Lab 5

Introduction

The underlying theme of this lab is angle modulation, in particular frequency modulation (FM). The details of both modulation and demodulation are investigated. Particular emphasis will be placed on the modulation process using a voltage controlled oscillator. The spectrum of an FM signal will also be examined. The use of the quadrature detector and/or the phase–locked loop for FM demodulation is also considered.

• Voltage Controlled Oscillator and FM Frequency Deviation Constant

Studying a voltage controlled oscillator (VCO) is a good way to get introduced to frequency modulation concepts. A VCO block diagram representing a single channel of the 33600A is shown in Figure 1.

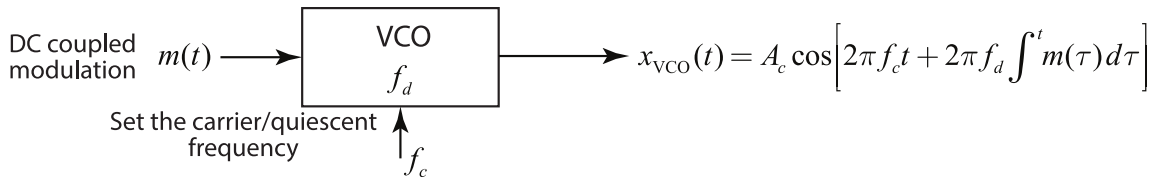


Figure 1: VCO block diagram with a separate input to set the carrier quiescent frequency broken out.

In this figure the traditional VCO model is changed from a single control voltage input consisting of a fixed bias voltage to set the center frequency f_c plus a time varying bias, to be two inputs, a modulation input $m(t)$ and a center frequency input f_c. As you work through this lab at times Channel 1 will be the VCO and at other times Channel 2 will be the VCO.

When the instantaneous frequency of a sinusoidal carrier waveform is proportional to a message, $m(t)$, it can be expressed as

$$f_i(t) = f_c + f_d\, m(t) \tag{1}$$

where f_c is the carrier frequency, $m(t)$ is the modulating signal, and f_d is the frequency deviation constant with units of Hz/volt.

Since frequency is the time derivative of phase, or instantaneous phase is the integral of instantaneous frequency, the FM waveform can be expressed as

$$
\begin{aligned}
x_c(t) &= A_c \cos[\theta_c(t)] \\
&= A_c \cos\left[2\pi\left(f_c t + f_d \int^t m(\tau)\, d\tau\right)\right]
\end{aligned} \tag{2}
$$

When $m(t) = A_{\mathrm{DC}} = a$ constant, the instantaneous frequency becomes

$$
\begin{aligned}
f_i &= \frac{d}{dt}[f_c t + f_d\, A_{\mathrm{DC}} t] \\
&= f_c + f_d\, A_{\mathrm{DC}}
\end{aligned} \tag{3}
$$

In summary, a dc voltage produces a frequency that is offset from the carrier frequency by $f_d\, A_{\mathrm{dc}}$ Hz.

Laboratory Exercises

In this first exercise you will characterize Channel 2 of the Keysight 33600A as a VCO. Configure Channel 1 of the 33600A as a variable voltage source to serve as an input to Channel 2. Note a bench power supply could also be used for this purpose, but the fact that one of the waveforms produced by the 33600A is DC is very convenient. Figure 2 depicts the setup with Channel 1 first set to produce a variable *DC* waveform. Second, Channel 2 is setup to produce a 50 MHz sinusoid at –15 dBm. It becomes a VCO by setting the generator for FM modulation using external modulation via the rear panel connector. On the front panel of the 33600A set the modulation mode (*Mod* button) to FM and then set *Freq Dev = 5* MHz and finally configure *Source External* as *Mod In = 5V*. When in external FM mode the deviation value is no longer the peak deviation, instead it is a combination of *Freq Dev*, *Mod IN*, and the amplitude of external input voltage that controls the deviation of the Channel 2 source. The applied voltage moves the frequency of the source above and below the nominal center frequency of 50 MHz. Reference all data that you take in the lab to the rear panel input, hereafter referred to as v_{mod}.

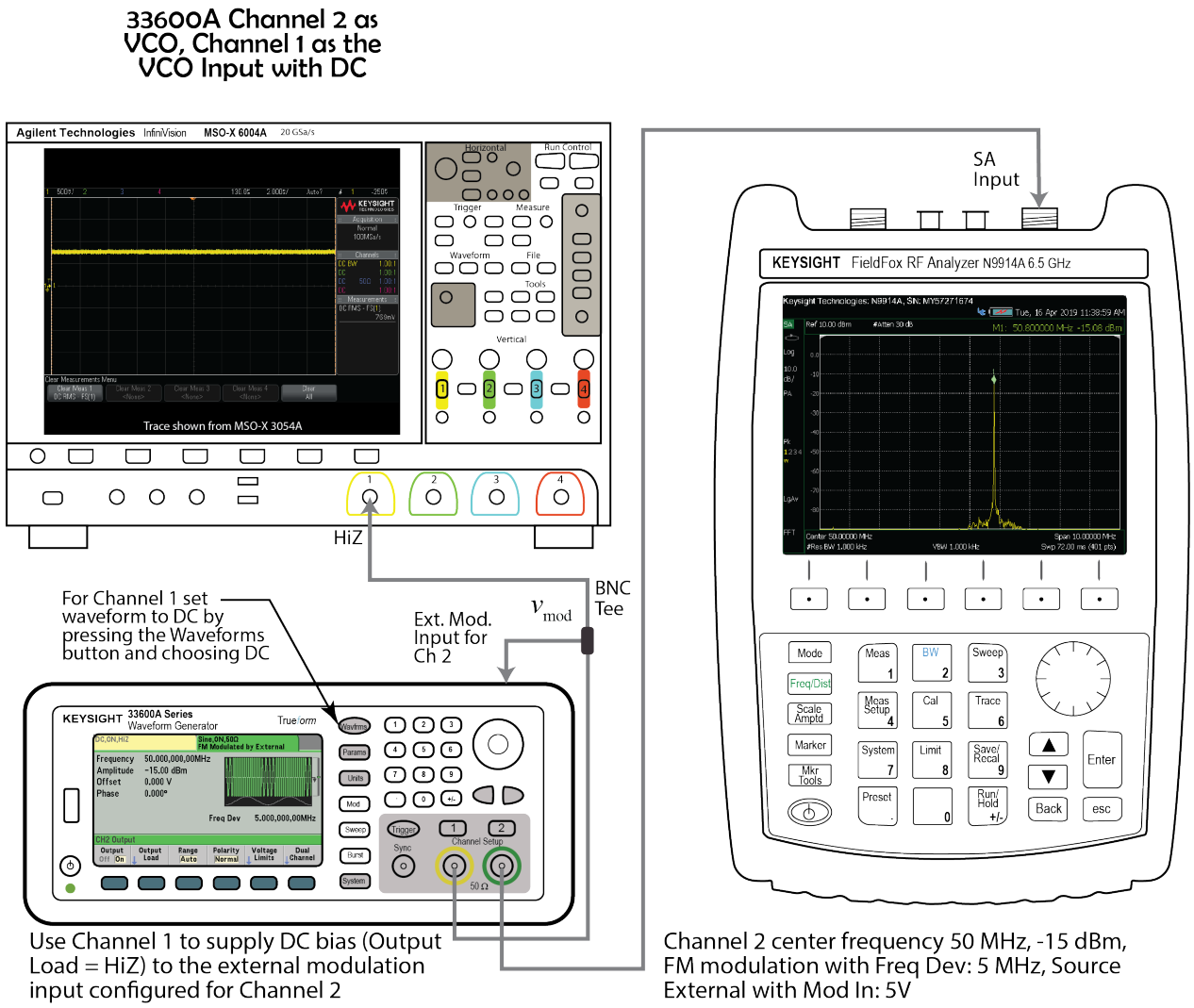


Figure 2: Configuring Channel 2 of the Keysight 33600A as a VCO and using Channel 1 as an adjustable DC source to then obtain the a VCO *tuning curve*.

In Figure 2 I also show the scope attached to v_{mod} so you can check the DC voltage and later the signal applied to the VCO input. Note for the DC measurements you may want to use the bench DVM to measure the control voltage. To make the voltage display on Channel 1 of the 33600A be sure to set the output load to HiZ.

Enter the frequency versus voltage data you collect into a Python *list* for plotting in the Jupyter Notebook Sample. The applied voltage range of interest is $-2 < A_{\text{DC}} < 2$ volts as applied at the v_{mod} input, with the carrier center frequency is set to 50 MHz. Take data points about 0.25 volts apart. Clearly establish that you have a linear plot.. With the data in the Jupyter notebook you can easily *fit* a line to the data using `polyfit()` and `polyval()`, create a plot of the frequency (use the Keysight N9914A spectrum analyzer marker) versus voltage and the slope on Hz/V or MHz/V. A Python sample using simulated measurement data is contained in the sample notebook and repeated below:

```
v_sim = arange(-2,2+.25,.25) # in Volts
f_sim = 50+v_sim*3 + 0.1*randn(len(v_sim)) # in MHz
```

```
v_sim
```

```
array([-2.  , -1.75, -1.5 , -1.25, -1.  , -0.75, -0.5 , -0.25,  0.  ,
        0.25,  0.5 ,  0.75,  1.  ,  1.25,  1.5 ,  1.75,  2.  ])
```

```
f_sim
```

```
array([44.14764527, 44.83113494, 45.45664035, 46.2873477 , 47.04520317,
       47.62064896, 48.45482124, 49.22576942, 50.10098242, 50.56772122,
       51.49870311, 52.42327137, 52.94645917, 53.70339293, 54.5731309 ,
       55.23217258, 55.90645751])
```

```
# Save data in the Python lists:
#v_data = [-2.0, -1.75, -1.5, -1.25, ...] # voltage data in Volts
#f_data = [44.147, 44.831, 45.457, ...] # Frequency data in MHz
# Here we load the simulated data from above as a placeholder
v_data = v_sim
f_data = f_sim
p = polyfit(v_data,f_data,1)
f_data_fit = polyval(p,v_data)
print('Slope = %4.2f MHz/V, Offset = %4.2f MHz' % (p[0],p[1]))
```

```
Slope = 2.98 MHz/V, Offset = 50.00 MHz
```

The polynomial fit to the data is held in the array `p`, which is a first degree polynomial relating to the model

$$\hat{f}_{\text{VCO}} = f_d \cdot v_{\text{mod}} + f_c \tag{4}$$
$$= p[0]v_{\text{mod}} + p[1] \tag{5}$$

```
plot(v_data,f_data,'r.')
plot(v_data,f_data_fit)
xlabel(r'VCO Input Voltage (V)')
ylabel(r'VCO Output Frequency (MHz)')
title(r'VCO Tuning Curve with $f_d = 2.9814$ MHz/V, $f_c = 50$ MHz')
legend((r'Raw Data',r'Fit: $f = v_{in}\cdot f_d + f_c$'))
grid();
```

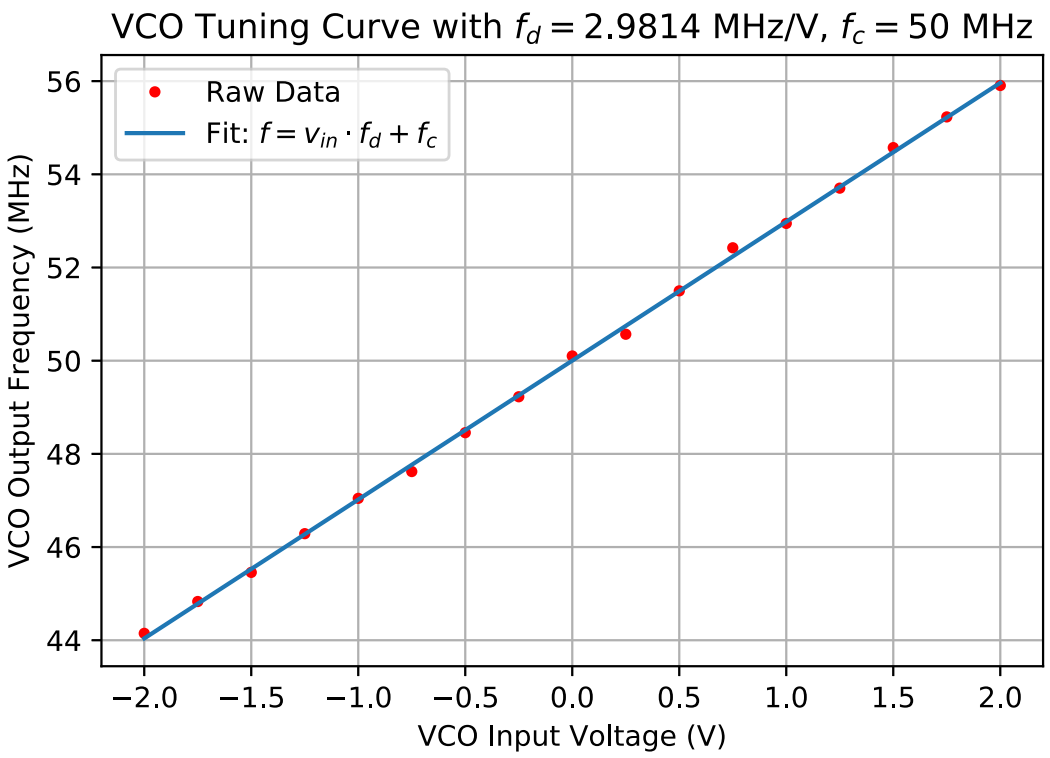


Figure 3: Sample VCO frequency versus voltage curve fit from simulated measurement data.

Note: When the Keysight 33600A is configured as a VCO, it will have essentially a *perfect* or linear tuning curve. The theoretical values of f_d and f_c are formed from the generator parameters `Freq Dev`, `Mod In` value 1V or 5V, and the center frequency `fc`. Can you deduce the theoretical expression?

• FM Modulation Index – β

If we let $m(t) = A_m \sin(2\pi f_m t)$ in the earlier equation given for $x_c(t)$, then

$$\begin{aligned} x_c(t) &= A_c \cos\left[2\pi f_c t + \frac{A_m f_d}{f_m}\cos(2\pi f_m t)\right] \\ &= A_c \cos\left[2\pi f_c t + \beta \cos(2\pi f_m t)\right]. \end{aligned} \tag{6}$$

In the above equation, $A_m\, f_d/f_m = \beta$ is referred to as the modulation index. It is an important parameter for characterizing the FM signal. Note that β increases with increasing amplitude of the modulating signal, and decreases with increasing frequency, f_m.

The equations given here represent sinusoidal frequency modulation, or tone modulation with frequency f_m. The equation is periodic and has a Fourier series. The series is rather complicated, however, and has coefficients which are functions of β instead of being fixed constants as in the case, for example, of a square wave. The Fourier series representation of sinusoidal frequency modulation is given by

$$\begin{aligned} x_c(t) = {} & A_c J_0(\beta)\, \cos\, \omega_c t \\ & + A_c J_1(\beta)[\cos(\omega_c + \omega_m)t - \cos(\omega_c - \omega_m)t] \\ & + A_c J_2(\beta)[\cos(\omega_c + 2\omega_m)t + \cos(\omega_c - 2\omega_m)t] \\ & + A_c J_3(\beta)[\cos(\omega_c + 3\omega_m)t \cos(\omega_c - 3\omega_m)t]\ \ldots \end{aligned} \tag{7}$$

This equation shows that for tone modulation the spectrum consists, theoretically at least, of sidebands spaced f_m apart out to infinite frequency. The function $J_n(\beta)$ is called the n–th order Bessel function, of the first kind, with argument β. The Bessel functions are graphed and tabulated in most math handbooks, and the value of the function for any given argument, β, can easily be found.

For $\beta = 0,\ J_0(\beta) = 1$ and all of the remaining coefficients are zero. This says that the only term present in the series in the unmodulated case is the carrier frequency, as it should be. For β very small the first pair of sidebands will come and go according to the value of their corresponding coefficients in the equation above.

Laboratory Exercises

1. Connect the output of the Keysight 33600A function generator Channel 2 to the Keysight N9914A spectrum analyzer. Continue to use a carrier frequency of 50 MHz and center the carrier signal on the analyzer screen. Set the frequency span to about 200 kHz. The general setup is shown in Figure 4 as a continuation of Figure 2.

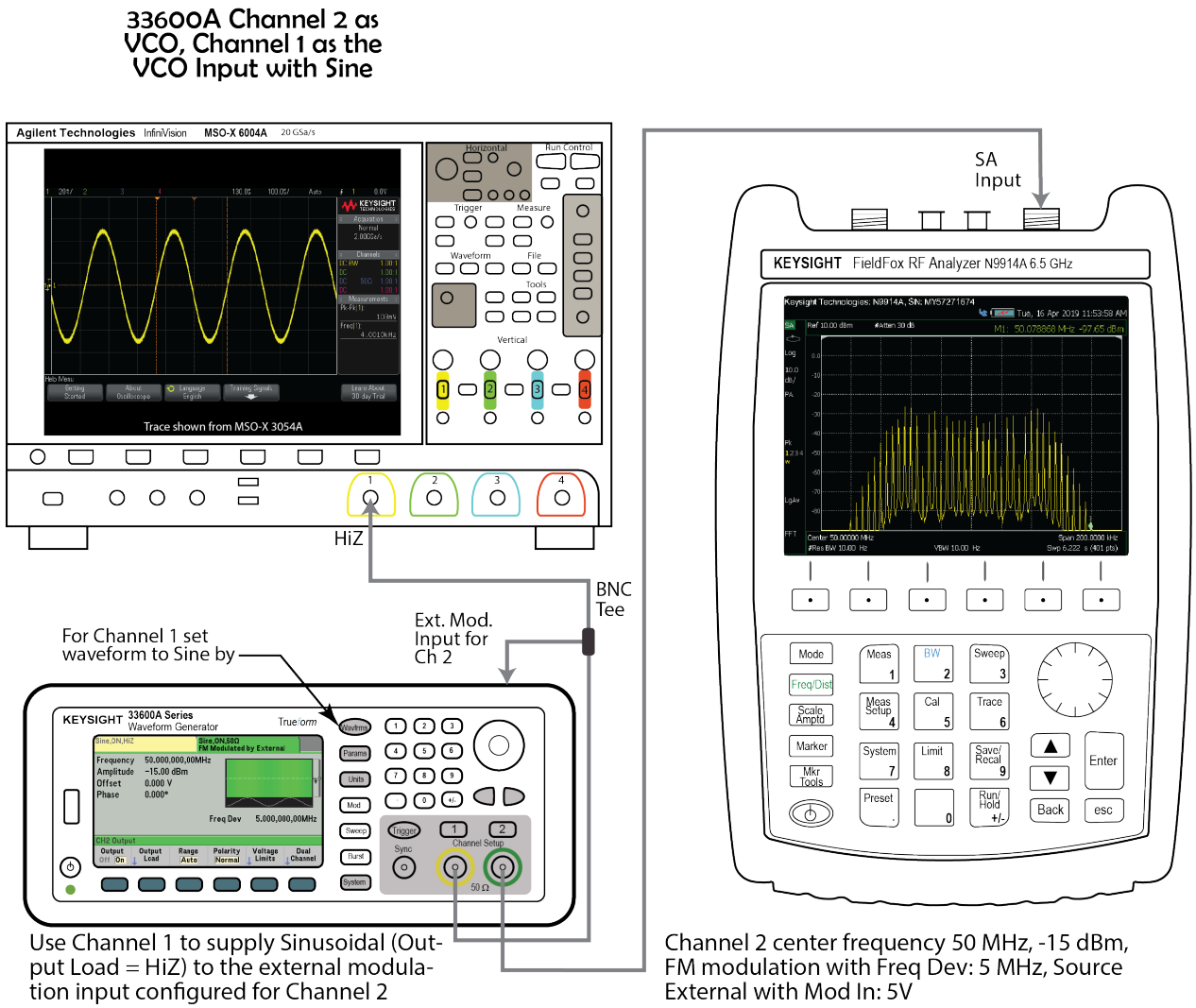


Figure 4: Sinusoidal FM using the combination of the 33600A Channel 2 as a VCO and the 33600A Channel 1 as a modulation source.

2. Set the v_{mod} input to Channel 2 to a frequency of 10 kHz. Start with the modulating signal at zero amplitude and slowly increase the level. Watch the Channel 1 output on a scope to observe the v_{mod} waveform and also observe the frequency spectrum of the Channel 2 output. As the input level is increased, one pair of sidebands and then a second and a third pair will appear. Reduce the input until only the first pair is present. On a dB scale we will call this the 10% point or the when the second pair of sidebands is down 20dB relative to the first pair. Take data to calculate the value of β at this point. The condition where only one sideband of the modulating frequencies is present is known as *narrow–band* FM. The maximum modulation index for sinusoidal narrow–band FM is usually assumed to be around $\beta = 0.5$ or less. Would you agree?

Note: You will notice in the above that the high VCO sensitivity achieving *narrow band* FM requires the modulation amplitude to be very small, in fact at the lower limit of the Channel 1 generator amplitude. The noise being picked up on the VCO input is also noticeable as the sidebands are *quivering*. In the next part internal modulation will be used and noise problem will go away. As a rule wideband tuning range VCOs are very susceptible to noise and interference ingress on the tuning line.

3. In the remaining parts to this exercise it will be more convenient to use internal FM modulation, so repeat part 1 using internal modulation by a 10 kHz sinusoid. You no longer have the ability to set the amplitude of the modulation, instead $\beta = \text{Freq Dev}/f_m$. Verify that this is the case as you will now have to adjust *Freq Dev* on the generator to get the equivalent modulation index of part 1. It should be that $A_m f_d$ in the β expression is replaced by the generator deviation constant *Freq Dev*. Do you agree?

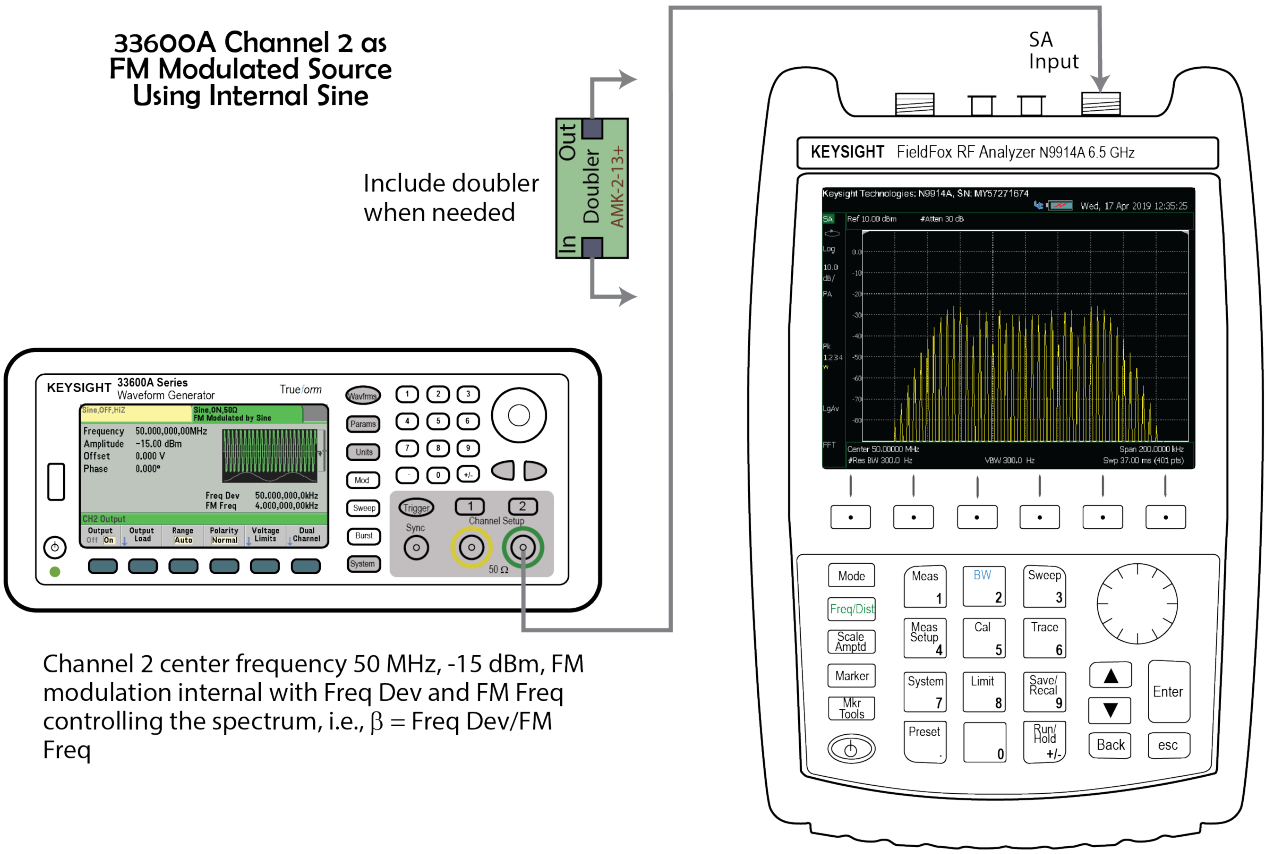


Figure 5: Internal FM modulation used in the remainder of this portion of the lab and beyond.

4. Increase *Freq Dev* slowly, observing the FM spectrum and noting the appearance of several additional sidebands on the analyzer screen. Use the 10 dB per division vertical scale. At some point, after three or four sidebands have appeared, the carrier frequency line will begin to decrease in amplitude. Adjust the modulation amplitude until the carrier term is gone. Calculate β for this condition and compare your experimental value with theory. Note that the zeros of the $J_n(\beta)$ Bessel functions are tabulated in mathematical handbooks and communication theory texts such as in [1]. A short table of zeros is given in Table 1.

Table 1: A short table of Bessel function zeros.

	$J_n(x) = 0$				
$n = 0$	2.4048	5.5201	8.6537	11.7915	14.9309
$n = 1$	3.8317	7.0156	10.1735	13.3237	16.4706
$n = 2$	5.1356	8.4172	11.6198	14.7960	17.9598
$n = 3$	6.3802	9.7610	13.0152	16.2235	19.4094
$n = 4$	7.5883	11.0647	14.3725	17.6160	20.8269
$n = 5$	8.7715	12.3386	15.7002	18.9801	22.2178
$n = 6$	9.9361	13.5893	17.0038	20.3208	23.5861
$n = 7$	11.0864	14.8213	18.2876	21.6415	24.9349
$n = 8$	12.2251	16.0378	19.5545	22.9452	26.2668

5. Now use one–half the modulation frequency of that used above and, by adjusting the input signal amplitude, duplicate the conditions observed in 2. How do the amplitudes compare for these two cases? Does this agree with the equation for β?
6. Return to the frequency and external amplitude of 2. Increase the input signal level slowly and note the signal amplitudes for which $J_1(\beta),\ J_2(\beta)$, etc., go to zero. At what value of β does the carrier term go to zero a second time? Compare your results with theory.

Simulation of the measurements you have been taking is dicussed in the Jupyter notebook sample. In Chapter 4 of [1] you learn that a frequency modulated carrier takes the form

$$x_c(t) = A_c \cos\left[2\pi f_c t + 2\pi f_d \int^t m(\lambda)\, d\lambda\right] \tag{8}$$

where A_c is the carrier amplitude, $m(t)$ the message signal, here a NRZ data stream, and f_d is the modulator deviation constant having units of Hz per unit of $m(t)$. In a discrete–time implementation and with the carrier at f_0, *complex baseband* FM takes the form

$$x_c[n] = A_c \exp\left[2\pi f_d \sum_{k=0}^{n} m[k]\frac{1}{f_s}\right], \tag{9}$$

where the integration is replaced by the running sum, `cumsum` in Python's numpy. In the plot below uses a log scale and is calibrated in dBm. Since the signal is a complex sinusoid you can view the spectrum as a single–sided spectrum for the case of a real sinusoid with the plot being made with $f \rightarrow (f - f_c)$.

Two related simulation examples that obtain the spectrum dBm as an equivalent to a single–sided spectrum are given below. The first is for a carrier power of –30 dBm with $\beta = 0.1$ and the second is for –30 dBm with $\beta = 3.8317$. Use these examples as a reference in creating simulation models for the measurements you have already taken and those that follow.

Narrow Bandwidth FM β is Small

```
fs = 100000
fc = 0
N_samp = 100000
t = arange(N_samp)/fs
Pc_dBm = -30
fd = 1000 #Hz/v
Am = 0.1
fm = 1000
m = Am*cos(2*pi*fm*t) # sinusoid
#m = Am*sign(cos(2*pi*fm*t)) # square wave
xc = sqrt(10**((Pc_dBm-
30)/10))*exp(1j*2*pi*fd*cumsum(m)/fs)*exp(1j*2*pi*fc*t)
f, Sx = ss.simple_SA(xc,N_samp,2**14,fs)
plot(f,10*log10(Sx)+30)
#psd(xc,2**12,48000);
title(r'Complex Baseband FM Spectrum - $\beta = 0.1$')
ylabel(r'PSD (dBm)')
xlabel(r'Frequency (Hz)')
xlim([-8000,8000])
ylim([-80,-20]);
grid();
```

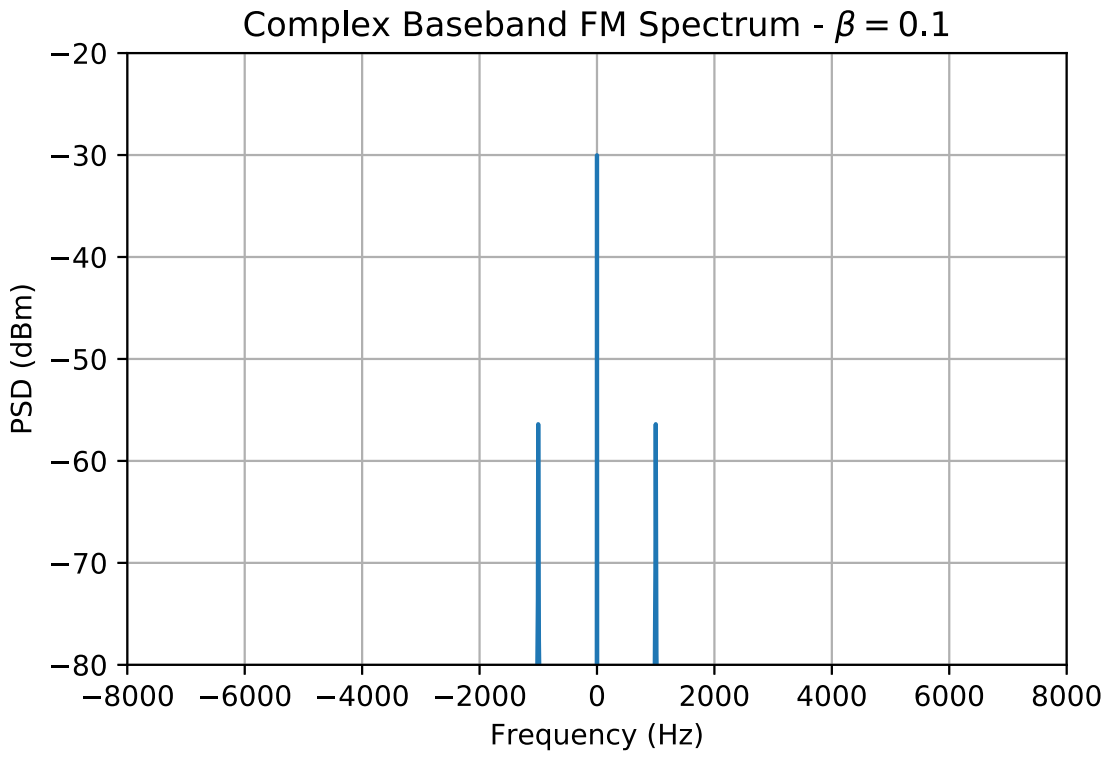


Figure 6: Python simulated single-sided spectrum for β small (narrowband FM).

- Medium Bandwidth $\beta = 3.8317$ To Eliminate the First Sideband Pair

```
fs = 100000
fc = 0
N_samp = 100000
t = arange(N_samp)/fs
Pc_dBm = -30
fd = 1000 #Hz/v
Am = 3.8317
fm = 1000
m = Am*cos(2*pi*fm*t) # sinusoid
#m = Am*sign(cos(2*pi*fm*t)) # square wave
xc = sqrt(10**((Pc_dBm-
30)/10))*exp(1j*2*pi*fd*cumsum(m)/fs)*exp(1j*2*pi*fc*t)
f, Sx = ss.simple_sa(xc,N_samp,2**14,fs)
plot(f,10*log10(Sx)+30)
#psd(xc,2**12,48000);
title(r'Complex Baseband FM Spectrum - $\beta = 3.8317$')
ylabel(r'PSD (dBm)')
xlabel(r'Frequency (Hz)')
xlim([-8000,8000])
ylim([-80,-20]);
grid();
```

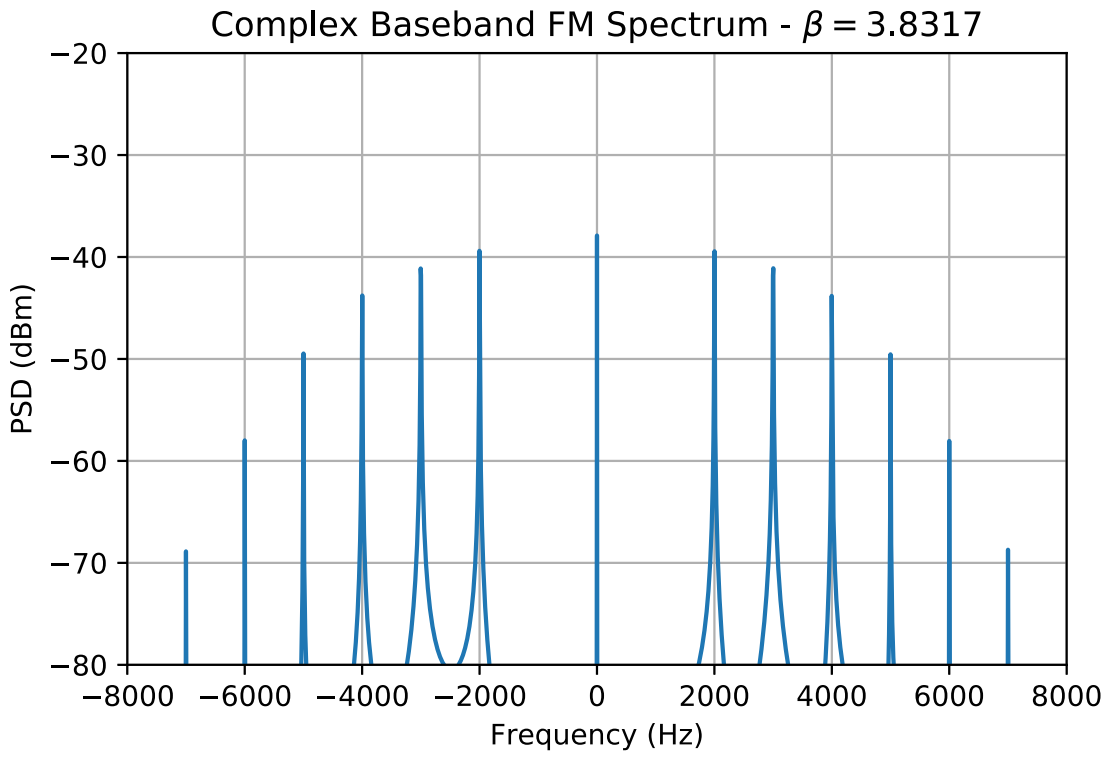


Figure 7: Python simulated single-sided spectrum for $\beta = 3.8317$, which makes the first sideband pair null to zero..

7. The parameter β is sometimes written as $\beta = \Delta f / f_m$. That is, $f_d A_m = \Delta f$, and Δf is called the frequency deviation. It is the maximum instantaneous frequency or the peak swing in carrier frequency from its unmodulated value. The bandwidth occupied by an FM signal is related to the peak deviation, but not in a rigorous fashion. Set the peak deviation, $f_d A_m = \Delta f$, at a fixed value that gives sidebands out to about 100 kHz (*Freq Dev* = 75 kHz works well with $f_m = 10$ kHz) on each side of the carrier and use about 50 kHz per division on the analyzer, which is a span of 500 kHz, here centered at 50 MHz. Observe and comment on the spectrum you see.
8. Decrease the modulating frequency without changing Δf, and notice the effect on the spectrum. Measure the bandwidth at several different modulating frequencies, e.g., 10 kHz, 5 kHz, and 1 kHz. If you are using the 10dB per division vertical scale you can define spectral bandwidth in terms of the band of frequencies for which the spectrum is say 10 to 20 dB down from its peak value. Calculate the relationship between Δf and bandwidth for each. Carson's rule for FM by a sinusoidal signal of frequency f_m states that the 98% containment bandwidth is approximately

$$\mathrm{BW} = 2\, f_m(\beta + 1) = 2(\Delta f + f_m) \tag{10}$$

   Would you agree with this approximation?

9. As a cross-check export corresponding `.csv` files for the three f_m and Δf combinations. The Jupyter notebook sample contains a helper class for processing the `.csv` files into spectrum plots and calculates the 98% fractional bandwidth as obtained from the actual measurements.

Example Import

See the Lab 5 Jupyter notebook sample for details on the `FieldFox_capture` class. Here the capture is at $f_c = 30$ MHz, –10 dBm, $f_m = 10$ kHz and *Freq Dev* = 50 kHz.

```
# Create an instance of the capture class for a given csv file:
sin_FM = FieldFox_capture('SIN_10K50K30Mfc.csv')
```

```
# Check the total power of the capture:
sin_FM.total_power_dBm()
```

```
Total power in capture = -10.17 dBm
```

```
# Calculate the 98% bandwidth:
sin_FM.bandwidth(fc=30e6)
```

```
98 percent BW = 0.121 MHz
```

```
# Carson's rule check:
print('BW = %4.3f MHz' % (2*(.05 + 0.01),))
```

```
BW = 0.120 MHz
```

```
sin_FM.spectrum_plot(30e6)
ylim([-80,-10]);
```

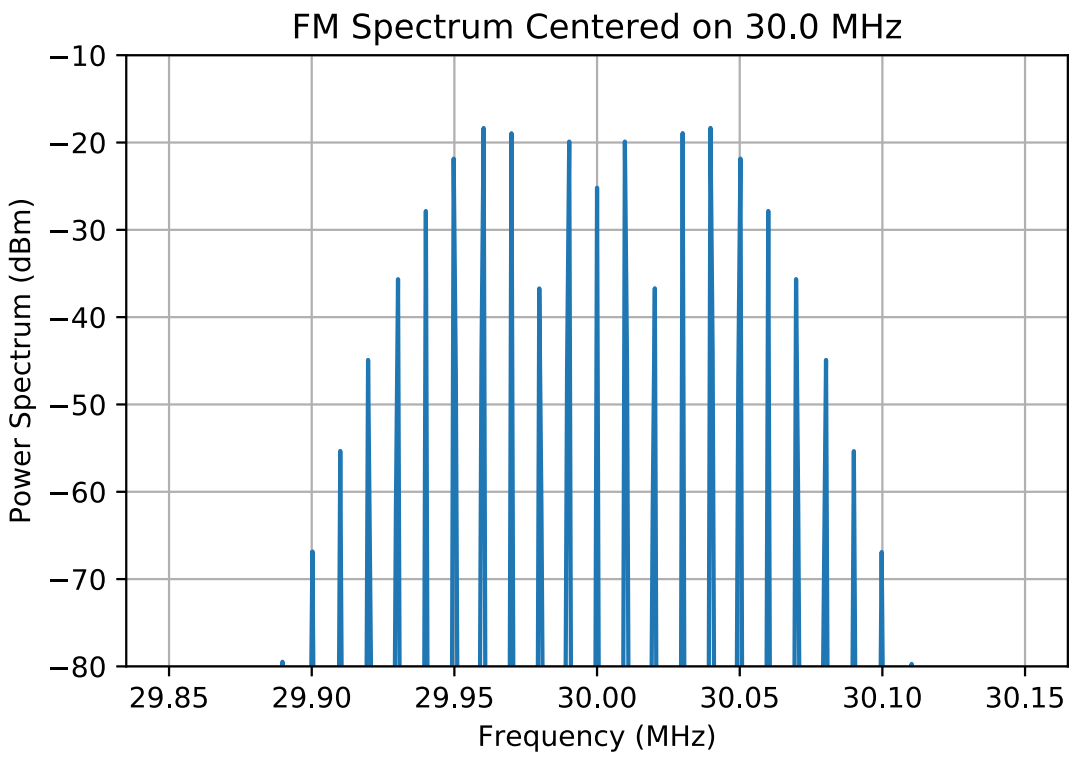


Figure 8: Power spectrum plot from imported file `SIN_10K50K30Mfc.csv`.

FM with Other than Sinusoidal Signals

The Fourier series (or transform) for an FM waveform is mathematically tractable for only a few special cases, the sinusoidal case being one of them. For signals which are more complicated, the detailed structure of the spectrum cannot be analyzed. Only a few rules relating modulating frequency, bandwidth, and peak deviation can be used to describe the frequency domain representation of FM for the general case.

Suppose $m(t)$ is a zero average value square wave of amplitude$\pm A_m$. Then the instantaneous frequency of the resulting FM signal jumps from $(f_c - f_d A_m)$ to $(f_c + f_d A_m)$ or from $(f_c - \Delta f)$ to $(f_c + \Delta f)$. Mathematically we can write this as

$$x_c(t) = \sum_{n=-\infty}^{\infty} p(t - nT_m) \tag{11}$$

where

$$p(t) = A_c \left\{ \Pi\left(\frac{t - T_m/4}{T_m/2}\right) \cos[2\pi(f_c + \Delta f)t] + \Pi\left(\frac{t - 3T_m/4}{T_m/2}\right) \cos[2\pi(f_c - \Delta f)t] \right\} \tag{12}$$

For this special case the power spectral density of $x_c(t)$ is relatively easy to obtain. Clearly $S_x(f)$ will consist of delta functions spaced at multiples of $1/T_m$. The envelope of $S_x(f)$ is proportional to $|P(f)|^2$. This model will be further developed in the Jupyter notebook sample sometime in the future (not used Spring 2019).

Laboratory Exercises

1. Using a setup similar to Figure 5 set the Channel 2 carrier to 50 MHz at –15 dBm and configure the internal modulation to be a square wave at $f_m = 10$ kHz.
2. Observe the spectrum on the analyzer as *Freq Dev* $= \Delta f$ is increased over the values of 20, 100, 500, and 1000 kHz. Center the spectrum analyzer on 50 MHz and set the span as needed to a clear view of what I call the *suspension bridge* spectrum.Compare your measured results to a Python simulation. A sample result, taken from the Jupyter notebook sample is shown below:

```
fs = 100000
fc = 0
N_samp = 500000
Nfft = 2**15
t = arange(N_samp)/fs
Pc_dBm = -30
```

```
fd = 1000 #Hz/v
Am = 20
fm = 500
#m = Am*cos(2*pi*fm*t) # sinusoid
m = Am*sign(cos(2*pi*fm*t)) # square wave
xc = sqrt(10**((Pc_dBm-
30)/10))*exp(1j*2*pi*fd*cumsum(m)/fs)*exp(1j*2*pi*fc*t)
f, Sx = ss.simple_sa(xc,N_samp,Nfft,fs)
plot(f/1e3,10*log10(Sx)+30)
#psd(xc,2**12,48000);
title(r'Complex Baseband FM Spectrum - Square wave')
ylabel(r'PSD (dBm)')
xlabel(r'Frequency (kHz)')
#xlim([-8000,8000])
ylim([-80,-40]);
grid();
```

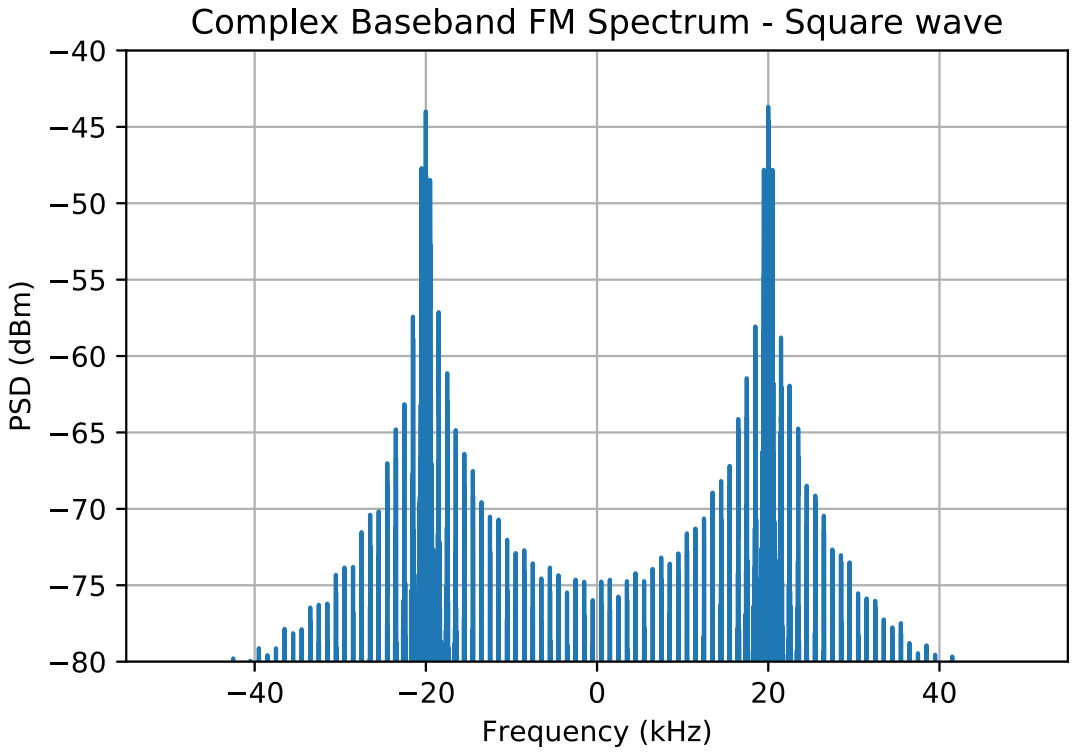


Figure 9: Square wave spectrum (*suspension bridge*) simulation where the frequency axis is shifted about zero.

Note as the peak deviation increases more and more sidebands need to be plotted and that will require `Nfft` to be an even high power of two. A higher power of two in turn means that the number of simulation sample points, `N_samp`, will have to be increased.

3. Does Carson's rule seem to hold for this signal if you define BW as the –15 dB spectrum width below the peaks?

4. More calculate the BW by exporting the four `.csv` spectrum captures to the Jupyter notebook and making use of the `FieldFox_capture` class as you did for the sinusoidal FM case.

Inserting the RF Board Doubler

Here you verify the behavior FM following the RF board doubler. From Chapter 4 of [1] you know that the theoretical action of a frequency doubler is to place a two in the argument of an angle modulated carrier, i.e.,

$$\text{Doubler}\{x_c(t)\} = x_{2c}(t) = A_{2c}\cos\left[2\left(2\pi f_c t + 2\pi f_d \int^t m(\lambda)\,d\lambda\right)\right] \tag{13}$$

The real doubler also outputs a weak fundamental term and weak higher–order harmonic terms. In this exercise the focus is on the second–harmonic carrier output by the doubler.

Laboratory Exercises

Verify the doubling math to be a reality by configuring a test setup similar to that shown in Figure 5, except now you will be inserting the RF Board doubler.

1. Set $f_c = 35$ MHz, *Freq Dev* = 100 kHz and choose a convenient value for f_m to validate the math model.
2. Observe the doubler input spectrum at 35 MHz and note the spacing of the spectral lines is indeed $\pm n f_m$, $n = 1, 2, \ldots$. What is the 98% containment bandwidth of the signal?
3. Observe the doubler output spectrum at 70 MHz and note that the spacing of the spectral lines is still $\pm n f_m$, $n = 1, 2, \ldots$. Is this correct? What has changed? Explain. What is the 98% containment bandwidth of the signal.

FM Demodulation

To recover the information contained in an FM signal requires obtaining the signals instantaneous frequency. For the signal

$$x_c(t) = A_c\,\cos[\omega_c t + \phi(t)] = A_c\,\cos\left[\omega_c t + k_f \int^t m(\alpha)\,d\alpha\right] \tag{14}$$

the instantaneous radian frequency is

$$\omega_i(t) = \omega_c + \frac{d\phi(t)}{dt} = \omega_c + k_f\,m(t) \tag{15}$$

where $k_f = 2\pi f_d$. An ideal discriminator produces output

$$y_D(t) = \frac{1}{2\pi} K_D \frac{d\phi(t)}{dt} = K_D f_d m(t). \tag{16}$$

Practical implementation of the ideal FM discriminator can be done using analog circuit design or using digital signal processing. In this part of the lab we will consider the use of a slope detector and an analog phase–locked loop (PLL) with sinusoidal phase detector.

Slope Detection

Slope detection of FM makes use of the gain slope of a filter to convert frequency deviation into amplitude modulation. This make use of a *quasi–static* approximation, that is given a constant or static frequency passing through a filter produces a steady–state amplitude at the filter output according to

$$y_{\text{out}}(t) = A_c \left| H(f_{\text{static}}) \right| \cos \left[2\pi f_{\text{static}}(t) t + \angle H(f_{\text{static}}(t)) \right] \tag{17}$$

where $H(f)$ is frequency response of the filter. The quasi–static approximation assumes that the filter output response quickly settles to steady–state magnitude and phase, with respect to FM carrier instantaneous frequency

$$f_{\text{inst}} = f_c + f_d m(t) \tag{18}$$

This is reasonable as the filter is operating in the realm of the carrier frequency and the maximum frequency of $m(t)$ is not greater than W Hz. Thus it follows that

$$y_{\text{AM}}(t) = A_c |H(f_{\text{inst}}(t)| \cos \left[2\pi f_{\text{inst}}(t) t + \angle H(f_{\text{inst}}(t)) \right] \tag{19}$$

For the the FM to AM conversion to work well the gain slope of the filter needs to be of the form

$$\left| H\big(f_{\text{inst}}(t)\big) \right| \simeq a \cdot f_{\text{inst}}(t) + b \tag{20}$$

where a and b are real constants. The relationship of (40) is of the proper form to produce AM that can be demodulated using an envelope detector.

Laboratory Exercises

1. Configure the test set–up of Figure 10 using the RF Board 1 MHz lowpass filter as a means to convert FM to AM. Due to component variations your filter response will be slightly different from the screenshots embedded in Figure 10.

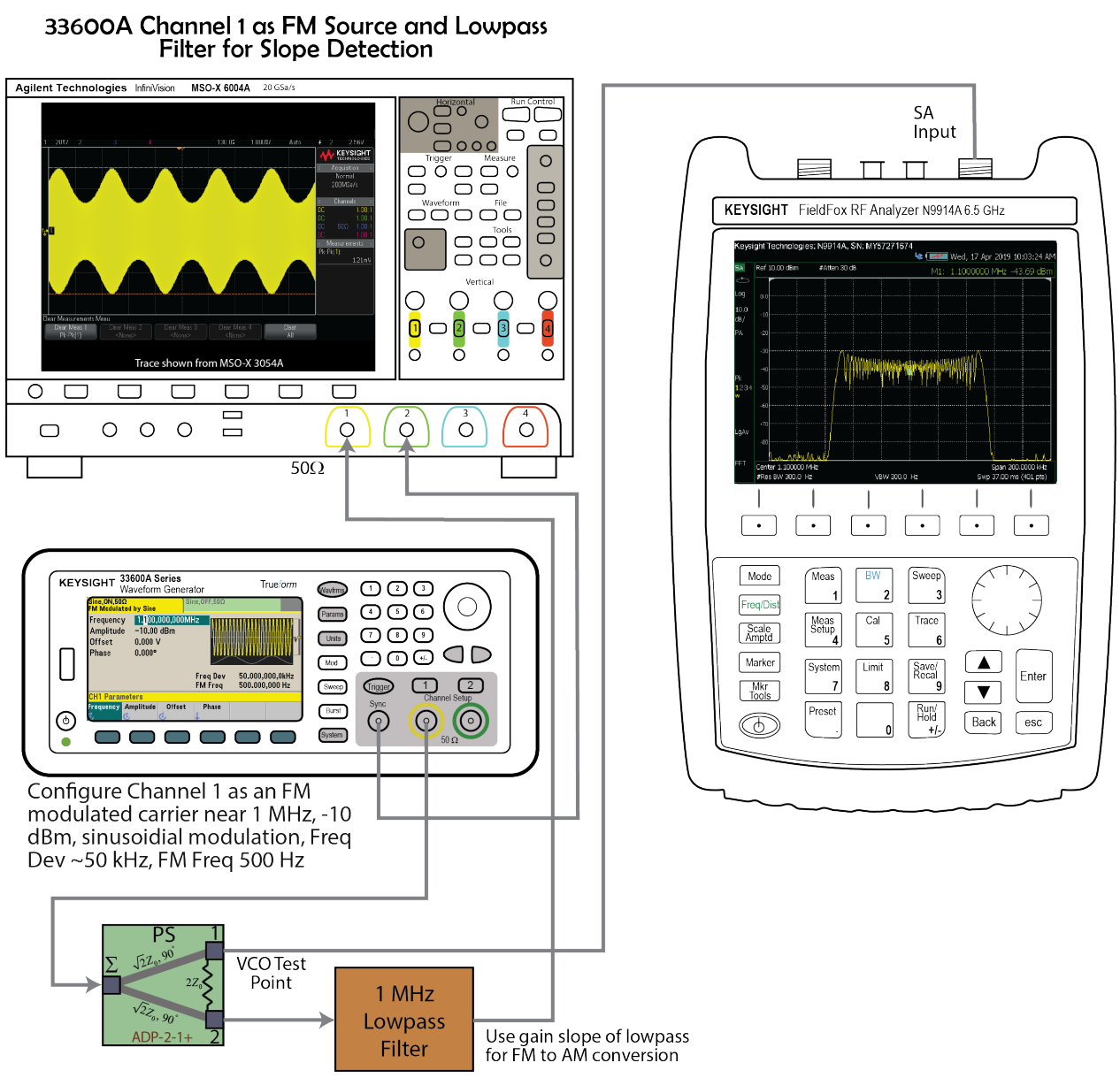


Figure 10: Slope detection using the gain slope of the 1 MHz lowpass to convert FM to AM.

2. Obtain results similar to those shown in Figure 10 by setting the carrier frequency somewhere above the filter cutoff, e.g. around 1.1 MHz, and use sinusoidal tone modulation with $f_m = 500$ Hz. Adjust the *Freq Dev to be around 50 kHz.
3. Optional for bonus points create a Python model of the lowpass filter as a 5th–order digital Chebyshev type 1 filter driven by a sinusoidal FM signal source. Observe the output and see a similar AM signal is obtained. The sampling rate will need to be around 10–100 Msps, so the computational burden will be rather high. You will only need, say two cycles of a 500 Hz message sinusoid. At $f_s = 100$ MHz this requires just $2 \times 10^8/500 = 400,000$ samples.

First–Order Analog PLL

An analog phase–locked loop (PLL) with sinusoidal phase detector, is shown in Figure 11, for FM demodulation.

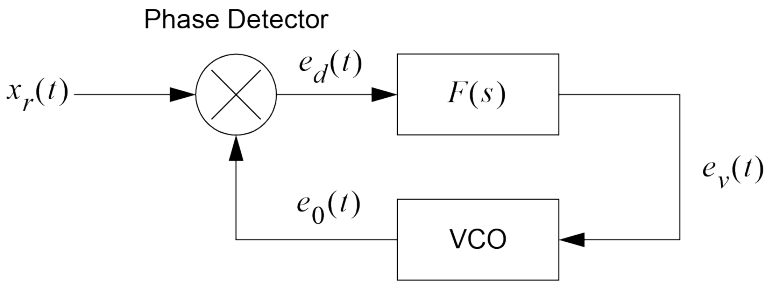


Figure 11: General PLL diagram employing a sinusoidal phase detector.

For modeling purposes we let

$$x_r(t) = A_c \sin\left[2\pi f_c t + \theta(t)\right] \tag{21}$$

$$e_o(t) = A_v \cos\left[2\pi f_c t + \hat{\theta}(t)\right]. \tag{22}$$

Note that frequency error may also be included in $\phi(t) = \theta(t) - \hat{\theta}(t)$. Assuming the double frequency term is removed, we can write

$$e_d(t) = \frac{1}{2} A_c A_v K_d \sin\left[\theta(t) - \hat{\theta}(t)\right]. \tag{23}$$

The VCO, see Figure 12, converts voltage to frequency deviation relative the VCO quiescent frequency f_0. The VCO output instantaneous frequency in Hz is

$$f_{\text{VCO}}(t) = f_0 + \frac{K_v}{2\pi} e_v(t) = f_0 + \frac{1}{2\pi} \cdot \frac{d\hat{\theta}(t)}{dt}. \tag{24}$$

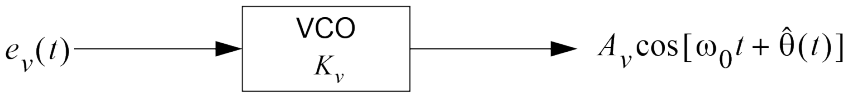


Figure 12: VCO model for a PLL.

The frequency deviation in radians/s is

$$\text{VCO Frequency Deviation} = \frac{d\hat{\theta}(t)}{dt} = K_v e_v(t) \tag{25}$$

In mathematical terms we now close the loop by connecting the phase detector output to the VCO through a convolution of the loop filter impulse response

$$\frac{d\hat{\theta}(t)}{dt} = \frac{A_c A_v K_d K_v}{2} \int^t f(t-\lambda) \sin\left[\theta(\lambda) - \hat{\theta}(\lambda)\right] d\lambda \tag{26}$$

This equation can be represented in block diagram form as the nonlinear feedback control model of Figure 13.

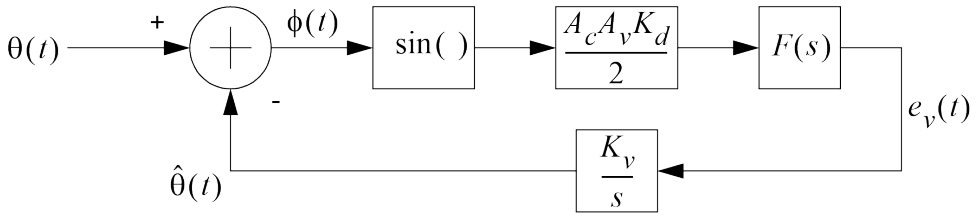


Figure 13: Non–linear baseband PLL model.

When the loop is in lock, with small phase error, i.e.

$$\sin[\phi(t)] = \sin[\theta(t) - \hat{\theta}(t)] \simeq \theta(t) - \hat{\theta}(t) = \phi(t), \tag{27}$$

we can linearize the loop. This linearizing leads to the s–domain PLL model shown in Figure 14.

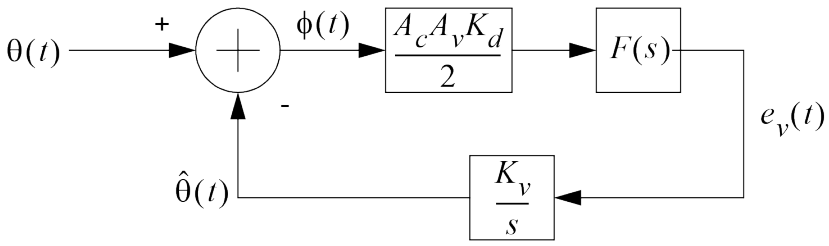


Figure 14: Linear baseband PLL model.

Working from the block diagram we can solve for $\Theta(s)$ in terms of $\Phi(s)$

$$\hat{\Theta}(s) = \frac{K_t}{s}\left[\Theta(s) - \hat{\Theta}(s)\right] F(s)$$

$$\text{or } \hat{\Theta}(s)\left[1 + \frac{K_t}{s}F(s)\right] = \frac{K_t}{s}\Theta(s)F(s), \tag{28}$$

where we can lump the phase detector gain into $K_d' = A_c A_v K_d/2$, thus

$$K_t = K_d' K_v \text{ rad/s} \tag{29}$$

Finally, the closed–loop transfer function, $H(s) = \hat{\Theta}(s)/\Theta(s)$, can be written as

$$H(s) = \frac{\hat{\Theta}(s)}{\Theta(s)} = \frac{K_t F(s)}{s + K_t F(s)}. \tag{30}$$

For a first–order PLL $F(s) = 1$, then we have

$$H(s) = \frac{K_t}{s + K_t} \tag{31}$$

We are finally in a position to consider the details of how the first–order PLL recovers the FM message signal $m(t)$. With FM the phase deviation at the PLL input of Figure 14, is

$$\Theta(s) = \frac{f_d M(s)}{s}, \tag{32}$$

where $M(s) = \mathcal{L}\{m(t)\}$. The VCO control voltage input is

$$\begin{aligned} E_v(s) &= \Theta(s) \cdot \frac{s}{K_v} \cdot H(s) \\ &= \frac{k_v M(s)}{s} \cdot \frac{s}{K_v} \cdot \frac{K_t}{s + K_t} \\ &= \frac{f_d}{K_v} \cdot \frac{K_t}{s + K_t} \cdot M(s). \end{aligned} \tag{33}$$

The 3dB bandwidth in Hz of the FM demodulator is just the loop gain divided by 2π

$$\text{Demodulator 3dB Bandwidth} = \frac{K_t}{2\pi} \text{ Hz} \tag{34}$$

The linear analysis assumes that the loop is in lock. The first–order PLL is in lock if $d\hat{\theta}(t)/dt = 0$. The governing relationship for the loop to be in lock is the nonlinear differential equation of (25). For the case of the first–order loop we have

$$\frac{d\hat{\theta}(t)}{dt} = K_t \sin[\phi(t)]. \tag{35}$$

Suppose the loop is in lock for $t < 0$ and the input phase deviation undergoes a step change in frequency, i.e.,

$$\frac{d\theta(t)}{dt} = \Delta\omega\, u(t), \tag{36}$$

where we assume $\Delta\omega > 0$. Combining this with (25), we can write

$$\frac{d\phi(t)}{dt} = \Delta\omega\, u(t) - K_t \sin[\phi(t)]. \tag{37}$$

A plot of $d\phi(t)/dt$ versus $\phi(t)$ is known as the *phase plane* plot. The phase plane plot for a first–order PLL having $\Delta\omega > 0$ is shown in Figure Phase_Plane. At $t = 0$ the phase plane operating point jumps to $d\phi(t)/dt|_{t=0} = \Delta\omega$. Since dt is also positive, we conclude that $d\phi$ is also positive. If $d\phi(t)/dt$ should become negative $d\phi$ is also negative, which drives the operating point to the

`stable lock point` which again has $d\phi(t)/dt = 0$ (a frequency error of zero). Due to the finite loop gain, K_t, there is a steady–state phase error ϕ_{ss} when the loop finally settles. We conclude that for the loop to lock, or in this case remain locked, the phase plane curve must cross the $d\phi/dt = 0$ axis.

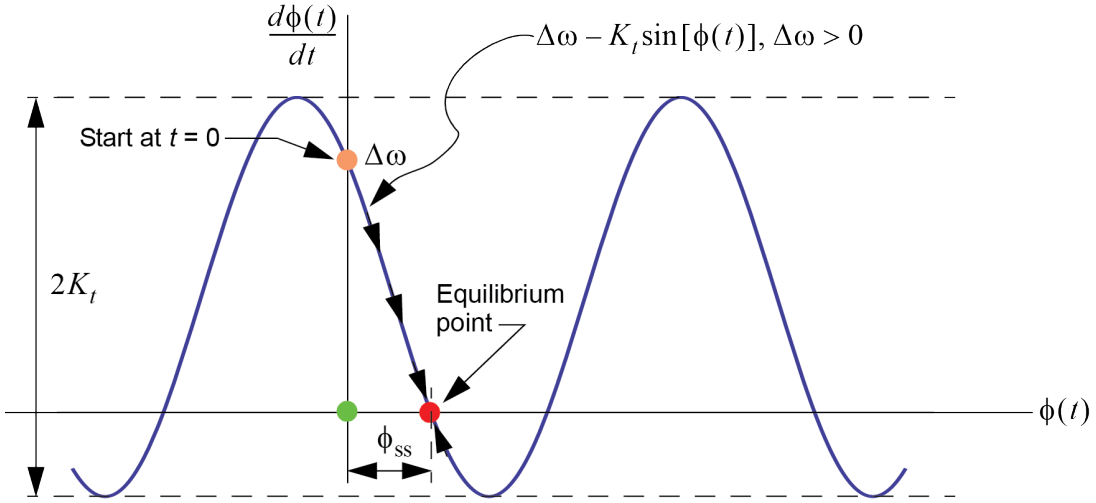


Figure 15: Phase plane plot for first–order PLL with a frequency step of $\Delta\omega > 0$.

The maximum $\Delta\omega$ the loop can handle is K_t rad/s, so the *total lock range* of the PLL is

$$\text{Total Lock Range in Hz: } f_0 - \frac{K_t}{2\pi} \leq f \leq f_0 + \frac{K_t}{2\pi}, \tag{38}$$

where we recall that f_0 is the VCO quiescent frequency. For a given $\Delta\omega$ within the lock range, the steady–state phase error is

$$\phi_{ss} = \sin^{-1}\left(\frac{\Delta\omega}{K_t}\right) \tag{39}$$

Laboratory Exercises

Consider the test setup shown in Figure 16 below:

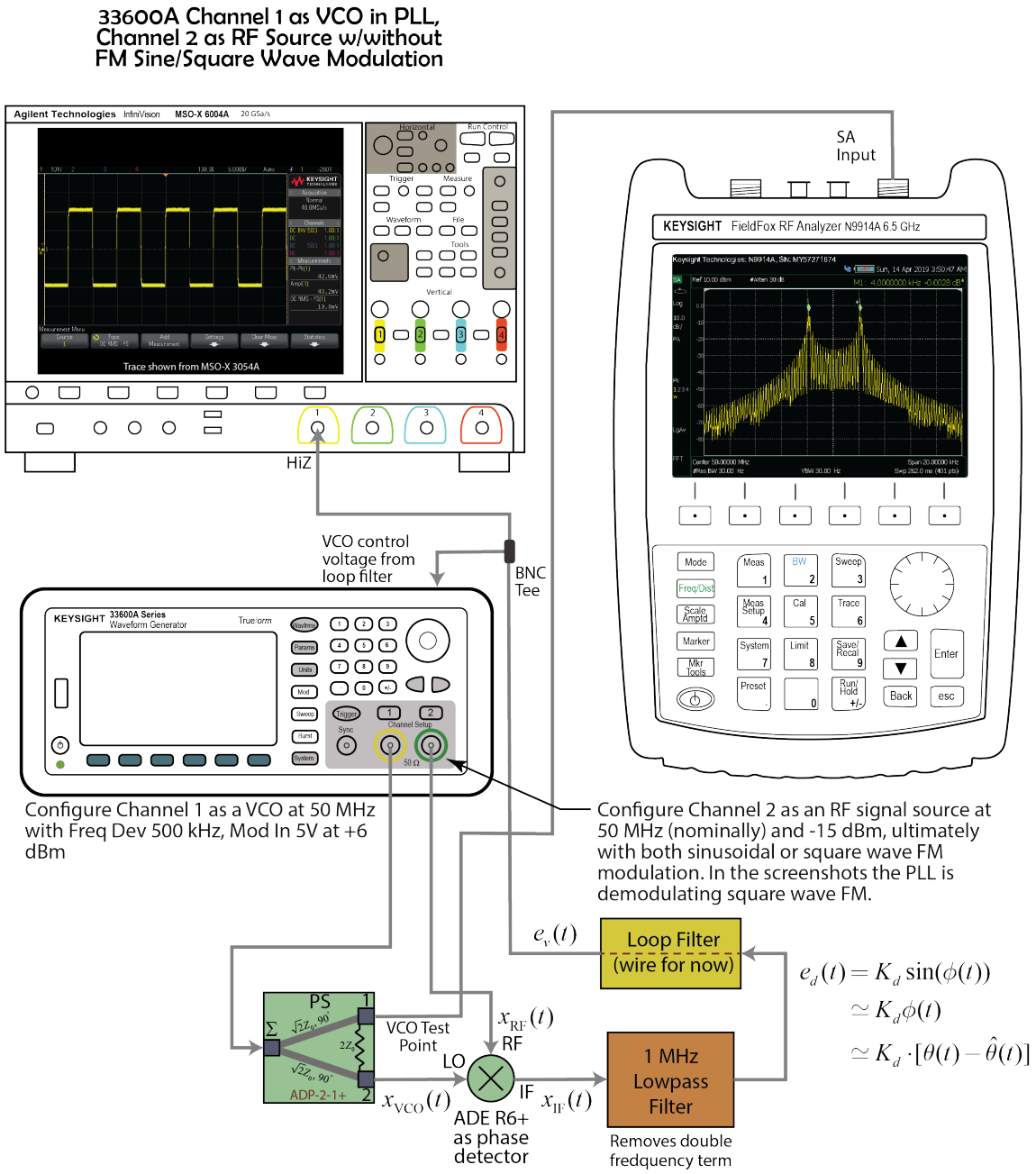


Figure 16: Instrument configuration for building a first–order PLL using one of the mixers on the RF board.

1. Configure the test set–up of Figure 16. In particular Channel 1 is now used as a VCO centered at 50 MHz and +6 dBm power with *Freq Dev* of 500 kHz and external *Mod In* at +5V. **Note**: You earlier characterized Channel 1 as a VCO with a *Freq Dev* of 5 MHz. With the deviation reduced to 500 kHz the sensitivity K_v in V/Hz will be reduced by a factor of 10 (why?). Channel 2 is configured initially as an unmodulated carrier at nominally 50 MHz and –15 dBm power level. Note that the loop filter in this case is just a wire from the phase detector directly to the VCO input (back panel of the 33600). Initially do not connect the phase detector output to the VCO input, but do probe with the scope to see the difference frequency at the phase detector output. When the input signal and VCO are slightly offset in frequency, you observe

the difference frequency (*beat note*) on the scope. It is of the form

$$e_d(t) = K_d' \sin(2\pi f_\Delta t), \tag{40}$$

which is actually of the same form of the phase detector output when the loop is locked, that is $K_d' \sin(\phi)$. Note as mentioned earlier the phase detector gain coefficient K_d' is equivalent to $A_c A_v K_d/2$ shown in Figures 13 and 14. The phase detector gain, in volts per radian, is thus the peak voltage you observe on the scope. Record the value of K_d' you observe for PLL closed–loop analysis.

2. Calculate K_t using the measured value of K_d' and the previously measured value of the VCO sensitivity, now scaled by 1/10. Formally you need to have K_v in rads/s, but since we are most interested in the lock range in Hz, K_v in Hz/v and hence K_t in Hz is sufficient.
3. Now close the loop by connecting the VCO to the phase detector output. Verify that the loop is locked by observing the phase detector output on the scope using DC coupling. The beat note should be gone and you should see a DC level. You might need to lock the loop by tuning the Channel 2 signal (RF input) in small steps above or below the nominal 5 MHz set value. Once the loop locks you will notice that the DC level you observe moves up and down with the frequency tuning of the input signal. Next measure the lock range of the PLL by varying the frequency of the input signal above and below 50 MHz. Take relatively small steps to insure that the loop does not jump out of lock before reaching the true upper and lower lock range limits. For the first–order PLL this should be twice the open loop gain in Hz, that is twice the product of the peak phase detector output voltage times the VCO sensitivity K_v in Hz/v. See if your calculations agree with your observation. Python nonlinear loop simulation using `sk_dsp_comm.synchronization.pll1` shows what happens when the frequency of the input signal exceeds the lock range.

```
import sk_dsp_comm.synchronization as pll
```

1st–Order PLL Nonlinear Simulation
Three modulation types can inserted by commenting and uncommenting code: frequency step, sinusoidal FM and sqaure wave FM.

Frequency Step Exceeds Lock Range

```
# Sampling rate is well above the expected loop bandwidth K_t/(2pi) of ~4500
Hz.
fs = 100000
Kt_Hz = 4500
t = arange(0,2.5,1/fs)
# FM modulation m(t) with fD = 1 Hz/volt so peak deviation is max{m(t)}
Df = 4501 # peak deviation in Hz
# Freq Step
```

```
m = Df*ss.step(t-.5)
# Sinusoidal FM
fm = 100
#m = Df*cos(2*pi*fm*t)*ss.step(t-.5)
# Squarewave FM
#m = Df*sign(cos(2*pi*fm*t))*ss.step(t-.5)
# Accumulate freq to phase to form input to baseband PLL
phi = 2*pi*cumsum(m)/fs
theta_hat, ev, psi = pll.PLL1(phi,fs,1,1,Kt_Hz,0.707,1)
```

```
plot(t,ev)
#plot(t,psi)
xlabel(r'Time (s)')
ylabel(r'VCO Control Voltage ($K_v$ in Hz is one)')
xlim([0.48,0.55])
title(r'VCO Control Voltage as Frequency Deviation in Hz')
grid();
```

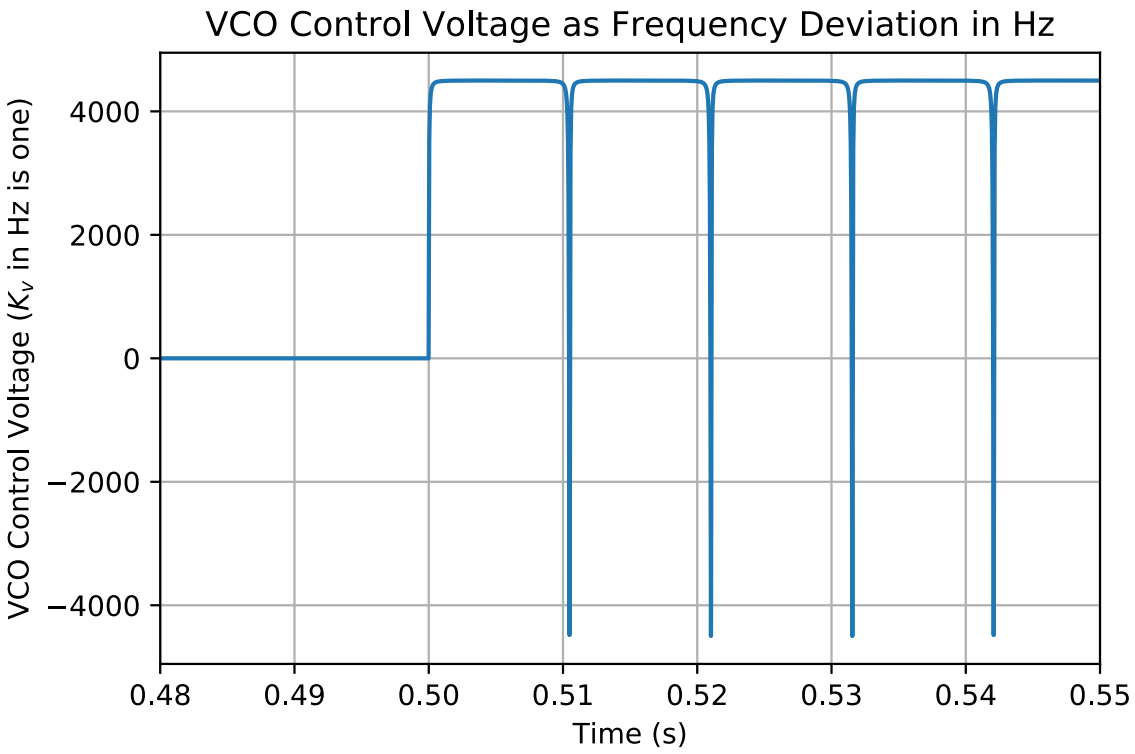


Figure 17: Python first-order PLL phase detector output/VCO input when frequency offset exceeds the lock range (K_t of 4500 Hz in this case).

```
#plot(t,ev)
plot(t,psi)
xlabel(r'Time (s)')
ylabel(r'Phase $\psi$ (rad)')
#xlim([0.48,0.55])
title(r'Unwrapped Phase: Slipping Cycles')
grid();
```

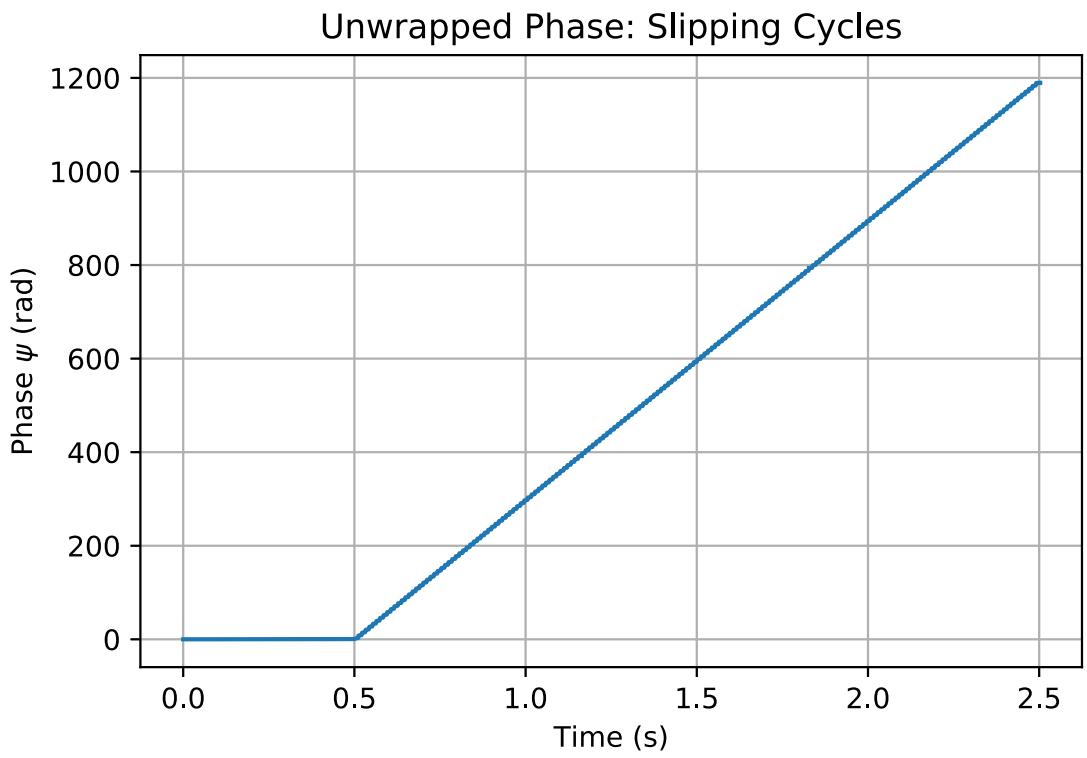


Figure 18: Python first–order PLL phase error output showing cycle slipping when frequency offset exceeds the lock range (of 4500 Hz in this case).

Case Study 1: Sinusoidal FM

4. Now apply FM modulation by properly configuring the modulation options for Channel 2 of the 33600. Note the screen shots in Figure 16 are for square wave modulation with the scope showing the VCO control voltage the spectrum analyzer showing the VCO spectrum as it is phase locked to the input RF signal. On Channel 2 set *Freq Dev* to 2 kHz, waveform to sine at $f_m = 100$ Hz. Assuming the closed–loop bandwidth is greater than 100 Hz (is should be around 4500 Hz for the setting discussed in Figure 16), the PLL should be tracking the FM input signal, that is the VCO control voltage (phase detector output voltage) will follow the modulation. Verify this on the scope. This waveform is the demodulated FM signal. Compare your results with the Python simulation shown in Figure 18. If the modulation is switched to a square wave you should see the loop properly tracking the modulation, but should be able to observe the rise time/fall time is controlled by the finite loop bandwidth.

Case Study 2: Square wave FM

```
# Sampling rate is well above the expected loop bandwidth K_t/(2pi) of ~4500 Hz.
fs = 100000
Kt_Hz = 2000
t = arange(0,2.5,1/fs)
# FM modulation m(t) with fD = 1 Hz/volt so peak deviation is max{m(t)}
Df = 400 # peak deviation in Hz
# Freq Step
#m = Df*ss.step(t-.5)
# Sinusoidal FM
fm = 50
#m = Df*cos(2*pi*fm*t)*ss.step(t-.5)
# Squarewave FM
m = Df*sign(cos(2*pi*fm*t))*ss.step(t-.5)
# Accumulate freq to phase to form input to baseband PLL
phi = 2*pi*cumsum(m)/fs
theta_hat, ev, psi = pll.PLL1(phi,fs,1,1,Kt_Hz,0.707,1)
```

```
plot(t,ev)
#plot(t,psi)
xlabel(r'Time (s)')
ylabel(r'VCO Control Voltage ($K_v$ in Hz is one)')
xlim([0.502,0.52])
title(r'VCO Control Voltage as Frequency Deviation in Hz')
grid();
```

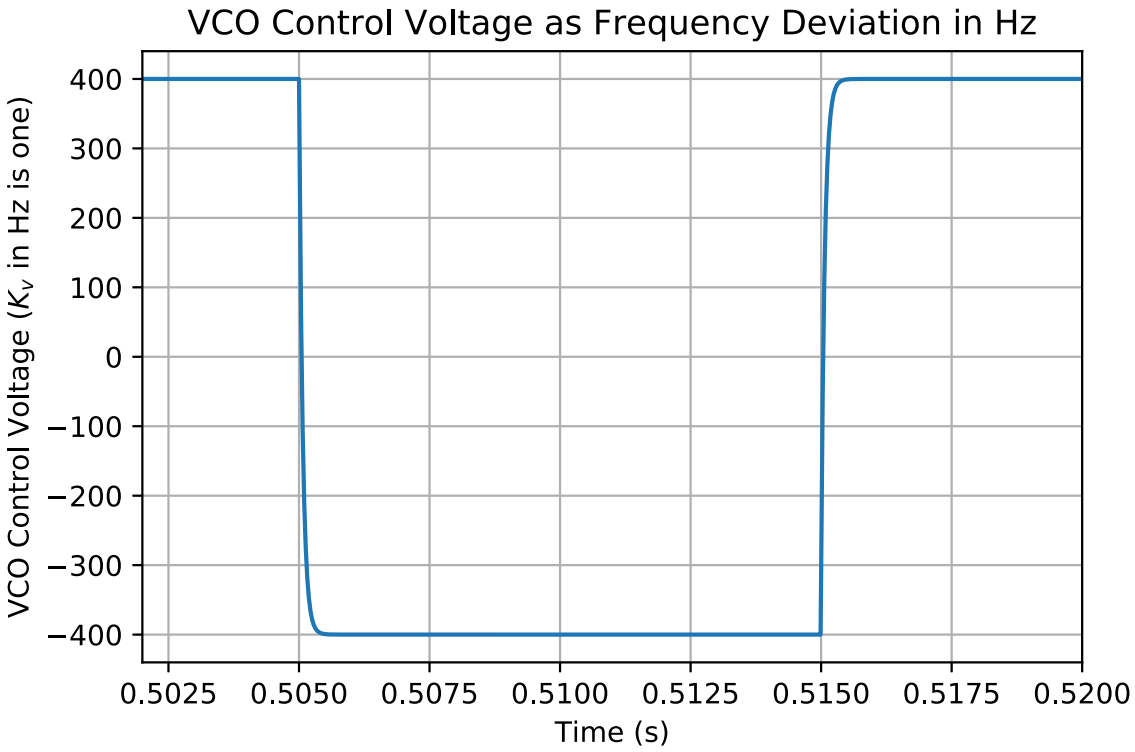
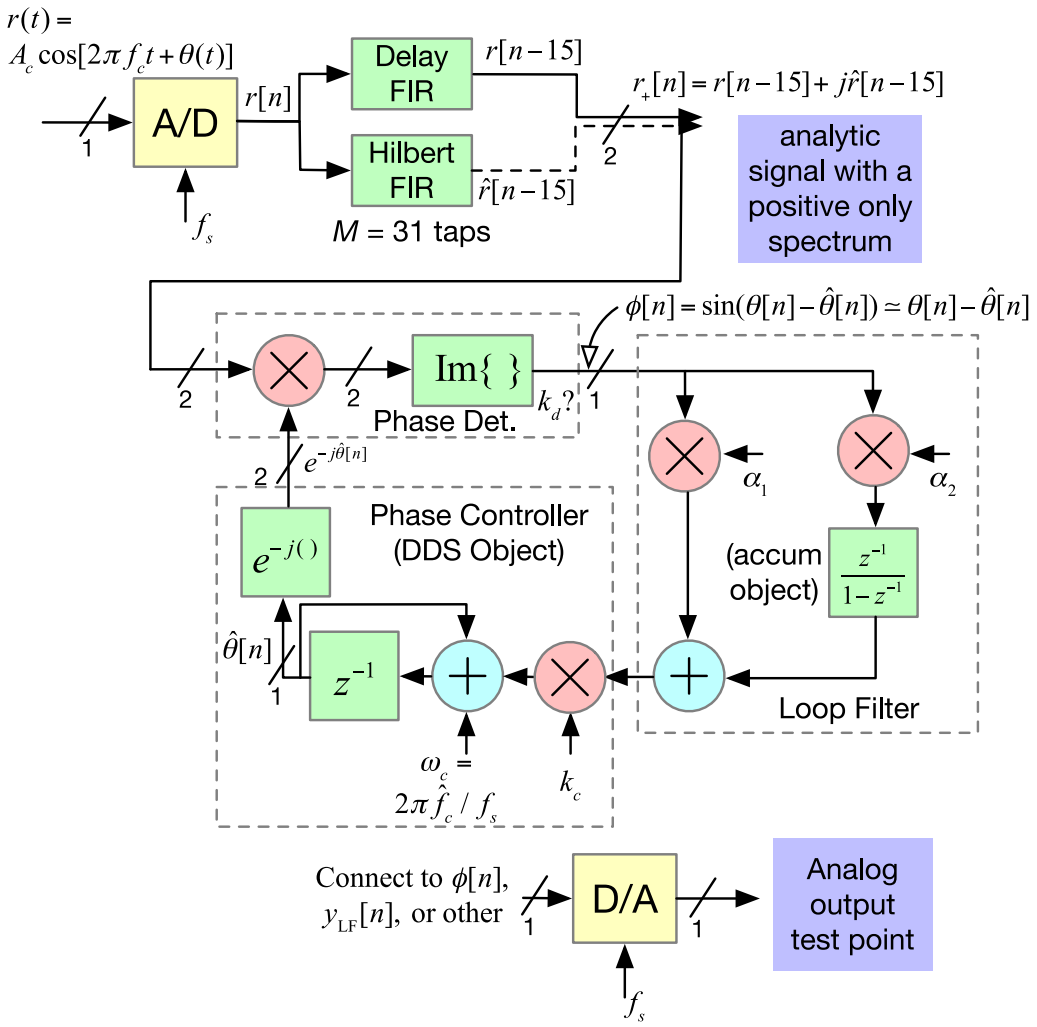


Figure 19: Python first–order PLL demodulated FM/phase detector output/VCO input with square wave FM..

5. Verify that if you increase the *Freq Dev* of Channel 2 too far, the PLL will loose lock. Decrease the gain of the PLL by decreasing *Freq Dev* on Channel 1 (this will decrease K_t by 1/2) from 500 kHz to 250 kHz. By halving the loop gain the lock range is halved and the PLL will likely fail to track the input FM unless *Freq Dev* is reduced for Channels.

Digital PLL Using Pyaudio_helper

Below is a block diagram of a DSP–based PLL built around `pyaudio_helper`. This portion of the experiment is under development, but code, found in the sample Jupiter notebook, does function.



References

1. Rodger Ziemer and William Tranter, Principles of Communications, 8th edition, Wiley, 2014.
2. ?

Appendix

What do we need?