Amplitude modulation of audio signals using the discrete-time Fourier Transform

In this paper, I am going to examine the amplitude modulation of a carrier signal using the discrete-time Fourier Transform. The numerical examples will use the computer procedure described in the earlier paper titled *A VisualBasic subroutine which calculates a fast Fourier Transform*.

For the application I have in mind, the frequency of the carrier signal will not be very high. It will be a high audio frequency, say $15{,}000$ Hz, well below even the lowest radio frequencies in common use. I want to set things up so that the waveform of the carrier signal itself can be adequately represented using the same sampling frequency that is used to sample the audio signal to be transmitted. This is going to require a relatively high sampling frequency.

Suppose we choose a sampling frequency that is just high enough that there are ten sampling periods in one complete period of the carrier. The $15{,}000$ Hz carrier has a period of $67\mu s$. The sampling period must therefore be $6.7\mu s$ or less. Sampling at this frequency generates $1/6.7\mu s = 149{,}254$ samples per second. Since the discrete-time Fourier Transform depends on the sample size being a power of two, the most suitable sample size would be $262{,}144 = 2^{18}$, for sample durations of one second.

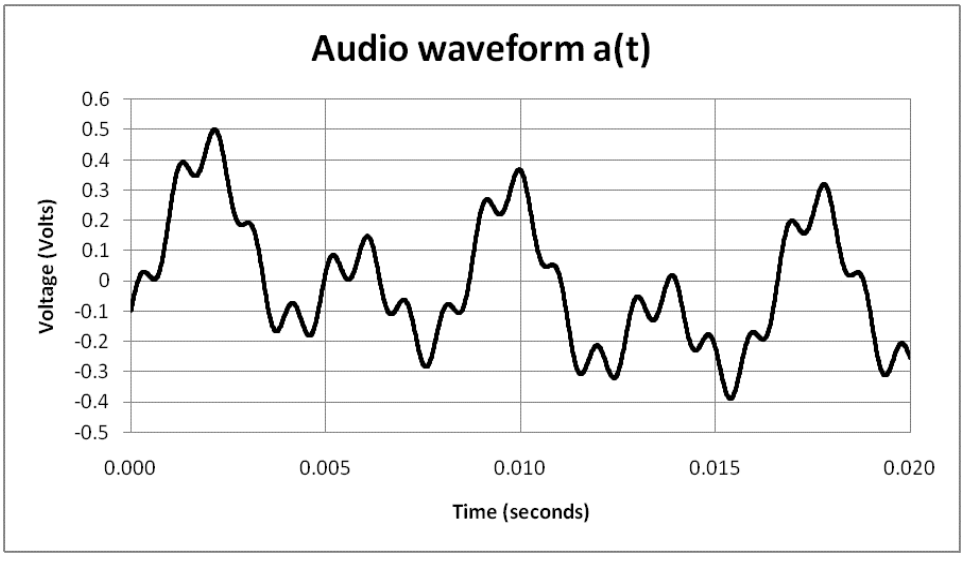
For example purposes, I am going to use a simple audio signal, with just enough content to be interesting. Here is what I propose:

Cosine component at 32 Hz with amplitude 0.1 Volts
Sine component at 128 Hz with amplitude 0.15 Volts
Cosine component at 256 Hz with amplitude -0.2 Volts
Sine component at 1024 Hz with amplitude 0.075 Volts

This audio signal can be written as a function of continuous time as:

$$a(t) = \begin{bmatrix} 0.1\cos(2\pi 32t) + \cdots \\ \cdots + 0.15\sin(2\pi 128t) + \cdots \\ \cdots - 0.2\cos(2\pi 256t) + \cdots \\ \cdots + 0.075\sin(2\pi 1024t) \end{bmatrix} \quad (1)$$

The following graph shows the first 20ms of this sample signal.



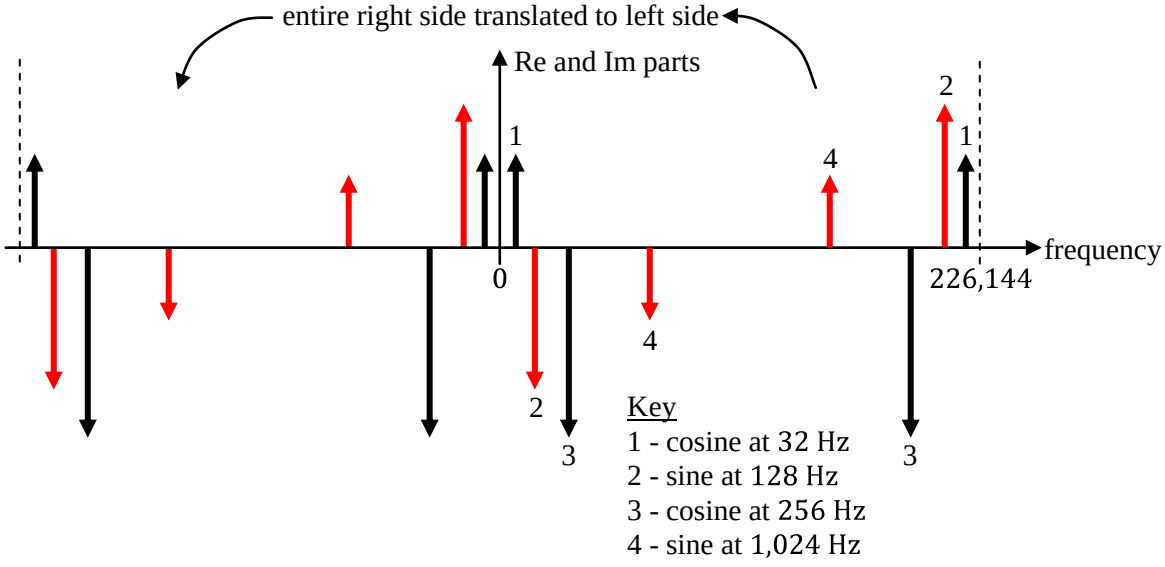
We generate 262,144 samples of this signal during the first second by using:

$$x_n = \begin{bmatrix} 0.1\cos\left(2\pi 32\dfrac{n}{262{,}144}\right)+\cdots \\ \cdots+0.15\sin\left(2\pi 128\dfrac{n}{262{,}144}\right)+\cdots \\ \cdots-0.2\cos\left(2\pi 256\dfrac{n}{262{,}144}\right)+\cdots \\ \cdots+0.075\sin\left(2\pi 1024\dfrac{n}{262{,}144}\right) \end{bmatrix} \text{ for } n=0,1,2,\cdots,262{,}142,262{,}143 \quad (2)$$

We next apply the forward transform (subroutine DTFT_Foward() in the computer procedure) to this sample of voltages. The resulting values are post-processed by: (i) removing the artifacts of machine imprecision by setting all values less than 10^{-8} exactly to zero, and (ii) by dividing all values by 262,144. The resulting frequency components are:

```
K=32:          DTFTRe =  0.05      DTFTIm =  0
K=128:         DTFTRe =  0         DTFTIm = -0.075
K=256:         DTFTRe = -0.1       DTFTIm =  0
K=1,024:       DTFTRe =  0         DTFTIm = -0.0375
K=261,120:     DTFTRe =  0         DTFTIm =  0.0375
K=261,888:     DTFTRe = -0.1       DTFTIm =  0
K=262,016:     DTFTRe =  0         DTFTIm =  0.075
K=262,112:     DTFTRe =  0.05      DTFTIm =  0
```

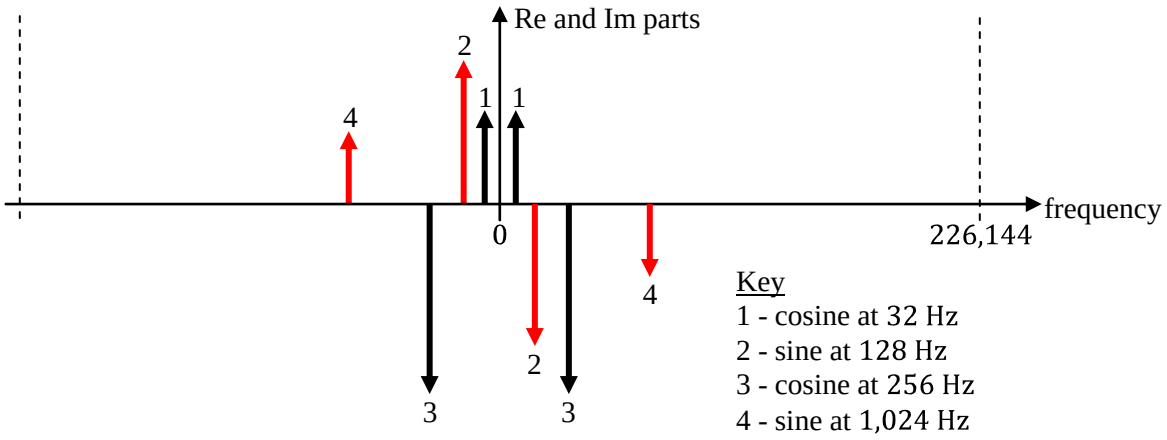
The spectrum is nice and simple, but only because the frequencies I chose for the components of the audio signal were nice round numbers. I have graphed the frequency spectrum as follows. Real components (which arise from the cosine terms) are rendered in black; imaginary components (which arise from the sine terms) are rendered in red. I have shown the "basic" range of frequency components, from zero to 262,144 Hz, and also the repetition of this basic range one complete cycle to the left.



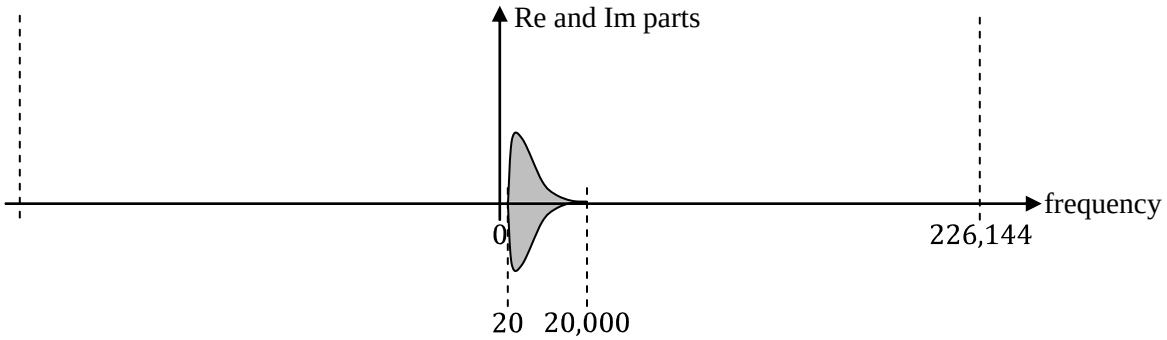
The DTFT_Forward() transform generates two frequency components for each trigonometric term. One component appears at the expected frequency, say $f = 32$ Hz. The other appears at the frequency $226{,}144 - f$. These higher frequency components arise because of a basic assumption that must be made to enjoy the benefits of a fast Fourier Transform, namely, that the set of sample data is repeated

indefinitely into both positive and negative times with a period of 226,144 sample times. This gives rises to repeating cycles of frequency components. Because of this periodicity, all of the frequencies which appear in the extended frequency spectrum are valid representations of the time-domain samples.

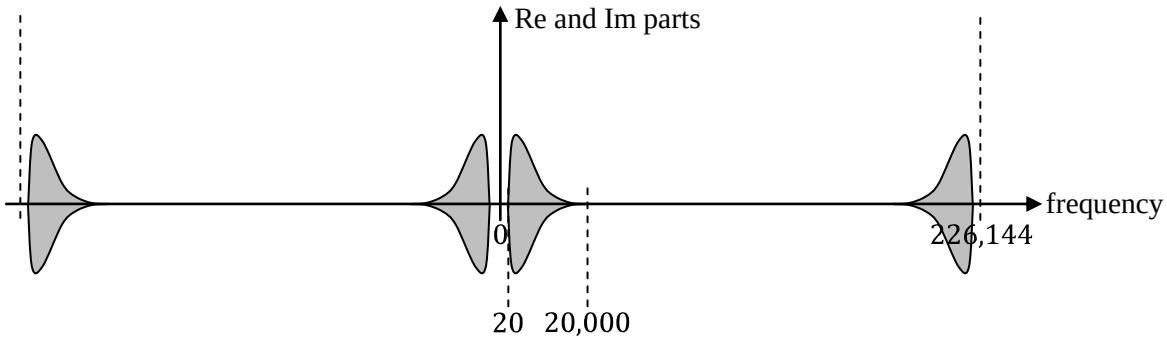
It is customary to focus one's attention only on that part of the spectrum which is centered around zero. The following graph shows what is called the "baseband" of the sample data.



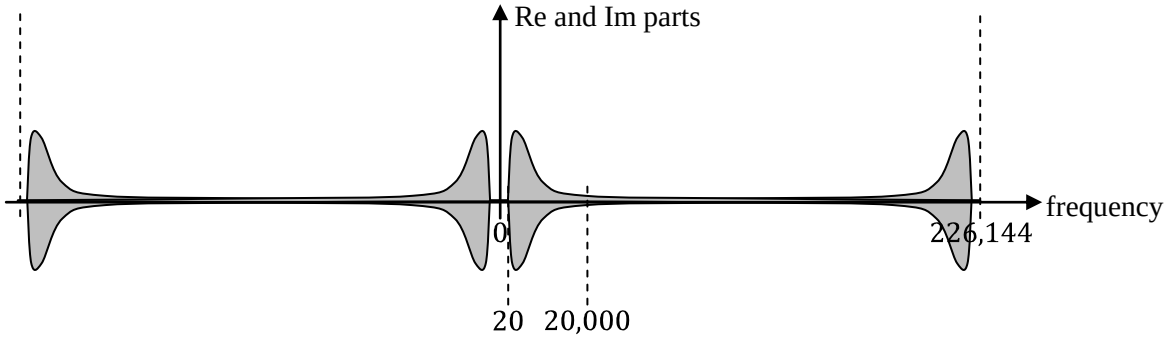
It is said that a good human ear can hear sounds in the range 20 Hz to 20,000 Hz. The principal frequency components in a high-quality sound track might therefore be described graphically as follows. The area shaded in grey is intended to represent the envelope containing the many frequency components, both Real and Imaginary, which would be found in such a sound track. The components would range from 20 Hz up to 20,000 Hz. It is the case with real-life sources of sound that substantially all of the energy is to be found in the lower frequencies. That is why I have shown the envelope being larger at its low-frequency end.



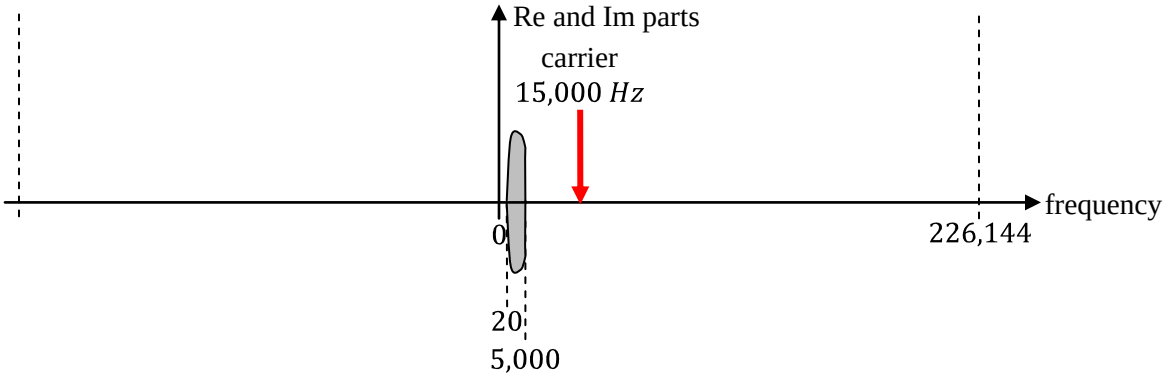
If the sound track has a finite duration, but if the track is assumed to be repeated over-and-over in both positive and negative times, then the Fourier Transform will contain reflections and translations of the basic spectrum shown above. We could graph them notionally somewhat like this:



Next, assume the sound track is digitally sampled, rather than treated as a continuous waveform. Then there will be frequency components all the way up to 226,144 Hz. The nice trigonometric functions I used in the sample waveform in Equation (1) did not result in this phenomenon, but real audio signals include frequencies which are not integral fractions of 226,144 Hz , and the discrete-time Fourier Transform will generate many high-order components as it tries to compensate for non-periodicity in the raw data. A more realistic graph of the frequency components an FFT routine would generate for sampled real-world audio signals is as follows. Fortunately, the spurious components typically decrease in magnitude as the frequency increases, so the "tails" of the lobes will be small.



Before doing any processing on a frequency spectrum like this, I am going to "filter out" all frequencies less than, say, 5,000 *Hz*. Having done a Fourier Transform, we know the specific frequency components, so this kind of filtering is easy to do. We can simply ignore, or set to zero, all components at frequencies greater than 5,000 Hz. We can also filter out all the negative frequencies by ignoring them, too. After this filtering, the spectrum we will have for the sampled sound track should look something like this:



It appears from the figure that filtering at 5,000 Hz is draconian, and chops off a great deal of the information about the signal. In practice, the situation is not usually this severe. Although the human ear may work all the way up to 20,000 Hz, the human voice does not. The top key on a piano is tuned to 4,186.01 Hz. Classical opera is written for sopranos who can reach 1,174.70 Hz. (The precision arises from the piano key to which their voices are matched.) The public telephone system is filtered at about 1,500 Hz. Filtering at 5,000 Hz still captures almost all of the sounds we experience in real-life.

The reason I need to filter the audio signal is to avoid overlap with the carrier, which will be at 15,000 Hz or so. I have marked the carrier frequency with a red arrow in the preceding figure. For reasons I will describe in the next section, we are going to need a good clear separation between the carrier frequency and the audio signals with which we will modulate it.

Amplitude modulation

Of all the modulation schemes, the simplest is the Product Modulator. A Product Modulator takes two signals and, in the time-domain, multiplies them together. If we let the carrier waveform be $c(t)$ and the information waveform be $m(t)$, then the Product Modulator takes their instantaneous values at any instant of time and multiplies them together. The result, which I will call $f(t)$, is:

$$f(t) = m(t) \times c(t) \quad (3)$$

It is usually the case that the frequency of the carrier is much higher than the frequency or frequencies of the information to be conveyed. When that is the case, one says that the information signal modulates the carrier signal, which is why I used an "m" as the symbol for the information waveform $m(t)$. From a mathematical point-of-view, however, the two signals are equal participants in the process. Neither signal dominates the other in any way.

The curious result of using multiplication to modulate is the effect it has on trigonometric waveforms. Product modulation gives rise to frequency components whose frequencies are the difference between the input frequencies, as well as their sum. This is most easily demonstrated assuming the input waveforms are continuous and indefinite in time, for example:

$$c(t) = C\sin(2\pi f_c t) \quad (4A)$$
$$m(t) = M\cos(2\pi f_m t + \varphi) \quad (4B)$$

To permit a bit of generality, I have used a cosine function to represent the information signal. By including an arbitrary phase angle φ, though, it represents in principle any combination of sine and cosine waveforms at the specified frequency f_m. The result of the Product Modulator is the following:

$$f(t) = MC\sin(2\pi f_c t)\cos(2\pi f_m t + \varphi) \quad (5)$$

This can be expanded using a couple of identities for the sum and difference of arguments of the sine function, namely:

$$\begin{array}{lll} \text{Identity \#1:} & \sin(A+B) & = \sin A\cos B + \cos A\sin B \\ \text{Identity \#2:} & \sin(A-B) & = \sin A\cos B - \cos A\sin B \\ \text{Adding:} & \sin(A+B)+\sin(A-B) & = 2\sin A\cos B \end{array} \quad (6)$$

Using Equation (6) to expand Equation (5), we get:

$$\begin{aligned} f(t) &= \tfrac{1}{2}\{MC\sin(2\pi f_c t + 2\pi f_m t + \varphi) + MC\sin(2\pi f_c t - 2\pi f_m t + \varphi)\} \\ &= \tfrac{1}{2}MC\{\sin[2\pi(f_c + f_m)t + \varphi] + \sin[2\pi(f_c - f_m)t - \varphi]\} \end{aligned} \quad (7)$$

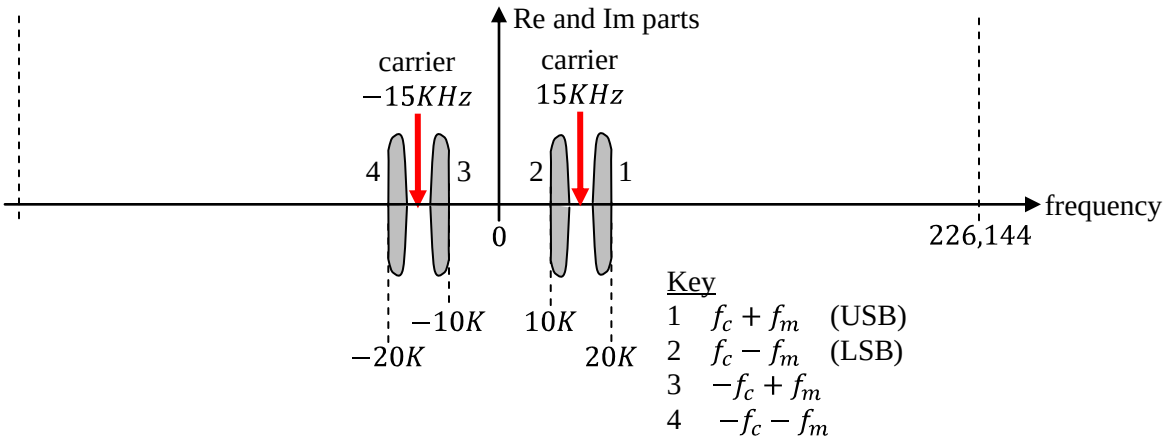
sum of frequencies — difference of frequencies

The expansion in Equation (7) is not unique. An identity of the sine function is that $\sin A = -\sin(-A)$. We could write Equation (7) using negative frequencies, as follows:

$$f(t) = -\tfrac{1}{2}MC\{\sin[2\pi(-f_c - f_m)t - \varphi] + \sin[2\pi(-f_c + f_m)t + \varphi]\} \quad (8)$$

two more frequency sums and differences

For a given information signal frequency f_m, the Product Modulator generates four counterparts. The concept is most easily understood if we show what happens in the frequency domain to the audio signal we looked at in the previous section. After using the Product Modulator to modulate the 15,000 Hz carrier, we will find something like this in the frequency domain:



There will be four copies of the frequency spectrum of the filtered sound track. Two will be "reflections" around the positive carrier frequency (15,000 Hz) and the other two will be reflections around the negative carrier frequency (−15,000 Hz). The upper pair, centered on the positive carrier frequency, is usually called the "passband" to distinguish it from the "baseband" signal.

The envelope on the positive side of the positive carrier frequency, which is identified as the USB in the figure, is called the "upper sideband". Its counterpart on the negative side of the positive carrier frequency is called the "lower sideband". It is identified as LSB in the figure.

First numerical example

In this numerical example, I am going to try to construct something directly in the frequency domain, and then use the inverse DTFT transform to see if it produces the expected (and desired) waveform in the time domain. I am going to try to modulate the audio signal described by the formula in Equation (1).
I will use all four of the "lobes" which appear in the graph above. The frequency spectrum we extracted above for this audio signal was this:

```
K=32:              DTFTRe =  0.05       DTFTIm =  0
K=128:             DTFTRe =  0          DTFTIm = -0.075
K=256:             DTFTRe = -0.1        DTFTIm =  0
K=1,024:           DTFTRe =  0          DTFTIm = -0.0375
K=261,120:         DTFTRe =  0          DTFTIm =  0.0375
K=261,888:         DTFTRe = -0.1        DTFTIm =  0
K=262,016:         DTFTRe =  0          DTFTIm =  0.075
K=262,112:         DTFTRe =  0.05       DTFTIm =  0
```

I will first filter out the high-frequency components, being those for which $k > 5{,}000$. Then, adding in the shift by 15,000 Hz, we will have these four components. For now, I will leave the magnitudes alone.

```
K=15,000 + 32:     DTFTRe =  0.05       DTFTIm =  0
K=15,000 + 128:    DTFTRe =  0          DTFTIm = -0.075
K=15,000 + 256:    DTFTRe = -0.1        DTFTIm =  0
K=15,000 + 1,024:  DTFTRe =  0          DTFTIm = -0.0375
```

These four components fall within the upper sideband lobe (USB) in the above graph. They were all obtained by shifting the basic frequencies upwards by the carrier frequency. In this case, these four components are all there is, so they constitute the entire USB. As I have said, I will include the other three lobes as well. The lower sideband (LSB) will be the following four components:

```
K=15,000 - 32:        DTFTRe =  0.05        DTFTIm = 0
K=15,000 - 128:       DTFTRe =  0           DTFTIm = 0.075
K=15,000 - 256:       DTFTRe = -0.1         DTFTIm = 0
K=15,000 - 1,024:     DTFTRe =  0           DTFTIm = 0.0375
```

Note that the Imaginary parts must be algebraically reversed for the LSB. They correspond to sine terms, which are asymmetric around the carrier frequency. The Real parts correspond to cosine terms which, being symmetric around the carrier frequency, are not negated.

The other two lobes contain the reflections of these eight components into negative frequencies. Since the DTFT_Inverse() procedure operates only on positive values of the index k, we cannot use these negative frequencies *per se*. What we do instead is use the frequencies which are a factor 2π greater, which corresponds to a graphical shift by $N = 262{,}144$ to the right. For example, a component at the negative frequency `-15,000 + 128` has exactly the same effect in a transformation as it does at the positive frequency `262,144 + (-15,000 + 128)`.

Just so there is no doubt about the 16 frequency components I used as inputs to the inverse transfer in this example, I have listed here the code from Form1 which prepares the input vectors FreqRe() and FreqIm() for the inverse transform DTFT_Inverse().

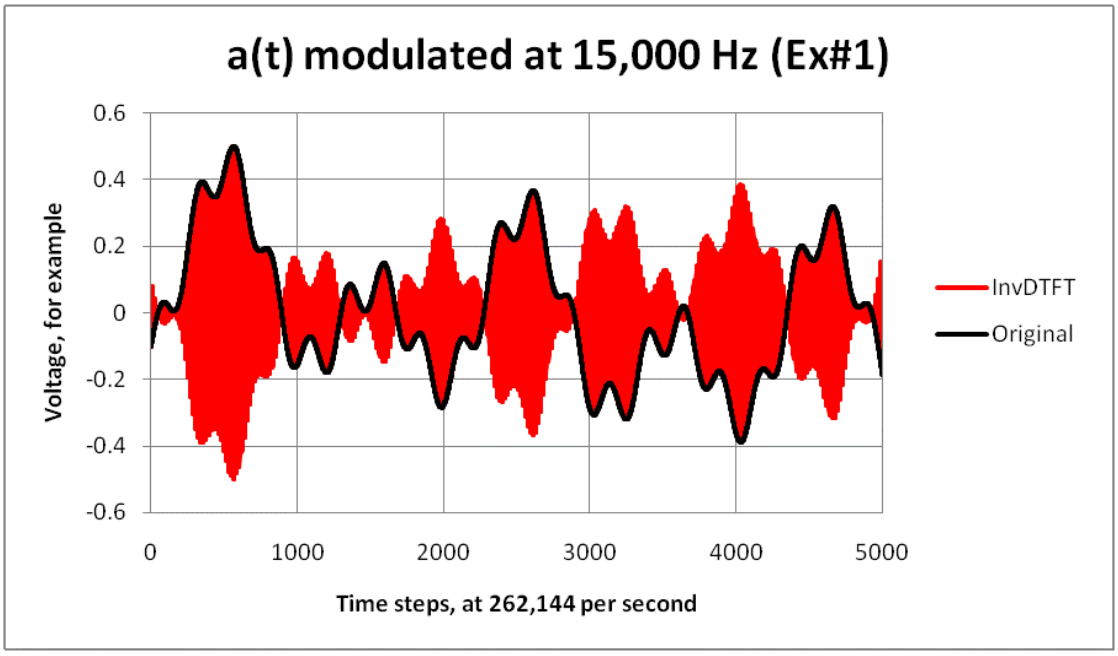
```
NI = CInt(2 ^ 18)
' First numerical example
' Set the frequency domain data directly
For I As Int32 = 0 To (NI - 1) Step 1
    FreqRe(I) = 0
    FreqIm(I) = 0
Next I
' Cosine at 32 Hz, listed from low frequency to high frequency
FreqRe(15000 - 32) = 0.1 / 4
FreqRe(15000 + 32) = 0.1 / 4
FreqRe(NI - (15000 + 32)) = 0.1 / 4
FreqRe(NI - (15000 - 32)) = 0.1 / 4
' Sine at 128 Hz
FreqIm(15000 - 128) = 0.15 / 4
FreqIm(15000 + 128) = -0.15 / 4
FreqIm(NI - (15000 + 128)) = 0.15 / 4
FreqIm(NI - (15000 - 128)) = -0.15 / 4
' Cosine at 256 Hz
FreqRe(15000 - 256) = -0.2 / 4
FreqRe(15000 + 256) = -0.2 / 4
FreqRe(NI - (15000 + 256)) = -0.2 / 4
FreqRe(NI - (15000 - 256)) = -0.2 / 4
' Sine at 1024 Hz
FreqIm(15000 - 1024) = 0.075 / 4
FreqIm(15000 + 1024) = -0.075 / 4
FreqIm(NI - (15000 + 1024)) = 0.075 / 4
FreqIm(NI - (15000 - 1024)) = -0.075 / 4
```

Cosine is an even function, so its algebraic sign does not change with reflections around the carrier frequency or translations upwards or downwards by the sample size. The sine function, on the other

hand, is an odd function, and its algebraic sign is changed (with respect to the signs in the USB) for each reflection and/or translation.

Note one more thing. Since each trigonometric term is now represented by four separate frequency components, I have divided their original amplitudes by a factor of four.

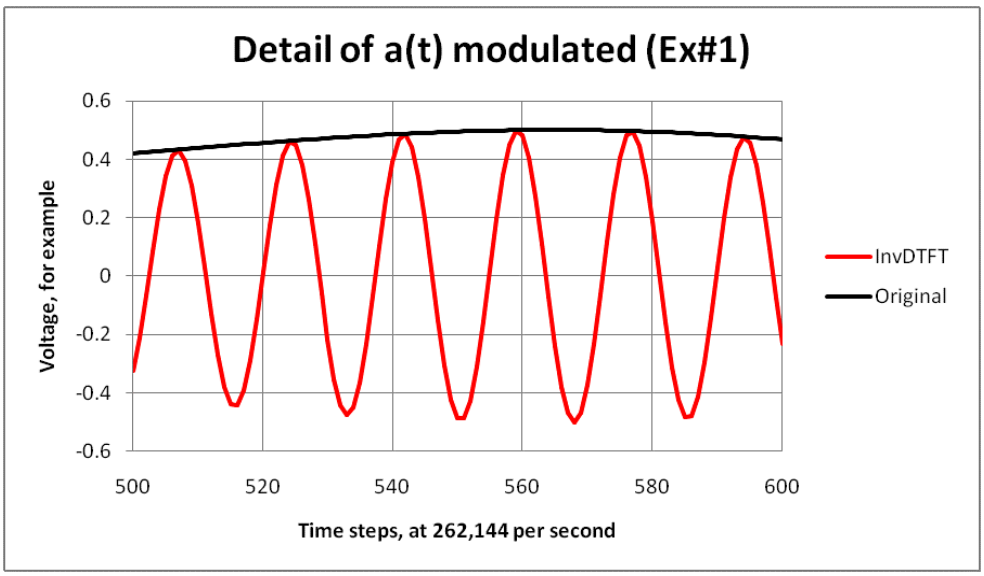
To determine whether this frequency spectrum represents a modulated signal in the time domain, I applied the inverse transform. The following graph shows (in red) the time sequence generated by the inverse transform from the frequency spectrum. For the sake of comparison, the original waveform is also shown (in black).



Only the Real parts of the results of the inverse transform are plotted in this graph. The Imaginary parts are ignored – they are all zero. The carrier frequency is so high that the waveforms of its individual cycles cannot be distinguished in this graph. They merge together into a red mass. What is important is that the envelope of the modulated carrier follows the original audio signal. Although this particular type of modulation is not exactly what I want, this example does show two things: (i) that it is possible to manipulate the Fourier Transform of the audio signal in a way that simulates modulation, and (ii) that it is then possible to synthesize from the manipulated spectrum a real output stream that represents the modulated carrier.

The graph above shows the first $5{,}000$ points in the time sequence. I chose to display the first $5{,}000$ points for a reason. Since a full second of real time corresponds to $262{,}144$ points, these $5{,}000$ points represent approximately $5{,}000/262{,}144 = 0.0191$ seconds. This length of time is quite close to the 20ms of the sample waveform shown in the very first graph in this paper. I wanted to make it clear that the results of the inverse transform can be interpreted in seconds of real time.

The following graph shows a detail of the preceding graph. It shows the period between the 500^{th} and 600^{th} time step. The scale used in this detailed graph is large enough to allow the waveform of the carrier (in red) to be inspected. It is quite a good reproduction of a pure sine wave.



As a reality check, observe that there are about six complete cycles of the carrier in this 100-time step sample. The period of the carrier can be estimated from this wavelength as $(100/6)/262{,}144 = 63.6\mu s$, which corresponds to a frequency of $1/63.6\mu s = 15{,}720$ Hz. The difference from the expected carrier frequency of 15,000 Hz is due to the fact that the span of the graph above is not exactly six complete cycles of the carrier, but a little bit less.

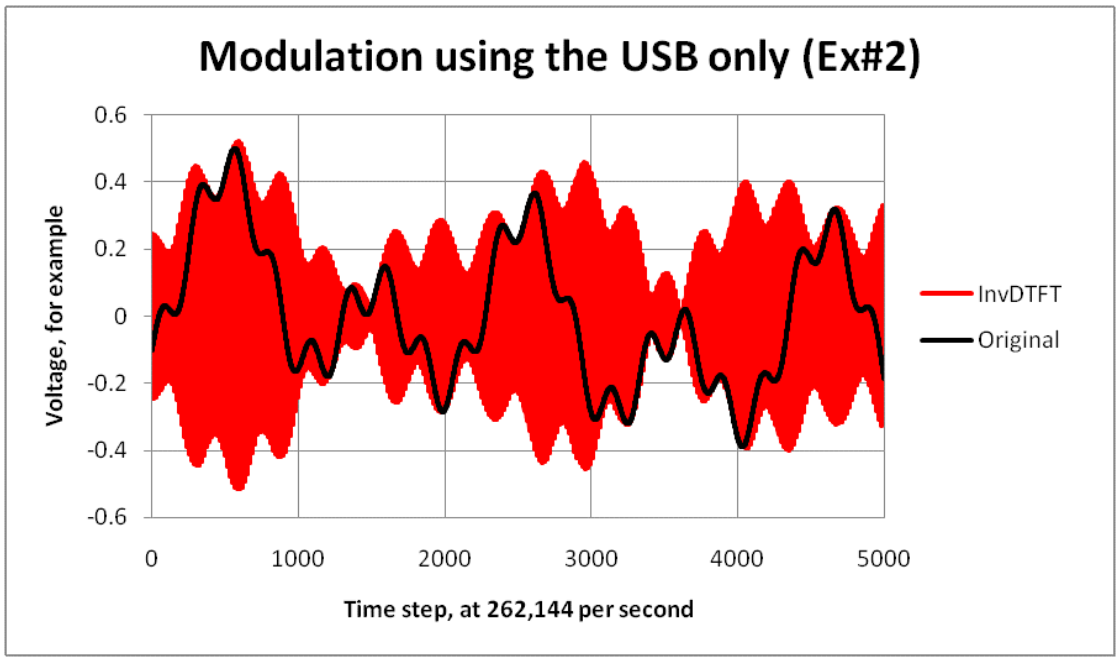
Second numerical example

In this numerical example, I am going to repeat the procedure in the first example but using the frequency components in only two of the four lobes. I will start with the USB and shift it upwards by the carrier frequency. To obtain the other lobe, I will reflect the shifted USB through the $f = 0$ axis, and then translate it upwards by precisely one cycle (equal to the sample size). The frequency spectrum for this example was coded as follows:

```
' Second numerical example - Use the USB and its reflection only
' Set the frequency domain data directly
For I As Int32 = 0 To (NI - 1) Step 1
    FreqRe(I) = 0
    FreqIm(I) = 0
Next I
' Cosine at 32 Hz, listed from low frequency to high frequency
FreqRe(15000 + 32) = 0.1 / 2
FreqRe(NI - (15000 + 32)) = 0.1 / 2
' Sine at 128 Hz
FreqIm(15000 + 128) = -0.15 / 2
FreqIm(NI - (15000 + 128)) = 0.15 / 2
' Cosine at 256 Hz
FreqRe(15000 + 256) = -0.2 / 2
FreqRe(NI - (15000 + 256)) = -0.2 / 2
' Sine at 1024 Hz
FreqIm(15000 + 1024) = -0.075 / 2
FreqIm(NI - (15000 + 1024)) = 0.075 / 2
```

Since each frequency component is mentioned twice in the spectrum, the desired amplitudes for the component waveforms in the time domain were divided by two.

The following graph shows the time sequence generated by the inverse transform. As before, only the Real parts are shown. The Imaginary parts are all zero.



The carrier is modulated, but it is not modulated in the same way as four lobes did it. The envelope of the modulated carrier does not follow the audio signal's waveform. The "information" has not been lost, rather, it cannot be recovered from the modulated carrier as easily as it can be in the first example.

Aside: In both the examples so far, the Imaginary parts produced by the inverse transform were identically equal to zero. That is not happenstance. If a frequency spectrum is constructed with symmetric Real components and asymmetric Imaginary components, then the time sequence generated by the inverse transform will be entirely Real. Symmetric Real components in the frequency domain correspond to pure cosine waveforms in the time domain. Asymmetric Imaginary components in the frequency domain correspond to pure sine waveforms in the time domain. This equivalence, or duality, exists whenever one transforms from one domain to the other, in either direction.

Double Sideband Modulation

So far, the "modulation" technique I have described is simply the use of a Product Modulator, which is not much more than a mathematical multiplication sign. One can modulate more effectively than that. One of the simplest enhancements is to add the carrier signal to the result of the Product Modulator, a technique which is called Double Sideband Modulation. I will use the same symbols and waveforms as above. If $f(t)$ is the result of the Product Modulator, and if $g(t)$ is the result of the Double Sideband Modulator, then the relationships between them are as follows:

$$\begin{aligned}
\text{Carrier:} \quad & c(t) = C\sin(2\pi f_c t) \\
\text{Audio:} \quad & m(t) = M\cos(2\pi f_m t + \varphi) \\
\text{PM:} \quad & f(t) = MC\sin(2\pi f_c t)\cos(2\pi f_m t + \varphi) \\
\text{DSM:} \quad & g(t) = MC\sin(2\pi f_c t)\cos(2\pi f_m t + \varphi) + C\sin(2\pi f_c t) \qquad (9)
\end{aligned}$$

Adding the carrier signal does not look like too big a change, but it does have important consequences. Let's first collect the terms as follows:

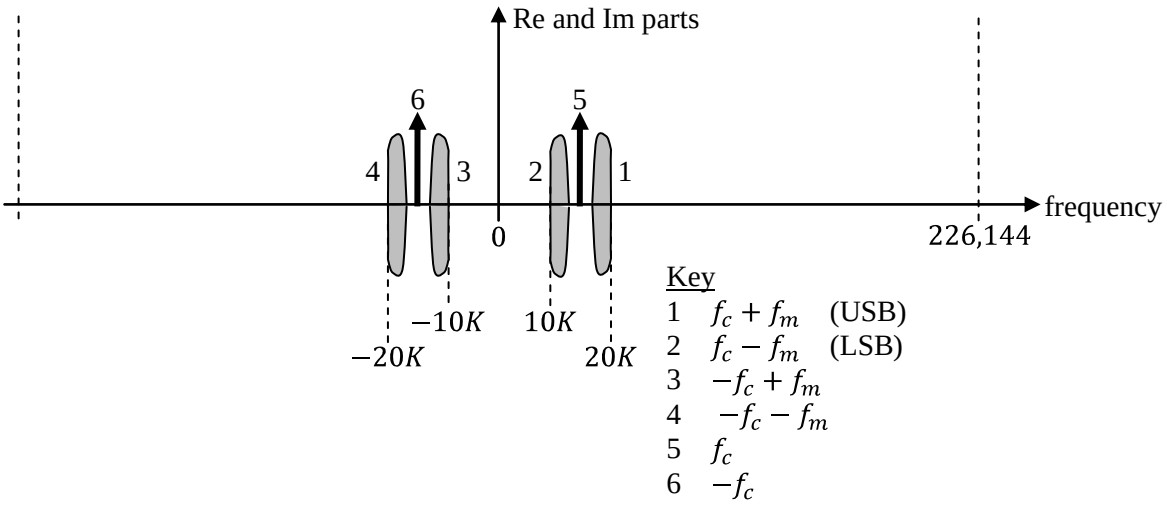
$$g(t) = C[1 + M\cos(2\pi f_m t + \varphi)]\sin(2\pi f_c t) \quad (10)$$

I have not said anything about the amplitudes C and M of the carrier and audio waveforms, respectively, other than to suggest in some of the graphs that they could be measured in Volts. But their relative values are important. To give us some control over the relative amplitudes, I am going to define a number called the modulation index α and to introduce it into the equation as follows[1]:

$$g(t) = C[1 + \alpha M\cos(2\pi f_m t + \varphi)]\sin(2\pi f_c t) \quad (11)$$

A big value of α emphasizes the audio signal's relative importance; a small value reduces it. An optimal value is one that prevents the value in square brackets from ever being algebraically negative. I say "an" optimal value rather than "the" optimal value because a real-world signal will contain many components $M\cos(2\pi f_m t + \varphi)$ at many frequencies f_m, and we want the index α to prevent a negative value at all frequencies, not just individually but for the aggregate signal as well.

What does adding the carrier signal do to the frequency spectrum? The Fourier Transforms, including the DTFT variant, are all "linear". If the input consists of two things added together, the output will consist of their individual transforms, simply added together. Adding a pure sine or cosine carrier waveform to the output of the Product Modulator will result in the transform of the carrier being added to the transform of the Product Modulator. The transform of the carrier is easy – it will be a single frequency component (and its reflections and translations). I can show the revised frequency spectrum using one of the figures from above.



The lobes labeled 1 through 4 are the transform of the modulating signal. The carrier signal is represented by single-frequency spikes labeled 5 and 6. That the two spikes are symmetrically-located means they correspond to a cosine-based carrier. A sine-based carrier would have asymmetric spikes.

What is not shown to scale on the graph are the relative sizes of the lobes and the two carrier spikes. We can control the relative sizes through our choice of the modulation index α.

[1] The modulation index is usually represented by the symbol k, but using that symbol in the midst of so many references to the frequency index would lead to nothing but trouble.

Third numerical example

In this example, I am going to do another direct synthesis, not unlike those in the first two examples. I will use the Double Sideband Modulation technique, and apply it to the sample audio waveform we used before. I am setting out as a goal that the synthesized time sequence have its "information content" at the same magnitude as before. The question is: what magnitude should be used for the carrier?

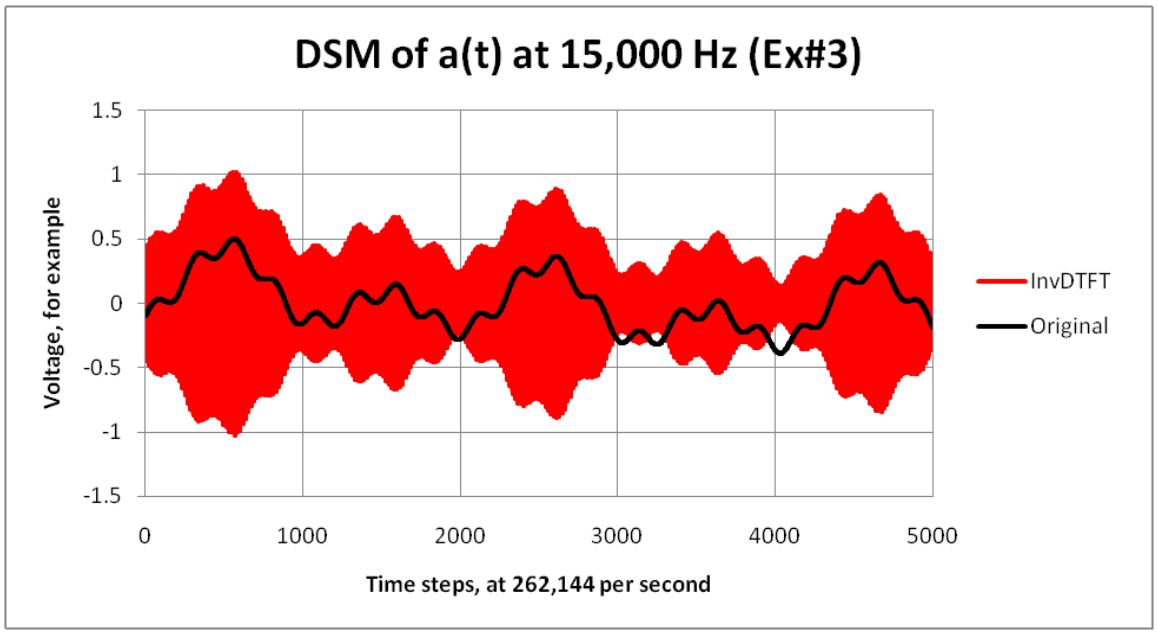
In the sample audio waveform, the cosine term at 32 Hz has an amplitude of 0.1 Volts. If the frequency spectrum which is sent to the inverse DTFT transform uses four lobes, it which this component is represented four times, then its magnitude in the spectrum should be $0.1/4$. The time sequence which the inverse transform produces will then have this cosine component with an amplitude of 0.1 Volts, as desired.

The amplitudes of the four trigonometric terms in the sample audio waveform are 0.1, 0.15, 0.2 and 0.075 Volts, respectively. They are going to be synthesized with those magnitudes. We can calculate the very highest and very lowest voltage that are possible when they are added together. It does not matter which are cosine terms and which are sine terms, or what their phases are, the "worst case" occurs when all four add constructively. The maximum voltage the sample audio signal can attain is $0.1 + 0.15 + 0.2 + 0.075 = 0.525$ Volts. The maximum negative voltage it can reach is -0.525 Volts.

We want to set the amplitude of the carrier (in the time domain) to be this magnitude, or greater. Suppose we decide to set it to 0.525 Volts. In the frequency spectrum, the carrier will be represented twice, so its amplitude in the frequency spectrum needs to be set to $0.525/2$. These are the two specific components we need to add to the frequency spectrum of the first numerical example above to implement the Double Sideband Modulation. Here is the code that sets the spectrum.

```
' Third numerical example
For I As Int32 = 0 To (NI - 1) Step 1
    FreqRe(I) = 0
    FreqIm(I) = 0
Next I
' Cosine at 32 Hz, listed from low frequency to high frequency
FreqRe(15000 - 32) = 0.1 / 4
FreqRe(15000 + 32) = 0.1 / 4
FreqRe(NI - (15000 + 32)) = 0.1 / 4
FreqRe(NI - (15000 - 32)) = 0.1 / 4
' Sine at 128 Hz
FreqIm(15000 - 128) = 0.15 / 4
FreqIm(15000 + 128) = -0.15 / 4
FreqIm(NI - (15000 + 128)) = 0.15 / 4
FreqIm(NI - (15000 - 128)) = -0.15 / 4
' Cosine at 256 Hz
FreqRe(15000 - 256) = -0.2 / 4
FreqRe(15000 + 256) = -0.2 / 4
FreqRe(NI - (15000 + 256)) = -0.2 / 4
FreqRe(NI - (15000 - 256)) = -0.2 / 4
' Sine at 1024 Hz
FreqIm(15000 - 1024) = 0.075 / 4
FreqIm(15000 + 1024) = -0.075 / 4
FreqIm(NI - (15000 + 1024)) = 0.075 / 4
FreqIm(NI - (15000 - 1024)) = -0.075 / 4
' Carrier at positive and reflected frequency
FreqRe(15000) = 0.525 / 2
FreqRe(NI - 15000) = 0.525 / 2
```

And here is the time sequence which the inverse transform generates.

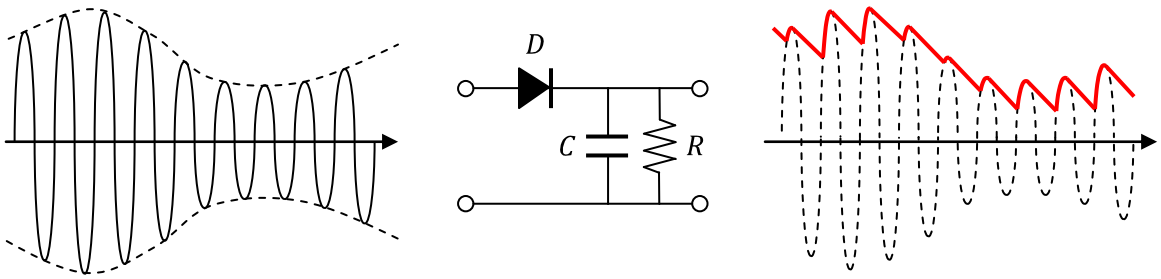


This is perfect for my application. The envelope of the modulated carrier exactly matches the original audio waveform. The envelope "matches" the original, but is displaced upwards by a constant voltage of 0.525 Volts. At no point does the envelope pass below zero. (In fact, during the 20ms or so of the time sequence shown, the four trigonometric waveforms never add constructively, so the envelope never does touch zero.) If the envelope passes below zero, things get inverted and the envelope stops matching the original waveform. That condition is called "overmodulation". It is the business of the modulation index to be big enough to prevent that from happening. (In this example, I did not specify the modulation index directly, but set things up in such a way that $\alpha = 1$ for the sample audio signal taken as a whole.)

There is a reason why it is so handy to have the modulation envelope match the original signal. Actually, there are two reasons.

The first is that some carrier frequencies travel very well. A shout does not get very far, but a radio wave modulated to the frequency content of the shout can travel around the world.

The second reason relates to what happens at the receiving end. We create a time sequence with the expectation that it is going to be sent somewhere, or to something, and the audio signal extracted from what arrives there. Lots of devices – and not just radios and other electromagnetic equipment – are well-fitted to extracting information from an envelope. If a device relies on any phenomenon that responds differently "in one direction" than the other, it can can used for this purpose. Since radio is the most common example, I will briefly describe its basic workings.



The received signal is shown on the left. Assume it is a voltage waveform. It is processed by the circuit shown in the center, which produces the voltage waveform rendered in red on the right. It is diode D that responds different to the flow of current in one direction versus the other. When the incoming voltage level is higher than the voltage drop on capacitor C, the diode allows current to flow through. This current "flows into" the capacitor, where the stored charge increases. As the stored charge increases, the voltage drop over the capacitor also increases. If the value of the capacitor is relatively small, its voltage drop will respond quickly to the incoming current. The voltage drop over the capacitor, which is the output of this little circuit, will "climb up" the slope of the incoming voltage peak.

Once the voltage of the incoming carrier falls below the voltage drop over the capacitor, the diode will be reverse-biased. In that mode, it will not allow any more current to flow into the capacitor. Something else will happen. The charge stored in the capacitor will start to leak out, flowing through resistor R into the common lead. This loss of charge will cause the voltage drop over the capacitor to fall. The combination of the values of the capacitor and resistor determines how fast the capacitor's voltage drop will decrease. The values are chosen so that the voltage falls a bit, but not too much, between successive peaks in the carrier. When the values of the capacitor and resistor are chosen properly, the output of this little circuit will track the peak values of the envelope of the incoming signal. In the figure, it appears as if the output is jagged, not smooth. That's true. But when the carrier frequency is high, these little jig jags are small compared to the signal.

In the early days of radio, before transistors and diodes came into use, the role of the diode was filled by something else. A piece of crystalline rock, galena or some other quartz, could be touched by the whisker of a cat, and the junction would be unidirectional to electric current. These were early types of "crystal radios".

Transmitting without the carrier

When a waveform is transmitted, a certain amount of power must be expended to transmit each frequency component in the spectrum. In the third numerical example, the transmitted waveform included the upper sideband components, the lower sideband components and two instances of the carrier. The frequency composition of the two sidebands is the same – each is simply a reflection of the other. Why use up power transmitting two versions of essentially the same thing? Why transmit the carrier at all? It does not contain any information at all about the signal. In a moment, I will tell you one good reason why you might chose to send the whole kit-and-caboodle[2].

The drive to reduce the power required to transmit a certain amount of information led to various alternative methods of modulation, or of processing the modulated carrier before transmission.

One of the first was to remove the carrier from the waveform before transmission. This was called Double Sideband Modulation - Suppressed Carrier. This is exactly the waveform we dealt with in the first numerical example above. The frequency spectrum we used included all four lobes, shifted in frequency, but no carrier. The time sequence which the inverse transform produced had some of the features of normal AM modulation, but the envelope of the modulated carrier was not the same as the original audio signal. In any event, removing the need to transmit the carrier wave reduces the power required by a factor of two.

The next step, of course, is only to transmit one of the two lobes. This results in USB or LSB transmission, depending on whether the upper or lower sideband, respectively, is the one transmitted. We

[2] In case you were wondering, a caboodle is a big bag or hamper in which you can lug around all of your stuff, which is itself called your kit.

have seen this, too, in the second numerical example. In that example, we used only the upper sideband. The envelope of the modulated carrier looked even less like the original signal than the one in the first example.

The problem is this. If the carrier is not transmitted, the receiver has to generate its own version of the carrier, which it then uses to reconstruct the original signal. This means that a simple "envelope detecting" circuit like the one I described above cannot deal with suppressed carrier or sideband-only transmitters. In this day and age, the price of the extra hardware (or software) needed in the more complex receivers is not that much. It is so low in fact, that one would be very hard pressed to find a manufacturer of radios which only receive the AM band.

Applications which depend on envelope detection are stuck. The carrier has to be transmitted along with both sidebands if that is all the receiver can handle.

The receiver I am interested in is the human ear. It is no diode and can't detect radio waves at any frequency. Under some circumstances, though, it may (and I hope can) demodulate AM signals. That will be the topic of the next paper.

Jim Hawley
© June 2015

If you found this description helpful, please let me know. If you spot any errors or omissions, please send an e-mail. Thank you.