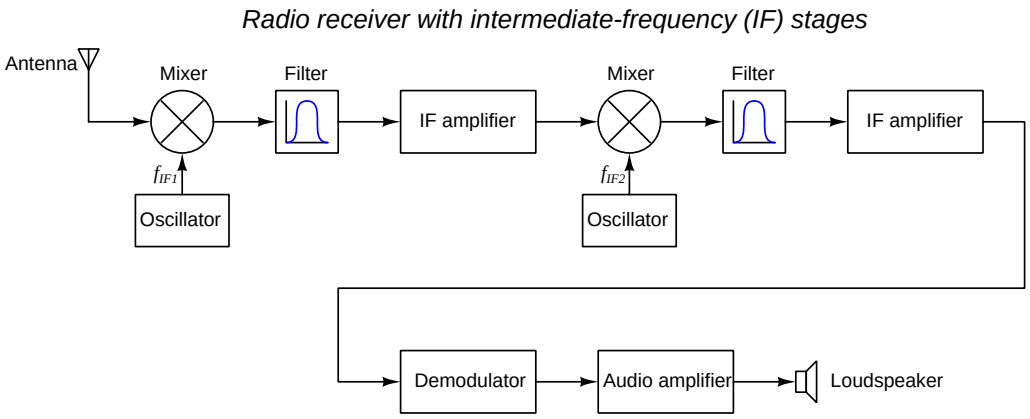
Frequency-domain (spectrum analyzer) plot:



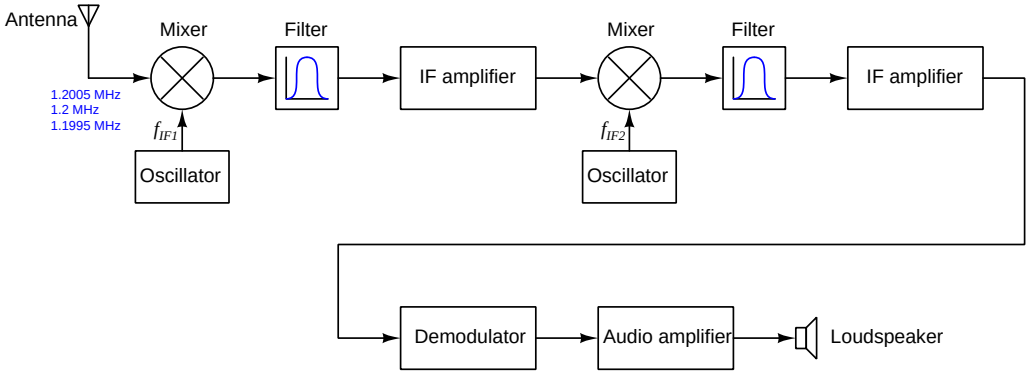
Note how many more sidebands we see due to the harmonics intrinsic to the square wave.

2.5 Example: downconverting AM receiver

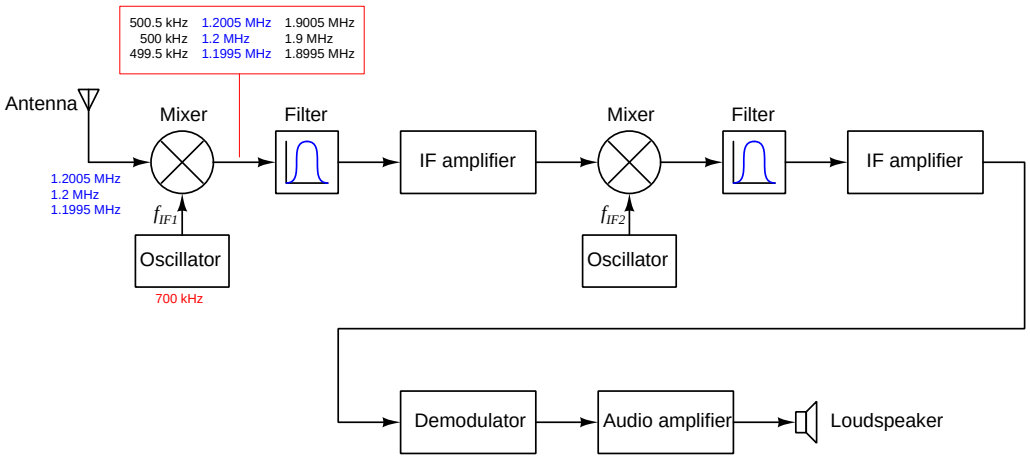
The following block diagram shows a radio receiver using multiple stages of frequency downconversion:



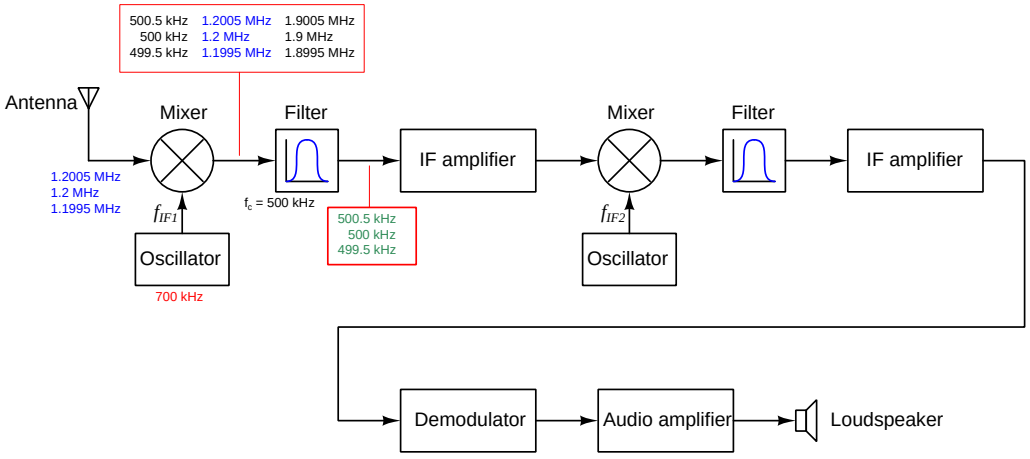
Let's assume the antenna picks up a simple AM signal with a carrier frequency of 1200 kHz modulated by a simple sine-wave tone of 500 Hz. This means the incoming signal consists of a 1.2 MHz carrier as well as 1200500 Hz and 1199500 Hz sidebands:



If we set the first local oscillator to a frequency of 700 kHz, that oscillator's frequency will create sum and difference frequencies with the carrier and sidebands of the antenna's signal. This collection of frequencies resulting from the first-stage mixer and local oscillator are shown on the diagram inside the red box. Each of the frequencies from the antenna are shown in blue, while each of the sum and difference frequencies resulting from mixing with the 700 kHz local oscillator are shown to either side of the blue figures in black text:



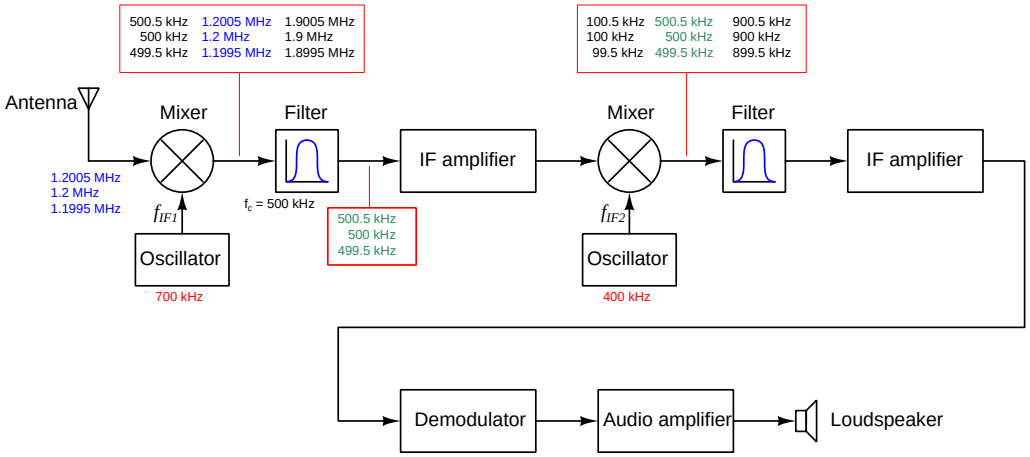
Our goal here is *frequency downconversion*, which means we're interested in keeping the lowest of these frequencies and discarding the rest. This means we'll need to tune the center-frequency of the first band-pass filter to 500 kHz, allowing it to pass that low frequency (in addition to the two immediately-adjacent frequencies) while blocking all others. The filter's output frequencies are shown in green text, inside another red box:



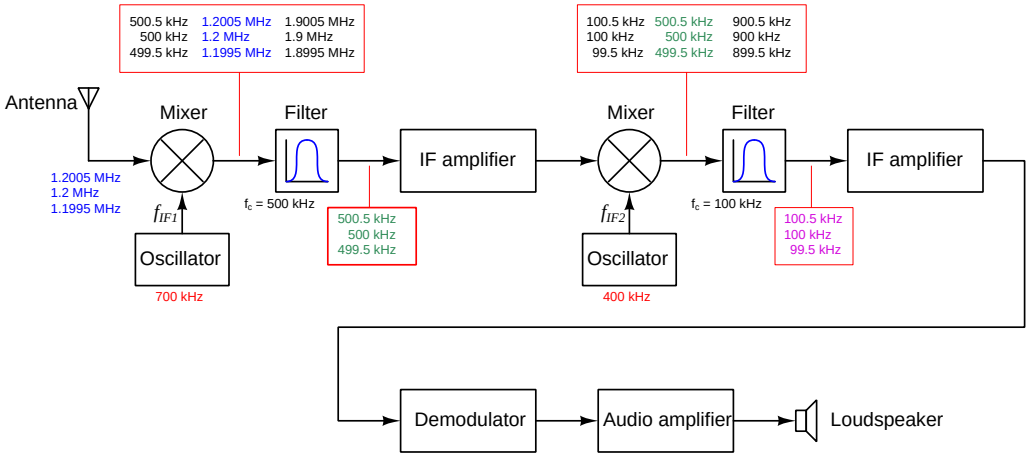
Note how the sidebands in the downconverted frequency spectrum are still ± 500 Hz on either

side of the center frequency. This means the essential information contained in the broadcast 500 Hz sine-wave audio tone is still present in the downconverted signal, just every frequency in that spectrum being shifted lower than they were at the antenna.

Next we amplify these signals and pass them through another mixer with its own local oscillator set to 400 kHz, resulting in another group of sum and difference frequencies shown below:



The next band-pass filter will have its center-frequency set to 100 kHz to pass through the lowest of these while blocking the rest:



Once again, the essential information contained in the original 500 Hz audio tone is still present in this set of downconverted frequencies, as we still have sidebands that are $\pm$ 500 Hz away from the center frequency. This shows no important information was lost in the frequency downconversion process.

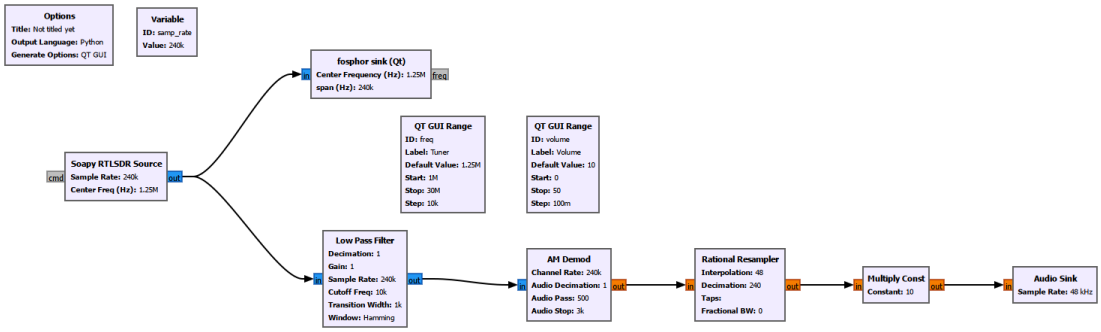
The final-stage IF amplifier then boosts these signals' amplitude and feeds them to the AM demodulator which then extracts the 500 Hz audio information from the modulated (100 kHz carrier) signal and sends that audio-frequency signal to the audio amplifier and finally to the loudspeaker.

Frequency downconversion allows each IF amplifier stage to operate on a relatively narrow bandwidth of signals which simplifies their design and improves gain performance. Likewise, the demodulator can now operate at a much lower carrier frequency than the original signal captured by the antenna, simplifying its design and improving its performance as well.

2.6 Example: GNU Radio Companion SDR demodulation flowgraphs

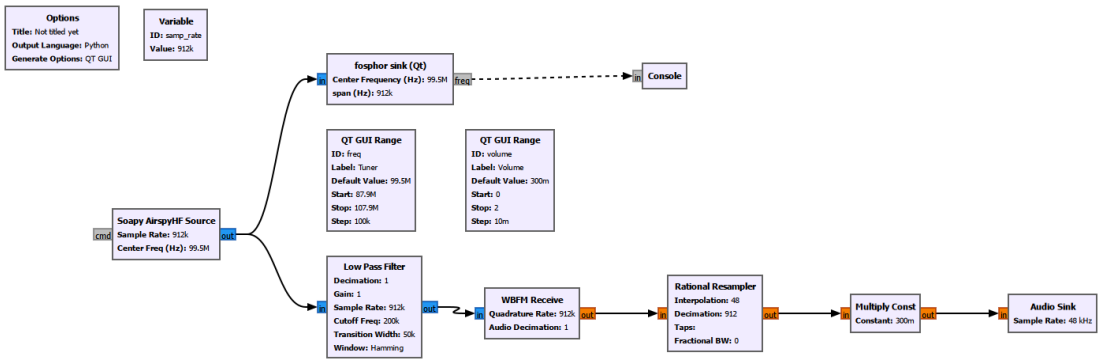
`GNU Radio Companion` is free and open-source computer software useful for creating digital signal processing systems necessary to decode/demodulate the I-Q data received by a software-defined radio (SDR) unit. While the actual code running in `GNU Radio` is either Python or C++, the "Companion" side of this software allows the user to place graphical function blocks on the computer screen which are then translated into appropriate Python or C++ code to implement the digital signal processing functions. These block diagrams are called *flowgraphs* in the lingo of `GNU Radio Companion`.

Broadcast AM receiver flowgraph



This flowgraph takes in I-Q data from an RTL-SDR receiver dongle via a USB cable connection to the computer, using the `Soapy RTLSDR Source` function block with its sampling rate set for 240 kHz (as the `samp_rate` variable within a `Variable` block). The complex data stream is fed to a `fosphor sink` display showing a frequency spectrum and "waterfall" signal strength history, as well as to a digital low pass fiter block cutting off at 10 kHz (to capture audio signals typically broadcast in AM). The `freq` variable sets the center tuning frequency for the RTL-SDR source as well as for the `fosphor` display sink, this variable being adjustable as a "Tuner" slide control appearing adjacent to the spectrum/waterfall display when the flowgraph is compiled and running. An `AM Demod` function block demodulates the AM signal and converts it from complex data into floating-point data, which is then passed to a `Rational Resampler` function block which interpolates and decimates the signal to downconvert its sample rate to 48 kHz so as to be compatible with most personal computer audio subsystems. Another GUI slide control sets the `volume` variable which is referenced within the `Multiply Const` function block to provide variable amplification/attenuation of the audio-frequency signal. An `Audio Sink` function block passes this audio signal sampled at 48 kHz to the computer's audio output.

Broadcast (wideband) FM receiver flowgraph



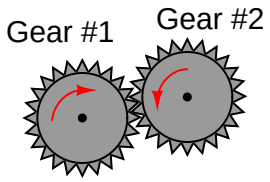
This flowgraph is built for a different SDR dongle, in this case an *AirspyHF* which has greater resolution than the RTL-SDR but not as wide of an RF frequency range. A `Soapy AirspyHF Source` function block handles the interfacing with the USB-connected SDR receiver hardware and outputs a complex-number I-Q data stream. Much of this flowgraph is the same as the AM receiver previously discussed, the major difference of course being a wideband FM demodulator function block called `WBFM Receive`, and a higher sample rate suitable for the AirspyHF SDR unit.

2.7 Example: frequency-mixing in gear systems

Rotating machinery produces vibrations whose frequencies are a function of the rotating components' speeds. Rotary speed is customarily expressed in units of *revolutions per minute* or *RPM*, though frequency is more commonly expressed in *cycles per second* or *Hertz*.

If a rotating component is off-balance, such that it produces a vibration oscillating once per rotation, the component's rotating speed divided by 60 yields the fundamental frequency of that vibration, since there is one cycle of vibration per rotation and 60 seconds within every minute of time. For example, an off-balance shaft rotating at 1800 RPM generates a fundamental vibration at 30 Hz.

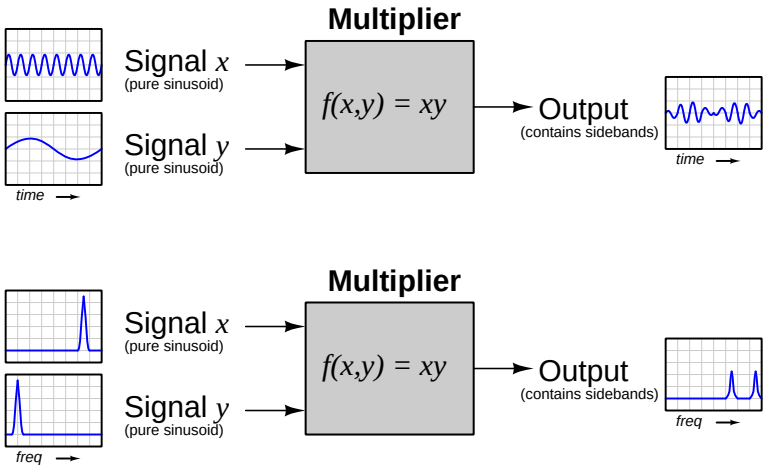
When rotating *gears* mesh together, the repeated contact between those gears' teeth produces vibrations as well, except these vibrations' fundamental frequency is equal to the shaft rotational speed multiplied by the number of teeth on the gear. Consider the gear-set shown in the illustration below, having 24 teeth each. If they both rotate at 1800 RPM, then the frequency of their teeth must be 24 times 30 Hz, or 720 Hz:



Something very interesting occurs if the shaft supporting one of these gears is slightly bent so that the gears mesh tightly during parts of their rotation but loosely during others. This will have the effect of modulating the intensity (amplitude) of the tooth vibration once per shaft revolution, so that the high-frequency vibrations produced by the teeth grow louder and softer at a frequency equal to the shaft speed. One can think of the bent shaft as imposing a sinusoidal *multiplying* factor to the tooth vibrations, which is mathematically the same as *mixing* applied to electronic signals.

Like all multiplicative mixing processes, a pair of gears suffering a bent shaft will produce vibrations at the shaft frequency (f_{shaft}), at the tooth frequency (f_{tooth}), and *sideband* frequencies of $f_{tooth} - f_{shaft}$ and $f_{tooth} + f_{shaft}$. The intensity of these sidebands as viewed on a frequency-domain plot (i.e. spectrum analyzer display) indicate the depth of the amplitude-modulation and therefore the severity of the shaft's bend.

Instead of this frequency-mixing occuring within an electronic multiplier circuit, it occurs naturally between the gears as a mechanical phenomenon. If we consider the "multiplier" in the following block diagrams to represent this natural function of meshing gear teeth, with signal x being the tooth noise and signal y being the bent shaft's oscillation, we may visualize the effects of this mixing in both the time domain (upper) and frequency domain (lower):



Machinery health experts know to look for sidebands and other such effects when analyzing the vibrations of complex mechanisms, and realize their diagnostic value in locating causes of abnormal vibration signatures.

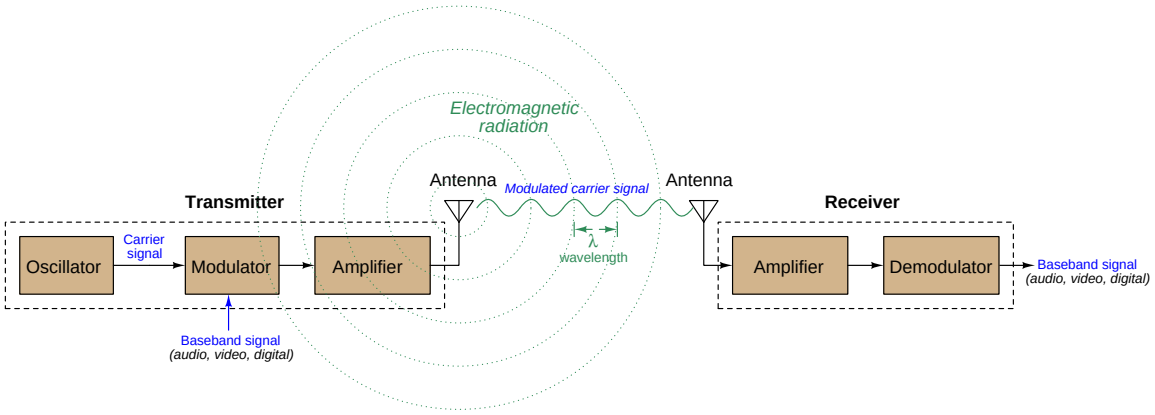
Chapter 3

Tutorial

3.1 Introduction to modulation

Many types of electronic recording and communication systems make use of an important concept called *modulation*, whereby information is impressed in one form or another upon a relatively high-frequency sinusoidal waveform. *Demodulation* is the reverse process, whereby the original information is extracted from the modulated signal. The signal representing the information in its raw form is called the *baseband*, while the high-frequency sinusoid is called the *carrier*.

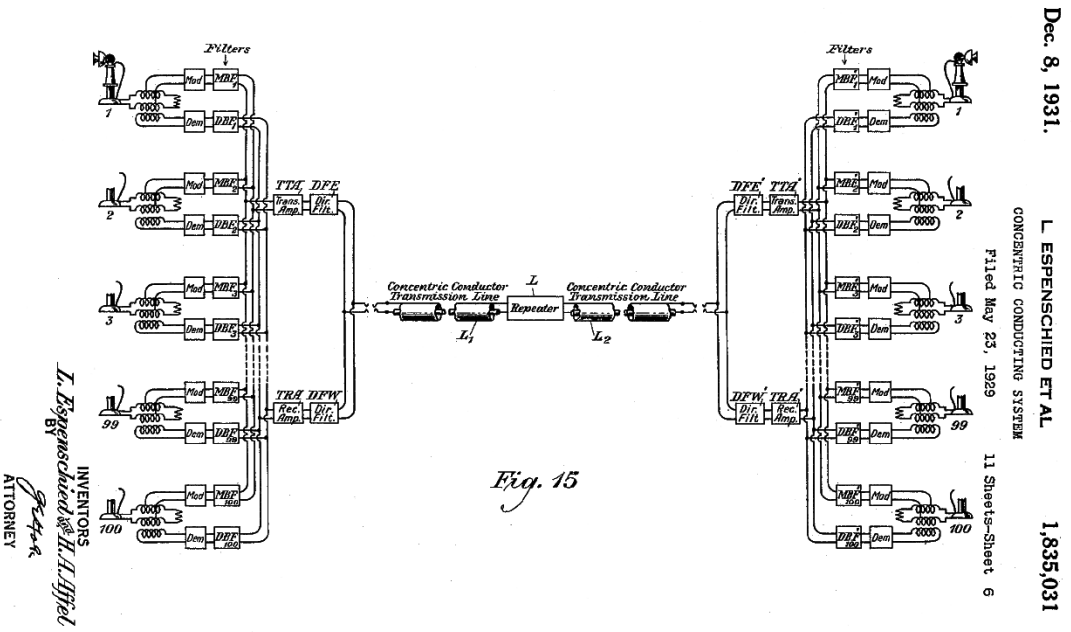
Both the basic concept of modulation and the need for modulation may be understood through some practical examples. First, consider the phenomenon of *radio communication*: where information such as human speech and music may be conveyed long distances through empty space by means of electromagnetic waves. The production of those electromagnetic waves relies on energizing an *antenna* structure with alternating current (AC) of such frequency that the physical length of the antenna elements will be some simple fraction of the wave's length. For any antenna of practical dimensions, this requires extremely high AC frequencies, far greater than the frequencies audible to human ears[1]. Thus, the physical requirements of electromagnetic radiation necessitates *modulation* of a high-frequency "carrier" signal suitable for the antenna with the "baseband" speech or music we wish to broadcast. A similar application of modulation for the sake of the communication channel is when we must convey an inherently analog (i.e. continuously-variable) signal over a digital channel limited to "high" and "low" logic states – in such cases we would choose a form of pulse-based modulation where the *timing* of the digital (high/low) pulses convey the analog data.



[1] For example, a "whip" antenna consisting of a straight metal wire positioned perpendicular to the Earth's surface must be energized by AC having a wavelength approximately four times that of the wire's length. A "whip" antenna two meters in length would therefore require AC with a wavelength of eight meters. Calculating frequency (f) based on the given wavelength (λ) and the speed of light in open space ($c \approx 3 \times 10^8$ m/s), we obtain $f = \frac{c}{\lambda} = 37.5$ MHz. Thus, we would need a carrier frequency of 37.5 *million* cycles per second in order to effectively drive this antenna to produce radio waves. By contrast, the human range of hearing extends only from approximately 20 Hz to 20 kHz.

A second application of modulation is as a means to *multiplex* different signals over a common communication channel. Consider a "trunked" telephone system where a single pair of conductors within a cable must carry multiple audio conversations with no "crosstalk" or other interference between those conversations. If we take the audio information from each conversation and use it to modulate carrier signals of widely different frequency, the modulated signals may be easily filtered from one another which enables them to be combined together at the transmitting end of the trunk cable and separated from one another at the receiving end.

A United States patent granted in the year 1931 (US patent number 1,835,031 for Lloyd Espenschied and Herman Affel) shows such a trunked telephone system, with modulating ("Mod") and demodulating ("Dem") circuits at each telephone unit interfacing it with the larger system. A single coaxial-style signal cable ("concentric conductor transmission line") conveys a multitude of simultaneous telephone conversations over long distances to other telephone units:



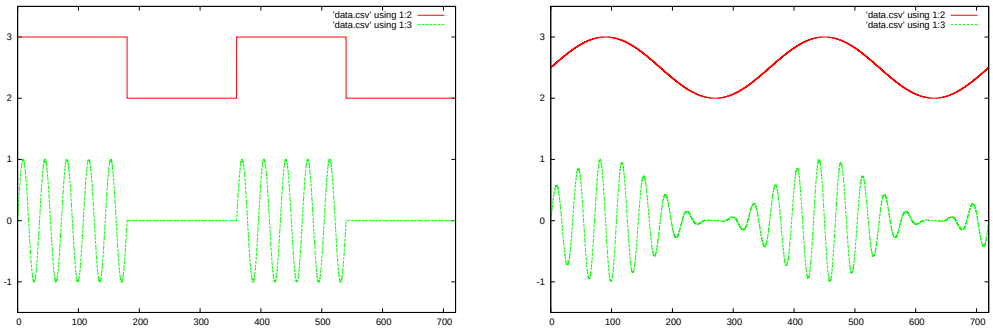
Signals in this telephone system are routed to their proper destinations through the use of band-pass filters ("BF"), each one tuned to a unique carrier frequency so as to pass only one audio data stream while blocking all others. Such band-pass filtering would be useless in separating baseband (audio) signals from one another since all humans speak within the same narrow range of frequencies. Only by modulating carrier signals with frequencies far above what humans can hear is this system able to assign individual vocal tones to specific frequencies that may be discriminated by the use of filters.

For example, consider one telephone conversation modulating a 50 kHz carrier signal and another conversation modulating a 75 kHz carrier signal. The audio frequencies of those two conversations would surely overlap the same range of frequencies and thus be inseparable by band-pass filtering if they were to be superimposed on the same cable, but their respective modulated carriers – being

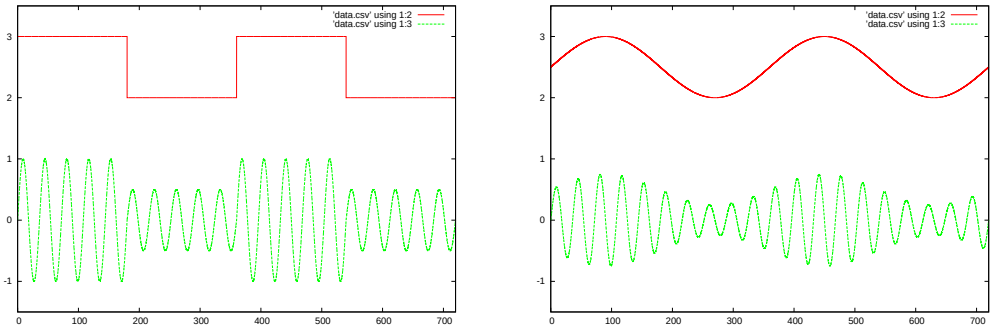
separated from one another by 25 kHz – are easily separated by band-pass filters (one tuned to 50 kHz and the other to 75 kHz). This enables those two modulated-carrier signals (as well as many others) to be superimposed on the common coaxial cable and later separated from each other into individual conversations by filtering.

3.2 Amplitude modulation

One of the most basic forms of modulation is *amplitude modulation*, abbreviated simply as *AM*. This is where the baseband signal controls the amplitude of a higher-frequency carrier signal. Examples of amplitude modulation using both digital (left, also known as *On-Off Keying* or *OOK*) and analog (right) baseband signals appear below, with the baseband signal shown in red and the modulated carrier in green:



Both of these examples show amplitude modulation with 100% *depth*: that is, the amplitude of the modulated carrier extends all the way to zero. Lesser modulation depths are also possible, as shown below with examples of 50% amplitude modulation depth (the digital variety on the left also known as *Amplitude Shift Keying* or *ASK*):



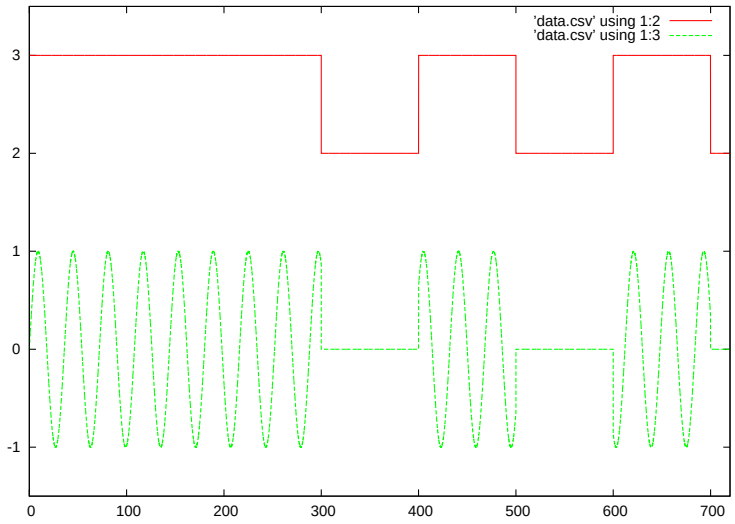
Amplitude modulation used in the context of radio communication utilizes a carrier frequency high enough to radiate electromagnetic energy from an antenna structure, while the baseband signal controls the strength of those emissions over time. An AM radio receiver works by detecting the "envelope" of the received signal which represents the original (baseband) audio information; i.e. the music and speech broadcast by an AM transmitter. In the analog AM examples shown above, this baseband data is a simple sinusoid. However, the baseband data may be as complex as one desires, so long as the highest-order harmonic frequencies are well below that of the carrier.

The earliest forms of AM radio communication were digital in nature, the baseband data consisting of a human operator pressing a "key" switch on and off according to the patterns defined by *Morse code*. This was simply called *continuous wave* (CW) communication, where the carrier signal was either on or off.

International Morse Code (English letters and Arabic numerals only)

A •—	J •———	S •••	0 —————
B —•••	K —•—	T —	1 •————
C —•—•	L •—••	U ••—	2 ••———
D —••	M ——	V •••—	3 •••——
E •	N —•	W •——	4 ••••—
F ••—•	O ———	X —••—	5 •••••
G ——•	P •——•	Y —•——	6 —••••
H ••••	Q ——•—	Z ——••	7 ——•••
I ••	R •—•		8 ———••
			9 ————•

For example, the following graph shows the CW signal for the Morse code sequence representing the letter "D" using On-Off Keying (OOK) which is just Amplitude Shift Keying (ASK) with 100% modulation depth:

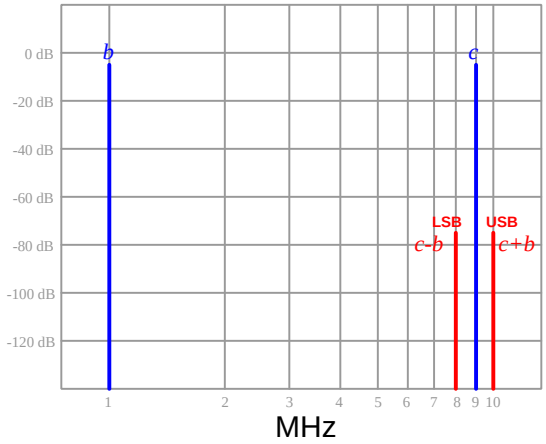


When viewed from a frequency-domain perspective, the result of amplitude-modulating a carrier frequency with some lower baseband frequency are two new frequencies called *sidebands*. If c represents the carrier frequency and b represents the baseband frequency, the two sideband frequencies will be the sum and the difference between the two:

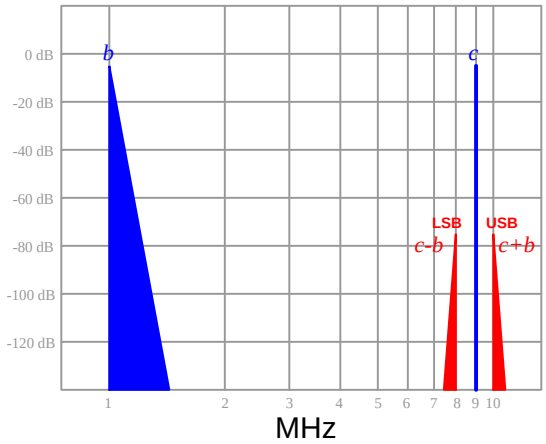
$$\text{Upper sideband frequency} = c + b$$

$$\text{Lower sideband frequency} = c - b$$

To illustrate, the spectrum of an AM signal generated from 9 MHz carrier (c) and 1 MHz baseband (b) sinusoids reveals a lower sideband (LSB) frequency of 8 MHz and an upper sideband (USB) frequency of 10 MHz:



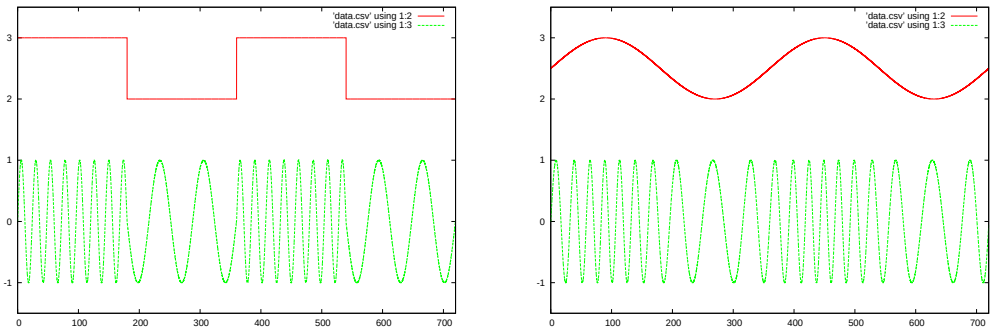
If the baseband signal is a complex mix (or *band*) of frequencies rather than a single sinusoid, the sidebands will be replicas of the baseband signal's spectrum profile:



The amplitude of these sidebands depends on the depth of modulation: the greater the depth, the stronger the sidebands. If the baseband signal amplitude falls to zero, only the carrier will exist at the output (with no sidebands at all). This suggests only the sidebands contain the information represented by the original (baseband) signal – the carrier conveys no information of its own. For this reason, some AM systems are designed to suppress the carrier, and often one of the sidebands as well, in an effort to occupy less bandwidth over a crowded communication channel and to expend less energy transmitting the same information. The term "single sideband" (SSB) describes one such system, where both the carrier and one of the two sidebands is completely eliminated from the output of the system.

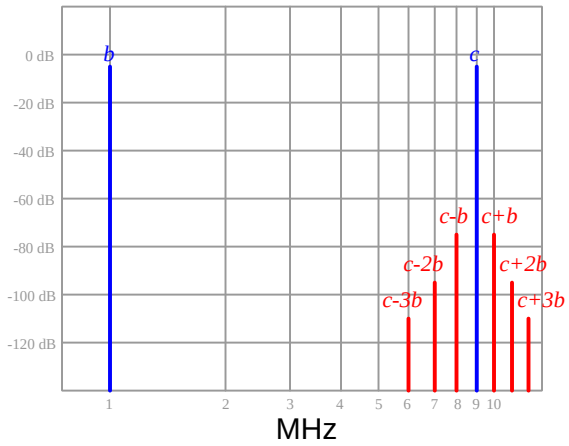
3.3 Frequency modulation

With *frequency modulation*, abbreviated simply as *FM*, the baseband signal controls the frequency of the carrier signal. Examples of frequency modulation using both digital (left, also known as *Frequency Shift Keying* or *FSK*) and analog (right) baseband signals appear below. Again, the baseband signal is shown in red and the modulated carrier in green:



Compared to amplitude modulation, frequency modulation enjoys greater immunity to noise but occupies more bandwidth. The noise immunity comes from the fact that the baseband's information is not contained in the modulated signal's amplitude at all, and therefore any corruptions to that amplitude will have no effect on the interpreted signal. A thought experiment helps illustrate this concept of noise immunity. Imagine AM and FM radio signals broadcast during a lightning storm. When lightning strikes, the resulting pulse of electromagnetic energy will appear as a transient overlaid on the RF waveform. An AM radio receiver will interpret this transient as part of the audio information because it views the "envelope" of the modulated wave as the encoded information, and the transient pulse is part of that envelope. The FM radio receiver, however, only looks at changes in the received signal's frequency as representing audio information and ignores amplitude. Therefore, the lightning strike will register as noise on the AM receiver but not on the FM receiver.

The bandwidth requirements of FM are not as intuitive to understand as its noise immunity. When viewed on a spectrum analyzer, we find that a simple sinusoid baseband signal fed into a frequency modulator produces a *harmonic series* of sidebands to either side of the carrier. The number and relative amplitudes of these harmonics vary with the depth of modulation, as does the amplitude of the carrier[2]:

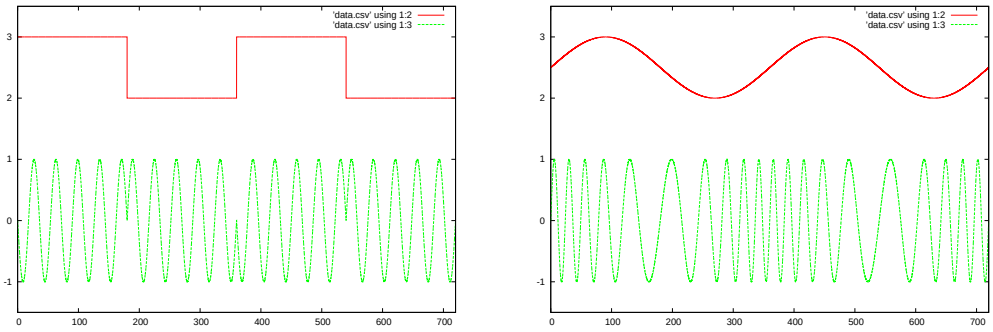


Bear in mind that the array of harmonics shown above is the result of the baseband signal being a simple sinusoid (i.e. a fundamental frequency only, with no harmonics). When the baseband itself consists of a complex spectrum of frequencies (e.g. speech, music, video, digital data), the FM sidebands become even more complex as they contain multiple harmonics *for each of the frequencies contained in the baseband signal*. This is the reason why FM radio broadcast frequencies must be further separated from each other on the electromagnetic spectrum than AM broadcast frequencies, all other factors being equal: the "spreading" of the baseband spectrum into even wider sidebands occupies a larger amount of available spectrum than AM.

[2]In fact, for some modulation depths, the carrier disappears completely leaving only sidebands! This is a desirable goal, as the carrier itself conveys no information and thus constitutes unnecessary energy dissipation. Only the sidebands represent the information being broadcast, and so a sideband-only broadcast is ideal from the perspective of energy efficiency. It is also an advantage for radio communication integrity, as regulatory restrictions limit the total amount of RF power a transmitter is allowed to broadcast, which means a modulation depth resulting in 100% of the energy going into information-carrying sidebands (and 0% going into information-void carrier) gets you more range and more signal-to-noise ratio for any given maximum power limit.

3.4 Phase modulation

Phase modulation, abbreviated simply as *PM*, is where the baseband signal controls the phase angle with respect to the unmodulated carrier, and is very closely related to frequency modulation[3]. Examples of phase modulation using both digital (left, also known as *Phase Shift Keying* or *PSK*) and analog (right) baseband signals appear below:



If you closely compare these phase-modulated waveforms against the frequency-modulated waveforms shown in the previous section, you will note a distinct difference between the digital versions but very little difference between the analog versions. For sinusoidal baseband waveforms, the only difference between PM and FM is where the modulated waveform attains its highest and lowest frequencies: in the analog FM example the slowest frequency occurred at the negative peak of the baseband waveform and the highest frequency occurred at the positive peak; here with analog PM we see the high- and low-frequency limits occurring at the zero-crossing points of the baseband waveform[4].

The spectrum of a phase-modulated signal is not unlike that of a frequency-modulated signal: for a sinusoidal baseband the result is a harmonic series of sideband frequencies to either side of the carrier. However, phase modulation does tend to generate a *wider* set of sidebands than frequency modulation for some baseband signals. This makes sense when we compare the time-domain waveforms for digital PM versus digital FM and see that with PM it is possible to create more abrupt changes in the time domain, and we know that abrupt time-domain waveshapes equate to higher harmonic frequencies.

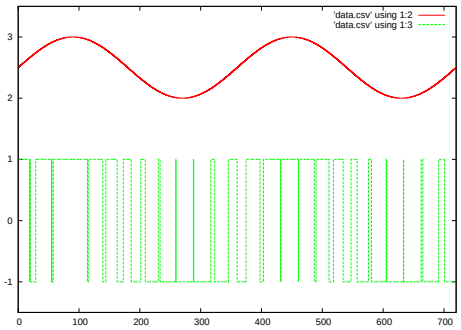
[3] In fact, you can think of FM as being the time-integral of PM, and PM as being the time-derivative of FM. Consider a DC baseband signal's effect on a PM waveform versus an FM waveform: the constant value of this DC modulating signal would cause the PM signal's phase to deviate by a constant angle compared to the unmodulated carrier, but the FM signal would oscillate at a different frequency from the carrier which is equivalent to a phase shift that is constantly growing over time.

[4] As mentioned in the previous footnote, the difference between PM and FM is one of time-integration or time-differentiation of the baseband signal. Recall from elementary calculus that the time-derivative of a sine wave is a cosine wave (i.e. a sine wave phase-shifted by 90 degrees), and so if we compare sinusoidally-modulated PM and FM waveforms we should not be surprised to see this 90 degree phase shift.

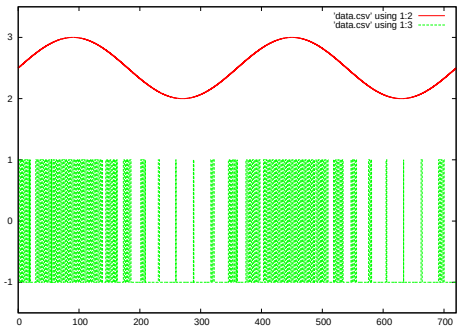
3.5 Pulse modulation

Traditional modulation techniques such as AM, FM, and PM impress the information contained by a baseband signal onto a sinusoidal carrier waveform, but other modulation schemes exist. One such alternative is to modulate baseband data into a series of *pulses* suitable for transmission over a digital medium supporting only high/low states (e.g. optical fiber with gate-driven LEDs as light sources and photodiodes driving logic gates as light detectors).

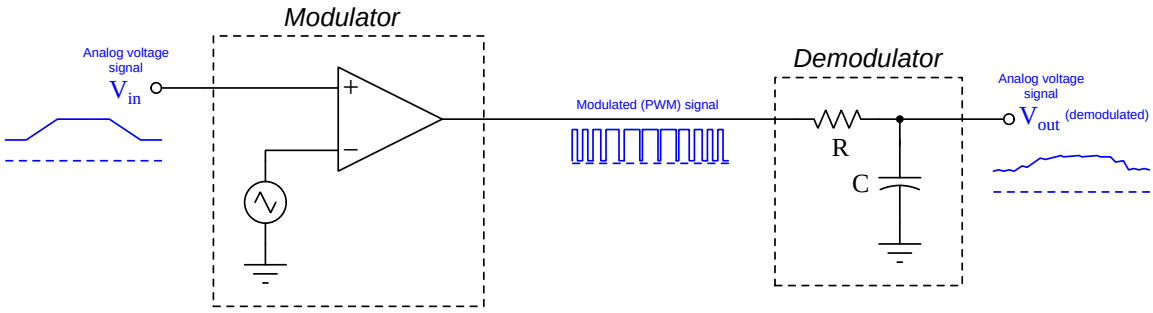
A very common form of pulse-based modulation is *pulse-width modulation*, abbreviated *PWM*. In this modulation technique, the duty cycle of a pulse waveform (but not its frequency) varies with the instantaneous value of the baseband signal. PWM signals are very easy to demodulate, since their time-average value represents the original baseband signal. We often find pulse-width modulation used as a method of electronic power control, rapidly pulsing power on and off to a load, with the load's inherent time lag performing the averaging function necessary to translate the PWM duty cycle into an analog power value.



Closely related to pulse-width modulation is *pulse-density modulation*, abbreviated *PDM*. This is where all pulses are the same width, but the density of their occurrence over time varies with the instantaneous amplitude of the baseband signal. In other words, instead of the pulse duty cycle varying, the pulse *frequency* varies to create approximately the same effect:



Pulse width modulation signals may be produced by comparing the analog baseband signal against a higher-frequency carrier signal using a *comparator*. A simple low-pass filter circuit is all that is necessary to demodulate the PWM signal back into baseband:



Any analog DC voltage applied to V_{in} generates a PWM signal from the comparator which will be "averaged" by the RC low-pass filter (also called a *passive integrator*) to become a "smoothed" pulsing signal resembling the original DC voltage signal. The cutoff frequency of this filter must be chosen to be much lower than the PWM frequency, in order to filter out as much of the harmonics as possible and only leave the baseband signal as V_{out}.

Although different circuitry is necessary to generate a pulse density modulator (PDM), demodulation works just the same: simply use a low-pass filter to extract the baseband information from the modulated carrier.

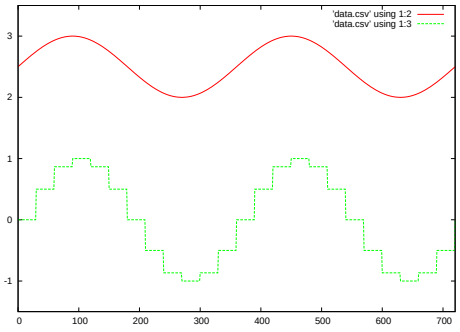
A similar application for either PWM or PDM is extracting an analog output signal from a *microcontroller*[5] or other digital circuit capable only of on/off states. Pulse modulation may be thought of as a clever way to encode analog information in any signal capable only of discrete amplitude states: even though the voltage or current is limited to "on" and "off" states, we may represent a continuously-variable signal in the form of the *duty cycle* ratio between "on" time versus total "on + off" time.

[5] A "microcontroller" is a microprocessor combined with all necessary peripheral circuitry to form a functional computer on a single wafer ("chip") of semiconductor suitable for integrated circuit packaging. Many microcontrollers lack a digital-to-analog converter, but with judicious pulsing of an output pin and a passive low-pass filter network attached, the same effect may be realized: achieving analog signal capability in a device only able to switch between "high" and "low" states.

3.6 Sampling modulation

There are many applications where a low-frequency waveform is represented as a series of "steps" rather than as a smooth and continuous wave. One such application is *analog-to-digital sampling* where a continuously-variable analog signal is sampled at some high rate by an analog-to-digital conversion circuit, the resulting digital signal consisting of a set of data points representing the analog signal's amplitude values at those specific points in time when sampling occurs. A very similar application is *digital-to-analog synthesis* where a set of digitized values representing amplitudes of a waveform are converted into analog signal values, resulting in a wave comprised of "stair-stepped" values over time. Yet another application is in AC-to-DC controlled power rectification where semiconductor devices such as transistors or thyristors are used to precisely control the conversion of AC power into DC power, the AC current drawn by these power converter circuits resembling the same sort of stair-stepped waveshape that we see in digital-to-analog synthesis[6].

An example of a sine wave being sampled twelve (12) times per cycle is shown below:



Unlike most types of modulation where the intent is to impress some form of information or intelligence onto a carrier waveform, the stair-stepping impressed a low-frequency signal by a sampling system (or within an AC-DC converter circuit) is an unavoidable consequence of that system's operation.

Sampling is described here as a form of modulation due to its effects on the resulting signal's frequency spectrum. As it so happens, the frequency spectrum of a sampled signal contains the "pure" signal's frequency as well as *sidebands* of the sampling frequency. In the case of our 12-step-sampled sine wave, the sampled waveform would have a frequency spectrum containing the fundamental as well as 11th, 13th, 23rd, 25th, 35th, and 37th harmonics, etc., each successive sideband-pair being weaker in magnitude.

[6] In the case of controlled rectification circuits, the AC current waveform has a fundamental frequency equal to that of the AC power system's frequency, but containing the same number of "steps" as the number of rectifying elements in the converter circuit. Thus, a 12-pulse converter will exhibit an AC input current that is roughly a sine wave with each cycle consisting of twelve distinct step-values.

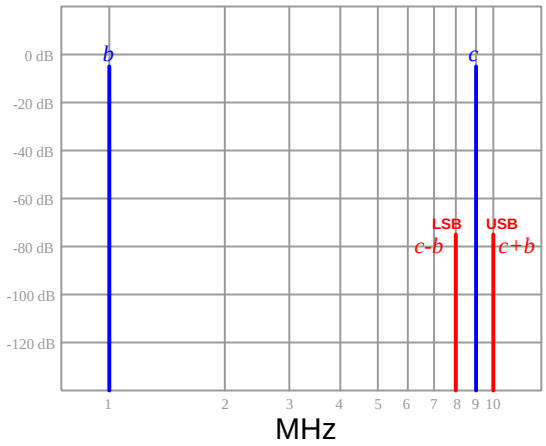
3.7 Frequency-shifting

In the earlier section on amplitude modulation (AM), we encountered the phenomenon whereby amplitude-modulating a carrier frequency with some lower baseband frequency created two new frequencies called *sidebands*[7]. If c represents the carrier frequency and b represents the baseband frequency, the two sideband frequencies will be the sum and the difference between the two:

$$\text{Upper sideband frequency} = c + b$$

$$\text{Lower sideband frequency} = c - b$$

A spectrum display of a 9 MHz carrier (c) sinusoid modulated by a 1 MHz baseband (b) sinusoids reveals a lower sideband (LSB) frequency of 8 MHz and an upper sideband (USB) frequency of 10 MHz:

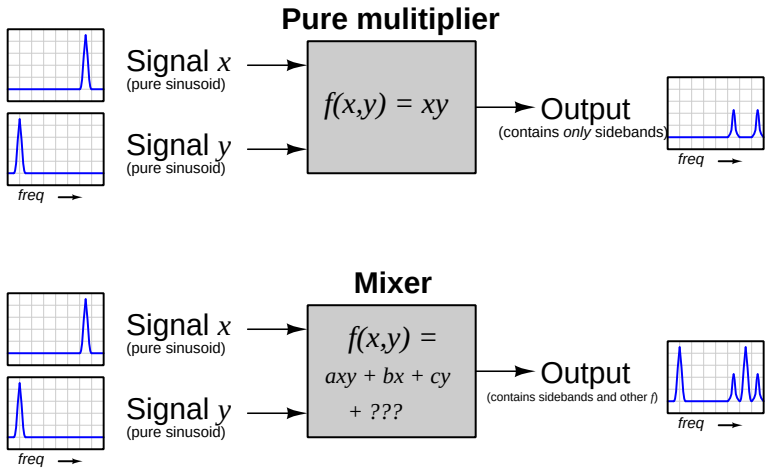


Each of these sidebands contain all the information inherent to the baseband signal, but exist at a higher frequency than the baseband. In other words, we could filter out the carrier and one of the two sidebands to leave just one sideband, and not lose any of the information contained in the baseband signal. This is an extremely useful property of amplitude modulation, as *it allows us to alter the frequency of an information-bearing signal without altering the information itself.*

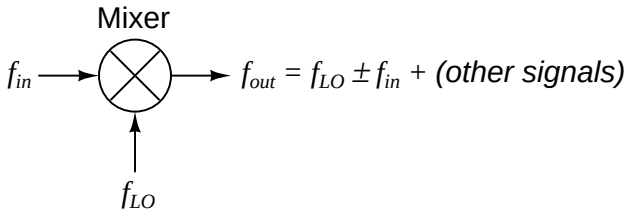
This principle is known by another name: *heterodyning*, from the Greek words *heteros* ("other") and *dyne* ("force"), referring to the modulation of one signal (or force) by another. The mathematical foundation of heterodyning is the *multiplication* of signals' instantaneous values, which is precisely what happens in amplitude modulation: the baseband signal's value at any point in time serves as a multiplying factor for the amplitude of the carrier waveform. As the baseband signal rises in value, the carrier's amplitude becomes multiplied by that value and grows in size; as the baseband value falls, the carrier's amplitude attenuates. This is why an amplitude-modulated signal has an "envelope" shape identical to the baseband waveform, because the baseband signal determines how far the modulated waveform is allowed to peak.

[7] The mathematical basis for the creation of sideband frequencies is explored in section 5.1 starting on page 70.

Any electronic circuit capable of multiplying the instantaneous values of two AC signals is therefore capable of performing amplitude modulation. However, constructing an electronic circuit capable of pure multiplication – especially at high signal frequencies – is a daunting task. In practice it is common to use a circuit less precise than a true multiplier (but which contains a product term somewhere in its nonlinear function), resulting in the expected sidebands *plus* other frequencies (usually the baseband and carrier). We refer to such a pseudo-multiplier circuit as a *mixer*:



Not all mixers are created equal. Generally speaking, the simpler the mixer circuit, the greater the number of unwanted[8] frequencies in the output signal spectrum. *Balanced* mixer designs, for example, cancel out more of these unwanted frequencies than *unbalanced* mixers, but in truth all mixers end up outputting more than just the sideband frequencies produced by a true multiplier. For this reason mixers are usually followed by *filter* networks designed to attenuate all but the desired frequency(ies). The generic block-diagram symbol for a mixer circuit is a circle with a large "X" symbol inside, taking an input signal and modulating (mixing) it with a second signal from a *local oscillator* (LO):



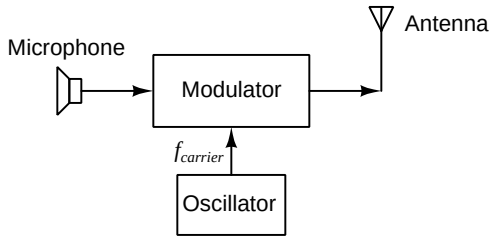
In a simple amplitude modulation system, f_{in} would be the baseband signal, f_{LO} the carrier frequency, and f_{out} the AM signal ready to be transmitted over the communication channel (e.g. as voltage/current signals on a two-conductor cable or as electromagnetic waves through space).

[8] A technical term used to describe some undesired effect resulting from practical limitations of a device or of a system is *artifact*. In this case we would say that the presence of the baseband and carrier (and potentially other frequencies as well) in the output signal is an artifact of the imperfect mixing process, not present in the output of a true multiplier.

However, heterodyning is not limited to just amplitude modulation, even though AM is based on heterodyning. We may use mixers to "heterodyne" *any* frequency-based communications signal to either raise or lower that signal frequency as desired. Thus, mixing (also known as *frequency conversion*) is a general-purpose technique applicable to multiple modulation types where it is advantageous to process the signal in different ways at different frequencies.

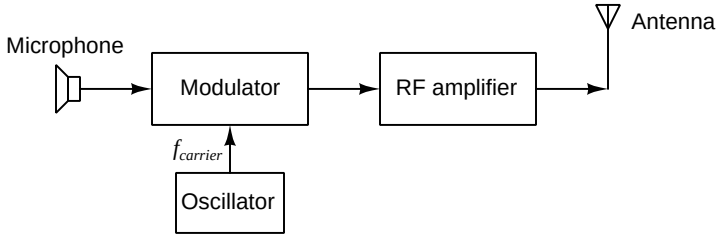
Consider a generic radio transmitter, where audio information sensed by a microphone is converted into RF (radio-frequency) electrical energy to power an antenna. A block diagram for an extremely simple radio transmitter (of *any* modulation type) is shown here:

Elementary radio transmitter



While a transmitter built like this might function as a proof-of-concept system – consisting solely of a microphone, modulator circuit, and antenna – it would likely lack enough power to be practical. *Amplification*, therefore, is necessary to build a practical radio transmitter, as shown in the following block diagram where we place an RF amplifier between the modulator and the antenna:

Radio transmitter with single-stage amplification

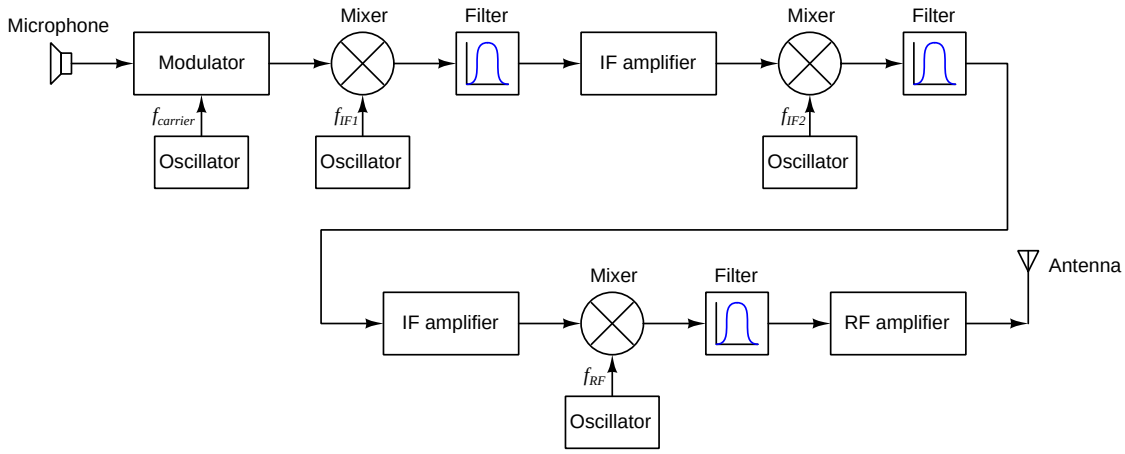


However, amplifiers tend to have limited bandwidth (i.e. a limited range of signal frequencies at which they exhibit useful gain values) as well as limited gain within their recommended bandwidth[9], and so the power output from a transmitter such as this to its antenna will likely be meager. A more practical strategy would be to use multiple stages of amplification rather than a single stage, and have each stage operating at a progressively higher frequency until we reach the RF necessary for the antenna.

[9] Recall the *gain-bandwidth product* of an amplifier circuit, a fixed value approximately equal to the product of the amplifier's gain and the frequency for that gain. For an RF amplifier which must function at a very high frequency, the achievable gain will necessarily be low for any given value of GBW product.

In order to tailor a cascaded set of amplifiers to a different frequency range in order to maximize the collective gain, we must shift the frequency of the modulated signal upward in stages. This is done with a series of mixers, each fed by its own local oscillator. This frequency-shifting strategy results in the audio-frequency (AF) signal being shifted upward to become an "intermediate-frequency" (IF) signal, which is shifted upward again to eventually become radio-frequency (RF) suitable for emission at the antenna:

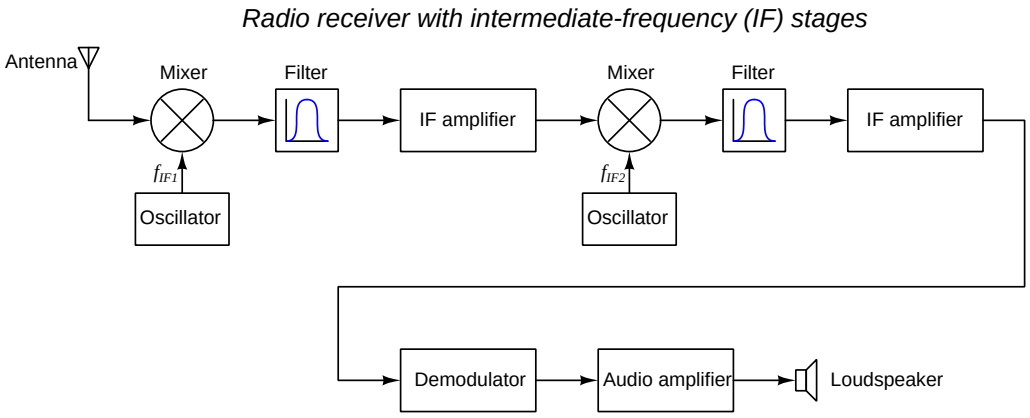
Radio transmitter with intermediate-frequency (IF) stages



Each of the local oscillator inputs has a progressively higher frequency ($f_{carrier}$ is the lowest, followed by f_{IF1}, f_{IF2}, and finally the highest which is f_{RF}). Each mixer outputs a pair of sideband frequencies $f_{LO} \pm f_{in}$ as well as some other "artifact" frequencies due to the imperfections of the mixer, and so a filter stage immediately following each mixer selects the one sideband of interest to be passed along for amplification. This use of mixers to increase the frequency of the information-bearing signal is called *upconversion*.

The fact that mixers are imperfect multipliers, generating artifact signals that must be filtered out, permits a practical simplification in the local oscillator. Instead of using a sinusoidal oscillator to generate f_{LO}, it becomes permissible to use a *square-wave* oscillator instead. Square waves are just an infinite series of sinusoids consisting of a fundamental frequency and only odd-numbered harmonics in descending amplitude, and so the filtering necessary for proper mixing ends up eliminating those unwanted harmonics as well. This is advantageous because square-wave oscillators are simpler in design and operation than sinusoidal oscillators, and in digital systems it is natural to use the *clock* signal as this frequency (or as the basis for it, using counter circuits to divide clock frequency into the desired LO frequency). The use of square-wave oscillators makes possible *single-chip* mixer construction where the local oscillator is an embedded portion of that integrated circuit.

Mixers may also be used in radio receivers of all modulation types, to perform the reverse operation: *downconversion*. This time, the RF signal intercepted by the receiver antenna is progressively mixed with lower frequencies to down-shift the signal frequency for optimum amplification, as shown in the following block diagram:

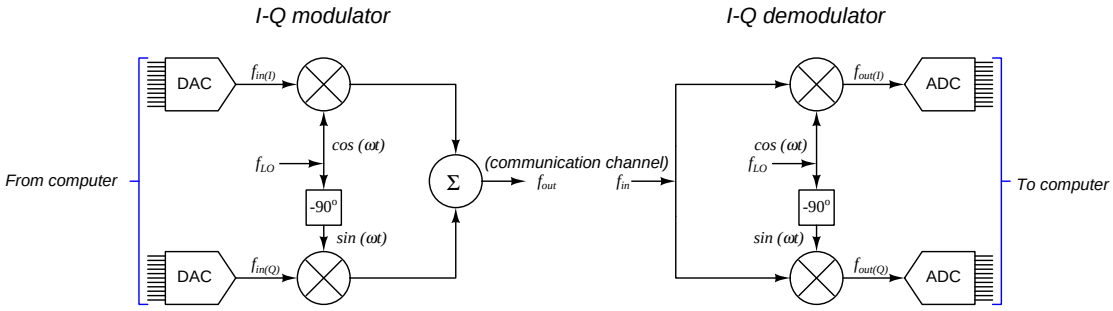


This time, f_{IF1} is the higher of the local oscillator frequencies, set to a frequency value such that the difference between it and the incoming RF signal from the antenna is the down-converted frequency suitable for the first stage of amplification. If you would like to see an example downconverter receiver with frequency values shown, refer to the Case Tutorial "Example: downconverting AM receiver" section.

Not only does upconversion and downconversion help optimize amplifier performance by limiting the bandwidth at which the amplifiers must function, but it also places the modulation and demodulation circuitry at the lowest-frequency values possible in each system. All other factors being equal, signal processing at low frequencies is much less troublesome and expensive than signal processing at high frequencies. This is especially true for *digital signal processing*, where a microprocessor must keep up with the pace of the incoming signals to digitize those signals and to perform mathematical operations on them. Thus, heterodyning is an extremely useful technique for simplifying the design and boosting the performance of any high-frequency communications system.

3.8 I-Q modulators

A relatively modern method of signal modulation popular in digital communication systems is the so-called *I-Q modulator*, the *I* and *Q* standing for *in-phase* and *quadrature*[10], respectively. In an I-Q modulation system, the waveform to be transmitted is divided into two portions, each one phase-shifted from one another by 90 degrees. These two signals are then mixed with similarly[11] phase-shifted versions of a local oscillator's (LO) signal, and the results added together to form a composite signal. Demodulation for an I-Q system follows a similar pattern, with the received signal being mixed with both in-phase and quadrature local oscillator (LO) signals to restore the original baseband I and Q data:



On the modulator side of this system, the baseband signal portions are upconverted by the mixers receiving the higher local oscillator frequency. On the demodulator side, the received signal is downconverted by the mixers and LO frequency to restore the original baseband signal frequency. In both cases, the frequency conversion occurs in just one mixing step, as the local oscillator is tuned to exactly match the carrier frequency of interest. This is known as *direct conversion* or *homodyning*[12] and requires no intermediate frequency (IF) stages in either transmitter or receiver.

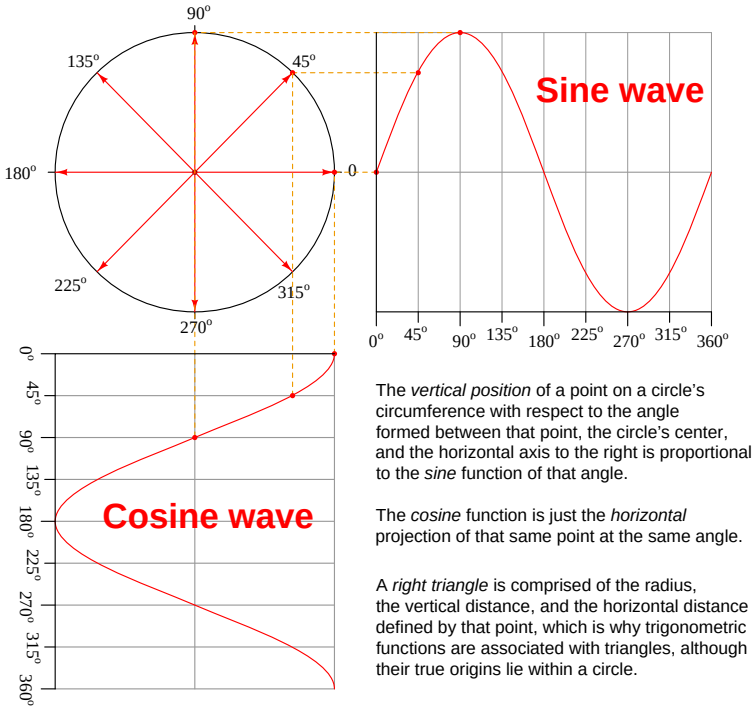
It is not uncommon to find the exact same circuit design used for both I-Q modulation and I-Q demodulation, that circuit usually taking the form of a single chip (IC). This efficient packaging lends itself very well to integration with digital circuitry to form the basis for *software-defined radio* units, where the heart of modulation and demodulation occurs in software rather than in hardware, the I-Q modulator simply acting as the interface between the digitally-encoded baseband information and the modulated RF.

[10] "Quadrature" is just a sophisticated term meaning one-quarter of a cycle out of phase, or 90^o out of phase.

[11] I-Q modulation is a technique lending itself better to digital than to analog due to this necessity of having otherwise identical signals be phase-shifted from each other by 90 degrees. It is relatively simple to design a phase-shifting network to produce a quadrature shift on a sinusoidal waveform of constant frequency, but if that signal's frequency must vary or if the signal consists of multiple frequencies, the task of maintaining a precisely 90 degree shift between the two versions of it becomes very complex (or practically impossible). Digital signals, however, are easier to phase-shift. A square wave of the type used for the local oscillator signal, for example, may be easily converted into in-phase and quadrature versions by means of a *ring counter*. Information that begins in digital form may similarly be phase-shifted by digitally-implemented time delays using shift registers, for which no simple equivalent exists for analog signals.

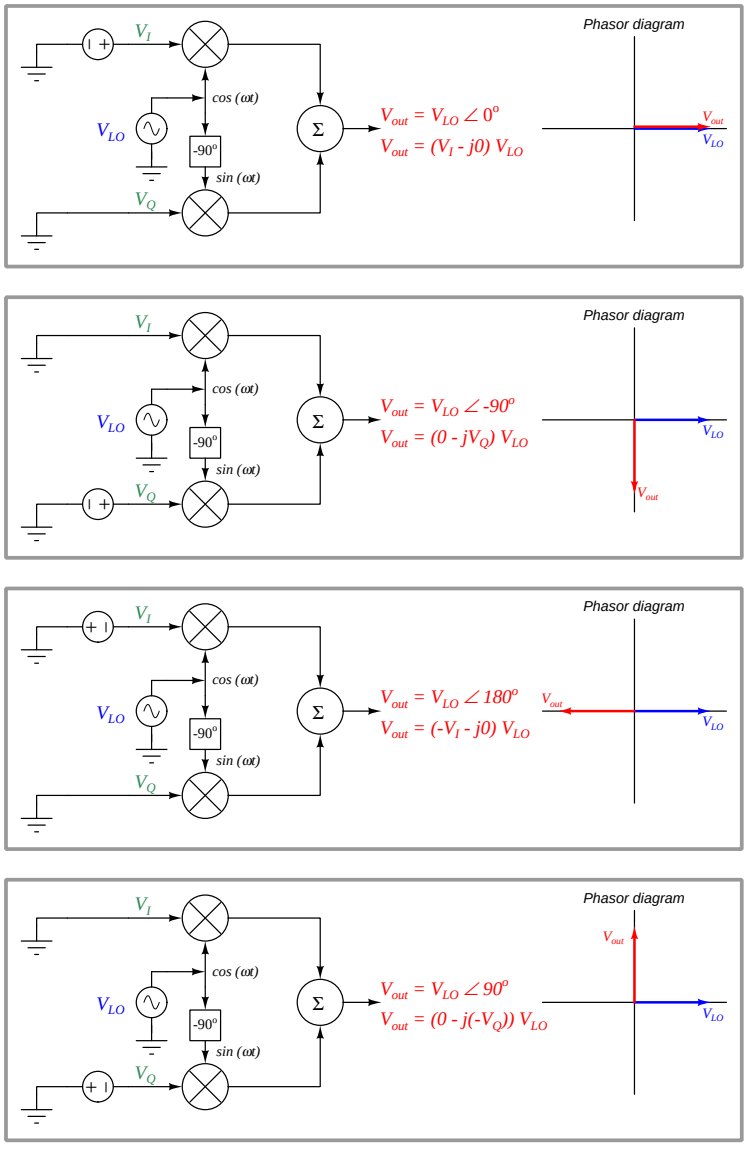
[12] This simply means mixing of *same* frequencies rather than mixing of *different* frequencies as with heterodyning. Direct conversion (homodyning) requires extremely precise and stable local oscillator circuits, which is why it has not seen popular adoption earlier but has lagged until crystal oscillator technology matured.

I-Q modulation should seem familiar to anyone accustomed to phasor representation of AC quantities: the I and Q signal portions are just the *real* and *imaginary* components of a phasor baseband signal. If we consider the baseband signal to be a simple sinusoid, the "real" portion is a cosine wave while the "imaginary" portion is a sine wave:



The same is true of the local oscillator signal, which is split into in-phase and quadrature-shifted portions before being sent to the mixers: the "real" portion of the baseband signal being mixed with the "real" portion of the local oscillator signal, and the "imaginary" portion of the baseband signal being mixed with the "imaginary" portion of the local oscillator signal. When summed together, the result is a waveform whose frequency *and* phase represent information contained in the baseband signal.

We may gain a deeper understanding of I-Q modulation by considering a set of "limiting cases" for the I and Q signals input to an I-Q modulator, exploring the phase relationship of the output signal (f_{out}) to the un-shifted local oscillator signal (f_{LO}). Our limiting cases will be simple DC or zero (null) signals as shown below:



In other words, the voltage signal at the I and Q inputs constitute the real and imaginary components of a complex coefficient (i.e. $(I - jQ)$) multiplying the local oscillator signal to produce the output. The polar magnitude of this complex coefficient determines the magnitude of the output, while the relative magnitudes of the I and Q signals determines the amount of phase-shift from the

local oscillator source.

I-Q modulation is extremely versatile, as it is able to perform amplitude modulation, frequency modulation, *and/or* phase modulation all by careful manipulation of the I and Q input signals:

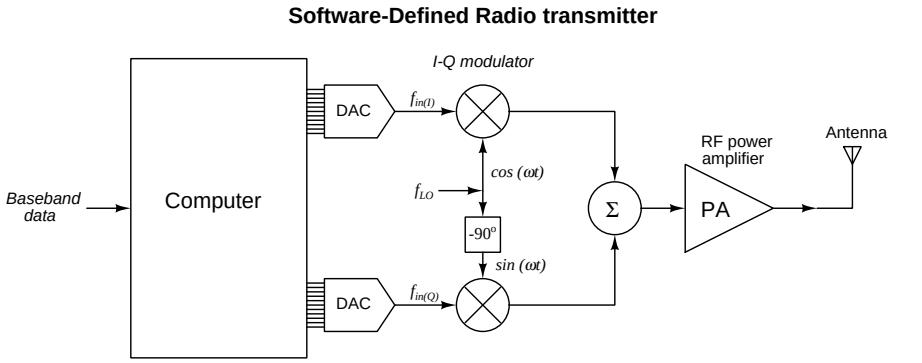
- **Amplitude modulation (simple AM)** – set the Q input to zero all the time, and send baseband information only into the I input. The I signal's instantaneous value will determine the output waveform's amplitude.
- **Single-Sideband Amplitude modulation (SSB AM)** – set the I input to the real value of the baseband signal, and the Q input to the imaginary value. The phasor sum of these results in cancellation of the lower sideband.
- **Phase modulation (PM)** – the baseband signal value ($b(t)$) sets the angle value for a cosine function and a sine function, both with constant peak amplitudes. The I input becomes $\cos(b(t))$ and the Q input becomes $\sin(b(t))$. The phasor sum of the I and Q values determines the phase shift of the output waveform in relation to the local oscillator signal.
- **Frequency modulation (FM)** – the baseband signal value ($b(t)$) sets frequency value for a cosine function and a sine function, both with constant peak amplitudes. Functionally, the way this works is for the baseband signal to set the *rate* at which the angle increments or decrements for those cosine and sine functions. Taking the continuous time-integral of the baseband value results in an incrementing or decrementing angle which becomes the argument for both the sine and cosine functions. The I input becomes $\cos(\int b(t)\, dt)$ and the Q input becomes $\sin(\int b(t)\, dt)$. With this configuration, the instantaneous value of the baseband signal sets how rapidly the output signal's phase rotates, resulting in frequency modulation.

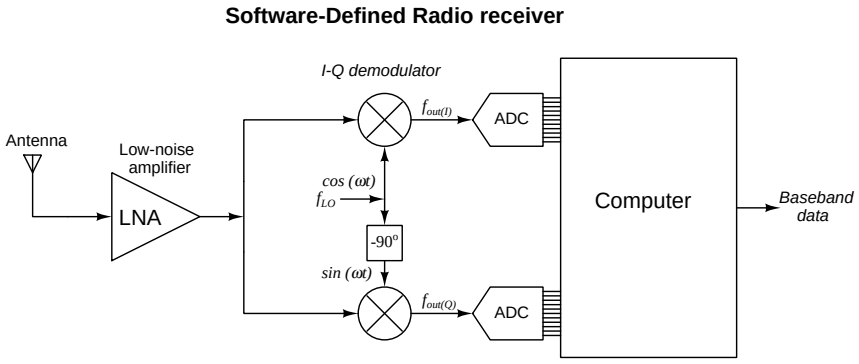
Beyond basic AM, PM, and FM, I-Q modulation is also able to encode binary (1 and 0) bit states as the I and Q inputs, the four combinations (00, 01, 10, and 11) each representing a different amount of phase shift for the modulated waveform. In this configuration the analog/digital converters are eliminated, with discrete logic-level signals (i.e. "high" and "low") directly driving the modulator mixers and logic signals output directly by the demodulator mixers.

3.9 Software-defined radio (SDR)

No modern discussion of radio modulation and demodulation would be complete without at least some mention of *Software-Defined Radio* or *SDR*, the basic principle being the use of digital computing technology to modulate (for transmission) and/or demodulate (for reception) radio signals rather than using analog circuitry to do so. Traditional radio transmitters use analog amplifiers, mixers, oscillators, and other circuits to modulate a radio-frequency (RF) carrier signal with baseband information, while traditional radio receivers use similar circuitry to extract baseband information from modulated RF signals received by an antenna. Software-defined radios, by contrast, use as few hardware components as possible to modulate/demodulate RF signals using I-Q techniques and then interface those I-Q signals to high-speed analog-digital converter circuits where a microprocessor, FPGA, or some other programmable digital device further processes the digitized signals.

Block diagrams showing basic SDR transmitter and receiver systems are shown below:

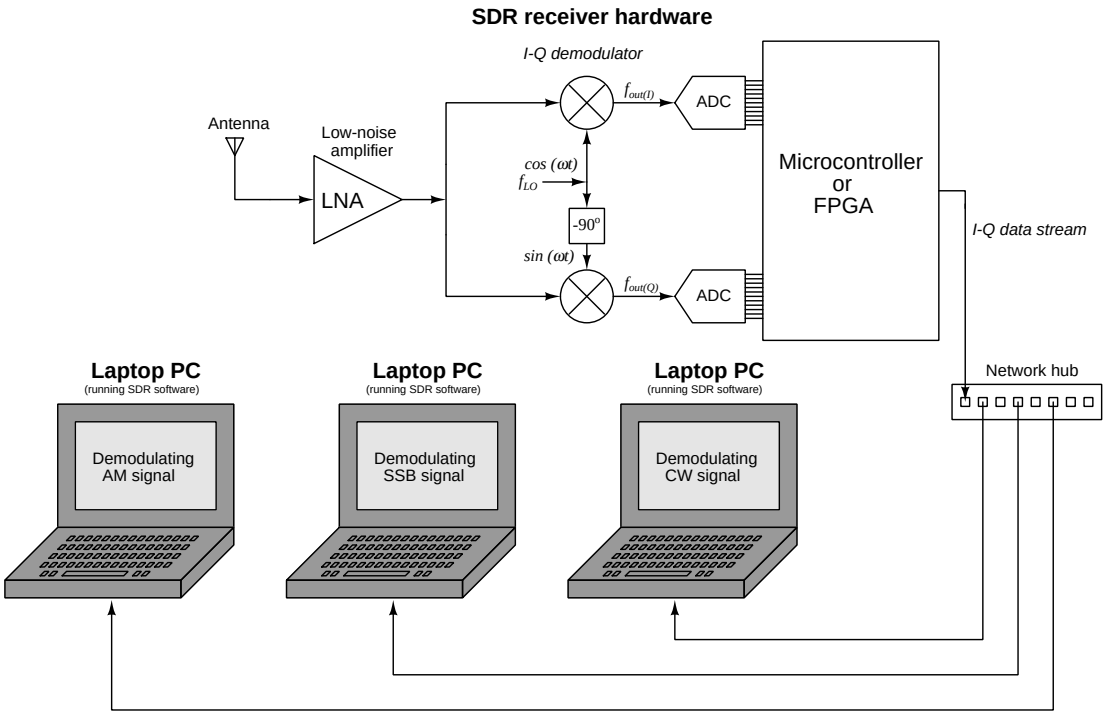




The advantage of using hardware only for I-Q modulation/demodulation and using software to process those in-phase (I) and quadrature (Q) signals is that any form of traditional signal modulation may be chosen and executed by that computing device without requiring any modification or adjustment to the hardware. For example, SDRs following the block diagram designs

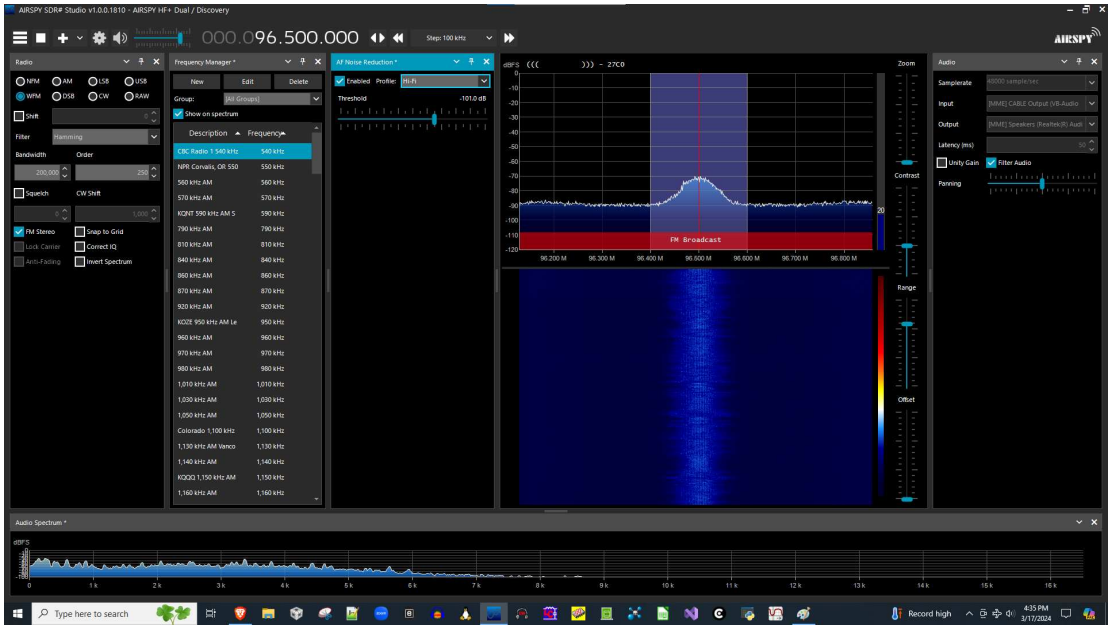
shown above could be programmed to implement FM, AM (dual-sideband, LSB, or USB) arbitrarily while using the exact same minimalist hardware.

Performing modulation in the digital domain rather than in analog hardware additionally permits *networked* radio reception. This is where a single SDR receiver outputs a continuous stream of live I-Q data to a shared network where multiple general-purpose computers running SDR signal-processing software receive that data stream and independently demodulate the raw I-Q data. With such a system it is possible for these multiple computers to tune into and decode different frequencies and modulation types all simultaneously from the same receiver! Principal limitations include the receiver's bandwidth and the network's data rate limit:



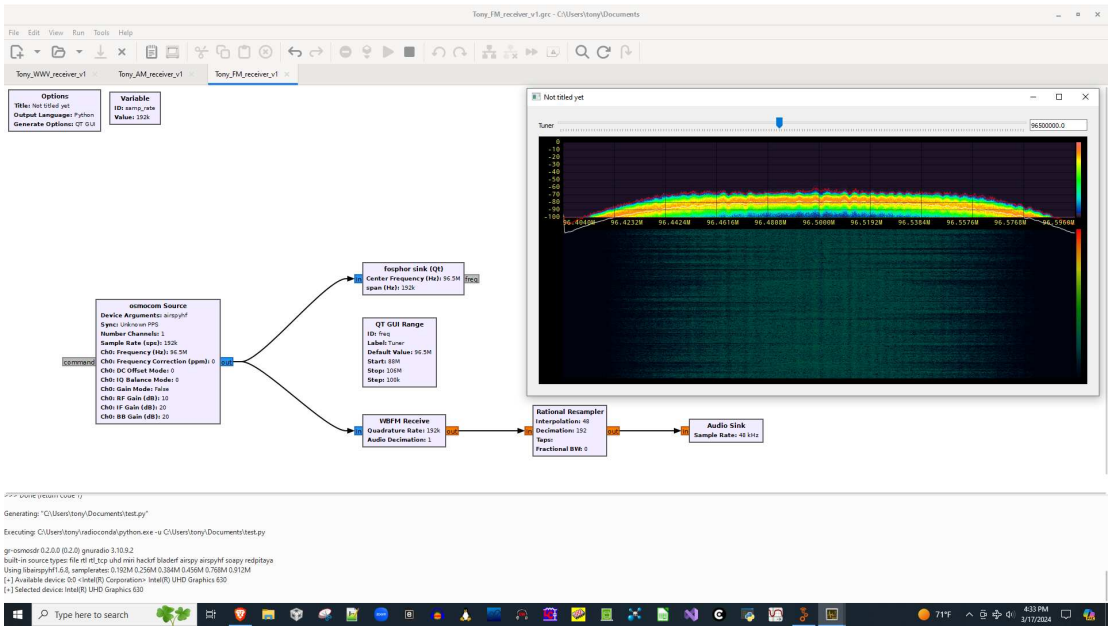
SDR also dramatically decreases the cost of entry into the world of radio and RF experimentation by leveraging ubiquitous personal-computing technology which is amply powerful to handle such signal processing tasks. Rather than having to purchase or build a complete hardware-based radio receiver or transmitter, one need only purchase or build an SDR unit with minimal amplification, filtering, mixing, and analog-digital conversion and let the software do the rest on computer hardware you already own.

Below is a screenshot of SDR software called `SDR#` tuning in to an FM broadcast signal at a frequency of 96.5 MHz. The SDR hardware used for this example is an Airspy HF+ Discovery unit costing less than $200 US dollars at the time of this writing (2024), using USB as the physical interface with any personal computer. The `SDR#` software happens to be running on a personal computer with the Microsoft Windows operating system:



As you can see in this screenshot, the `SDR#` software supports narrowband FM (NFM), wideband FM (WFM), AM, dual sideband AM (DSB), lower sideband AM (LSB), upper sideband AM (USB), and carrier wave (CW) demodulation on the I-Q data, as well as providing a "Raw" option where the I-Q data is left unconverted. Conveniences such as a Frequency Manager for channel pre-sets allows the user to easily switch between stations. Other features include noise reduction, a spectrum display, a colored "waterfall" display showing a brief history of signal strengths on the displayed spectrum, etc.

Of course, relegating most of the signal processing to the digital (software) domain means one has multiple options for software paired with any given SDR hardware. Below we see another screenshot showing the same Airspy HF+ Discovery receiver unit connected to the same personal computer but running a different piece of software called the `GNU Radio Companion`[13] which is intended to be more of an experimental software platform for SDR reception and/or transmission. Unlike `SDR#` which provides a very polished user interface intended for ease of operation, `GNU Radio Companion` provides a graphical programming environment where signal-processing function blocks may be linked together to form a working modulation or demodulation system:



In this particular case a function block called "osmocom Source" provides the I-Q data stream interface with the Airspy HF+ Discovery receiver and outputs data in complex floating-point 32-bit data form at a rate of 192,000 bits per second (192 kbps)[14]. A block called "fosphor sink" displays this data in the form of a spectrum while another block called "WBFM Receive" takes that same 32-bit FP data stream and performs frequency demodulation on the raw I-Q data. A "Rational Resampler" block decimates[15] the raw stream's 192 kbps sample rate down to 48 kbps which is more suitable for audio output, and finally an "Audio Sink" block exports the demodulated (baseband) audio data to the personal computer's speakers. The color of each function block's input and output

[13] This happens to be open-source software.

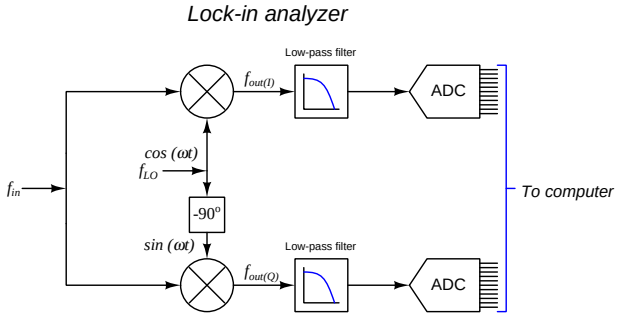
[14] The choice of data rate is constrained by the SDR receiver hardware. In this case, 192 kbps is the lowest sampling rate supported for the Airspy HF+ Discovery receiver.

[15] This colorful term refers to periodic undersampling of a previously sampled signal, essentially discarding most of the sampled data and passing along a minority of it. In this particular case the Airspy HF+ Discovery hardware unit has a minimum I-Q sampling rate of 192 kbps, but we need just 48 kbps for audio reproduction. Therefore, the resampler block's task is to "decimate" the 192 kbps data stream by a 4:1 ratio so that only one out of every four samples gets passed along to the Audio Sink block for audio playback.

tab is meaningful: for example, blue represents complex floating-point 32-bit data, whereas orange represents “real” 32-bit floating-point data.

3.10 Lock-in analyzers

I-Q demodulation finds interesting application in a class of measuring instruments known as *lock-in analyzers* or *lock-in amplifiers* used to measure the magnitude and phase angle of noisy AC signals. The following block diagram shows a lock-in analyzer where the local oscillator (f_{LO}) is set to the frequency of interest within the noisy input signal (f_{in}). This particular lock-in analyzer uses an analog mixing stage followed by analog-to-digital converters (ADCs) for final analysis within a digital computer, but all-digital lock-in analyzers also exist which perform the mixing and filtering functions digitally:



The low-pass filters block all but the zero-frequency DC signals resulting from the local oscillator's signal mixing with the one frequency of interest within the input signal. All other frequencies present within the input signal result in non-DC products when mixed with the local oscillator signal and are thereby screened out by the filters. The two analog-to-digital converters (ADCs) see DC signal voltages together[16] representing the phase shift between the frequency of interest and the local oscillator signal.

Mixing between the input signal and the local oscillator signal therefore provides a highly effective means of discriminating just one frequency of interest while ignoring all others (including noise), making lock-in analyzers exceptionally useful for signal-measurement applications challenged by excessive amounts of noise or other interference.

A real example of a lock-in analyzer is shown in the following photograph:

[16] The Pythagorean sum of the filtered I and Q signals (i.e. $\sqrt{V_I^2 + V_Q^2}$) represents the signal-of-interest's magnitude while the relative strengths of the I and Q signals represent the phase shift (i.e. $\theta = \tan^{-1} \frac{V_Q}{V_I}$).

Chapter 4

Historical References

This chapter is where you will find references to historical texts and technologies related to the module's topic.

Readers may wonder why historical references might be included in any modern lesson on a subject. Why dwell on old ideas and obsolete technologies? One answer to this question is that the initial discoveries and early applications of scientific principles typically present those principles in forms that are unusually easy to grasp. Anyone who first discovers a new principle must necessarily do so from a perspective of ignorance (i.e. if you truly *discover* something yourself, it means you must have come to that discovery with no prior knowledge of it and no hints from others knowledgeable in it), and in so doing the discoverer lacks any hindsight or advantage that might have otherwise come from a more advanced perspective. Thus, discoverers are forced to think and express themselves in less-advanced terms, and this often makes their explanations more readily accessible to others who, like the discoverer, comes to this idea with no prior knowledge. Furthermore, early discoverers often faced the daunting challenge of explaining their new and complex ideas to a naturally skeptical scientific community, and this pressure incentivized clear and compelling communication. As James Clerk Maxwell eloquently stated in the Preface to his book *A Treatise on Electricity and Magnetism* written in 1873,

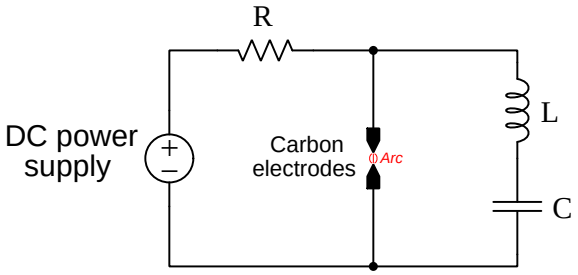
> It is of great advantage to the student of any subject to read the original memoirs on that subject, for science is always most completely assimilated when it is in its nascent state . . . `[page xi]`

Furthermore, grasping the historical context of technological discoveries is important for understanding how science intersects with culture and civilization, which is ever important because new discoveries and new applications of existing discoveries will always continue to impact our lives. One will often find themselves impressed by the ingenuity of previous generations, and by the high degree of refinement to which now-obsolete technologies were once raised. There is much to learn and much inspiration to be drawn from the technological past, and to the inquisitive mind these historical references are treasures waiting to be (re)-discovered.

4.1 Arc converter transmitters

A long-obsolete technology for producing radio-frequency oscillations called the *arc converter* used an electric arc powered by a DC voltage source, connected to suitable inductance and capacitance to form a resonant circuit which would oscillate due to the *negative resistance*[1] of the arc.

The basic concept of an *arc converter* oscillator is evident in the following schematic diagram:



This technology was popular during the 1920's for radio broadcast transmitters, as a means of producing high-frequency oscillations suitable for the generation of electromagnetic waves at high levels of power (often tens of kiloWatts or more!). This was at a time when vacuum tube technology had not yet advanced to the point of similar power output capability, making arc converters the technology of choice for long-distance broadcasting.

However, a high-power RF oscillator is by itself useless as a radio transmitter without some means to modulate its output. At that time, the dominant technology for microphones used granules of carbon which were compressed by the pressure of a diaphragm moved by air vibrations (sound waves). In other words, a 1920's-era carbon microphone was a *variable resistor*, and therefore could be used to directly modulate the flow of electric current through it.

[1] This is a phenomenon whereby the voltage dropped across a load decreases as current increases. Some devices such as *tunnel diodes* exhibit this characteristic, which is quite unlike the linear relationship of voltage to current that we see in plain resistive elements, and which Ohm's Law represents.

The following set of schematic diagrams taken from Figure 115 of C.F. Elwell's book *The Poulsen Arc Generator* shows various methods whereby such a carbon-granule microphone could be used to amplitude-modulate the output of such an arc converter:

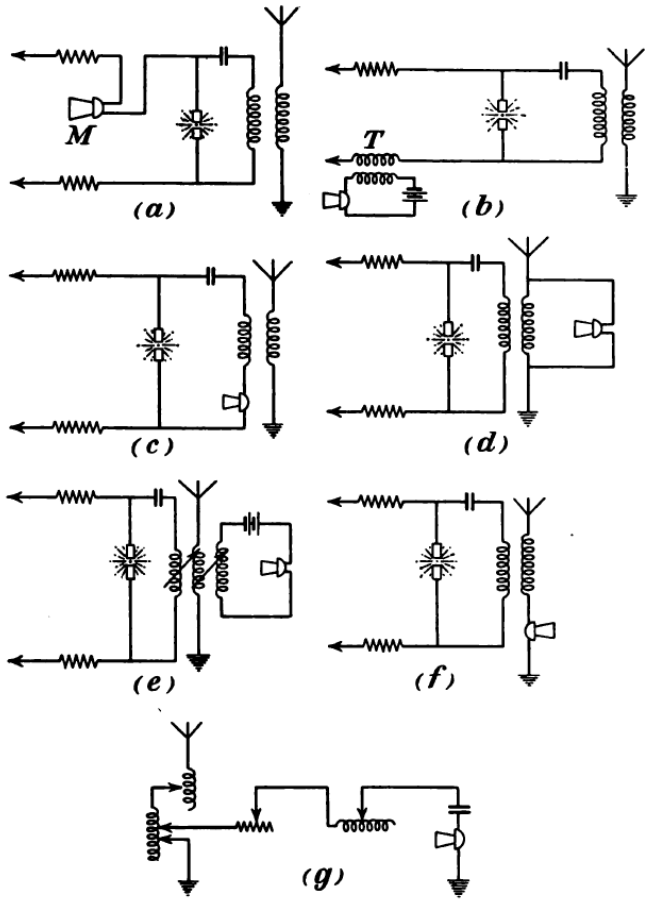


Fig. 115.—General Methods of Modulation of Aerial Current.

Method (a) places the microphone in series with the DC source (in each schematic, the DC voltage source connects to the two wire-ends marked by arrowheads), thereby directly modulating the amplitude of the DC feeding the arc. Method (b) is similar, but transformer-couples the microphone (with its own DC source) to the DC circuit of the arc converter. Methods (c) and (f) place the microphone in series with the RF signal itself, directly modulating that high-frequency signal's amplitude by the microphone's own varying resistance. Methods (d), (e), and (g) exploit the carbon microphone's ability to function as a crude load, modulating the RF signal's amplitude by means of direct loading: when sound waves compressed the microphone's carbon granules, the decreased resistance presented a "heavier" load to the RF circuit, causing more energy to be shunted away from the antenna and dissipated in the microphone itself as heat.

Later, early vacuum tube technology was coupled with arc converter technology to make absorption-style modulation possible at higher power levels than what a carbon microphone could directly dissipate. Here we see (from Figure 116 of Elwell's 1923 book) a circuit using two triode tubes and two coupling transformers to perform absorption-modulation in an arc converter radio transmitter:

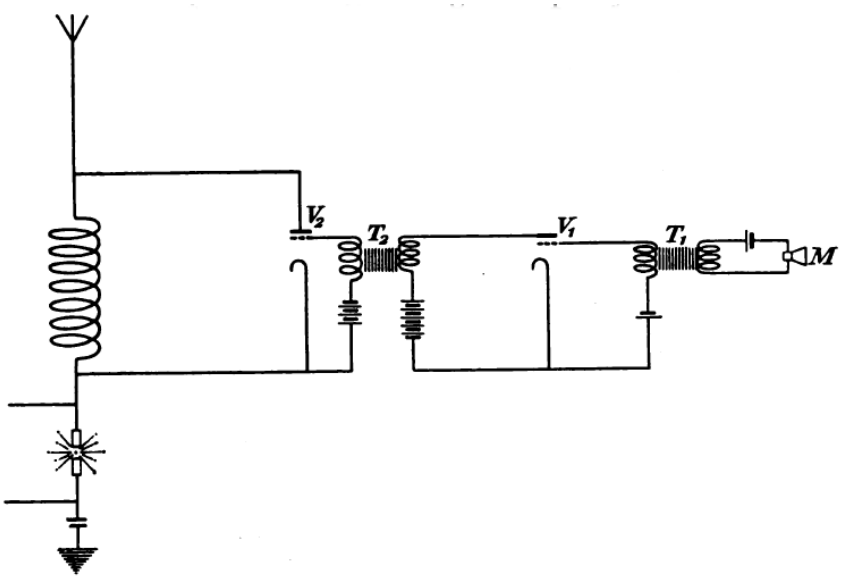


Fig. 116.—Method of Modulation by Absorption Control.

The two vacuum tubes in this schematic, labeled V_1 and V_2, are drawn without circular envelopes as became standard convention in later years, but you can see the cathode (inverted J-shape), grid (dashed line), and plate (solid line) of each. Tube V_2 served as the absorptive load for the arc converter's resonant circuit, that tube being driven by a signal from the microphone (M) driving the grid of amplifying tube V_1. Transformer coupling ensured good isolation between the high-power RF circuit and the microphone for operator safety.

4.2 Heterodyne radio reception

The following quotations come from an article entitled "The Heterodyne Receiver" by John V.L. Hogan (Manager of The International Radio Telegraph Company, and past President emeritus of the Institute of Radio Engineers) published in April 1921. The article was found in *The Electric Journal* volume 18 number 1 (copyright 1922), pages 116 to 119, and is reproduced here in its entirety. It explains in some detail how professor Reginald Fessenden's heterodyne innovation solved a vexing problem in radio telegraphy – the transmission and reception of *Morse Code* pulse signals – describes an earlier "chopper" technology inferior to the heterodyne technique, and along the way gives a fair overview of radio technology of that era in easy-to-understand language.

The fundamental problem addressed by Fessenden's invention is the efficient conversion of extremely high-frequency radio signals into audible tones which could be heard by human listeners and interpreted as the "dots" and "dashes" of Morse Code signals. The existing "chopper" technology of the time worked, but suffered multiple limitations. Fessenden applied a musical concept to radio which allowed the "down-conversion" of radio frequencies into audio frequencies without using a "chopper" device and thus escaping its technical limitations.

Please note that the author uses some now-obsolete terms such as *condensor* (i.e. capacitor), *capacity* (i.e. capacitance), *aerial* (i.e. antenna), and *detector* (i.e. diode).

> Of the many inventions in the applied science of radio signaling, a few stand far above all the others. The heterodyne receiver marks one of the highest peaks of achievement in wireless communication, and probably has done as much to advance the art as any single invention. When Fessenden devised this ingenious and eminently practical way of selecting and amplifying received radio signals, he established a system which, in conjunction with his continuous-wave transmitters, now bids fair to grow into substantially universal use.
>
> The coined name "heterodyne" is derived from the Greek *heteros* (other) and *dyne* (force), the new word implying that the receiver, which it designates, makes use of an "other force", i.e., a force other than that of the received signals. The heterodyne does truly utilize such a second source of energy, for it combines with the signal-producing effects of the received electromagnetic waves. The effects of another series of radio frequency oscillations which are locally generated at the receiving station. The invention is notable not so much for the fact that a local source of energy is used, as for the highly novel and effective ways in which the effects of the local source are combined with those of the incoming signals.
>
> In order to demonstrate the action of the heterodyne receiver, let us first look into the basic problem of receiving sustained-wave radio signals. The sustained or undamped wave is a progressive electromagnetic vibration of ultra-audible but infra-visible frequency. This wave is created by the surging of powerful alternating-currents, of similar radio frequency, in an elevated aerial wire system at the transmitting radio station. The sustained wave passes out radially over the earth's surface in all directions from the transmitter; as its energy is distributed over a larger and larger area the wave amplitude decays, and at any receiving point it is obviously very feeble as compared to its initial value. Nevertheless, such an electromagnetic wave, in passing a receiving aerial-wire system, is capable of setting up in the elevated conductors a small radio frequency alternating potential with respect to earth. If a circuit from aerial wire to earth is

provided, a feeble radio frequency current will flow; if the capacity reactance of the aerial wires with respect to earth is balanced-out (for the frequency of the arriving waves) by the inductive reactance of a tuning coil in the circuit, the current will build up by resonance to a maximum value.

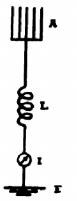


FIG. 1
SIMPLE
RECEIVING
SYSTEM

Fig. 1 shows an aerial wire system A connected to earth E through such a tuning coil L and a very sensitive current-indicator I. If we assume that radio waves of a frequency of 100000 per second (which is equivalent to a wave-length of 3000 meters, the wave velocity being 3×10^8 meters per second) strike the antenna A, it will necessarily follow that a small alternating voltage of this same frequency will be induced in the system. Since the circuit is in effect closed for currents of this frequency (because the capacitance of the aerial wires with respect to earth may be of the order of 0.001 microFarad), the voltage will result in a small 100000 cycle alternating current in the system. If the antenna capacitance of 0.001 microFarad is offset by making the inductance of the coil L approximately 2.5 milliHenrys, the circuit will have minimum reactance for 100000 cycles and consequently a maximum current will be developed. Under these conditions a powerful disturbance might produce as much as one milliampere of current through the indicator I, but the usual quantity would be measured in tens of microAmperes.

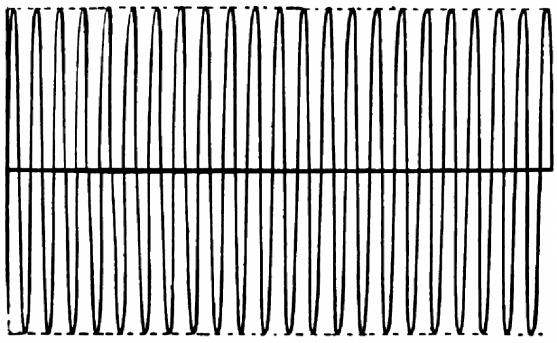


FIG. 2—SUSTAINED WAVE

An oscillogram of the antenna-to-ground current is shown in Fig. 2; a complete cycle occupies one one-hundred-thousandth of a second, so that the twenty-two cycles

represented in this figure consume only a little over one five-thousandth of one second. The amplitude is constant; hence the name "sustained" or "continuous" or "undamped" wave. Telegraph signaling with such waves is usually carried on by interrupting their continuity; a stream of waves is emitted (and hence received) for about one-twentieth of a second to represent a dot and for about three-twentieths second to indicate a dash. These signal trains of constant amplitude waves produce, in the receiver, groups of [page 116]

constant amplitude radio frequency current of similar duration. The problem of receiving radio telegraph messages thus resolves itself into that of observing the duration of these feeble alternating currents of exceedingly high frequency.

If the indicator of Fig. 1 were of the thermal type and capable of showing a substantial scale reading for a few millionths of an ampere, it might be used as a rather crude telegraph receiver. Could such an apparatus be secured, a short deflection would indicate a dot and a long deflection or pause a dash; telegrams in the Morse code could thus be spelled out slowly. In wire telegraphy aural reception was found to be far more satisfactory than visual operation; the same is true of radio telegraphy, and therefore we must find a way of generating sounds to indicate the radio frequency currents in the receiving circuits.

The telephone receiver is the most sensitive and most satisfactory device for producing sounds from electricity. However, a current of 100000 cycles per second frequency is many times too high to give a direct indication from a telephone receiver; even if it were possible to make the diaphragm vibrate at this rate, no sound would be heard because the upper limit of audibility is exceeded by some five times. Consequently some type of frequency transformation or lowering must be used.

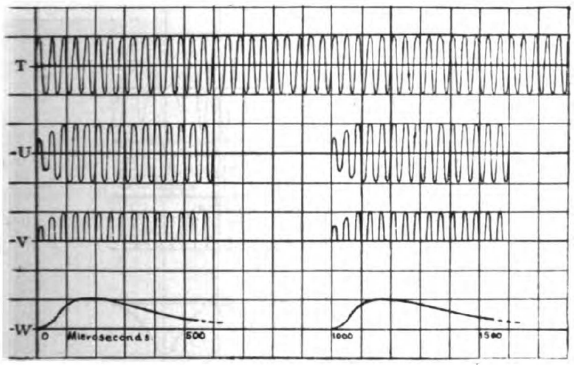


FIG. 3—OPERATION OF CHOPPER RECEIVER

Fig. 3 indicates oscillographically one of the most successful methods in use prior to general adoption of the heterodyne. On the uppermost axis T is shown the train of oscillations in the receiving antenna during part of a dot-signal. By inserting an interrupter or "chopper" somewhere in the circuit the oscillation-train is broken up into shorter groups about 0.001 second apart, as indicated on axis U. These shorter trains

are rectified or biased by passage through a distorting conductor (e.g. a crystal detector) as represented on axis V, and the rectified half waves are collected in a condenser and discharged as direct current pulses, axis W, through the windings of a telephone. Thus there will be current in pulses at 1000-per-second frequency in the telephone so long as waves are arriving. These will produce a musical vibration of the diaphragm; short and long tones of a pitch about two octaves above middle C will indicate dots and dashes. The pitch of the signal tone may be changed, by altering the interrupter speed, to a value most pleasing to the operator.

The "chopper" exists in a number of modifications, but all are handicapped in much the same way. Interrupter troubles are common, usually much of the arriving energy is wasted and there is little or no selective power inherent to the system. The prime defect is that all current impulses in the antenna circuit, practically regardless of their character or frequency, are "chopped-up" into the same musical tone as the signal. Thus the telephone receiver gives the same type of response to interfering signals as to that which it is desired to receive, and it becomes exceedingly difficult to interpret arriving messages under any but the best conditions.

The heterodyne receiver offers a violent and favorable contrast. Abandoning all devices of the interrupter class, Fessenden devised a frequency-reducing scheme based upon the principle of beats. It was well known that if two musical tones of slightly different frequency were simultaneously sounded, an auditor would hear not only the two notes, but also a flutter or amplitude variation whose rate would be equal to the difference in the tone frequencies. By the exercise of a great scientific imagination, Fessenden extended this concept to the range of radio frequencies, far above the limit of sound audition. His suggestion was not merely that, if a radio frequency effect of say 100000 cycles per second were caused to interact with another of say 101000 per second, a beat variation of 1000 per second (an audible frequency) would be produced; Fessenden went farther, and proposed the use of a generator located at the receiving station for the production of one of the two radio frequencies. By placing the second generator under the control of the receiving operator Fessenden made it a "frequency determining element" by means of which the operator at the receiving station could control the pitch of tone of the arriving signals and also that of interfering signals of different but adjacent radio frequencies. Further, by utilizing a radio frequency generator at the receiving station, Fessenden's heterodyne permitted a great economy in power. None of these features would have been possible had he not recognized that the generator of one of the two radio frequencies whose effects are [page 117]

combined could be allowed to run continuously, – i.e., that, since beat signals would be produced only when both frequencies affected the receiver, it was necessary to cut up only one of them into dots and dashes at the transmitting station.

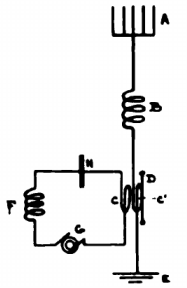


FIG. 4— DYNAMIC HETERODYNE

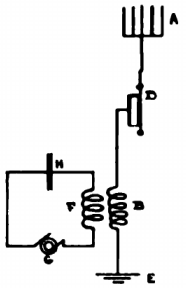


FIG. 5— ELECTROSTATIC HETERODYNE

Fig. 4 shows one of the earliest forms of heterodyne receiver. The aerial to ground oscillations flow from the antenna A through coil B and C' to earth E. Coil C' is of small dimensions and is carried on a mica diaphragm D. Near it (the assembly constituting a dynamometer heterodyne telephone) is mounted a fixed coil C, through which flows the local oscillatory current generated by the radio frequency alternator G in the circuit $FHCG$. By making the local frequency slightly different from the arriving frequency, the resultant force produced upon the diaphragm D by the interaction of the alternating-current fields of coils C and C' will produce a to-and-fro motion of a frequency equal to the difference between the two oscillation frequencies. Thus, if the received waves have a frequency of 100000 cycles per second and the local generator runs at 101000 cycles, the diaphragm will vibrate at 1000 cycles per second and give off an audible signal beat note so long as waves arrive from the transmitter.

A study of the dynamometer heterodyne will bring out the fact that the intensity of signal response is dependent not merely upon the strength of the received waves, but also on the strength of the local current. Thus the receiver gives strongly amplified tones from the desired signals; undesired signals of even slightly different frequencies will produce beat tones of a radically different pitch, and these are usually of proportionally smaller amplitude. Furthermore, the heterodyne receiver will give maximum amplification only when the received waves are purely sustained, for such sinusoidal oscillations are essential to maximum beat formation. This means that impulsive or irregular interfering disturbances, such as are produced by spark transmitters or by atmospheric (static) strays, will not be amplified nearly so much as the desired signals. All this is due to the unusual type of selective polarization utilized in these heterodyne receivers. By this function an exceedingly valuable means of discriminating between desired and interfering signals has been provided.

Fessenden pointed out that interaction between electrostatic fields might also be used for heterodyne reception. This arrangement, which utilizes voltage instead of current effects, is shown in Fig. 5. Here the antenna circuit passes from A through the electrostatic telephone D and inductance B to earth E. The electrostatic telephone consists of a conducting diaphragm supported close to a fixed plate, the two forming a condenser. When a voltage is impressed upon the two plates an attractive force is set up between them and the diaphragm moves toward the fixed conductor. By varying the charging

potential at audible frequency a tone may be produced.

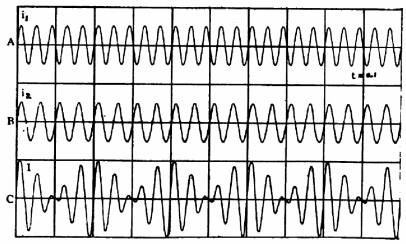


FIG. 6—FORMATION OF BEATS

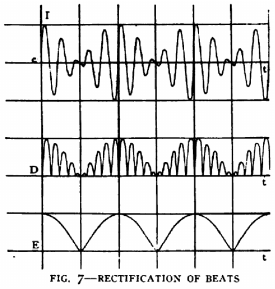


FIG. 7—RECTIFICATION OF BEATS

In using the electrostatic telephone for heterodyne reception, it is necessary to apply to the telephone not only the signal voltage but also the locally generated potential. This may conveniently be done by the inductive coupling between coils F and B of Fig. 5, coil F being in circuit with the local generator G and the tuning condenser H. Under these conditions the operation may be graphically represented as shown in Figs. 6 and 7 although, for convenience of drawing, the frequency ratio is greater than in radio practice. Referring to Fig. 6, the curve of axis A may be taken to represent the voltage impressed upon the condenser telephone by the arriving signal. The curve of axis B will then represent the potential due to the local generator. This is evidently of a different frequency, the ratio of A (frequency N_1) to B (frequency N_2) being 1.25 in these diagrams. The curve along axis C shows the algebraic sum of the potentials on A and B; by reason of the difference in frequencies this resultant potential fluctuates from maximum to minimum at the beat frequency or $N_1 - N_2$ cycles per second, as indicated.

The curve of axis C is repeated at the top of Fig. 7. Axis D shows the same pulsating radio frequency voltage in effect completely rectified; this is the process performed by the electrostatic telephone, for, although there is in it no electrical rectification or biasing, the device nevertheless produces a unidirectional mechanical force regardless of the polarity of the applied potential. Thus on axis D we have a graph of the mechanical [page 118]

force applied to the diaphragm of the condenser telephone, as it acts in its capacity of a perfect electromechanical rectifier. The diaphragm itself by reason of its inertia, cannot follow the rapid individual attractions, but will execute an averaged vibration somewhat as shown on axis E. This slow vibration is of the beat or audio frequency, and consequently gives rise to an audible signal tone, just as in the case of the dynamometer or electromagnetic heterodyne of Fig. 4.

The electrostatic heterodyne possesses the same amplifying and discriminating characteristics as the other forms, and is somewhat more sensitive than the dynamometer form. With high receiving aerials, and particularly with radio frequency amplifiers, it is not difficult to copy transoceanic signals with the electrostatic heterodyne. Without amplifiers, but using the large antenna of the Navy station at Arlington, Va., (which has a maximum height of 600 feet), and a small Poulsen arc as the local source of oscillations, signals have been received from San Francisco with the electrostatic telephone heterodyne.

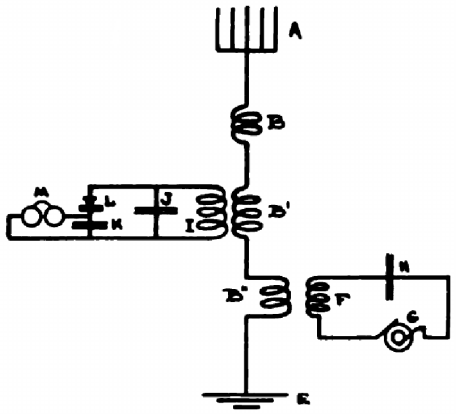


FIG. 8—TYPICAL CIRCUIT OF RECTIFIER HETERODYNE

A great increase in sensitiveness of the heterodyne receiver was secured by combining the local source of oscillations with an electrically rectifying radio receiver. A typical arrangement of this sort is shown in Fig. 8. Signal currents are impressed upon the rectifying detector L from the antenna A and across coupling B' I. An incoming wave of constant amplitude will produce a constant radio frequency potential across the detector, and this will result in a uniform direct current through the telephones M. When radio frequency currents of slightly different frequency, from generator G, are also impressed upon the detector by way of the inductive couplings F B" and B' I, a radically changed condition exists. The local voltages interact to produce potential beats across the detector; the rectifying system is subjected to fluctuating voltages such as appear on axis C of Fig. 7 so long as both radio frequencies are applied. These beat-voltages are inevitably rectified into a fluctuating or pulsating direct current, which passes through the telephone windings and produces a beat-tone signal.

Since the electrical rectifier-telephone combination is of great sensitiveness, its application to heterodyne reception has made it possible to receive selectively over great distances with comparatively small antenna structures. By proper choice of detector characteristics, the high powers of discrimination and amplification are also secured in this form of heterodyne, and, since its output is a varying audio frequency electrical current, it lends itself to combination with amplifiers, electrical tuning systems, etc.

Fig. 9 illustrates the soundproof receiving room of the Arlington Naval station, where some interesting early long distance work with the rectifier-heterodyne was done. On a special series of experiments, messages were copied several times every day from the U.S.S. Salem as she steamed to Gibraltar. By combining the signals with local oscillations generated from the small arc shown at the right of the photograph, the heterodyne amplification secured made it possible to read messages sent by the ship long after she had gone so far that her signals could not be understood when ordinary receivers were used. Developments of the heterodyne since this work in 1911 have kept it at the head of the list, and by its exceptional sensitiveness and selectivity, it has made possible the long-distance communication records which have been announced in the past few years. Although the type of transmitter used may vary largely, and while various auxiliary devices may be combined with the receiver, every exceptional radio performance of recent years has depended for its success upon the same principle. In every case the receiver has embodied some form of the rectifying heterodyne. [page 119]

4.3 Trunked telephony system

A United States patent (number 1,835,031) granted to Lloyd Espenschied and Herman Affel on 8 December 1931 documents in some detail a telephony system using frequency-division multiplexing to convey multiple conversations along a common coaxial cable. The patent's claims centered on the coaxial cable which was a novelty at that time, however the rest of the system as described in this patent gives us a fascinating look into communications technology of that era.

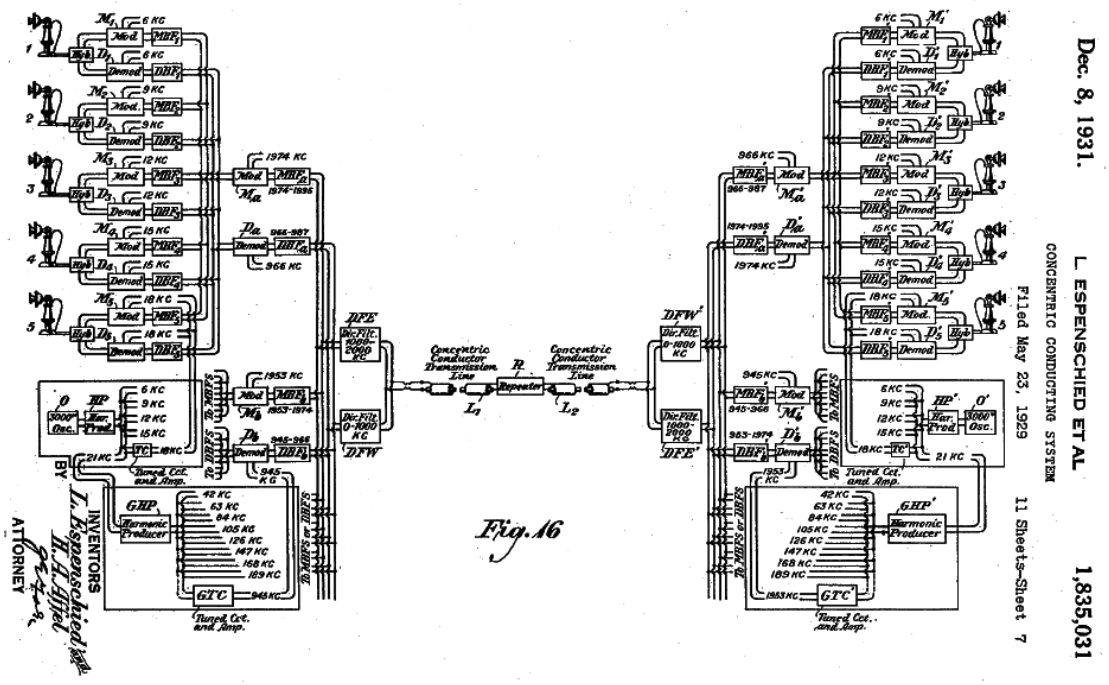


Fig. 16

Each telephone set connected to the rest of the system through a modulator and demodulator, each with a local oscillator. The frequencies of these local oscillators were set in 3 kHz increments beginning at 6 kHz, providing 3 kHz of sub-channel bandwidth for each telephone which is sufficient for ordinary human speech. Thus, the 0-3000 Hz speech range of each person speaking into their telephones would be shifted upward in frequency by modulator (mixer) circuits to 6000-9000 Hz, 9000-12000 Hz, 12000-15000 Hz, 15000-18000 Hz, and 18000-21000 Hz. The signals transmitted by the five telephones were combined to form a "group" signal modulated once more to shift their combined frequencies into the megaHertz range (west-to-east signals being shifted into a range 1-2 MHz; east-to-west signals being shifted to a maximum of 1 MHz).

Also worthy of note are the *harmonic producers* used to generate sinusoidal signals at 3 kHz frequency intervals. The patent gives no detail as to how these harmonic producers were built[2], but we may presume they asymmetrically distorted the 3 kHz oscillator's wave so as to produce both

[2]The patent's text simply says "By means of a harmonic producer HP of known type the required modulation and demodulation frequencies of 6, 9, 12, 15 and 18 kc. may be frequency as harmonics of the fundamental frequency of 3,000 cycles."

even-numbered and odd-numbered harmonics (i.e. 3 kHz producing 2nd harmonic of 6 kHz, 3rd harmonic of 9 kHz, 4th harmonic of 12 kHz, and so on). Each of these harmonics were selected by "tuned circuits" (band-pass filters) and suitably amplified for use as local oscillator signals at each modulator/demodulator.

The seventh harmonic of the 3 kHz oscillator (21 kHz) was then selected and re-distorted to produce harmonics at 21 kHz intervals, to be used as local oscillator signals at modulators and demodulators for each "group" of five telephone sets. The figure shows these group oscillator signals starting at 42 kHz and extending through 966 kHz, providing up to forty-five (45) groups of five telephones each.

It is impressive to see that signal mixing (modulation and demodulation) could be used to multiplex so many telephone channels over a single coaxial cable, all using circa-1929 analog vacuum-tube electronic technology!

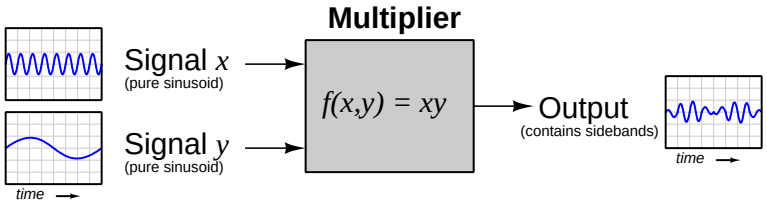
Chapter 5

Derivations and Technical References

This chapter is where you will find mathematical derivations too detailed to include in the tutorial, and/or tables and other technical reference material.

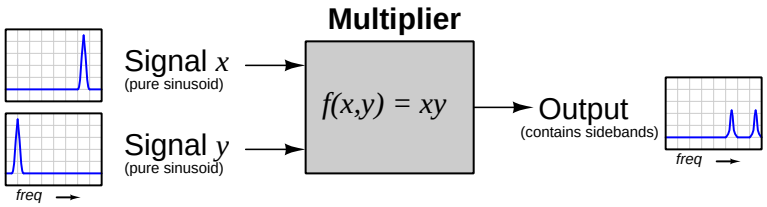
5.1 Mathematics of signal mixing

A common method for modulating signals is to multiply them together using a device constructed for that purpose. The amplitude of the lower-frequency signal serves to boost or attenuate the amplitude of the higher-frequency signal, so that the result is an *amplitude-modulated* signal where the signal's "envelope" follows the amplitude of the lower-frequency signal. A device constructed for the purpose of amplitude-modulating sinusoidal signals is commonly called a *mixer*[1].



A mixer's function as an amplitude modulator is easy enough to grasp when we view the signals from the perspective of the time domain, as shown by the blue-colored waveforms in the above illustration: in this example, signal y modulates the amplitude of signal x. What is far less intuitive to grasp is how these signals appear in the *frequency domain*, viewed as independent sine (or cosine) waves at different frequencies superimposed upon each other to comprise waveforms more complex than simple sinusoids.

When two sinusoidal signals are multiplied (i.e. "mixed" in the modulation sense of the word), the result is the production of additional frequencies called *sidebands*. The lower sideband signal's frequency is the difference between the two input signal frequencies, and the upper sideband signal's frequency is the sum of the two input signal frequencies. The mixer illustration will now be shown with signals represented in the frequency domain rather than in the time domain:



Proving just why sidebands are produced requires the application of some mathematics, specifically *trigonometry*.

[1] This terminology may be confusing for anyone accustomed to working with professional-grade audio equipment, where a "mixer" is a very different sort of electronic device, used to sum a set of audio-frequency signals to make a single (or multiple) channel of audio information.

We will begin our derivation by representing the mixer's two input signals as cosine functions, each with its own frequency (ω_x and ω_y). Since the value passed to a cosine function must be an angle and not a frequency, we will represent the two input signals as $\cos \omega_x t$ and $\cos \omega_y t$, respectively[2]. We may express the output signal as a time-based function, as follows:

$$f(x, y) = (\cos \omega_x t)(\cos \omega_y t)$$

Among the various mathematical identities in trigonometry is this one, relating the product of two cosine functions with cosines of the angular sum and difference:

$$\cos x \cos y = \frac{\cos(x - y) + \cos(x + y)}{2}$$

Substituting $\omega_x t$ for x and $\omega_y t$ for y yields the following result:

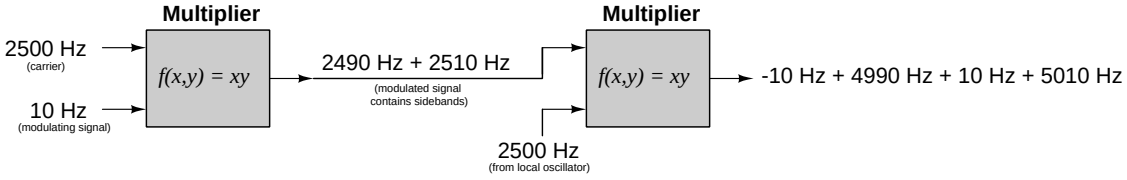
$$\cos(\omega_x t) \cos(\omega_y t) = \frac{\cos(\omega_x t - \omega_y t) + \cos(\omega_x t + \omega_y t)}{2}$$

Factoring out t in the angle terms on the right-hand side of this equation:

$$\cos(\omega_x t) \cos(\omega_y t) = \frac{\cos[(\omega_x - \omega_y)t] + \cos[(\omega_x + \omega_y)t]}{2}$$

This proves the multiplier's output signal consists of two sidebands. Given two sinusoidal input signals of differing frequency ($\cos(\omega_x t)$) and ($\cos(\omega_y t)$), the result is a *lower sideband* sinusoid having a frequency equal to the difference of the two input frequencies ($\cos[(\omega_x - \omega_y)t]$) and an *upper sideband* sinusoid having a frequency equal to the sum of the two input frequencies ($\cos[(\omega_x + \omega_y)t]$). The divisor of 2 merely means these sidebands will be weaker (i.e. less amplitude) than the input signals, and is irrelevant from the perspective of frequency.

For example, if signal x has a frequency of 2500 Hz and signal y has a frequency of 10 Hz, the two sidebands will have frequencies of 2490 Hz and 2510 Hz, respectively. This is what happens when a 2500 Hz *carrier* wave is amplitude-modulated by a 10 Hz *modulating* wave. If these modulated sideband signals happen to be fed into another multiplier along with a 2500 Hz *local oscillator* signal, the lower sideband wave of 2490 Hz will generate two sidebands of its own (−10 Hz and 4990 Hz) while the upper sideband wave will generate two sidebands of its own (10 Hz and 5010 Hz).



The two low-frequency signals (10 Hz and −10 Hz) carry the information embodied by the original modulating signal. In fact, these two signals are really identical to each other, since the

[2] ω represents frequency in angular units per second, typically radians per second. Time (t), of course, is typically measured in seconds. The product of radians per second and seconds is simply radians: an angle.

cosine function is an *even*[3] mathematical function and therefore $\cos(-10t) = \cos(10t)$. The two high-frequency signals (4990 Hz and 5010 Hz) are irrelevant to our purposes here and may be easily filtered out using a low-pass filter, if they aren't already severely attenuated by the bandwidth limitations of the circuitry.

Thus, we see how a multiplying function may be used to *amplitude-modulate* a signal onto a higher-frequency carrier, and how the same type of function may be used to *demodulate* that modulated signal to obtain the original signal's information. Another way of describing this is to say the first multiplier *upconverts* the 10 Hz signal into sidebands of much greater frequency, and then the second multiplier *downconverts* those sidebands to recover the original 10 Hz signal frequency.

The use of a multiplier to "downconvert" the frequency of a signal by mixing it with another signal of differing frequency is called *heterodyning*. This is the same phenomenon you hear with two audio tones of nearly-identical pitch played simultaneously[4]: the result as heard by your ears is a low-frequency "beat" pattern representing the difference in frequency between the two original tones. Heterodyning was first applied to radio technology as a means to impart an audible tone to an inaudible (radio-frequency) signal, in order to construct a radio receiver circuit capable of indicating the presence or absence of a carrier-wave signal broadcast by a transmitter using a keyswitch to switch on and off.

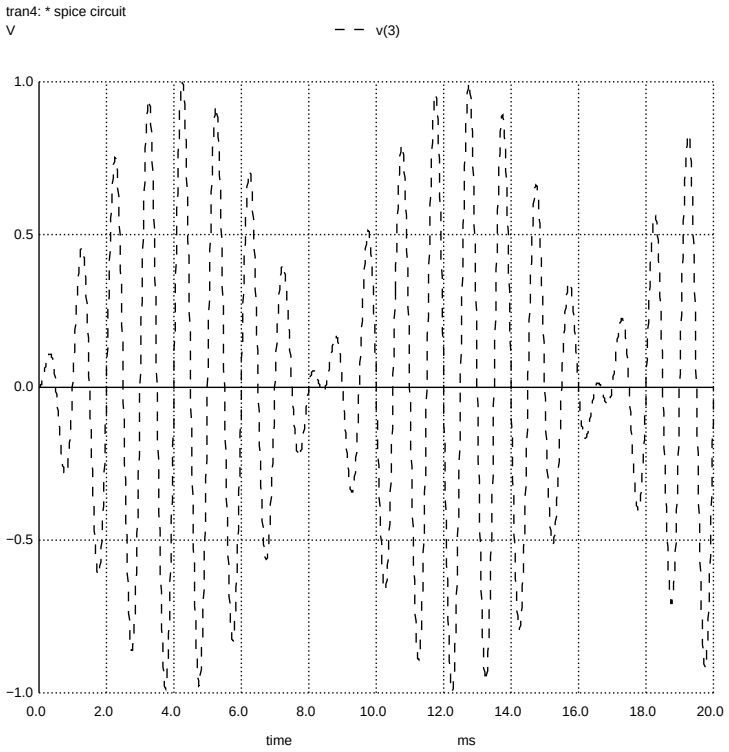
[3] Any mathematical function $f(x)$ is considered *odd* if $f(-x) = -f(x)$, as is the case with polynomial functions having only odd degree terms (e.g. $2x^7 + 8x^5 - x^3 + 9x$). Conversely, any mathematical function is considered *even* if $f(-x) = f(x)$, as with polynomial functions having only even (or zero) degree terms (e.g. $x^6 - 6x^4 + 3x^2 - 8$). The sine function is considered odd while the cosine function is even: for example, $\sin(-10) = -\sin(10)$ while $\cos(-10) = \cos(10)$.

[4] This is also the same phenomenon responsible for the perception of musical *harmony*: two or more tones played simultaneously that give rise to the impression of even more tones. This is why two or three tones of appropriate frequency played with each other sound so much richer than those same tones played one at a time: your auditory senses "mix" them together in such a way that you hear not only the original tones but also their respective sidebands. When mixed by a nonlinear system (in this case, your sense of hearing provides the nonlinearity necessary for true mixing to occur) the whole actually is greater than the sum of its parts.

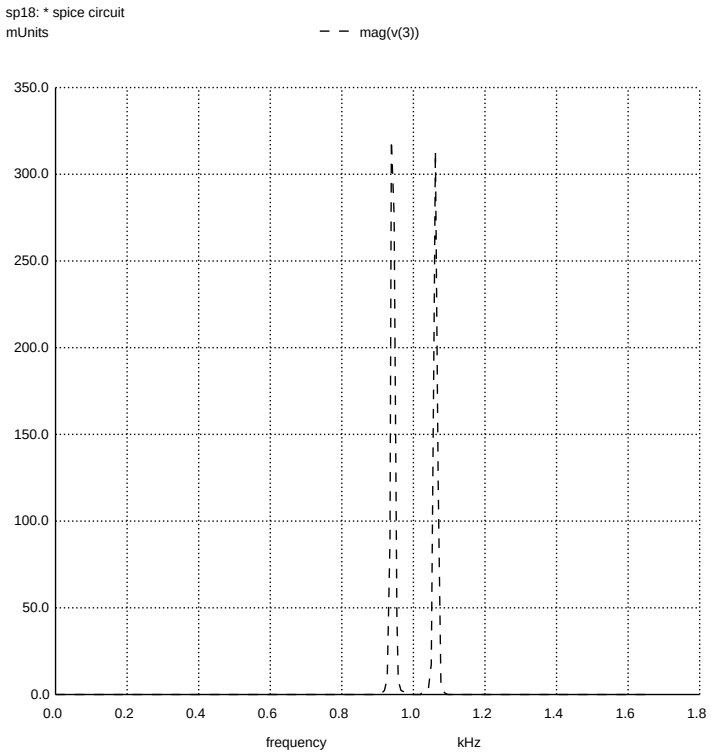
The following computer simulation (run in NGSPICE) shows a 60 Hz sinusoidal voltage (`v1`) and a 1 kHz sinusoidal voltage (`v2`) being mixed by a nonlinear dependent voltage source `b1` in the spice netlist. The three 5 kΩ resistors do nothing but satisfy SPICE's need for complete circuit loops:

```
* Mixer circuit
v1 1 0 sin (0 1 60 0 0)
v2 2 0 sin (0 1 1000 0 0)
b1 3 0 v=(v(1)*v(2))
r1 1 0 5000
r2 2 0 5000
r3 3 0 5000
.tran 0.01m 20m
.plot tran v(3)
.end
```

Plotted over a range of 0 to 20 milliseconds, the mixed signal (voltage between node 3 and ground) appears as a standard amplitude-modulated waveform:



A spectrum plot (generated using NGSPICE's `fft` command[5] shows the two sidebands at 940 Hz and 1060 Hz:



[5]In order to obtain a crisp plot of these two sidebands, the transient analysis parameters of the NGSPICE netlist had to be modified from the values giving a clean time-domain plot. Instead of analyzing over 0.01 millisecond intervals from 0 milliseconds to 20 milliseconds, the time values were shifted to 0.3 millisecond intervals from 0 milliseconds to 150 milliseconds. The interval shift from 0.01 ms to 0.3 ms shortened the frequency domain of the FFT plot, while the increased span from 20 ms to 150 ms made the peaks narrower and sharper. The modified "card" in the netlist file now reads `.tran 0.3m 150m`. The sequence of NGSPICE commands used to generate this FFT plot (after specifying the netlist filename and "running" the simulation with the `run` command) are as follows: `linearize v(3) ; fft v(3) ; plot mag(v(3))`.

5.2 Square-law mixing

Few electronic devices naturally *multiply* input signals[6], and so we must often exploit other types of device nonlinearities to construct mixer circuits.

An example of an exploitable nonlinearity is the *square-law* characteristic of certain electronic devices such as field-effect transistors. Recall the characteristic equation for a FET, where drain current is proportional to the square of applied gate-source voltage:

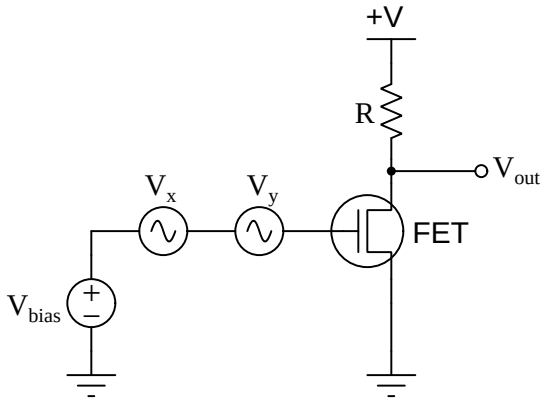
$$I_D = k(V_G - V_{TO})^2$$

Where,
I_D = Drain current, Amperes
k = Constant of proportionality
V_G = Gate-to-source voltage, Volts
V_{TO} = Threshold ("turn-on") voltage, Volts

For the sake of simplification, we may consider a FET to exhibit the following proportional relationship between applied gate-source voltage and drain current:

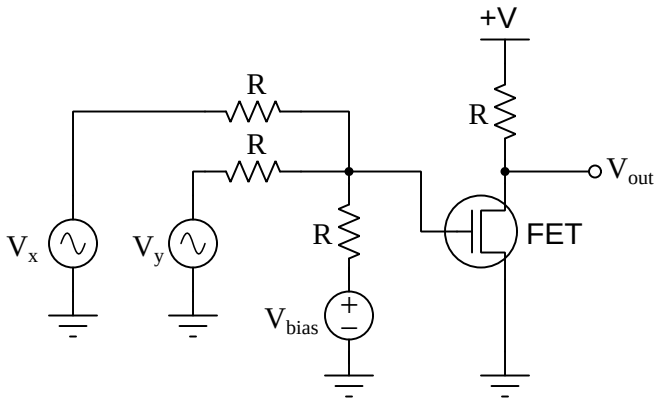
$$I_D \propto V_G^2$$

We may exploit the inherently quadratic nature of the FET in order to mix two sinusoidal signals of differing frequency by electrically *adding* those signals together and sending that superposition into the FET as a single input signal voltage between gate and emitter. An elementary circuit might look something like this:



[6] One notable exception is a Hall Effect device, where the output voltage is proportional to the product of the excitation current and the magnetic flux permeating the Hall Effect substrate. If the two input signals are directed to function as current through the device and current through a coil (producing the necessary magnetic field), the Hall Effect device's output signal will directly reflect the product of the two input signals. A major limitation of this technology for communication applications, though, is bandwidth. Strong magnetic fields are necessary through the Hall Effect substrate to obtain strong output signal voltages, and this requires a coil with a large inductance value (and along with that, typically high resistance and parasitic capacitance) which cannot be operated effectively at high frequencies.

Addition of input voltages V_x and V_y is achieved simply by virtue of their series connection, since voltages always add in series. Other possibilities exist for summing the two AC signals at the FET's input, such as the following resistor network:



The fact that a network of resistors can only *average* the two input signals rather than *sum* them is an irrelevant objection. What does matter is that somehow we end up with a mathematical *product* of the two input signals, regardless of how or if they may be attenuated.

Now we must mathematically explore how it might be possible to extract the product of two signals from their squared sum. First, we will represent our two input signal voltages as cosine waveforms with different frequencies (ω_x and ω_y). The superposition of these two signals will comprise the FET's gate voltage, and its output (i.e. drain current) will be proportional to the square of that superposition:

$$I_D \propto [\cos(\omega_x t) + \cos(\omega_y t)]^2$$

Algebraic expansion of this equation is as simple as expanding any squared binomial (e.g. $(a+b)^2 = a^2 + 2ab + b^2$):

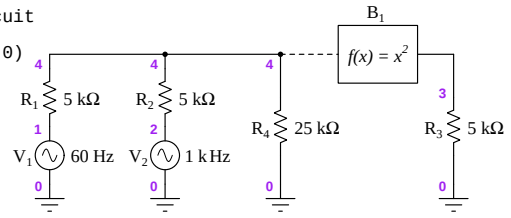
$$I_D \propto \cos^2(\omega_x t) + 2\cos(\omega_x t)\cos(\omega_y t) + \cos^2(\omega_y t)$$

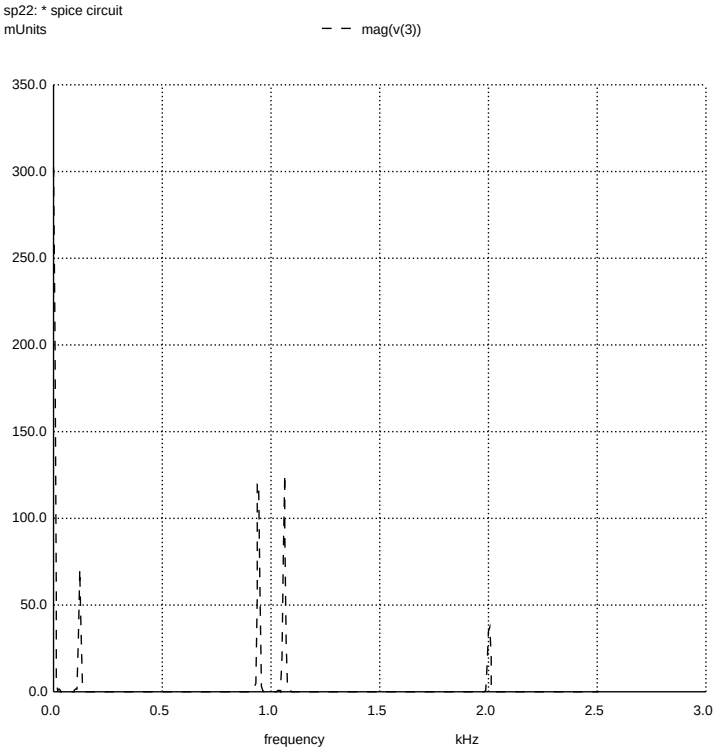
The middle term of this expanded equation clearly shows the two cosine-based signals multiplied by one another, which is precisely the mathematical relationship we require for mixing (i.e. amplitude modulation). Included with this product-of-signals term are two additional terms, each one being the square of its respective sinusoidal signal.

At this point the trigonometric identity $\cos^2 x = \frac{1+\cos(2x)}{2}$ proves useful to explore these additional terms. This identity tells us that squaring a cosine function results in another cosine function with a doubled angle. In the case of AC signals represented by cosines of continuously-changing angles, this doubling of the angle value is equivalent to a doubling of frequency. In other words, these cosine-squared terms are actually *second harmonics* of their respective x and y signal frequencies. This tells us the mixing will be impure, containing additional frequencies a pure multiplying mixer would never produce. With proper filtering after the mixer circuit stage, these harmonics may not pose any serious problem, but it is important to realize they will be present as an artifact of the FET's square-law nonlinearity.

A computer simulation (using NGSPICE) showing two sinusoidal waveforms – 60 Hz and 1 kHz – being superimposed in a passive averager network then mixed via a "square" function is shown in the following netlist and spectrum plot:

```
* Square-law mixer circuit
v1 1 0 sin (0 1 60 0 0)
v2 2 0 sin (0 1 1000 0 0)
b1 3 0 v=(v(4)^2)
r1 1 4 5000
r2 2 4 5000
r3 3 0 5000
r4 4 0 25000
.tran 0.2m 150m
.end
```

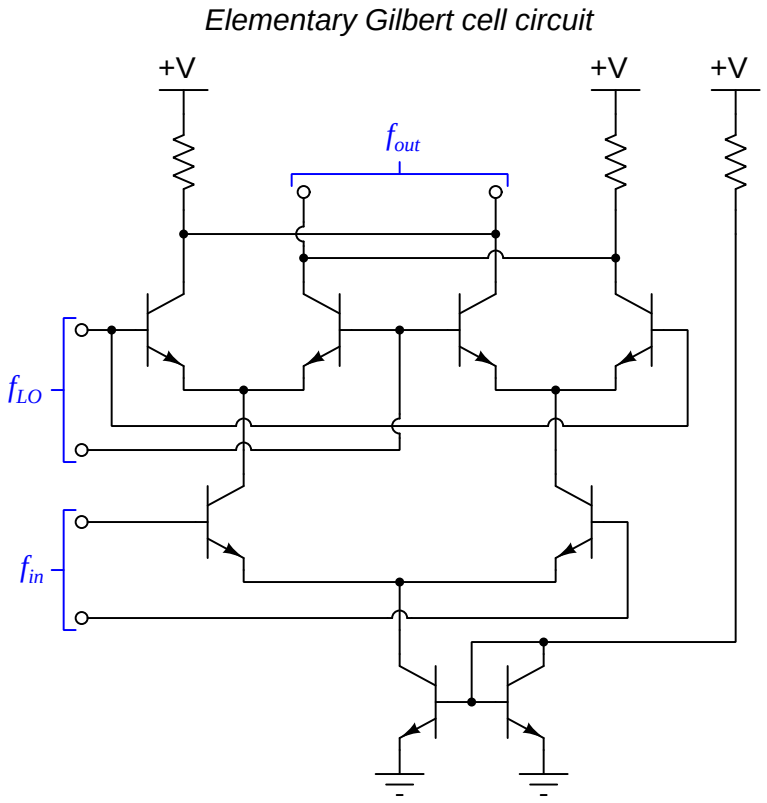




The two sidebands at 940 Hz and 1060 Hz are there, as are the two second-order harmonics of the 60 Hz signal (120 Hz) and the 1 kHz signal (2 kHz). A strong DC component (frequency = zero) also appears in the spectrum, the result of the $\frac{1}{2}$ constants in the two $\cos^2$ terms.

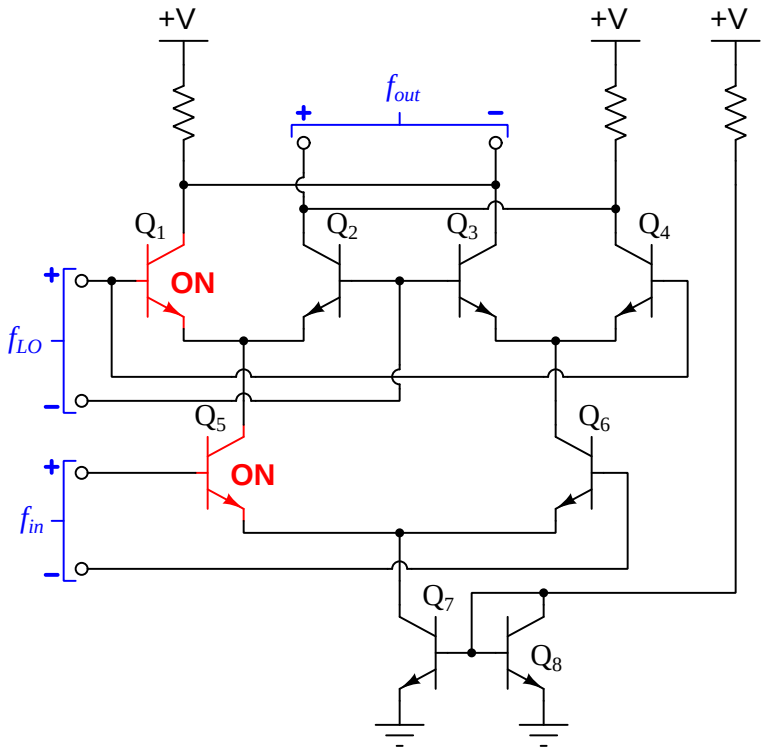
5.3 Gilbert cell circuit

A differential amplifier circuit commonly used for RF signal mixing is the so-called *Gilbert cell*, named after Barrie Gilbert. Its most basic configuration is a six-transistor network of differential amplifiers with a current source, as shown in the following schematic diagram:



Even in its simplified form the Gilbert cell circuit seems daunting at first observation. However, we may approach its analysis by simplifying the types of signals we send to its two differential inputs (f_{LO} and f_{in}): using "square-wave" signals rather than sinusoidal. This will be an example of the *limiting cases* problem-solving strategy in action, considering only the "limit" cases of peak AC signal values rather than their full analog range in order to simplify the circuit's action.

First we will consider the action of the Gilbert cell circuit with both input terminal pairs positive on top and negative on bottom. Transistor states will be assessed as either "on" or "off" in this limiting-case analysis:



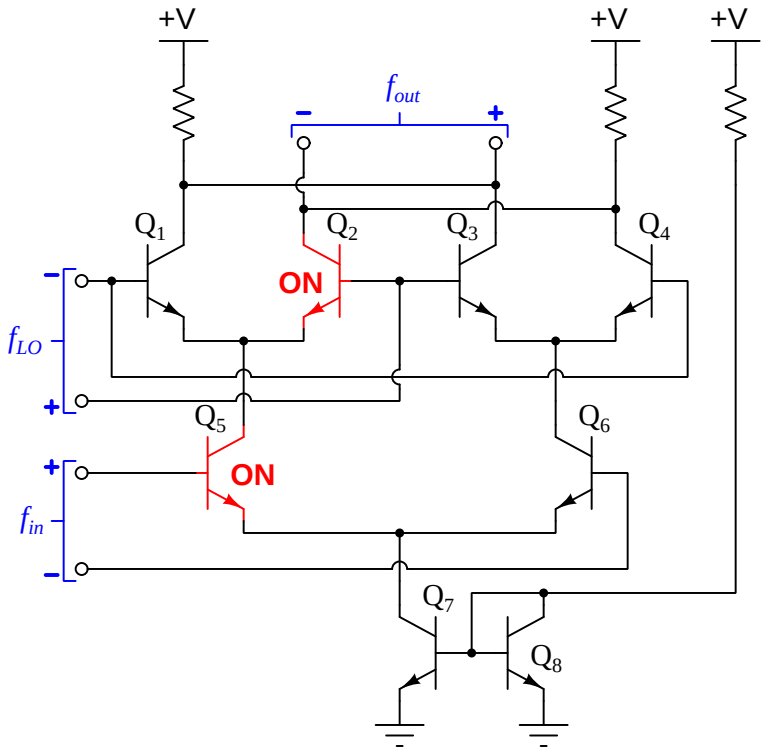
f_{in}'s polarity turns on Q_5 more than Q_6, allowing differential pair Q_1/Q_2 to function while shutting off differential pair Q_3/Q_4. f_{LO}'s polarity turns on Q_1, drawing down the electrical potential at the right-hand terminal of f_{out} while the left-hand terminal of f_{out} rises to full +V potential as a result of both Q_2 and Q_4 being non-conducting.

For the sake of analyzing this circuit as a *mixer* (ideally, a *multiplier*), we will arbitrarily consider all three signals (f_{LO}, f_{in}, and f_{out}) to be numerically "positive" in the polarities shown, and represent this state of affairs as the product of a positive quantity and another positive quantity:

$$(f_{LO}) \times (f_{in}) = (f_{out})$$

$$(+) \times (+) = (+)$$

Next we will reverse the polarity of the local oscillator signal (f_{LO}) and re-analyze the circuit in terms of "on" transistor states:

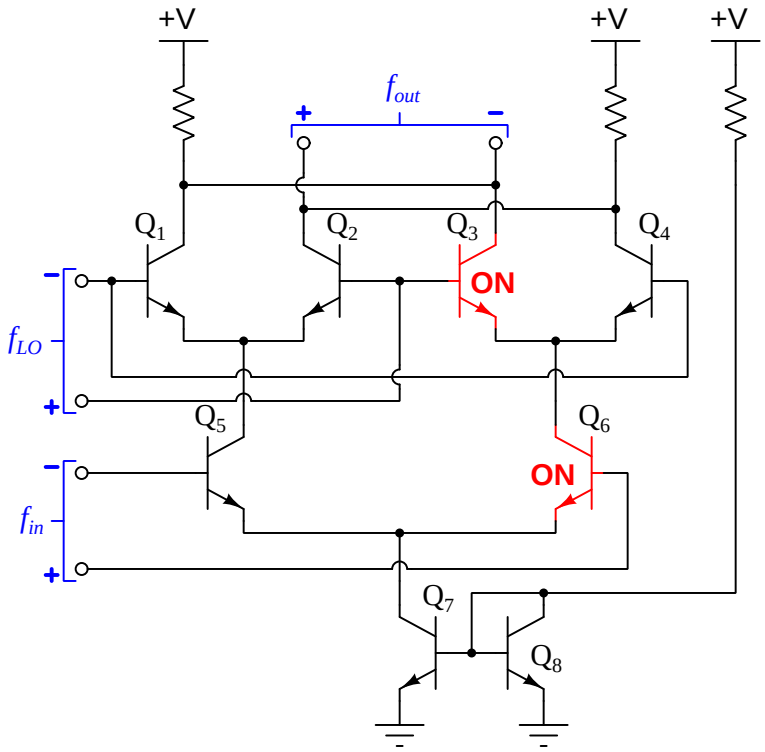


f_{in}'s polarity still has Q_5 on more than Q_6, allowing differential pair Q_1/Q_2 to function while shutting off differential pair Q_3/Q_4. The reversed polarity of f_{LO} turns on Q_2, drawing down the electrical potential at the left-hand terminal of f_{out} while the right-hand terminal of f_{out} rises to full +V potential. Represented mathematically:

$$(f_{LO}) \times (f_{in}) = (f_{out})$$

$$(-) \times (+) = (-)$$

Next we will reverse the polarity of the input signal (f_{in}) and re-analyze the circuit in terms of "on" transistor states:

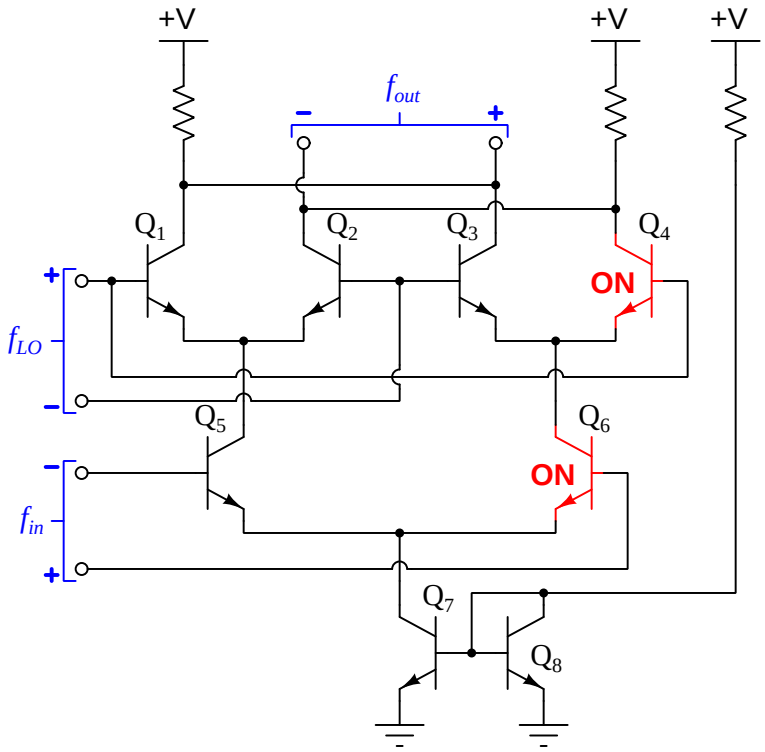


f_{in}'s polarity now turns on Q_6 more than Q_5, de-energizing differential pair Q_1/Q_2 while allowing differential pair Q_3/Q_4 to function. The polarity of f_{LO} turns on Q_3, drawing down the electrical potential at the right-hand terminal of f_{out} while the left-hand terminal of f_{out} rises to full +V potential. Represented mathematically:

$$(f_{LO}) \times (f_{in}) = (f_{out})$$

$$(-) \times (-) = (+)$$

Lastly we will reverse the polarity of the local oscillator again (f_{LO}) and re-analyze the circuit in terms of "on" transistor states:

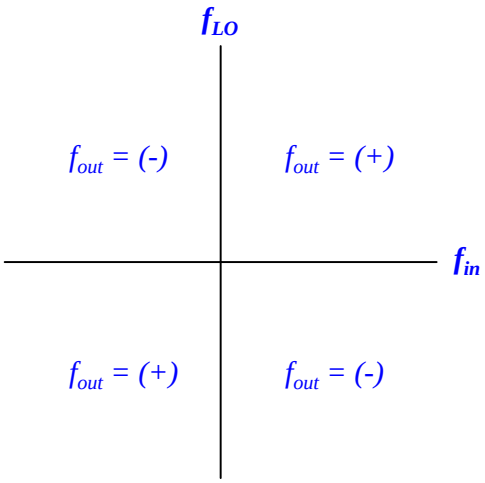


f_{in}'s polarity still holds Q_6 in the "on" state more than Q_5, maintaining differential pair Q_1/Q_2 in an inactive state while allowing differential pair Q_3/Q_4 to function. The reversed polarity of f_{LO} turns on Q_4, drawing down the electrical potential at the left-hand terminal of f_{out} while the right-hand terminal of f_{out} rises to full +V potential. Represented mathematically:

$$(f_{LO}) \times (f_{in}) = (f_{out})$$

$$(+) \times (-) = (-)$$

As we can see from these four limiting-case analyses, the Gilbert cell functions as a *four-quadrant multiplier*. This phrase simply means the circuit is able to effectively multiply all four combinations of mathematical signs for its two inputs. Shown on an x-y coordinate graph, each "quadrant" of the graph represents a different combination of input signal polarity and local oscillator polarity:



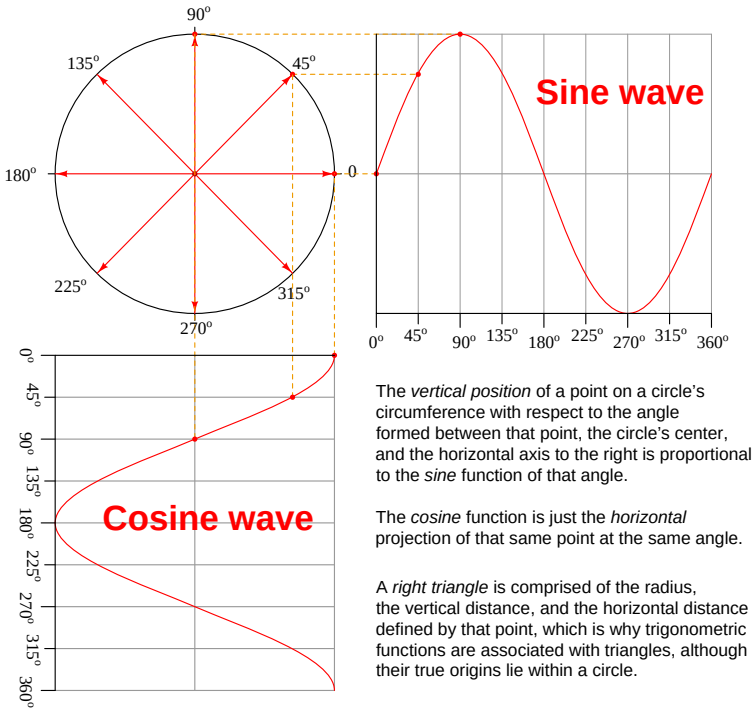
Multiplication is the fundamental property necessary for signal mixing, and a mixer circuit balanced such that it multiplies equally well in all four quadrants will function with minimal artifact frequencies present in its output.

When operated as RF mixers, the local oscillator signal (f_{LO}) is often a square-wave or an over-driven sinusoid rather than a pure sinusoid. Such signals are easier to generate on demand, and when the upper portion of the Gilbert cell is driven as such, those four transistors need not function linearly in order for the circuit to perform its task as a mixer. Only the lower transistors (operating with the "live load" current mirror) need function linearly in order to be able to process non-square-wave signals at f_{in}.

5.4 Trigonometry Reference

Trigonometry is a very useful mathematical tool for understanding modulation techniques operating on sinusoidal (sine-wave-shaped) signals. This reference lists a number of trigonometric identities and relations involving only sine and cosine functions. The variables x and y are used to denote angles. In AC circuits the angle of an oscillating signal is constantly changing, and so instead of using x or y to denote a constant angle, we often use the product of angular velocity and time (ωt) to denote the angle as a function of time[7].

5.4.1 The Unit Circle



5.4.2 Pythagorean identity for sine and cosine

$$\sin^2 x + \cos^2 x = 1$$

[7] Since ω is expressed in angular units per second (e.g. radians per second, or degrees per second) and t is expressed in seconds, the product of ω and t must cancel the unit of time and leave us with just an angle.

5.4.3 Odd and even functions

Any mathematical function $f(x)$ is considered *odd* if $f(-x) = -f(x)$, as is the case with polynomial functions having only odd degree terms (e.g. $2x^7 + 8x^5 - x^3 + 9x$). Conversely, any mathematical function is considered *even* if $f(-x) = f(x)$, as with polynomial functions having only even (or zero) degree terms (e.g. $x^6 - 6x^4 + 3x^2 - 8$). The sine function is considered odd while the cosine function is even: for example, $\sin(-10) = -\sin(10)$ while $\cos(-10) = \cos(10)$.

$$\sin(-x) = -\sin(x) \qquad \cos(-x) = \cos(x)$$

5.4.4 Sums and differences of angles

$$\sin(x \pm y) = (\sin x \cos y) \pm (\cos x \sin y) \qquad \cos(x \pm y) = (\cos x \cos y) \mp (\sin x \sin y)$$

5.4.5 Products and sums of sine and cosine functions

$$\sin x \sin y = \frac{\cos(x-y) - \cos(x+y)}{2} \qquad \cos x \cos y = \frac{\cos(x-y) + \cos(x+y)}{2}$$

$$\sin x \cos y = \frac{\sin(x+y) + \sin(x-y)}{2} \qquad \cos x \sin y = \frac{\sin(x+y) - \sin(x-y)}{2}$$

$$\sin x + \sin y = 2\sin\left(\frac{x+y}{2}\right)\cos\left(\frac{x-y}{2}\right)$$

$$\cos x + \cos y = 2\cos\left(\frac{x+y}{2}\right)\cos\left(\frac{x-y}{2}\right)$$

$$\sin x - \sin y = 2\cos\left(\frac{x+y}{2}\right)\sin\left(\frac{x-y}{2}\right)$$

$$\cos x - \cos y = -2\sin\left(\frac{x+y}{2}\right)\sin\left(\frac{x-y}{2}\right)$$

5.4.6 Squares of sine and cosine functions

$$\sin^2 x = \frac{1 - \cos(2x)}{2} \qquad \cos^2 x = \frac{1 + \cos(2x)}{2}$$

5.4.7 Doubled angles

$$\sin(2x) = 2\sin x \cos x \qquad \cos(2x) = \cos^2 x - \sin^2 x$$

5.4.8 Halved angles

$$\sin\left(\frac{x}{2}\right) = \pm\sqrt{\frac{1-\cos x}{2}} \qquad \cos\left(\frac{x}{2}\right) = \pm\sqrt{\frac{1+\cos x}{2}}$$

Chapter 6

Programming References

A powerful tool for mathematical modeling is text-based *computer programming*. This is where you type coded commands in text form which the computer is able to interpret. Many different text-based languages exist for this purpose, but we will focus here on just two of them, *C++* and *Python*.

6.1 Programming in C++

One of the more popular text-based computer programming languages is called *C++*. This is a *compiled* language, which means you must create a plain-text file containing C++ code using a program called a *text editor*, then execute a software application called a *compiler* to translate your "source code" into instructions directly understandable to the computer. Here is an example of "source code" for a very simple C++ program intended to perform some basic arithmetic operations and print the results to the computer's console:

```
#include <iostream>
using namespace std;

int main (void)
{
  float x, y;

  x = 200;
  y = -560.5;

  cout << "This simple program performs basic arithmetic on" << endl;
  cout << "the two numbers " << x << " and " << y << " and then" << endl;
  cout << "displays the results on the computer's console." << endl;

  cout << endl;

  cout << "Sum = " << x + y << endl;
  cout << "Difference = " << x - y << endl;
  cout << "Product = " << x * y << endl;
  cout << "Quotient of " << x / y << endl;

  return 0;
}
```

Computer languages such as C++ are designed to make sense when read by human programmers. The general order of execution is left-to-right, top-to-bottom just the same as reading any text document written in English. Blank lines, indentation, and other "whitespace" is largely irrelevant in C++ code, and is included only to make the code more pleasing[1] to view.

[1] Although not included in this example, *comments* preceded by double-forward slash characters (`//`) may be added to source code as well to provide explanations of what the code is supposed to do, for the benefit of anyone reading it. The compiler application will ignore all comments.

Let's examine the C++ source code to explain what it means:

- `#include <iostream>` and `using namespace std;` are set-up instructions to the compiler giving it some context in which to interpret your code. The code specific to your task is located between the brace symbols ({ and }, often referred to as "curly-braces").
- `int main (void)` labels the "Main" function for the computer: the instructions within this function (lying between the { and } symbols) it will be commanded to execute. Every complete C++ program contains a `main` function at minimum, and often additional functions as well, but the `main` function is where execution always begins. The `int` declares this function will return an *integer* number value when complete, which helps to explain the purpose of the `return 0;` statement at the end of the `main` function: providing a numerical value of zero at the program's completion as promised by `int`. This returned value is rather incidental to our purpose here, but it is fairly standard practice in C++ programming.
- Grouping symbols such as (parentheses) and {braces} abound in C, C++, and other languages (e.g. Java). Parentheses typically group data to be processed by a function, called *arguments* to that function. Braces surround lines of executable code belonging to a particular function.
- The `float` declaration reserves places in the computer's memory for two *floating-point* variables, in this case the variables' names being `x` and `y`. In most text-based programming languages, variables may be named by single letters or by combinations of letters (e.g. `xyz` would be a single variable).
- The next two lines assign numerical values to the two variables. Note how each line terminates with a semicolon character (`;`) and how this pattern holds true for most of the lines in this program. In C++ semicolons are analogous to periods at the ends of English sentences. This demarcation of each line's end is necessary because C++ ignores whitespace on the page and doesn't "know" otherwise where one line ends and another begins.
- All the other instructions take the form of a `cout` command which prints characters to the "standard output" stream of the computer, which in this case will be text displayed on the console. The double-less-than symbols (`<<`) show data being sent *toward* the `cout` command. Note how verbatim text is enclosed in quotation marks, while variables such as `x` or mathematical expressions such as `x - y` are not enclosed in quotations because we want the computer to display the numerical values represented, not the literal text.
- Standard arithmetic operations (add, subtract, multiply, divide) are represented as `+`, `-`, `*`, and `/`, respectively.
- The `endl` found at the end of every `cout` statement marks the end of a line of text printed to the computer's console display. If not for these `endl` inclusions, the displayed text would resemble a run-on sentence rather than a paragraph. Note the `cout << endl;` line, which does nothing but create a blank line on the screen, for no reason other than esthetics.

After saving this *source code* text to a file with its own name (e.g. `myprogram.cpp`), you would then *compile* the source code into an *executable* file which the computer may then run. If you are using a console-based compiler such as *GCC* (very popular within variants of the Unix operating system[2], such as Linux and Apple's OS X), you would type the following command and press the Enter key:

```
g++ -o myprogram.exe myprogram.cpp
```

This command instructs the `GCC` compiler to take your source code (`myprogram.cpp`) and create with it an executable file named `myprogram.exe`. Simply typing `./myprogram.exe` at the command-line will then execute your program:

```
./myprogram.exe
```

If you are using a graphic-based C++ development system such as Microsoft Visual Studio[3], you may simply create a new console application "project" using this software, then paste or type your code into the example template appearing in the editor window, and finally run your application to test its output.

As this program runs, it displays the following text to the console:

```
This simple program performs basic arithmetic on
the two numbers 200 and -560.5 and then
displays the results on the computer's console.

Sum = -360.5
Difference = 760.5
Product = -112100
Quotient of -0.356824
```

As crude as this example program is, it serves the purpose of showing how easy it is to write and execute simple programs in a computer using the C++ language. As you encounter C++ example programs (shown as source code) in any of these modules, feel free to directly copy-and-paste the source code text into a text editor's screen, then follow the rest of the instructions given here (i.e. save to a file, compile, and finally run your program). You will find that it is generally easier to

[2] A very functional option for users of Microsoft Windows is called *Cygwin*, which provides a Unix-like console environment complete with all the customary utility applications such as GCC!

[3] Using Microsoft Visual Studio community version 2017 at the time of this writing to test this example, here are the steps I needed to follow in order to successfully compile and run a simple program such as this: (1) Start up Visual Studio and select the option to create a New Project; (2) Select the Windows Console Application template, as this will perform necessary set-up steps to generate a console-based program which will save you time and effort as well as avoid simple errors of omission; (3) When the editing screen appears, type or paste the C++ code within the `main()` function provided in the template, deleting the "Hello World" `cout` line that came with the template; (4) Type or paste any preprocessor directives (e.g. `#include` statements, `namespace` statements) necessary for your code that did not come with the template; (5) Lastly, under the Debug drop-down menu choose either Start Debugging (F5 hot-key) or Start Without Debugging (Ctrl-F5 hotkeys) to compile ("Build") and run your new program. Upon execution a console window will appear showing the output of your program.

learn computer programming by closely examining others' example programs and modifying them than it is to write your own programs starting from a blank screen.

6.2 Programming in Python

Another text-based computer programming language called *Python* allows you to type instructions at a terminal prompt and receive immediate results without having to compile that code. This is because Python is an *interpreted* language: a software application called an *interpreter* reads your source code, translates it into computer-understandable instructions, and then executes those instructions in one step.

The following shows what happens on my personal computer when I start up the Python interpreter on my personal computer, by typing `python3`[4] and pressing the Enter key:

```
Python 3.7.2 (default, Feb 19 2019, 18:15:18)
[GCC 4.1.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The `>>>` symbols represent the prompt within the Python interpreter "shell", signifying readiness to accept Python commands entered by the user.

Shown here is an example of the same arithmetic operations performed on the same quantities, using a Python interpreter. All lines shown preceded by the `>>>` prompt are entries typed by the human programmer, and all lines shown without the `>>>` prompt are responses from the Python interpreter software:

```
>>> x = 200
>>> y = -560.5
>>> x + y
-360.5
>>> x - y
760.5
>>> x * y
-112100.0
>>> x / y
-0.35682426404995538
>>> quit()
```

[4] Using version 3 of Python, which is the latest at the time of this writing.

More advanced mathematical functions are accessible in Python by first entering the line `from math import *` which "imports" these functions from Python's math *library* (with functions identical to those available for the C programming language, and included on any computer with Python installed). Some examples show some of these functions in use, demonstrating how the Python interpreter may be used as a scientific calculator:

```
>>> from math import *
>>> sin(30.0)
-0.98803162409286183
>>> sin(radians(30.0))
0.49999999999999994
>>> pow(2.0, 5.0)
32.0
>>> log10(10000.0)
4.0
>>> e
2.7182818284590451
>>> pi
3.1415926535897931
>>> log(pow(e,6.0))
6.0
>>> asin(0.7071068)
0.78539819000368838
>>> degrees(asin(0.7071068))
45.000001524425265
>>> quit()
```

Note how trigonometric functions assume angles expressed in *radians* rather than *degrees*, and how Python provides convenient functions for translating between the two. Logarithms assume a base of e unless otherwise stated (e.g. the `log10` function for common logarithms).

The interpreted (versus compiled) nature of Python, as well as its relatively simple syntax, makes it a good choice as a person's first programming language. For complex applications, interpreted languages such as Python execute slower than compiled languages such as C++, but for the very simple examples used in these learning modules speed is not a concern.

Another Python math library is `cmath`, giving Python the ability to perform arithmetic on complex numbers. This is very useful for AC circuit analysis using *phasors*[5] as shown in the following example. Here we see Python's interpreter used as a scientific calculator to show series and parallel impedances of a resistor, capacitor, and inductor in a 60 Hz AC circuit:

```
>>> from math import *
>>> from cmath import *
>>> r = complex(400,0)
>>> f = 60.0
>>> xc = 1/(2 * pi * f * 4.7e-6)
>>> zc = complex(0,-xc)
>>> xl = 2 * pi * f * 1.0
>>> zl = complex(0,xl)
>>> r + zc + zl
(400-187.38811239154882j)
>>> 1/(1/r + 1/zc + 1/zl)
(355.837695813625+125.35793777619385j)
>>> polar(r + zc + zl)
(441.717448903332, -0.4381072059213295)
>>> abs(r + zc + zl)
441.717448903332
>>> phase(r + zc + zl)
-0.4381072059213295
>>> degrees(phase(r + zc + zl))
-25.10169387356105
```

When entering a value in rectangular form, we use the `complex()` function where the arguments are the real and imaginary quantities, respectively. If we had opted to enter the impedance values in polar form, we would have used the `rect()` function where the first argument is the magnitude and the second argument is the angle in radians. For example, we could have set the capacitor's impedance (`zc`) as $X_C \angle -90^o$ with the command `zc = rect(xc,radians(-90))` rather than with the command `zc = complex(0,-xc)` and it would have worked the same.

Note how Python defaults to rectangular form for complex quantities. Here we defined a 400 Ohm resistance as a complex value in rectangular form (400 +j0 Ω), then computed capacitive and inductive reactances at 60 Hz and defined each of those as complex (phasor) values ($0 - jX_c$ Ω and $0 + jX_l$ Ω, respectively). After that we computed total impedance in series, then total impedance in parallel. Polar-form representation was then shown for the series impedance (441.717 Ω $\angle -25.102^o$). Note the use of different functions to show the polar-form series impedance value: `polar()` takes the complex quantity and returns its polar magnitude and phase angle in *radians*; `abs()` returns just the polar magnitude; `phase()` returns just the polar angle, once again in radians. To find the polar phase angle in degrees, we nest the `degrees()` and `phase()` functions together.

The utility of Python's interpreter environment as a scientific calculator should be clear from these examples. Not only does it offer a powerful array of mathematical functions, but also unlimited

[5] A "phasor" is a voltage, current, or impedance represented as a complex number, either in rectangular or polar form.

assignment of variables as well as a convenient text record[6] of all calculations performed which may be easily copied and pasted into a text document for archival.

It is also possible to save a set of Python commands to a text file using a text editor application, and then instruct the Python interpreter to execute it at once rather than having to type it line-by-line in the interpreter's shell. For example, consider the following Python program, saved under the filename `myprogram.py`:

```
x = 200
y = -560.5

print("Sum")
print(x + y)

print("Difference")
print(x - y)

print("Product")
print(x * y)

print("Quotient")
print(x / y)
```

As with C++, the interpreter will read this source code from left-to-right, top-to-bottom, just the same as you or I would read a document written in English. Interestingly, whitespace *is* significant in the Python language (unlike C++), but this simple example program makes no use of that.

To execute this Python program, I would need to type `python myprogram.py` and then press the Enter key at my computer console's prompt, at which point it would display the following result:

```
Sum
-360.5
Difference
760.5
Product
-112100.0
Quotient
-0.35682426405
```

As you can see, syntax within the Python programming language is simpler than C++, which is one reason why it is often a preferred language for beginning programmers.

[6] Like many command-line computing environments, Python's interpreter supports "up-arrow" recall of previous entries. This allows quick recall of previously typed commands for editing and re-evaluation.

If you are interested in learning more about computer programming in *any* language, you will find a wide variety of books and free tutorials available on those subjects. Otherwise, feel free to learn by the examples presented in these modules.

6.3 Modeling signal modulation using C++

The following subsections show C++ programs modeling various types of modulation, each of them outputting a comma-separated variable (`.csv`) file which may be plotted using spreadsheet software or other mathematical visualizing software (e.g. `gnuplot`).

I happened to use `gnuplot` to generate the spectra. My gnuplot script is as follows, saved to a file named `script.txt`:

```
set datafile separator ","
set xrange [0:720]  # Sets bounds on the domain
set yrange [-1.5:3.5]  # Sets bounds on the range
set style line 1 lw 2 lc rgb "red"
set style line 2 lw 2 lc rgb "green"
plot 'data.csv' using 1:2 with lines ls 1, 'data.csv' using 1:3 with lines ls 2
```

All C++ programs were compiled using `g++` and run with text output redirected to a file named `data.csv` using the following command-line instructions:

```
g++ main.cpp ; ./a.out > data.csv
```

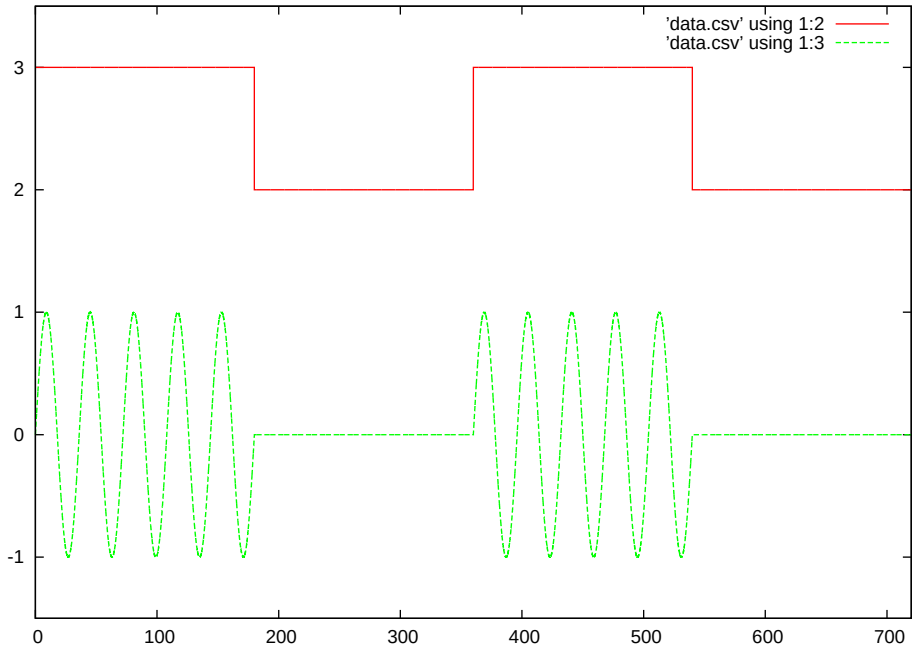
Then, after the comma-separated value file was populated with data from the C++ program's execution, I run `gnuplot` using the following command:

```
gnuplot -p script.txt
```

6.3.1 Digital amplitude-modulation

```
#include <iostream>
#include <cmath>
using namespace std;

float sinecalc (float);
float squarecalc (float);

int main (void)
{
  float x, inc;

  inc = 0.01;  // Increments for x (value of angle for baseband signal)

  for (x = 0 ; x <= 720 ; x = x + inc)
  {
    // Digital AM (ASK)
    cout << x << " , " << squarecalc(x) + 2 << " , "
      << squarecalc(x) * sinecalc(10*x) << endl;
  }

  return 0;
}

float sinecalc (float degrees)
{
  return sin (degrees * M_PI / 180);
}

float squarecalc (float degrees)
{
  if (degrees <= 180 || (degrees > 360 && degrees <= 540))
    return 1.0;

  else
    return 0.0;
}
```

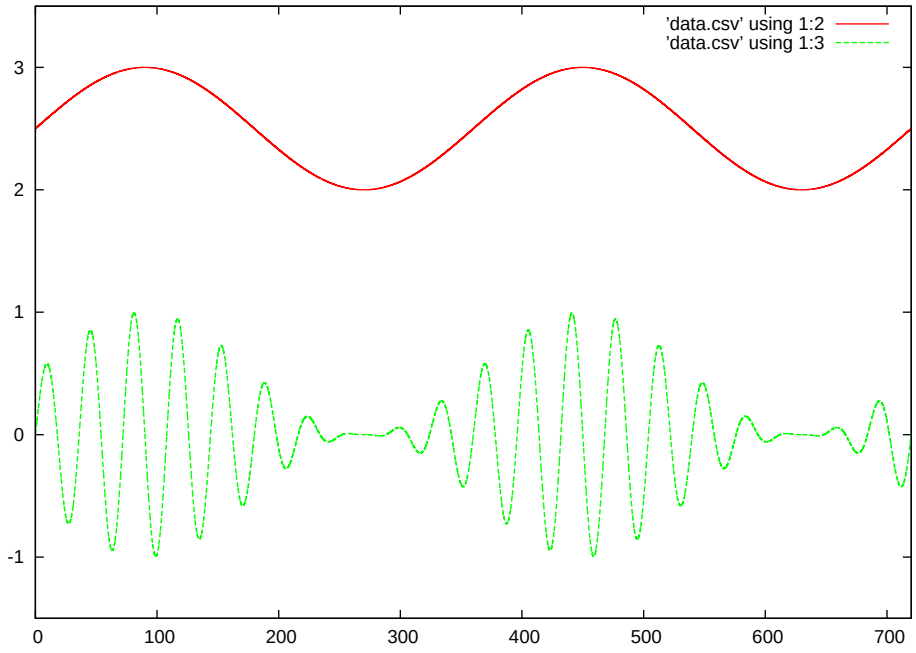
Upper waveform is the *baseband* (modulating) signal, lower waveform is the modulated *carrier*:



6.3.2 Analog amplitude-modulation

```
#include <iostream>
#include <cmath>
using namespace std;

float sinecalc (float);

int main (void)
{
  float x, inc;

  inc = 0.01;  // Increments for x (value of angle for baseband signal)

  for (x = 0 ; x <= 720 ; x = x + inc)
  {
    // Analog AM
    cout << x << " , " << 0.5*sinecalc(x) + 2.5 << " , "
      << (sinecalc(x)+ 1) * sinecalc(10*x) / 2 << endl;
  }

  return 0;
}

float sinecalc (float degrees)
{
  return sin (degrees * M_PI / 180);
}
```

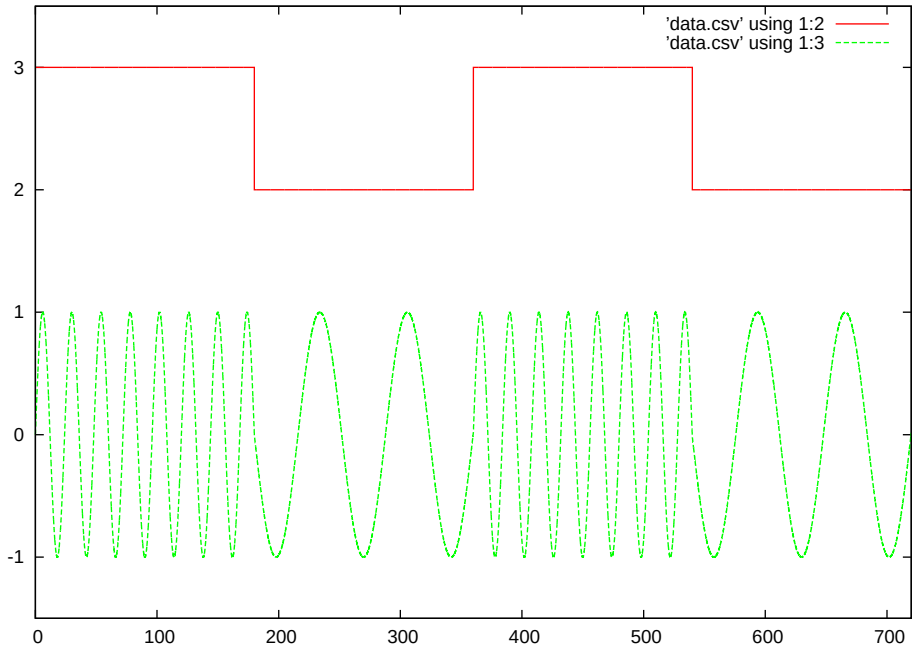
Upper waveform is the *baseband* (modulating) signal, lower waveform is the modulated *carrier*:



6.3.3 Digital frequency-modulation

```
#include <iostream>
#include <cmath>
using namespace std;

float sinecalc (float);
float squarecalc (float);

int main (void)
{
  float x, c, inc;

  inc = 0.01;  // Increments for x (value of angle for baseband signal)
  c = 0.0;  // Initial value for c (value of angle for carrier)

  for (x = 0 ; x <= 720 ; x = x + inc)
  {
    // Digital FM (FSK)
    cout << x << " , " << squarecalc(x) + 2 << " , " << sinecalc(10*c) << endl;
    c = c + ((squarecalc(x)+0.5) * inc);
  }

  return 0;
}

float sinecalc (float degrees)
{
  return sin (degrees * M_PI / 180);
}

float squarecalc (float degrees)
{
  if (degrees <= 180 || (degrees > 360 && degrees <= 540))
    return 1.0;

  else
    return 0.0;
}
```

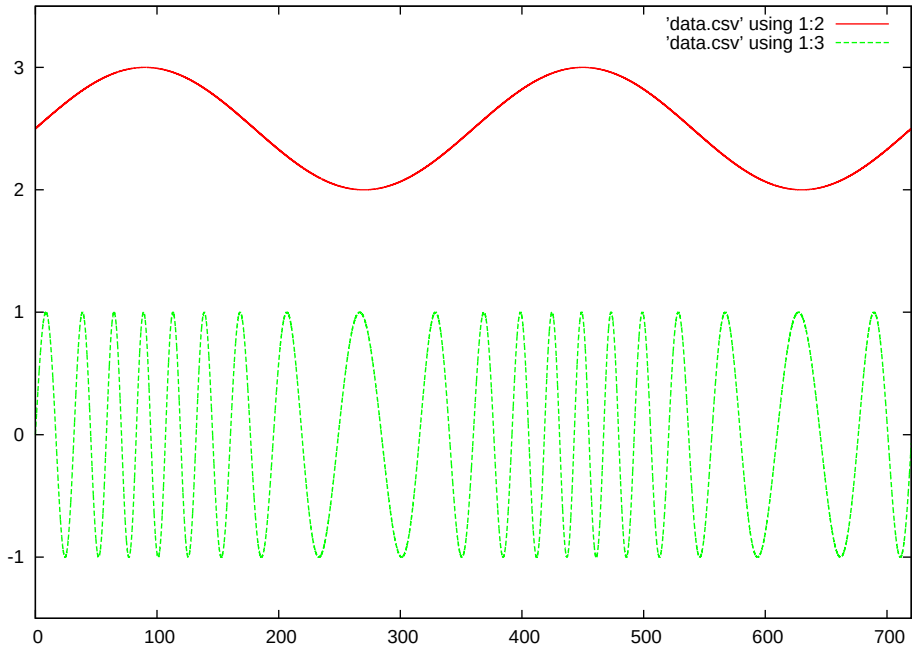
Upper waveform is the *baseband* (modulating) signal, lower waveform is the modulated *carrier*:



6.3.4 Analog frequency-modulation

```
#include <iostream>
#include <cmath>
using namespace std;

float sinecalc (float);

int main (void)
{
  float x, c, inc;

  inc = 0.01;  // Increments for x (value of angle for baseband signal)
  c = 0.0;  // Initial value for c (value of angle for carrier)

  for (x = 0 ; x <= 720 ; x = x + inc)
  {
    // Analog FM
    cout << x << " , " << 0.5*sinecalc(x) + 2.5 << " , "
      << sinecalc(10*c) << endl;
    c = c + ((0.5*sinecalc(x) + 1.0) * inc);
  }

  return 0;
}

float sinecalc (float degrees)
{
  return sin (degrees * M_PI / 180);
}
```

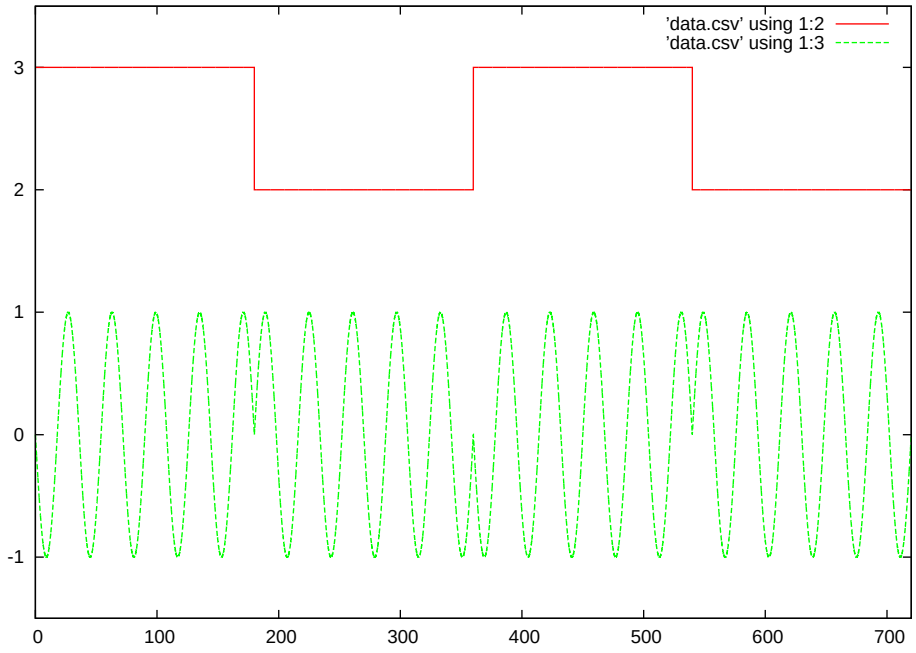
Upper waveform is the *baseband* (modulating) signal, lower waveform is the modulated *carrier*:



6.3.5 Digital phase-modulation

```
#include <iostream>
#include <cmath>
using namespace std;

float sinecalc (float);
float squarecalc (float);

int main (void)
{
  float x, inc;

  inc = 0.01;  // Increments for x (value of angle for baseband signal)

  for (x = 0 ; x <= 720 ; x = x + inc)
  {
    // Digital PM (PSK)
    cout << x << " , " << squarecalc(x) + 2 << " , "
      << sinecalc((10*x) + (180*squarecalc(x))) << endl;
  }

  return 0;
}

float sinecalc (float degrees)
{
  return sin (degrees * M_PI / 180);
}

float squarecalc (float degrees)
{
  if (degrees <= 180 || (degrees > 360 && degrees <= 540))
    return 1.0;

  else
    return 0.0;
}
```

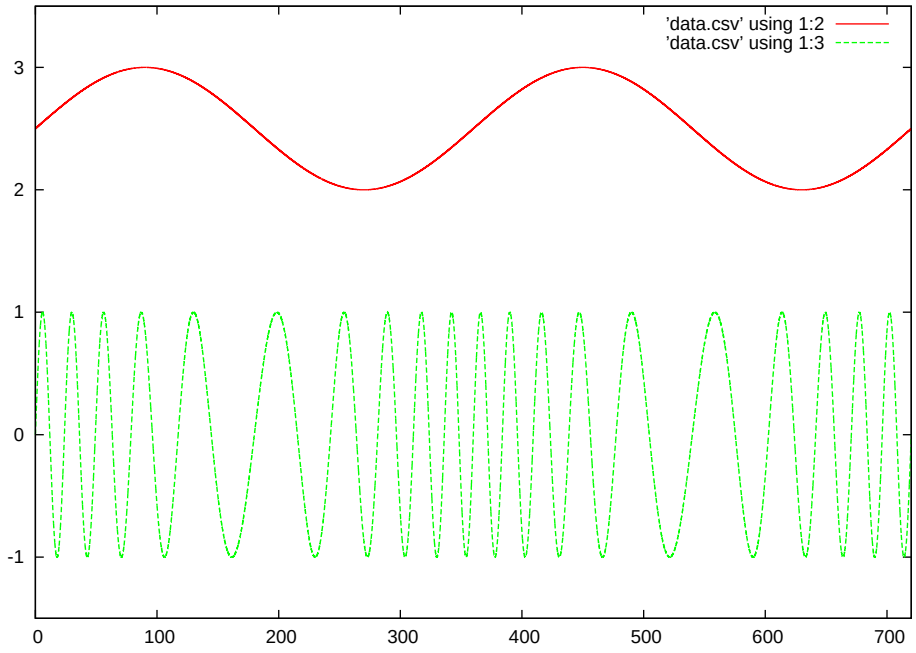
Upper waveform is the *baseband* (modulating) signal, lower waveform is the modulated *carrier*:



6.3.6 Analog phase-modulation

```
#include <iostream>
#include <cmath>
using namespace std;

float sinecalc (float);

int main (void)
{
  float x, inc;

  inc = 0.01;  // Increments for x (value of angle for baseband signal)

  for (x = 0 ; x <= 720 ; x = x + inc)
  {
    // Analog PM
    cout << x << " , " << 0.5*sinecalc(x) + 2.5 << " , "
      << sinecalc((10*x) + (300*sinecalc(x))) << endl;
  }

  return 0;
}

float sinecalc (float degrees)
{
  return sin (degrees * M_PI / 180);
}
```

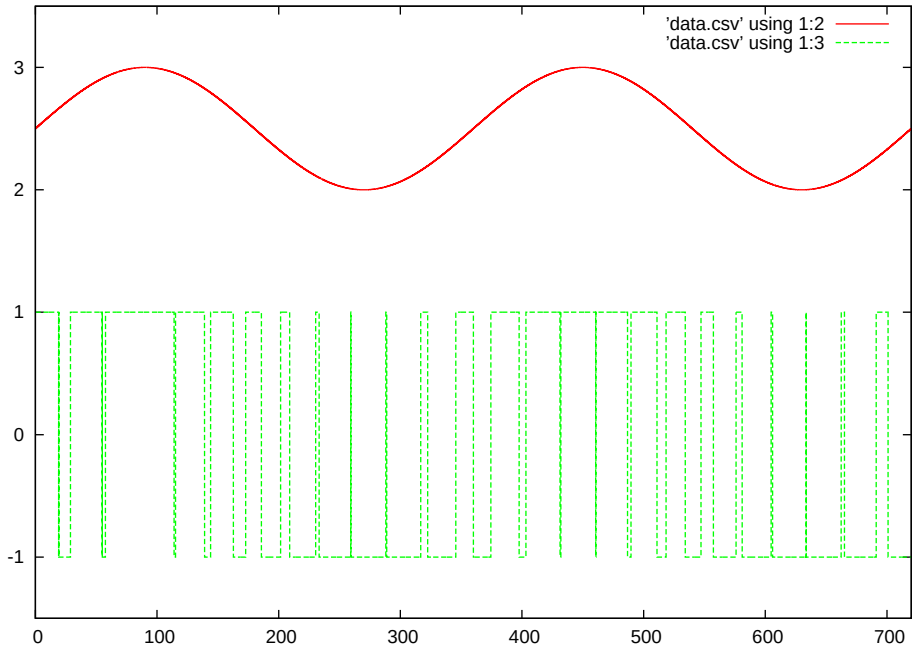
Upper waveform is the *baseband* (modulating) signal, lower waveform is the modulated *carrier*:



6.3.7 Pulse-width modulation

```
#include <iostream>
#include <cmath>
using namespace std;

float sinecalc (float);
float pwmcalc (float,float);

int main (void)
{
  float x, inc;

  inc = 0.01;  // Increments for x (value of angle for baseband signal)

  for (x = 0 ; x <= 720 ; x = x + inc)
  {
    // Pulse-width modulation (PWM)
    cout << x << " , " << 0.5*sinecalc(x) + 2.5 << " , "
      << pwmcalc(0.5*sinecalc(x) + 0.5, 25*x) << endl;
  }

  return 0;
}

float sinecalc (float degrees)
{
  return sin (degrees * M_PI / 180);
}

float pwmcalc (float value, float degrees)
{
  if ((value * 720) > ((int)(degrees) % 720))
    return 1.0;

  else
    return -1.0;
}
```

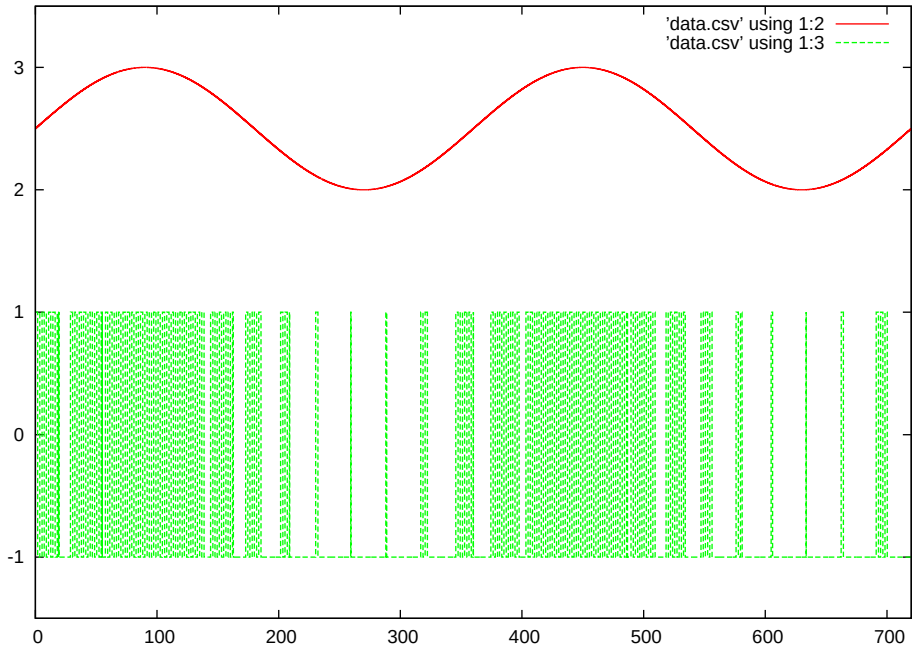
Upper waveform is the *baseband* (modulating) signal, lower waveform is the modulated *carrier*:



6.3.8 Pulse-density modulation

```
#include <iostream>
#include <cmath>
using namespace std;

float sinecalc (float);
float pdmcalc (float,float);

int main (void)
{
  float x, inc;

  inc = 0.01;  // Increments for x (value of angle for baseband signal)

  for (x = 0 ; x <= 720 ; x = x + inc)
  {
    // Pulse-density modulation (PDM)
    cout << x << " , " << 0.5*sinecalc(x) + 2.5 << " , "
      << pdmcalc(0.5*sinecalc(x) + 0.5, 25*x) << endl;
  }

  return 0;
}

float sinecalc (float degrees)
{
  return sin (degrees * M_PI / 180);
}

float pdmcalc (float value, float degrees)
{
  if (((value * 720) > ((int)(degrees) % 720)) && ((int)(degrees) % 90 < 45))
    return 1.0;

  else
    return -1.0;
}
```

Upper waveform is the *baseband* (modulating) signal, lower waveform is the modulated *carrier*:



Chapter 7

Questions

This learning module, along with all others in the ModEL collection, is designed to be used in an inverted instructional environment where students independently read[1] the tutorials and attempt to answer questions on their own *prior* to the instructor's interaction with them. In place of lecture[2], the instructor engages with students in Socratic-style dialogue, probing and challenging their understanding of the subject matter through inquiry.

Answers are not provided for questions within this chapter, and this is by design. Solved problems may be found in the Tutorial and Derivation chapters, instead. The goal here is *independence*, and this requires students to be challenged in ways where others cannot think for them. Remember that you always have the tools of *experimentation* and *computer simulation* (e.g. SPICE) to explore concepts!

The following lists contain ideas for Socratic-style questions and challenges. Upon inspection, one will notice a strong theme of *metacognition* within these statements: they are designed to foster a regular habit of examining one's own thoughts as a means toward clearer thinking. As such these sample questions are useful both for instructor-led discussions as well as for self-study.

[1] Technical reading is an essential academic skill for any technical practitioner to possess for the simple reason that the most comprehensive, accurate, and useful information to be found for developing technical competence is in textual form. Technical careers in general are characterized by the need for continuous learning to remain current with standards and technology, and therefore any technical practitioner who cannot read well is handicapped in their professional development. An excellent resource for educators on improving students' reading prowess through intentional effort and strategy is the book textitReading For Understanding – How Reading Apprenticeship Improves Disciplinary Learning in Secondary and College Classrooms by Ruth Schoenbach, Cynthia Greenleaf, and Lynn Murphy.

[2] Lecture is popular as a teaching method because it is easy to implement: any reasonably articulate subject matter expert can talk to students, even with little preparation. However, it is also quite problematic. A good lecture always makes complicated concepts seem easier than they are, which is bad for students because it instills a false sense of confidence in their own understanding; reading and re-articulation requires more cognitive effort and serves to verify comprehension. A culture of teaching-by-lecture fosters a debilitating dependence upon direct personal instruction, whereas the challenges of modern life demand independent and critical thought made possible only by gathering information and perspectives from afar. Information presented in a lecture is ephemeral, easily lost to failures of memory and dictation; text is forever, and may be referenced at any time.

General challenges following tutorial reading

- Summarize as much of the text as you can in one paragraph of your own words. A helpful strategy is to explain ideas as you would for an intelligent child: as simple as you can without compromising too much accuracy.
- Simplify a particular section of the text, for example a paragraph or even a single sentence, so as to capture the same fundamental idea in fewer words.
- Where did the text make the most sense to you? What was it about the text's presentation that made it clear?
- Identify where it might be easy for someone to misunderstand the text, and explain why you think it could be confusing.
- Identify any new concept(s) presented in the text, and explain in your own words.
- Identify any familiar concept(s) such as physical laws or principles applied or referenced in the text.
- Devise a proof of concept experiment demonstrating an important principle, physical law, or technical innovation represented in the text.
- Devise an experiment to disprove a plausible misconception.
- Did the text reveal any misconceptions you might have harbored? If so, describe the misconception(s) and the reason(s) why you now know them to be incorrect.
- Describe any useful problem-solving strategies applied in the text.
- Devise a question of your own to challenge a reader's comprehension of the text.

General follow-up challenges for assigned problems

- Identify where any fundamental laws or principles apply to the solution of this problem, especially before applying any mathematical techniques.
- Devise a thought experiment to explore the characteristics of the problem scenario, applying known laws and principles to mentally model its behavior.
- Describe in detail your own strategy for solving this problem. How did you identify and organized the given information? Did you sketch any diagrams to help frame the problem?
- Is there more than one way to solve this problem? Which method seems best to you?
- Show the work you did in solving this problem, even if the solution is incomplete or incorrect.
- What would you say was the most challenging part of this problem, and why was it so?
- Was any important information missing from the problem which you had to research or recall?
- Was there any extraneous information presented within this problem? If so, what was it and why did it not matter?
- Examine someone else's solution to identify where they applied fundamental laws or principles.
- Simplify the problem from its given form and show how to solve this simpler version of it. Examples include eliminating certain variables or conditions, altering values to simpler (usually whole) numbers, applying a limiting case (i.e. altering a variable to some extreme or ultimate value).
- For quantitative problems, identify the real-world meaning of all intermediate calculations: their units of measurement, where they fit into the scenario at hand. Annotate any diagrams or illustrations with these calculated values.
- For quantitative problems, try approaching it qualitatively instead, thinking in terms of "increase" and "decrease" rather than definite values.
- For qualitative problems, try approaching it quantitatively instead, proposing simple numerical values for the variables.
- Were there any assumptions you made while solving this problem? Would your solution change if one of those assumptions were altered?
- Identify where it would be easy for someone to go astray in attempting to solve this problem.
- Formulate your own problem based on what you learned solving this one.

General follow-up challenges for experiments or projects

- In what way(s) was this experiment or project easy to complete?
- Identify some of the challenges you faced in completing this experiment or project.

- Show how thorough documentation assisted in the completion of this experiment or project.
- Which fundamental laws or principles are key to this system's function?
- Identify any way(s) in which one might obtain false or otherwise misleading measurements from test equipment in this system.
- What will happen if (component X) fails (open/shorted/etc.)?
- What would have to occur to make this system unsafe?

7.1 Conceptual reasoning

These questions are designed to stimulate your analytic and synthetic thinking[3]. In a Socratic discussion with your instructor, the goal is for these questions to prompt an extended dialogue where assumptions are revealed, conclusions are tested, and understanding is sharpened. Your instructor may also pose additional questions based on those assigned, in order to further probe and refine your conceptual understanding.

Questions that follow are presented to challenge and probe your understanding of various concepts presented in the tutorial. These questions are intended to serve as a guide for the Socratic dialogue between yourself and the instructor. Your instructor's task is to ensure you have a sound grasp of these concepts, and the questions contained in this document are merely a means to this end. Your instructor may, at his or her discretion, alter or substitute questions for the benefit of tailoring the discussion to each student's needs. The only absolute requirement is that each student is challenged and assessed at a level equal to or greater than that represented by the documented questions.

It is far more important that you convey your *reasoning* than it is to simply convey a correct answer. For this reason, you should refrain from researching other information sources to answer questions. What matters here is that *you* are doing the thinking. If the answer is incorrect, your instructor will work with you to correct it through proper reasoning. A correct answer without an adequate explanation of how you derived that answer is unacceptable, as it does not aid the learning or assessment process.

You will note a conspicuous lack of answers given for these conceptual questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your conceptual answers, where applicable, is to use circuit simulation software to explore the effects of changes made to circuits. For example, if one of these conceptual questions challenges you to predict the effects of altering some component parameter in a circuit, you may check the validity of your work by simulating that same parameter change within software and seeing if the results agree.

[3] *Analytical* thinking involves the "disassembly" of an idea into its constituent parts, analogous to dissection. *Synthetic* thinking involves the "assembly" of a new idea comprised of multiple concepts, analogous to construction. Both activities are high-level cognitive skills, extremely important for effective problem-solving, necessitating frequent challenge and regular practice to fully develop.

7.1.1 Reading outline and reflections

"Reading maketh a full man; conference a ready man; and writing an exact man" – Francis Bacon

Francis Bacon's advice is a blueprint for effective education: reading provides the learner with knowledge, writing focuses the learner's thoughts, and critical dialogue equips the learner to confidently communicate and apply their learning. Independent acquisition and application of knowledge is a powerful skill, well worth the effort to cultivate. To this end, students should read these educational resources closely, journal their own reflections on the reading, and discuss in detail their findings with classmates and instructor(s). You should be able to do all of the following after reading any instructional text:

☑ Briefly SUMMARIZE THE TEXT in the form of a journal entry documenting your learning as you progress through the course of study. Share this summary in dialogue with your classmates and instructor. Journaling is an excellent self-test of thorough reading because you cannot clearly express what you have not read or did not comprehend.

☑ Demonstrate ACTIVE READING STRATEGIES, including verbalizing your impressions as you read, simplifying long passages to convey the same ideas using fewer words, annotating text and illustrations with your own interpretations, working through mathematical examples shown in the text, cross-referencing passages with relevant illustrations and/or other passages, identifying problem-solving strategies applied by the author, etc. Technical reading is a special case of problem-solving, and so these strategies work precisely because they help solve any problem: paying attention to your own thoughts (metacognition), eliminating unnecessary complexities, identifying what makes sense, paying close attention to details, drawing connections between separated facts, and noting the successful strategies of others.

☑ Identify IMPORTANT THEMES, especially GENERAL LAWS and PRINCIPLES, expounded in the text and express them in the simplest of terms as though you were teaching an intelligent child. This emphasizes connections between related topics and develops your ability to communicate complex ideas to anyone.

☑ Form YOUR OWN QUESTIONS based on the reading, and then pose them to your instructor and classmates for their consideration. Anticipate both correct and incorrect answers, the incorrect answer(s) assuming one or more plausible misconceptions. This helps you view the subject from different perspectives to grasp it more fully.

☑ Devise EXPERIMENTS to test claims presented in the reading, or to disprove misconceptions. Predict possible outcomes of these experiments, and evaluate their meanings: what result(s) would confirm, and what would constitute disproof? Running mental simulations and evaluating results is essential to scientific and diagnostic reasoning.

☑ Specifically identify any points you found CONFUSING. The reason for doing this is to help diagnose misconceptions and overcome barriers to learning.

7.1.2 Foundational concepts

Correct analysis and diagnosis of electric circuits begins with a proper understanding of some basic concepts. The following is a list of some important concepts referenced in this module's full tutorial. Define each of them in your own words, and be prepared to illustrate each of these concepts with a description of a practical example and/or a live demonstration.

Modulation

Demodulation

Baseband

Carrier

Analog signal

Digital signal

Multiplexing

Electromagnetic wave

Fundamental frequency

Harmonic frequency

Crosstalk

Sinusoidal decomposition (i.e. Fourier's Theorem)

Mixing

Heterodyning

Local oscillator

Up/downconversion

Sideband

Filter

Time domain

Frequency domain

Bandwidth

Amplification

Radio frequency (RF)

Intermediate frequency (IF)

Quadrature

Phasor

Phase angle

7.1.3 Carrier-less radio

Would it be possible, at least in principle, to build a radio transmitter with an extremely long antenna that could directly broadcast audio-frequency (rather than radio-frequency) signals, and to build receivers capable of intercepting those radio broadcasts?

Challenges

- Suppose multiple broadcast stations were built in this manner, each one broadcasting different sounds. How well could these stations be differentiated by listeners at the receiving ends?
- Draw a diagram of such a radio transmitter, assuming a median tone frequency of 3 kHz.

7.1.4 Reginald Fessenden's invention

Read the account written by John Hogan in 1922 about Reginald Fessenden's invention of heterodyne radio reception (refer to the Historical References chapter of this module, the section entitled "Heterodyne radio reception") and answer the following questions:

- Explain how the author describes the production and reception of radio waves.
- Explain how a simple ammeter may be used as a Morse code indicator.
- What sort of modulation is used for reception of Morse code by telephone receiver (i.e. loudspeaker)?
- Identify disadvantages of the "chopper" strategy, and how the heterodyne strategy improves upon it.
- Explain how the received RF signal is demodulated in Fessenden's design.

Challenges

- Heterodyning produces both a *difference* frequency and a *sum* frequency, but only the difference frequency is considered in this presentation. Explain why we are able to ignore the sum frequency in Fessenden's design.
- The author states "Axis D shows the same pulsating radio frequency voltage in effect completely rectified; this is the process performed by the electrostatic telephone, for, although there is in it no electrical rectification or biasing, the device nevertheless produces a unidirectional mechanical force regardless of the polarity of the applied potential. Thus on axis D we have a graph of the mechanical force applied to the diaphragm of the condenser telephone, as it acts in its capacity of a perfect electromechanical rectifier." Explain why such a "telephone" (speaker) essentially rectifies the AC signal impressed upon it.

7.1.5 Bat sonar detector

Bats emit ultrasonic (high audio frequency) tones to communicate with each other and also to "echo-locate" flying prey in the dark. These tones are too high in frequency for humans to directly hear.

Sketch a general diagram for a device that might *downconvert* bat tones into lower-frequency tones that human beings could hear.

Challenges

- An alternative strategy for making bat tones audible is to record them digitally, and then have the computer play back those recordings at a fraction of the recording speed. Explain how this technique works.
- Identify common bat vocalization frequencies as well as the frequency range for typical human hearing, and explain how these values would influence your detector's design.

7.2 Quantitative reasoning

These questions are designed to stimulate your computational thinking. In a Socratic discussion with your instructor, the goal is for these questions to reveal your mathematical approach(es) to problem-solving so that good technique and sound reasoning may be reinforced. Your instructor may also pose additional questions based on those assigned, in order to observe your problem-solving firsthand.

Mental arithmetic and estimations are strongly encouraged for all calculations, because without these abilities you will be unable to readily detect errors caused by calculator misuse (e.g. keystroke errors).

You will note a conspicuous lack of answers given for these quantitative questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. My advice is to use circuit simulation software such as SPICE to check the correctness of quantitative answers. Refer to those learning modules within this collection focusing on SPICE to see worked examples which you may use directly as practice problems for your own study, and/or as templates you may modify to run your own analyses and generate your own practice problems.

Completely worked example problems found in the Tutorial may also serve as "test cases[4]" for gaining proficiency in the use of circuit simulation software, and then once that proficiency is gained you will never need to rely[5] on an answer key!

[4] In other words, set up the circuit simulation software to analyze the same circuit examples found in the Tutorial. If the simulated results match the answers shown in the Tutorial, it confirms the simulation has properly run. If the simulated results disagree with the Tutorial's answers, something has been set up incorrectly in the simulation software. Using every Tutorial as practice in this way will quickly develop proficiency in the use of circuit simulation software.

[5] This approach is perfectly in keeping with the instructional philosophy of these learning modules: *teaching students to be self-sufficient thinkers*. Answer keys can be useful, but it is even more useful to your long-term success to have a set of tools on hand for checking your own work, because once you have left school and are on your own, there will no longer be "answer keys" available for the problems you will have to solve.

7.2.1 Miscellaneous physical constants

Note: constants shown in **bold** type are *exact*, not approximations. Values inside of parentheses show one standard deviation (σ) of uncertainty in the final digits: for example, the magnetic permeability of free space value given as $1.25663706212(19) \times 10^{-6}$ H/m represents a center value (i.e. the location parameter) of $1.25663706212 \times 10^{-6}$ Henrys per meter with one standard deviation of uncertainty equal to $0.0000000000019 \times 10^{-6}$ Henrys per meter.

Avogadro's number (N_A) = **6.02214076** $\times 10^{23}$ **per mole** (mol^{-1})

Boltzmann's constant (k) = **1.380649** $\times 10^{-23}$ **Joules per Kelvin** (J/K)

Electronic charge (e) = **1.602176634** $\times 10^{-19}$ **Coulomb** (C)

Faraday constant (F) = **96,485.33212...** $\times 10^{4}$ **Coulombs per mole** (C/mol)

Magnetic permeability of free space (μ_0) = $1.25663706212(19) \times 10^{-6}$ Henrys per meter (H/m)

Electric permittivity of free space (ϵ_0) = $8.8541878128(13) \times 10^{-12}$ Farads per meter (F/m)

Characteristic impedance of free space (Z_0) = 376.730313668(57) Ohms (Ω)

Gravitational constant (G) = $6.67430(15) \times 10^{-11}$ cubic meters per kilogram-seconds squared ($m^3/kg\text{-}s^2$)

Molar gas constant (R) = **8.314462618... Joules per mole-Kelvin** (J/mol-K) = 0.08205746(14) liters-atmospheres per mole-Kelvin

Planck constant (h) = **6.62607015** $\times 10^{-34}$ **joule-seconds** (J-s)

Stefan-Boltzmann constant (σ) = **5.670374419...** $\times 10^{-8}$ **Watts per square meter-Kelvin**^{4} ($W/m^2 \cdot K^4$)

Speed of light in a vacuum (c) = **299,792,458 meters per second** (m/s) = 186282.4 miles per second (mi/s)

Note: All constants taken from NIST data "Fundamental Physical Constants – Complete Listing", from `http://physics.nist.gov/constants`, National Institute of Standards and Technology (NIST), 2018 CODATA Adjustment.

7.2.2 Introduction to spreadsheets

A powerful computational tool you are encouraged to use in your work is a *spreadsheet*. Available on most personal computers (e.g. Microsoft Excel), *spreadsheet* software performs numerical calculations based on number values and formulae entered into cells of a grid. This grid is typically arranged as lettered columns and numbered rows, with each cell of the grid identified by its column/row coordinates (e.g. cell `B3`, cell `A8`). Each cell may contain a string of text, a number value, or a mathematical formula. The spreadsheet automatically updates the results of all mathematical formulae whenever the entered number values are changed. This means it is possible to set up a spreadsheet to perform a series of calculations on entered data, and those calculations will be re-done by the computer any time the data points are edited in any way.

For example, the following spreadsheet calculates average speed based on entered values of distance traveled and time elapsed:

	A	B	C	D
1	Distance traveled	46.9	Kilometers	
2	Time elapsed	1.18	Hours	
3	Average speed	= B1 / B2	km/h	
4				
5				

Text labels contained in cells `A1` through `A3` and cells `C1` through `C3` exist solely for readability and are not involved in any calculations. Cell `B1` contains a sample distance value while cell `B2` contains a sample time value. The formula for computing speed is contained in cell `B3`. Note how this formula begins with an "equals" symbol (`=`), references the values for distance and speed by lettered column and numbered row coordinates (`B1` and `B2`), and uses a forward slash symbol for division (`/`). The coordinates `B1` and `B2` function as *variables*[6] would in an algebraic formula.

When this spreadsheet is executed, the numerical value `39.74576` will appear in cell `B3` rather than the formula `= B1 / B2`, because 39.74576 is the computed speed value given 46.9 kilometers traveled over a period of 1.18 hours. If a different numerical value for distance is entered into cell `B1` or a different value for time is entered into cell `B2`, cell `B3`'s value will automatically update. All you need to do is set up the given values and any formulae into the spreadsheet, and the computer will do all the calculations for you.

Cell `B3` may be referenced by other formulae in the spreadsheet if desired, since it is a variable just like the given values contained in `B1` and `B2`. This means it is possible to set up an entire chain of calculations, one dependent on the result of another, in order to arrive at a final value. The arrangement of the given data and formulae need not follow any pattern on the grid, which means you may place them anywhere.

[6] Spreadsheets may also provide means to attach text labels to cells for use as variable names (Microsoft Excel simply calls these labels "names"), but for simple spreadsheets such as those shown here it's usually easier just to use the standard coordinate naming for each cell.

Common[7] arithmetic operations available for your use in a spreadsheet include the following:

- Addition (`+`)
- Subtraction (`-`)
- Multiplication (`*`)
- Division (`/`)
- Powers (`^`)
- Square roots (`sqrt()`)
- Logarithms (`ln()` , `log10()`)

Parentheses may be used to ensure[8] proper order of operations within a complex formula. Consider this example of a spreadsheet implementing the *quadratic formula*, used to solve for roots of a polynomial expression in the form of $ax^2 + bx + c$:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

	A	B
1	x_1	= (-B4 + sqrt((B4^2) - (4*B3*B5))) / (2*B3)
2	x_2	= (-B4 - sqrt((B4^2) - (4*B3*B5))) / (2*B3)
3	a =	9
4	b =	5
5	c =	-2

This example is configured to compute roots[9] of the polynomial $9x^2 + 5x - 2$ because the values of 9, 5, and -2 have been inserted into cells `B3`, `B4`, and `B5`, respectively. Once this spreadsheet has been built, though, it may be used to calculate the roots of *any* second-degree polynomial expression simply by entering the new a, b, and c coefficients into cells `B3` through `B5`. The numerical values appearing in cells `B1` and `B2` will be automatically updated by the computer immediately following any changes made to the coefficients.

[7] Modern spreadsheet software offers a bewildering array of mathematical functions you may use in your computations. I recommend you consult the documentation for your particular spreadsheet for information on operations other than those listed here.

[8] Spreadsheet programs, like text-based programming languages, are designed to follow standard order of operations by default. However, my personal preference is to use parentheses even where strictly unnecessary just to make it clear to any other person viewing the formula what the intended order of operations is.

[9] Reviewing some algebra here, a *root* is a value for x that yields an overall value of zero for the polynomial. For this polynomial ($9x^2 + 5x - 2$) the two roots happen to be $x = 0.269381$ and $x = -0.82494$, with these values displayed in cells `B1` and `B2`, respectively upon execution of the spreadsheet.

Alternatively, one could break up the long quadratic formula into smaller pieces like this:

$$y = \sqrt{b^2 - 4ac} \qquad z = 2a$$

$$x = \frac{-b \pm y}{z}$$

	A	B	C
1	x_1	= (-B4 + C1) / C2	= sqrt((B4^2) - (4*B3*B5))
2	x_2	= (-B4 - C1) / C2	= 2*B3
3	a =	9	
4	b =	5	
5	c =	-2	

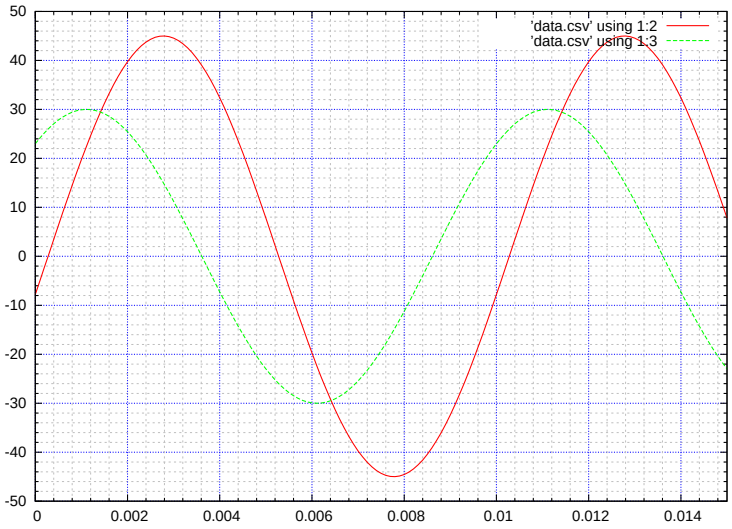
Note how the square-root term (y) is calculated in cell `C1`, and the denominator term (z) in cell `C2`. This makes the two final formulae (in cells `B1` and `B2`) simpler to interpret. The positioning of all these cells on the grid is completely arbitrary[10] – all that matters is that they properly reference each other in the formulae.

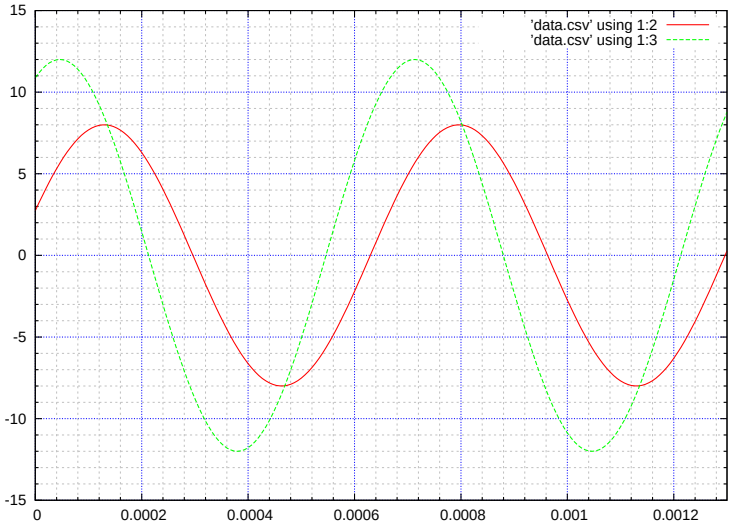
Spreadsheets are particularly useful for situations where the same set of calculations representing a circuit or other system must be repeated for different initial conditions. The power of a spreadsheet is that it automates what would otherwise be a tedious set of calculations. One specific application of this is to simulate the effects of various components within a circuit failing with abnormal values (e.g. a shorted resistor simulated by making its value nearly zero; an open resistor simulated by making its value extremely large). Another application is analyzing the behavior of a circuit design given new components that are out of specification, and/or aging components experiencing drift over time.

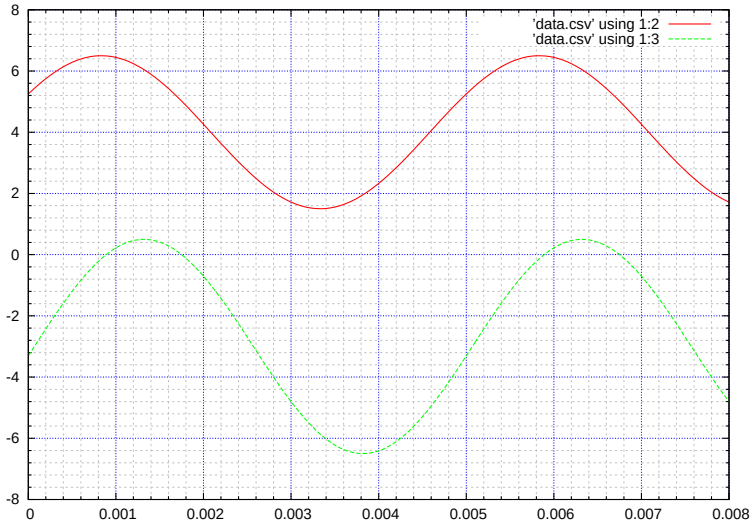
[10] My personal preference is to locate all the "given" data in the upper-left cells of the spreadsheet grid (each data point flanked by a sensible name in the cell to the left and units of measurement in the cell to the right as illustrated in the first distance/time spreadsheet example), sometimes coloring them in order to clearly distinguish which cells contain entered data versus which cells contain computed results from formulae. I like to place all formulae in cells below the given data, and try to arrange them in logical order so that anyone examining my spreadsheet will be able to figure out *how* I constructed a solution. This is a general principle I believe all computer programmers should follow: *document and arrange your code to make it easy for other people to learn from it.*

7.2.3 Oscillographs comparing sinusoids

Examine the following oscillographs, and from them determine the amplitudes of and phase shift separating the two sinusoids, as well as their frequency and period. In each case the vertical axis has a unit of *Volts* and the horizontal axis a unit of *seconds*:





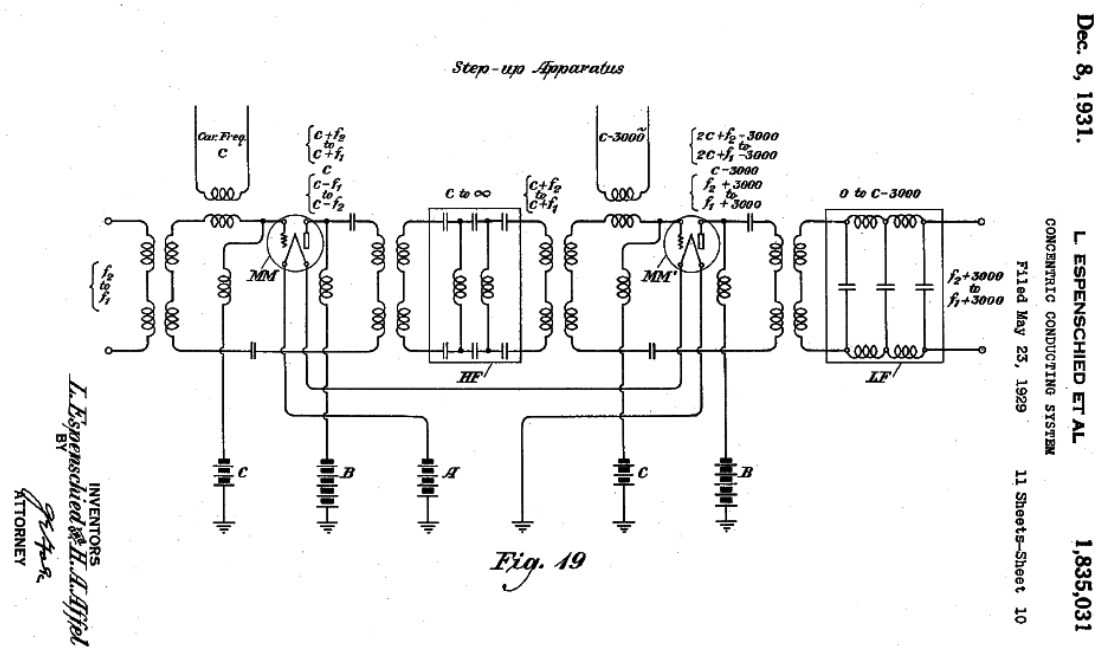


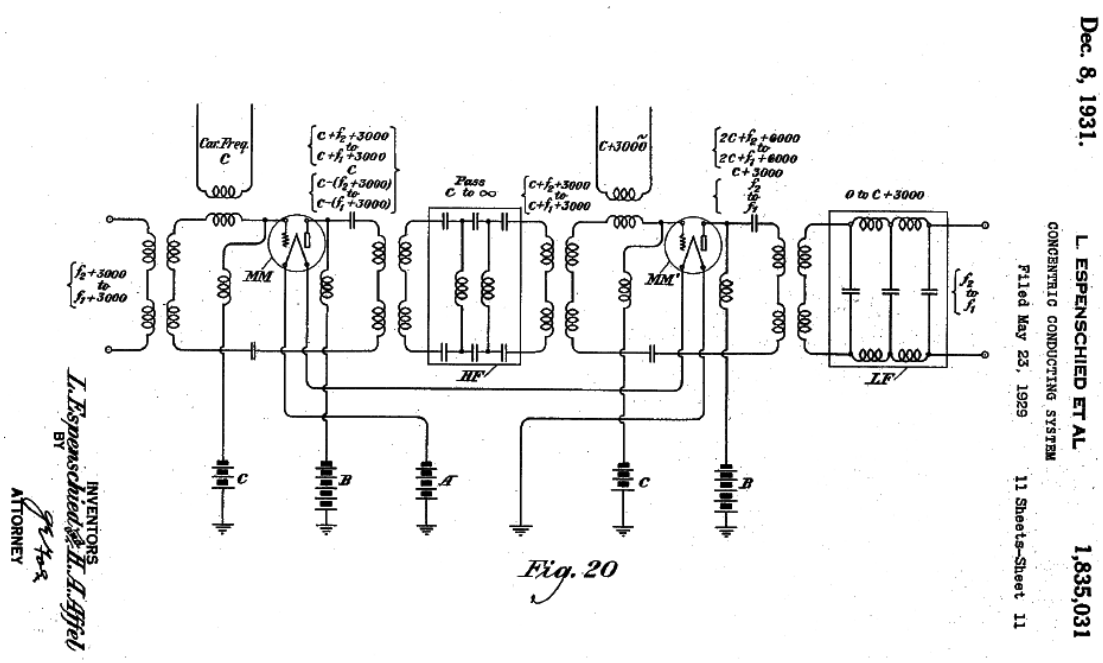
Challenges

- How can we tell both waveforms in each oscillograph have the same frequency, and why might this fact be important to us?

7.2.4 Telephony up/down converter circuits

Lloyd Espenschied and Herman Affel filed for a US patent in 1929 regarding a new type of high-frequency signal cable capable of conveying communication signals in the megaHertz range (*US Patent 1,835,031*, "Concentric Conducting System", application 23 May 1929, patent granted 8 December 1931). In this patent, they describe in detail a practical application for their coaxial ("concentric") cable involving the multiplexing of multiple telephone signals along a single cable. This multiplexing utilized frequency-shifting of the audio signals by means of vacuum tube mixer circuits such as those shown below in figures 19 and 20:





The range of frequencies for the baseband audio signal are represented in these diagrams as "f_1 to f_2". The carrier frequency is simply c.

Examine these two diagrams and explain the following:

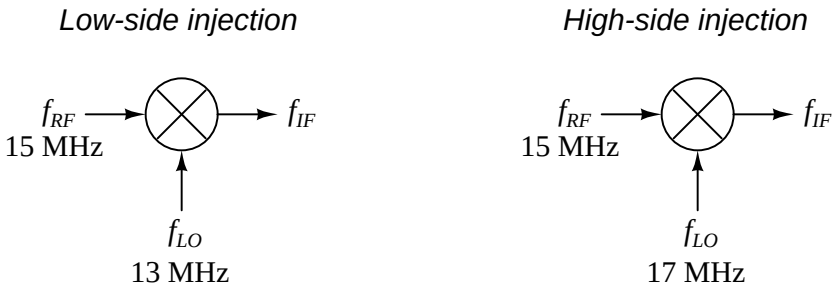
- What types of filter networks are used in these circuits, and what functions do they provide?
- How are the audio baseband frequencies "shifted" up or down at each stage of each circuit?
- How is it possible that the same amplifier circuit design is able to either up-convert or down-convert the signal frequencies.
- Assuming a pure sinusoidal tone of 2 kHz sung or played into the telephone receiver (microphone), calculate the signal frequencies at each stage of the up-converter and the down-converter.

Challenges

- Identify the functions of the DC power sources A, B, and C.

7.2.5 High-side versus low-side injection

The following diagrams show two different strategies for down-conversion of a 15 MHz RF signal into a 2 MHz IF signal:



Explain how each of them functions to achieve the same goal, and why one might be better than the other.

Challenges

- Is a -2 MHz signal also produced? Why or why not?
- Does the wave-shape of the local oscillator matter? Why or why not?

7.3 Diagnostic reasoning

These questions are designed to stimulate your deductive and inductive thinking, where you must apply general principles to specific scenarios (deductive) and also derive conclusions about the failed circuit from specific details (inductive). In a Socratic discussion with your instructor, the goal is for these questions to reinforce your recall and use of general circuit principles and also challenge your ability to integrate multiple symptoms into a sensible explanation of what's wrong in a circuit. Your instructor may also pose additional questions based on those assigned, in order to further challenge and sharpen your diagnostic abilities.

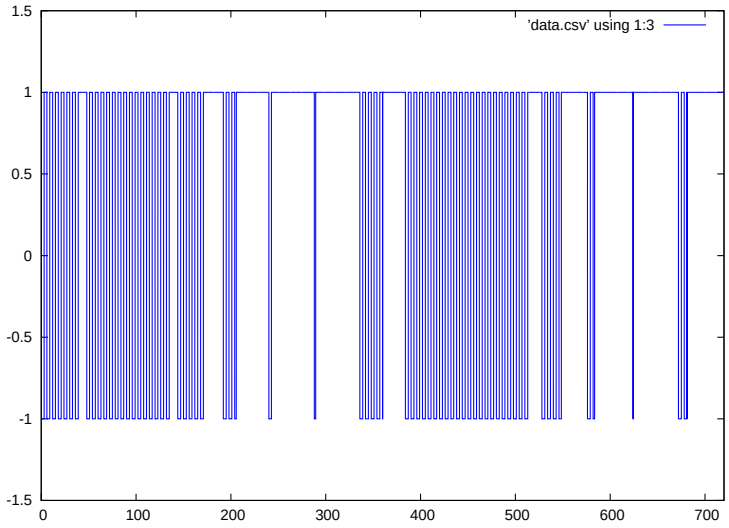
As always, your goal is to fully *explain* your analysis of each problem. Simply obtaining a correct answer is not good enough – you must also demonstrate sound reasoning in order to successfully complete the assignment. Your instructor's responsibility is to probe and challenge your understanding of the relevant principles and analytical processes in order to ensure you have a strong foundation upon which to build further understanding.

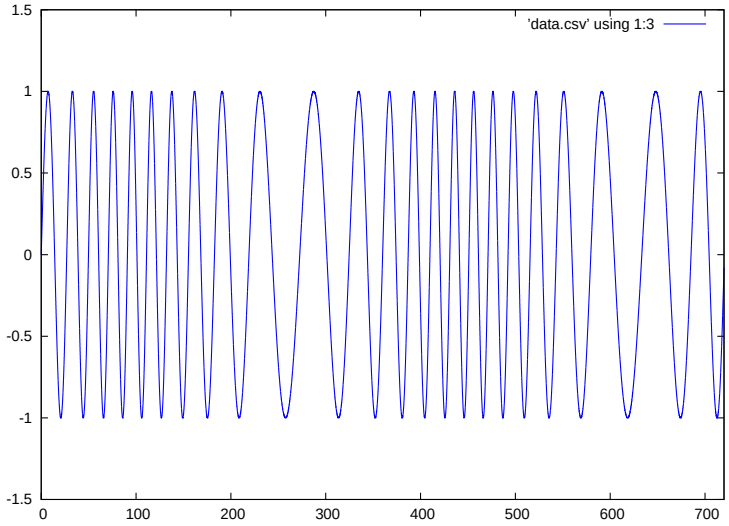
You will note a conspicuous lack of answers given for these diagnostic questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

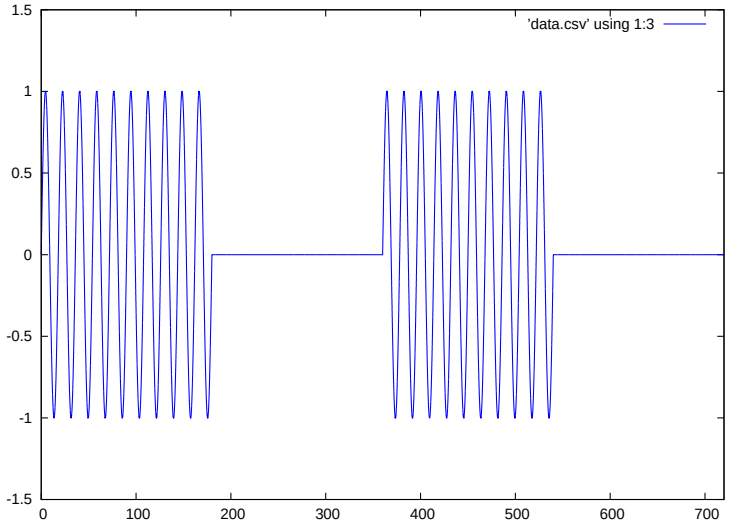
Another means of checking your diagnostic answers, where applicable, is to use circuit simulation software to explore the effects of faults placed in circuits. For example, if one of these diagnostic questions requires that you predict the effect of an open or a short in a circuit, you may check the validity of your work by simulating that same fault (substituting a very high resistance in place of that component for an open, and substituting a very low resistance for a short) within software and seeing if the results agree.

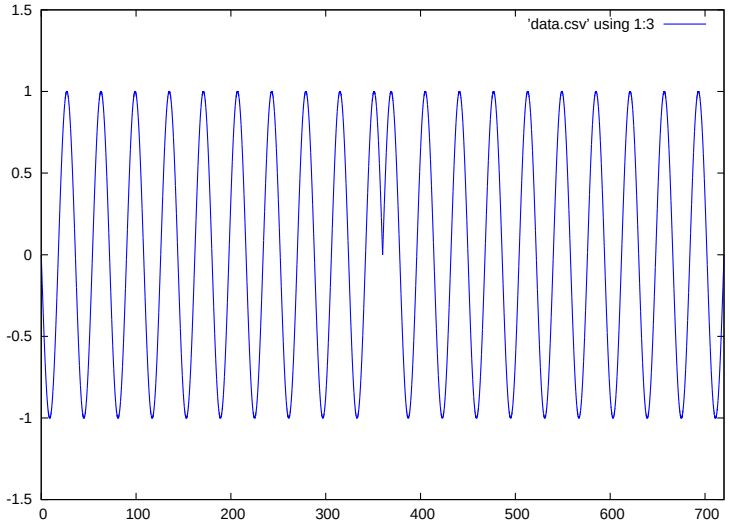
7.3.1 Identifying modulation types

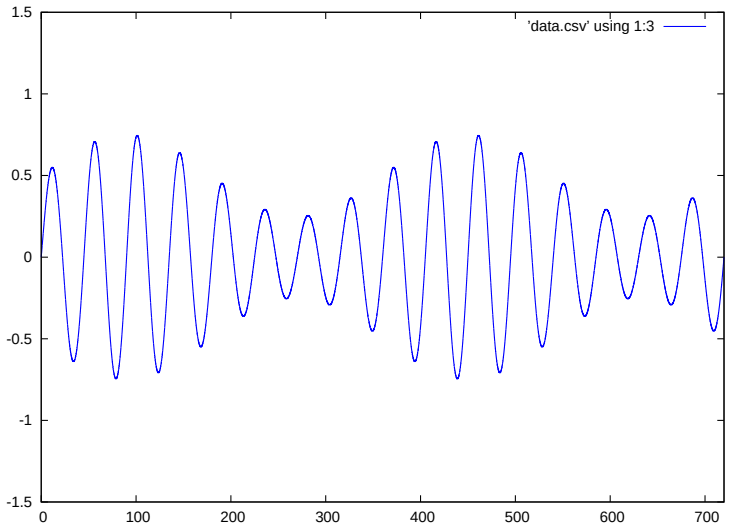
Identify the type of modulation, if possible, represented by each of the time-domain signal displays:






'data.csv' using 1:3
1.5
1
0.5
0
-0.5
-1
-1.5
0
100
200
300
400
500
600
700


'data.csv' using 1:3
1.5
1
0.5
0
-0.5
-1
-1.5
0
100
200
300
400
500
600
700

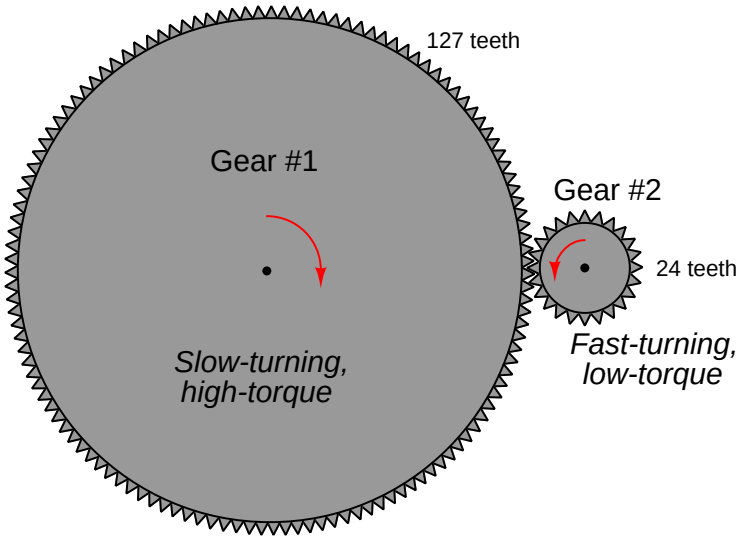


Challenges

- If possible, identify the depth of modulation for each of these waveforms.

7.3.2 Gear noise diagnosis

When mechanical gears mesh, the rotational speeds of their shafts are inversely proportional to the number of teeth on each of the meshing gears. For the gear set shown below, the speed ratio will be $\frac{127}{24}$:



Each shaft connected to the gear of course rotates at the same speed as its gear. If some portion of that rotating assembly is out-of-balance, it will create a sinusoidal vibration in the machine with a frequency equal to the shaft's speed (e.g. 1000 RPM = 16.67 Hz).

Suppose the larger of the two gears in the above illustration turns at 1500 RPM. Identify the vibrational frequency we would measure if one of these two shafts were out of balance.

Calculate the frequency of the noise produced by the meshing teeth of both gears.

If one of the two shafts is slightly *bent*, that shaft's gear will press more firmly into the other gear once per revolution, resulting in its gear tooth noise being *amplitude-modulated*. Suppose we measured sideband frequencies of 3150 Hz and 3200 Hz in the machine's vibration signature. Based on this measurement, which shaft is bent?

Challenges

- Calculate the sideband frequencies resulting from the other shaft being bent.

Appendix A

Problem-Solving Strategies

The ability to solve complex problems is arguably one of the most valuable skills one can possess, and this skill is particularly important in any science-based discipline.

- Study principles, not procedures. Don't be satisfied with merely knowing how to compute solutions – learn *why* those solutions work.
- Identify what it is you need to solve, identify all relevant data, identify all units of measurement, identify any general principles or formulae linking the given information to the solution, and then identify any "missing pieces" to a solution. Annotate all diagrams with this data.
- Sketch a diagram to help visualize the problem. When building a real system, always devise a plan for that system and analyze its function *before* constructing it.
- Follow the units of measurement and meaning of every calculation. If you are ever performing mathematical calculations as part of a problem-solving procedure, and you find yourself unable to apply each and every intermediate result to some aspect of the problem, it means you don't understand what you are doing. Properly done, every mathematical result should have practical meaning for the problem, and not just be an abstract number. You should be able to identify the proper units of measurement for each and every calculated result, and show where that result fits into the problem.
- Perform "thought experiments" to explore the effects of different conditions for theoretical problems. When troubleshooting real systems, perform *diagnostic tests* rather than visually inspecting for faults, the best diagnostic test being the one giving you the most information about the nature and/or location of the fault with the fewest steps.
- Simplify the problem until the solution becomes obvious, and then use that obvious case as a model to follow in solving the more complex version of the problem.
- Check for exceptions to see if your solution is incorrect or incomplete. A good solution will work for *all* known conditions and criteria. A good example of this is the process of testing scientific hypotheses: the task of a scientist is not to find support for a new idea, but rather to *challenge* that new idea to see if it holds up under a battery of tests. The philosophical

principle of *reductio ad absurdum* (i.e. disproving a general idea by finding a specific case where it fails) is useful here.

- Work "backward" from a hypothetical solution to a new set of given conditions.
- Add quantities to problems that are qualitative in nature, because sometimes a little math helps illuminate the scenario.
- Sketch graphs illustrating how variables relate to each other. These may be quantitative (i.e. with realistic number values) or qualitative (i.e. simply showing increases and decreases).
- Treat quantitative problems as qualitative in order to discern the relative magnitudes and/or directions of change of the relevant variables. For example, try determining what happens if a certain variable were to increase or decrease before attempting to precisely calculate quantities: how will each of the dependent variables respond, by increasing, decreasing, or remaining the same as before?
- Consider limiting cases. This works especially well for qualitative problems where you need to determine which direction a variable will change. Take the given condition and magnify that condition to an extreme degree as a way of simplifying the direction of the system's response.
- Check your work. This means regularly testing your conclusions to see if they make sense. This does *not* mean repeating the same steps originally used to obtain the conclusion(s), but rather to use some other means to check validity. Simply repeating procedures often leads to *repeating the same errors* if any were made, which is why alternative paths are better.

Appendix B

Instructional philosophy

"The unexamined circuit is not worth energizing" – Socrates (if he had taught electricity)

These learning modules, although useful for self-study, were designed to be used in a formal learning environment where a subject-matter expert challenges students to digest the content and exercise their critical thinking abilities in the answering of questions and in the construction and testing of working circuits.

The following principles inform the instructional and assessment philosophies embodied in these learning modules:

- The first goal of education is to enhance clear and independent thought, in order that every student reach their fullest potential in a highly complex and inter-dependent world. Robust reasoning is *always* more important than particulars of any subject matter, because its application is universal.
- Literacy is fundamental to independent learning and thought because text continues to be the most efficient way to communicate complex ideas over space and time. Those who cannot read with ease are limited in their ability to acquire knowledge and perspective.
- Articulate communication is fundamental to work that is complex and interdisciplinary.
- Faulty assumptions and poor reasoning are best corrected through challenge, not presentation. The rhetorical technique of *reductio ad absurdum* (disproving an assertion by exposing an absurdity) works well to discipline student's minds, not only to correct the problem at hand but also to learn how to detect and correct future errors.
- Important principles should be repeatedly explored and widely applied throughout a course of study, not only to reinforce their importance and help ensure their mastery, but also to showcase the interconnectedness and utility of knowledge.

These learning modules were expressly designed to be used in an "inverted" teaching environment[1] where students first read the introductory and tutorial chapters on their own, then individually attempt to answer the questions and construct working circuits according to the experiment and project guidelines. The instructor never lectures, but instead meets regularly with each individual student to review their progress, answer questions, identify misconceptions, and challenge the student to new depths of understanding through further questioning. Regular meetings between instructor and student should resemble a Socratic[2] dialogue, where questions serve as scalpels to dissect topics and expose assumptions. The student passes each module only after consistently demonstrating their ability to logically analyze and correctly apply all major concepts in each question or project/experiment. The instructor must be vigilant in probing each student's understanding to ensure they are truly *reasoning* and not just *memorizing*. This is why "Challenge" points appear throughout, as prompts for students to think deeper about topics and as starting points for instructor queries. Sometimes these challenge points require additional knowledge that hasn't been covered in the series to answer in full. This is okay, as the major purpose of the Challenges is to stimulate analysis and synthesis on the part of each student.

The instructor must possess enough mastery of the subject matter and awareness of students' reasoning to generate their own follow-up questions to practically any student response. Even completely correct answers given by the student should be challenged by the instructor for the purpose of having students practice articulating their thoughts and defending their reasoning. Conceptual errors committed by the student should be exposed and corrected not by direct instruction, but rather by reducing the errors to an absurdity[3] through well-chosen questions and thought experiments posed by the instructor. Becoming proficient at this style of instruction requires time and dedication, but the positive effects on critical thinking for both student and instructor are spectacular.

An inspection of these learning modules reveals certain unique characteristics. One of these is a bias toward thorough explanations in the tutorial chapters. Without a live instructor to explain concepts and applications to students, the text itself must fulfill this role. This philosophy results in lengthier explanations than what you might typically find in a textbook, each step of the reasoning process fully explained, including footnotes addressing common questions and concerns students raise while learning these concepts. Each tutorial seeks to not only explain each major concept in sufficient detail, but also to explain the logic of each concept and how each may be developed

[1] In a traditional teaching environment, students first encounter new information via *lecture* from an expert, and then independently apply that information via *homework*. In an "inverted" course of study, students first encounter new information via *homework*, and then independently apply that information under the scrutiny of an expert. The expert's role in lecture is to simply *explain*, but the expert's role in an inverted session is to *challenge*, *critique*, and if necessary *explain* where gaps in understanding still exist.

[2] Socrates is a figure in ancient Greek philosophy famous for his unflinching style of questioning. Although he authored no texts, he appears as a character in Plato's many writings. The essence of Socratic philosophy is to leave no question unexamined and no point of view unchallenged. While purists may argue a topic such as electric circuits is too narrow for a true Socratic-style dialogue, I would argue that the essential thought processes involved with scientific reasoning on *any* topic are not far removed from the Socratic ideal, and that students of electricity and electronics would do very well to challenge assumptions, pose thought experiments, identify fallacies, and otherwise employ the arsenal of critical thinking skills modeled by Socrates.

[3] This rhetorical technique is known by the Latin phrase *reductio ad absurdum*. The concept is to expose errors by counter-example, since only one solid counter-example is necessary to disprove a universal claim. As an example of this, consider the common misconception among beginning students of electricity that voltage cannot exist without current. One way to apply *reductio ad absurdum* to this statement is to ask how much current passes through a fully-charged battery connected to nothing (i.e. a clear example of voltage existing without current).

from "first principles". Again, this reflects the goal of developing clear and independent thought in students' minds, by showing how clear and logical thought was used to forge each concept. Students benefit from witnessing a model of clear thinking in action, and these tutorials strive to be just that.

Another characteristic of these learning modules is a lack of step-by-step instructions in the Project and Experiment chapters. Unlike many modern workbooks and laboratory guides where step-by-step instructions are prescribed for each experiment, these modules take the approach that students must learn to closely read the tutorials and apply their own reasoning to identify the appropriate experimental steps. Sometimes these steps are plainly declared in the text, just not as a set of enumerated points. At other times certain steps are implied, an example being assumed competence in test equipment use where the student should not need to be told *again* how to use their multimeter because that was thoroughly explained in previous lessons. In some circumstances no steps are given at all, leaving the entire procedure up to the student.

This lack of prescription is not a flaw, but rather a feature. Close reading and clear thinking are foundational principles of this learning series, and in keeping with this philosophy all activities are designed to *require* those behaviors. Some students may find the lack of prescription frustrating, because it demands more from them than what their previous educational experiences required. This frustration should be interpreted as an unfamiliarity with autonomous thinking, a problem which must be corrected if the student is ever to become a self-directed learner and effective problem-solver. Ultimately, the need for students to read closely and think clearly is more important both in the near-term and far-term than any specific facet of the subject matter at hand. If a student takes longer than expected to complete a module because they are forced to outline, digest, and reason on their own, so be it. The future gains enjoyed by developing this mental discipline will be well worth the additional effort and delay.

Another feature of these learning modules is that they do not treat topics in isolation. Rather, important concepts are introduced early in the series, and appear repeatedly as stepping-stones toward other concepts in subsequent modules. This helps to avoid the "compartmentalization" of knowledge, demonstrating the inter-connectedness of concepts and simultaneously reinforcing them. Each module is fairly complete in itself, reserving the beginning of its tutorial to a review of foundational concepts.

This methodology of assigning text-based modules to students for digestion and then using Socratic dialogue to assess progress and hone students' thinking was developed over a period of several years by the author with his Electronics and Instrumentation students at the two-year college level. While decidedly unconventional and sometimes even unsettling for students accustomed to a more passive lecture environment, this instructional philosophy has proven its ability to convey conceptual mastery, foster careful analysis, and enhance employability so much better than lecture that the author refuses to ever teach by lecture again.

Problems which often go undiagnosed in a lecture environment are laid bare in this "inverted" format where students must articulate and logically defend their reasoning. This, too, may be unsettling for students accustomed to lecture sessions where the instructor cannot tell for sure who comprehends and who does not, and this vulnerability necessitates sensitivity on the part of the "inverted" session instructor in order that students never feel discouraged by having their errors exposed. *Everyone* makes mistakes from time to time, and learning is a lifelong process! Part of the instructor's job is to build a culture of learning among the students where errors are not seen as shameful, but rather as opportunities for progress.

To this end, instructors managing courses based on these modules should adhere to the following principles:

- Student questions are always welcome and demand thorough, honest answers. The only type of question an instructor should refuse to answer is one the student should be able to easily answer on their own. Remember, *the fundamental goal of education is for each student to learn to think clearly and independently.* This requires hard work on the part of the student, which no instructor should ever circumvent. Anything done to bypass the student's responsibility to do that hard work ultimately limits that student's potential and thereby does real harm.

- It is not only permissible, but encouraged, to answer a student's question by asking questions in return, these follow-up questions designed to guide the student to reach a correct answer through their own reasoning.

- All student answers demand to be challenged by the instructor and/or by other students. This includes both correct and incorrect answers – the goal is to practice the articulation and defense of one's own reasoning.

- No reading assignment is deemed complete unless and until the student demonstrates their ability to accurately summarize the major points in their own terms. Recitation of the original text is unacceptable. This is why every module contains an "Outline and reflections" question as well as a "Foundational concepts" question in the Conceptual reasoning section, to prompt reflective reading.

- No assigned question is deemed answered unless and until the student demonstrates their ability to consistently and correctly apply the concepts to *variations* of that question. This is why module questions typically contain multiple "Challenges" suggesting different applications of the concept(s) as well as variations on the same theme(s). Instructors are encouraged to devise as many of their own "Challenges" as they are able, in order to have a multitude of ways ready to probe students' understanding.

- No assigned experiment or project is deemed complete unless and until the student demonstrates the task in action. If this cannot be done "live" before the instructor, video-recordings showing the demonstration are acceptable. All relevant safety precautions must be followed, all test equipment must be used correctly, and the student must be able to properly explain all results. The student must also successfully answer all Challenges presented by the instructor for that experiment or project.

Students learning from these modules would do well to abide by the following principles:

- No text should be considered fully and adequately read unless and until you can express every idea *in your own words, using your own examples.*
- You should always articulate your thoughts as you read the text, noting points of agreement, confusion, and epiphanies. Feel free to print the text on paper and then write your notes in the margins. Alternatively, keep a journal for your own reflections as you read. This is truly a helpful tool when digesting complicated concepts.
- Never take the easy path of highlighting or underlining important text. Instead, *summarize* and/or *comment* on the text using your own words. This actively engages your mind, allowing you to more clearly perceive points of confusion or misunderstanding on your own.
- A very helpful strategy when learning new concepts is to place yourself in the role of a teacher, if only as a mental exercise. Either explain what you have recently learned to someone else, or at least *imagine* yourself explaining what you have learned to someone else. The simple act of having to articulate new knowledge and skill forces you to take on a different perspective, and will help reveal weaknesses in your understanding.
- Perform each and every mathematical calculation and thought experiment shown in the text on your own, referring back to the text to see that your results agree. This may seem trivial and unnecessary, but it is critically important to ensuring you actually understand what is presented, especially when the concepts at hand are complicated and easy to misunderstand. Apply this same strategy to become proficient in the use of *circuit simulation software*, checking to see if your simulated results agree with the results shown in the text.
- Above all, recognize that learning is hard work, and that a certain level of frustration is unavoidable. There are times when you will struggle to grasp some of these concepts, and that struggle is a natural thing. Take heart that it will yield with persistent and varied[4] effort, and never give up!

Students interested in using these modules for self-study will also find them beneficial, although the onus of responsibility for thoroughly reading and answering questions will of course lie with that individual alone. If a qualified instructor is not available to challenge students, a workable alternative is for students to form study groups where they challenge[5] one another.

To high standards of education,

Tony R. Kuphaldt

[4] As the old saying goes, "Insanity is trying the same thing over and over again, expecting different results." If you find yourself stumped by something in the text, you should attempt a different approach. Alter the thought experiment, change the mathematical parameters, do whatever you can to see the problem in a slightly different light, and then the solution will often present itself more readily.

[5] Avoid the temptation to simply share answers with study partners, as this is really counter-productive to learning. Always bear in mind that the answer to any question is far less important in the long run than the method(s) used to obtain that answer. The goal of education is to empower one's life through the improvement of clear and independent thought, literacy, expression, and various practical skills.

Appendix C

Tools used

I am indebted to the developers of many open-source software applications in the creation of these learning modules. The following is a list of these applications with some commentary on each.

You will notice a theme common to many of these applications: a bias toward *code*. Although I am by no means an expert programmer in any computer language, I understand and appreciate the flexibility offered by code-based applications where the user (you) enters commands into a plain ASCII text file, which the software then reads and processes to create the final output. Code-based computer applications are by their very nature *extensible*, while WYSIWYG (What You See Is What You Get) applications are generally limited to whatever user interface the developer makes for you.

The `GNU/Linux` computer operating system

> There is so much to be said about Linus Torvalds' `Linux` and Richard Stallman's `GNU` project. First, to credit just these two individuals is to fail to do justice to the *mob* of passionate volunteers who contributed to make this amazing software a reality. I first learned of `Linux` back in 1996, and have been using this operating system on my personal computers almost exclusively since then. It is *free*, it is completely *configurable*, and it permits the continued use of highly efficient `Unix` applications and scripting languages (e.g. shell scripts, Makefiles, `sed`, `awk`) developed over many decades. `Linux` not only provided me with a powerful computing platform, but its open design served to inspire my life's work of creating open-source educational resources.

Bram Moolenaar's `Vim` text editor

> Writing code for any code-based computer application requires a *text editor*, which may be thought of as a word processor strictly limited to outputting plain-ASCII text files. Many good text editors exist, and one's choice of text editor seems to be a deeply personal matter within the programming world. I prefer `Vim` because it operates very similarly to `vi` which is ubiquitous on `Unix`/`Linux` operating systems, and because it may be entirely operated via keyboard (i.e. no mouse required) which makes it fast to use.

Donald Knuth's TEX typesetting system

Developed in the late 1970's and early 1980's by computer scientist extraordinaire Donald Knuth to typeset his multi-volume magnum opus *The Art of Computer Programming*, this software allows the production of formatted text for screen-viewing or paper printing, all by writing plain-text code to describe how the formatted text is supposed to appear. TEX is not just a markup language for documents, but it is also a Turing-complete programming language in and of itself, allowing useful algorithms to be created to control the production of documents. Simply put, TEX *is a programmer's approach to word processing*. Since TEX is controlled by code written in a plain-text file, this means anyone may read that plain-text file to see exactly how the document was created. This openness afforded by the code-based nature of TEX makes it relatively easy to learn how other people have created their own TEX documents. By contrast, examining a beautiful document created in a conventional WYSIWYG word processor such as Microsoft `Word` suggests nothing to the reader about *how* that document was created, or what the user might do to create something similar. As Mr. Knuth himself once quipped, conventional word processing applications should be called WYSIAYG (What You See Is *All* You Get).

Leslie Lamport's LATEX extensions to TEX

Like all true programming languages, TEX is inherently extensible. So, years after the release of TEX to the public, Leslie Lamport decided to create a massive extension allowing easier compilation of book-length documents. The result was LATEX, which is the markup language used to create all ModEL module documents. You could say that TEX is to LATEX as `C` is to `C++`. This means it is permissible to use any and all TEX commands within LATEX source code, and it all still works. Some of the features offered by LATEX that would be challenging to implement in TEX include automatic index and table-of-content creation.

Tim Edwards' `Xcircuit` drafting program

This wonderful program is what I use to create all the schematic diagrams and illustrations (but not photographic images or mathematical plots) throughout the ModEL project. It natively outputs PostScript format which is a true vector graphic format (this is why the images do not pixellate when you zoom in for a closer view), and it is so simple to use that I have never had to read the manual! Object libraries are easy to create for `Xcircuit`, being plain-text files using PostScript programming conventions. Over the years I have collected a large set of object libraries useful for drawing electrical and electronic schematics, pictorial diagrams, and other technical illustrations.

`Gimp` graphic image manipulation program

Essentially an open-source clone of Adobe's `PhotoShop`, I use `Gimp` to resize, crop, and convert file formats for all of the photographic images appearing in the ModEL modules. Although `Gimp` does offer its own scripting language (called `Script-Fu`), I have never had occasion to use it. Thus, my utilization of `Gimp` to merely crop, resize, and convert graphic images is akin to using a sword to slice bread.

`SPICE` circuit simulation program

`SPICE` is to circuit analysis as TEX is to document creation: it is a form of markup language designed to describe a certain object to be processed in plain-ASCII text. When the plain-text "source file" is compiled by the software, it outputs the final result. More modern circuit analysis tools certainly exist, but I prefer `SPICE` for the following reasons: it is *free*, it is *fast*, it is *reliable*, and it is a fantastic tool for *teaching* students of electricity and electronics how to write simple code. I happen to use rather old versions of `SPICE`, version 2g6 being my "go to" application when I only require text-based output. `NGSPICE` (version 26), which is based on Berkeley `SPICE` version 3f5, is used when I require graphical output for such things as time-domain waveforms and Bode plots. In all `SPICE` example netlists I strive to use coding conventions compatible with all `SPICE` versions.

Andrew D. Hwang's `ePiX` mathematical visualization programming library

This amazing project is a `C++` library you may link to any `C/C++` code for the purpose of generating PostScript graphic images of mathematical functions. As a completely free and open-source project, it does all the plotting I would otherwise use a Computer Algebra System (CAS) such as `Mathematica` or `Maple` to do. It should be said that `ePiX` is *not* a Computer Algebra System like `Mathematica` or `Maple`, but merely a mathematical *visualization* tool. In other words, it won't determine integrals for you (you'll have to implement that in your own `C/C++` code!), but it can graph the results, and it does so beautifully. What I really admire about `ePiX` is that it is a `C++` programming library, which means it builds on the existing power and toolset available with that programming language. Mr. Hwang could have probably developed his own stand-alone application for mathematical plotting, but by creating a `C++` library to do the same thing he accomplished something much greater.

`gnuplot` mathematical visualization software

Another open-source tool for mathematical visualization is `gnuplot`. Interestingly, this tool is *not* part of Richard Stallman's GNU project, its name being a coincidence. For this reason the authors prefer "gnu" *not* be capitalized at all to avoid confusion. This is a much "lighter-weight" alternative to a spreadsheet for plotting tabular data, and the fact that it easily outputs directly to an X11 console or a file in a number of different graphical formats (including PostScript) is very helpful. I typically set my `gnuplot` output format to default (X11 on my Linux PC) for quick viewing while I'm developing a visualization, then switch to PostScript file export once the visual is ready to include in the document(s) I'm writing. As with my use of `Gimp` to do rudimentary image editing, my use of `gnuplot` only scratches the surface of its capabilities, but the important points are that it's *free* and that it *works well*.

`Python` programming language

Both Python and C++ find extensive use in these modules as instructional aids and exercises, but I'm listing Python here as a *tool* for myself because I use it almost daily as a *calculator*. If you open a Python interpreter console and type `from math import *` you can type mathematical expressions and have it return results just as you would on a hand calculator. Complex-number (i.e. *phasor*) arithmetic is similarly supported if you include the complex-math library (`from cmath import *`). Examples of this are shown in the Programming References chapter (if included) in each module. Of course, being a fully-featured programming language, Python also supports conditionals, loops, and other structures useful for calculation of quantities. Also, running in a console environment where all entries and returned values show as text in a chronologically-ordered list makes it easy to copy-and-paste those calculations to document exactly how they were performed.

Appendix D

Creative Commons License

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or

limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

A. reproduce and Share the Licensed Material, in whole or in part; and

B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures.

For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

A. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors,

whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully

be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at `creativecommons.org/policies`, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at `creativecommons.org`.

Appendix E

References

The ARRL Antenna Book, Eleventh Edition, The American Radio Relay League, Inc., Newington, CT, 1968.

Bryant, James "Multipliers vs. Modulators", *Analog Dialogue*, Volume 47, Number 06, pp. 1-2, June 2013.

Elwell, C.F., *The Poulsen Arc Generator*, Van Nostrand Company, New York, NY, 1923.

Espenschied, Lloyd, and Affel, Herman, *US Patent 1,835,031*, "Concentric Conducting System", application 23 May 1929, patent granted 8 December 1931.

Gilbert, Barrie "A Precise Four-Quadrant Multiplier with Subnanosecond Response", *IEEE Journal of Solid-State Circuits*, Volume SC-3, Number 4, pp. 365-373, December 1968.

Hogan, John V.L., "The Heterodyne Receiver", *The Electric Journal*, Volume 18, Number 1, pp. 116-119, The Electric Journal, Pittsburgh, January-December 1922.

Horak, Ray, *Telecommunications and Data Communications Handbook*, John Wiley & Sons, Inc., New York, NY, 2007.

Lee, R.C.T. and Chiu, Mao-Ching and Lin, Jung-Shan, *Communications Engineering*, John Wiley & Sons (Asia) Pte Ltd, Singapore, 2007.

Marki, Ferenc, and Marki, Christopher, "Mixer Basics Primer – A Tutorial for RF & Microwave Mixers", Marki Microwave Incorporated, Morgan Hill, CA, 2010.

"MC1496, MC1496B Balanced Modulators/Demodulators", revision 10, publication order number MC1496/D, ON Semiconductor, Semiconductor Components Industries LLC, Denver, CO, October 2006.

Tomasi, Wayne, *Electronic Communications Systems*, third edition, Prentice-Hall Incorporated, Upper Saddle River, NJ, 1998.

Appendix F

Version history

This is a list showing all significant additions, corrections, and other edits made to this learning module. Each entry is referenced by calendar date in reverse chronological order (newest version first), which appears on the front cover of every learning module for easy reference. Any contributors to this open-source document are listed here as well.

16 April 2025 – added a new Case Tutorial section showing some useful GNU Radio Companion flowgraphs for software-defined radio.

27 February 2025 – typo correction courtesy of Jacob Stormes: "where is is" should have been "where it is".

26 February 2025 – minor edits to the "Example: downconverting AM receiver" Case Tutorial section.

17 December 2024 – added a new Tutorial section on lock-in analyzers, and added some index entries to other Tutorial sections.

17 October 2024 – added a new Case Tutorial section showing a practical frequency downconversion example for an AM receiver.

1 October 2024 – minor edits to the "Example: balanced mixer frequencies" section of the Case Tutorial chapter to make it clearer where sinusoidal signals are involved. Also added another Challenge question to the "High-side versus low-side injection" Quantitative Reasoning question.

26 September 2024 – divided the Introduction chapter into sections, one with recommendations for students, one with a listing of challenging concepts, and one with recommendations for instructors. Also divided the Full Tutorial into sections.

13 April 2024 – corrected a mis-statement about simultaneous band reception using software-defined radio (SDR). Edited image_7158 as well.

17 March 2024 – added a new Tutorial section on Software-Defined Radio (SDR), as well as some

new questions to the Introduction chapter.

10 March 2024 – added questions to the Introduction chapter.

6 March 2024 – edited the Historical References section on "Heterodyne radio reception" to include explanations of obsolete terms.

3 March 2024 – added some instructor notes.

10 January 2024 – edited text in the "Amplitude modulation" section of the Tutorial chapter to make use of the term *band* to describe a spectrum of frequencies present in one complex signal.

19 October 2023 – added a new illustration to the Tutorial showing carrier and baseband signals within a radio transmitter/receiver system.

21 June 2023 – added a new Tutorial section on "sampling" modulation.

4 June 2023 – removed an unused variable in both of the amplitude-modulation (AM) C++ programs shown in the "Modeling signal modulation using C++" sections of the Programming References chapter.

13 May 2023 – added Case Tutorial section on on frequency-mixing within gear mechanisms.

28 November 2022 – placed questions at the top of the itemized list in the Introduction chapter prompting students to devise experiments related to the tutorial content.

16 March 2022 – minor edits to the Tutorial.

21 December 2021 – added an intrductory section to the Tutorial.

6-7 December 2021 – minor edits to the Tutorial, correcting a typo and also adding concept from the module on Pulse Width Modulation to describe demodulation of PWM and PDM signals.

17 November 2021 – provided acronyms for the various forms of digital modulation (ASK, OOK, FSK, and PSK).

22 October 2021 – minor edit to image_4685, repositioning a V_{out} label for more clarity.

10 May 2021 – commented out or deleted empty chapters.

26 April 2021 – minor edits to the Historical References chapter, helping to explain the context of Reginald Fessenden's *heterodyne* radio receiver invention.

6-7 December 2020 – Added an Historical Reference section showing amplitude-modulation techniques used for legacy arc-converter radio transmitters. Also, corrected a kHz/MHz typo.

2-3 December 2020 – significantly edited the Introduction chapter to make it more suitable as a pre-study guide and to provide cues useful to instructors leading "inverted" teaching sessions. Also

made minor edits to the Tutorial and added a new Case Tutorial section showing the AD633 analog multiplier being used as a mixer for amplitude modulation.

22 November 2020 – added more to the I-Q modulation section of the Tutorial, giving limiting-case examples using DC inputs.

6 November 2020 – added a Case Tutorial chapter, containing results from a simple diode-based mixer circuit.

16 August 2020 – added more content to the Technical References chapter, and also added a Diagnostic Reasoning question on gear noise.

14 August 2020 – corrected an error in the Tutorial chapter regarding I-Q style frequency modulation.

8-12 August 2020 – added content to Tutorial as well as questions.

11 May 2020 – added C++ code example modeling different types of modulation.

12 November 2019 – added NGSPICE analysis showing mixer function as amplitude modulator.

11 November 2019 – added more content to the Technical References chapter.

10 November 2019 – document first created.

Index

ω, 71, 84

Adding quantities to a qualitative problem, 142
AF, 43
Airspy HF+ Discovery SDR receiver, 51
AM, 31
Amplitude demodulation, 72
Amplitude modulation, 31, 72
Amplitude shift keying, 31, 32
Angular velocity, 71, 84
Annotating diagrams, 141
Antenna, 28
Artifact, 41, 83
ASK, 31, 32
Audio frequency, 43

Balanced mixer, 8, 41
Baseband, 28
Beat pattern, 72

C++, 88
Capacity, 59
Checking for exceptions, 142
Checking your work, 142
Code, computer, 149
Compiler, C++, 88
Computer programming, 87
Condensor, 59
Continuous-wave communication, 32
Conversion, frequency, 42
Cosine, 71, 84
Cutoff frequency, 38
CW, 32

Decimation, 53
Degrees, 84
Depth, modulation, 31
Detector, crystal, 59
Digital signal processing, 23, 44
Dimensional analysis, 141
Direct conversion, 45
Domain, frequency, 70
Domain, time, 70
Downconversion, 44, 72
DSP, 23
Duty cycle, 38

Edwards, Tim, 150

Filter, low-pass, 38
FM, 34
Four-quadrant multiplier, 83
Frequency conversion, 42
Frequency domain, 70
Frequency modulation, 34
Frequency shift keying, 34
Frequency, audio, 43
Frequency, intermediate, 43
Frequency, radio, 43
FSK, 34

Gain-bandwidth product, 42
Gilbert cell, 78
Gilbert, Barrie, 78
GNU Radio Companion, 52
Graph values to solve a problem, 142
Greenleaf, Cynthia, 115

Harmonic, 76
Harmonic frequency, 31
Harmony, musical, 72
Heterodyning, 40, 42, 44, 72
How to teach with these modules, 144
Hwang, Andrew D., 151

Identify given data, 141

Identify relevant principles, 141
IF, 43
Instructions for projects and experiments, 145
Interference, 34
Intermediate frequency, 43
Intermediate results, 141
Interpreter, Python, 92
Inverted instruction, 144

Java, 89

Keying, amplitude, 31, 32
Keying, frequency, 34
Keying, on-off, 31
Keying, phase, 36
Knuth, Donald, 150

Lamport, Leslie, 150
Leakage, mixer, 8
Light, speed of, 28
Lightning, 34
Limiting cases, 79, 142
Low-pass filter, 38
Lower sideband, 33, 40

Maxwell, James Clerk, 55
Metacognition, 120
Microcontroller, 38
Mixer, 70
Mixer, balanced, 8
Modulation, 28
Modulation depth, 31
Modulation, amplitude, 31
Modulation, frequency, 34
Modulation, phase, 36
Moolenaar, Bram, 149
Morse code, 32, 59
Multiplier, four-quadrant, 83
Murphy, Lynn, 115
Musical harmony, 72

Negative resistance, 56
Noise immunity, FM, 34

Ohm's Law, 56
On-off keying, 31
OOK, 31

Open-source, 149
Oscillator, square wave versus sinusoidal, 43

PDM, 37
Phase modulation, 36
Phase shift keying, 36
PM, 36
Problem-solving: annotate diagrams, 141
Problem-solving: check for exceptions, 142
Problem-solving: checking work, 142
Problem-solving: dimensional analysis, 141
Problem-solving: graph values, 142
Problem-solving: identify given data, 141
Problem-solving: identify relevant principles, 141
Problem-solving: interpret intermediate results, 141
Problem-solving: limiting cases, 79, 142
Problem-solving: qualitative to quantitative, 142
Problem-solving: quantitative to qualitative, 142
Problem-solving: reductio ad absurdum, 142
Problem-solving: simplify the system, 141
Problem-solving: thought experiment, 34, 141
Problem-solving: track units of measurement, 141
Problem-solving: visually represent the system, 141
Problem-solving: work in reverse, 142
Programming, computer, 87
PSK, 36
Pulse density modulation, 37
Pulse width modulation, 37
PWM, 37
Python, 92

Quadrant, 83
Qualitatively approaching a quantitative problem, 142

Radians, 84
Radio, 28
Radio frequency, 43
Reading Apprenticeship, 115
Reductio ad absurdum, 142–144
RF, 43

Sampling modulation, 39

Schoenbach, Ruth, 115
Scientific method, 120
SDR, 23, 45, 49
SDR# software, 51
Sideband, 33, 40, 70
Simplifying a system, 141
Sine, 84
Single sideband AM, 34
Socrates, 143
Socratic dialogue, 144
Software-Defined Radio, 49
Software-defined radio, 23, 45
Source code, 88
Speed of light, 28
SPICE, 115
Square law, 75
SSB, 34
Stallman, Richard, 149

Thought experiment, 34, 141
Time domain, 70
Torvalds, Linus, 149
Trigonometry, 70, 84
Tunnel diode, 56

Unbalanced mixer, 41
Units of measurement, 141
Upconversion, 43, 72
Upper sideband, 33, 40

Visualizing a system, 141

Wavelength, 28
Whitespace, C++, 88, 89
Whitespace, Python, 95
Work in reverse to solve a problem, 142
WYSIWYG, 149, 150