# 9

This chapter shows how to use Circuit Envelope to measure time and frequency of an output signal when the input is a modulated source such as GSM, CDMA, etc.

# Lab 9: Circuit Envelope Simulations

# OBJECTIVES

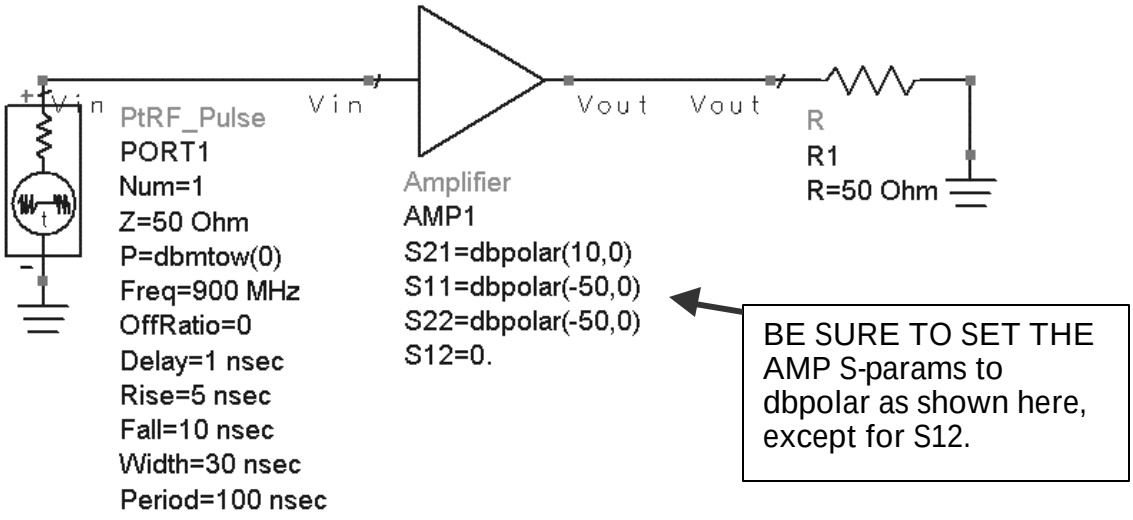
- Learn basic Circuit Envelope set up and simulation
- Simulate the response of a behavioral amp with a filter
- Simulate the Mixer with the Envelope Simulator

**About this 2 part lab:** Part A uses a behavioral amplifier to demonstrate basic Circuit Envelope simulation using a modulated signal and then measures the output envelope response in both time and frequency. Part B uses the mixer circuit where you can apply the techniques and perform more complex measurements.

# PROCEDURE Part A: CE basics with a behavioral amp

## 1. Create a new schematic design: ckt_env_basic

This amplifier circuit will be used to cover the basics of envelope simulation. Build the circuit shown here using the following steps:




a. Insert a behavioral amplifier (**Amp**) from the **System-Amps & Mixers** palette. Set the S-parameters as shown where S21 is 10 dB of gain with 0 phase (db and phase are separated by a comma). Next, S11 and S22 are –50 (dB return loss), and 0 phase. Finally, S12 can remain set to 0 to indicate no reverse leakage.

b. Insert a **pulsed RF source** (Sources-Modulated) and set it to 0 dBm at 900 MHz. Edit the following settings and be sure to check the display box: Off Ratio = 0, Delay=1 ns, Rise time=5 ns, Fall time = 10 ns, Pulse Width = 30 ns, and the period is 100 ns.

c. Insert a 50 ohm resistor, node names, grounds, and wire as needed.

## 2. Insert the Envelope Simulation controller

a. Set the frequency to 900 MHz and Order = 1. Later on, you will add distortion and increase the order.

b. Set the stop time to 50 ns. This is enough time to see the entire pulse width, including the rise, fall, and delay.

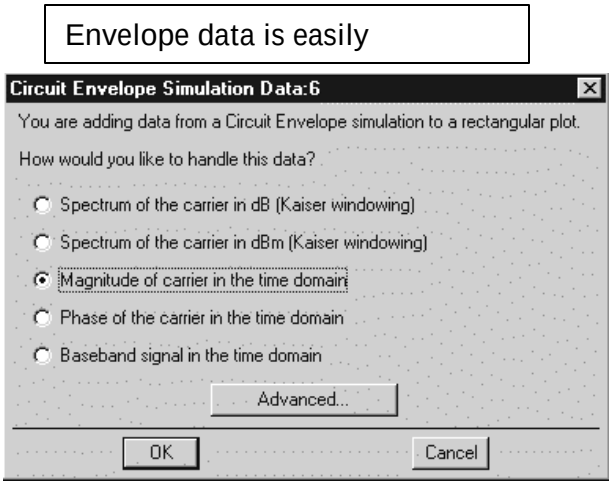
c. Set the step time to 1 ns. This means the signal will be sampled every 1 nanosecond. This means that you will get 51 points of time sampled data.


## 3. Simulate and plot the response

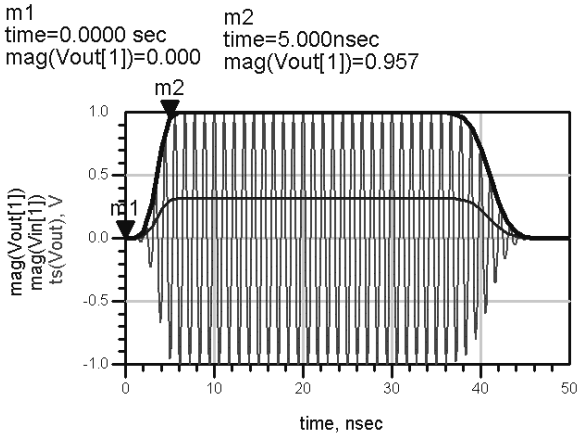
a. Simulate and watch the status window. You will see each time step calculated until the final result of 50 ns.

b. Open a new data display and name it **ckt_env_basic**. Plot Vin and Vout in a rectangular plot as the Magnitude of the Carrier in the time domain (per the dialog box).

c. Also, add a third trace which is Vout but edit it as: **ts (Vout)** which gives the composite waveform. The index [1] in the other two mag traces gives you the magnitude of the 900 MHz carrier.

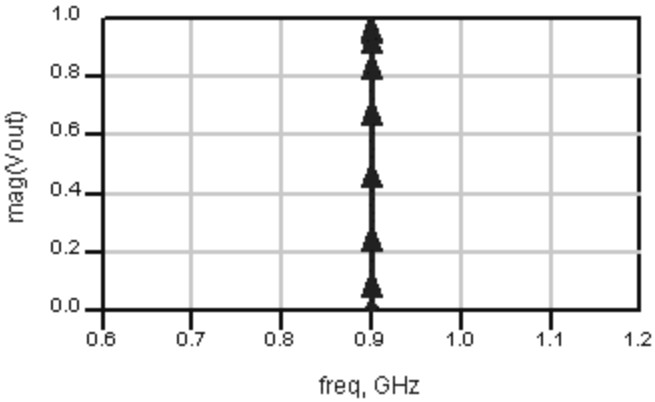



d. Put two markers on the plot to verify the rise time of 5 ns.



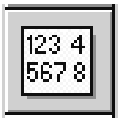
e. In a separate plot, insert mag of Vout again, and edit the trace to remove the indexing: **mag (Vout)**. Also, edit the **Plot Options**, and turn off Auto Scale: set X axis from 600 to 1200 MHz to center the trace.

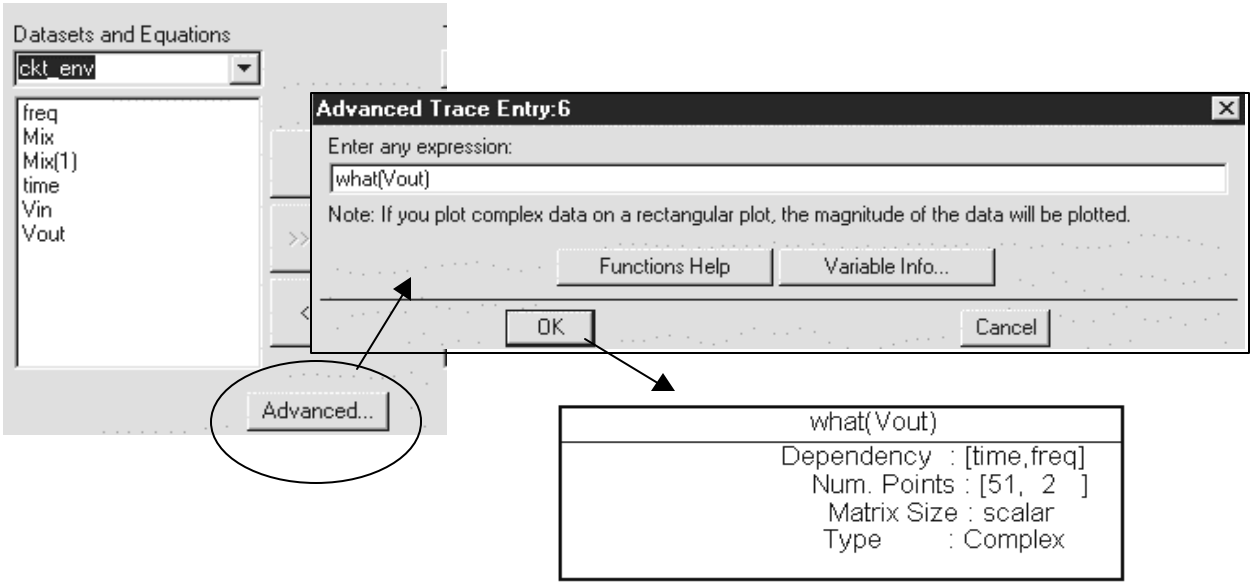
   Without the index, you get the magnitude of the fundamental (900 MHz) in the frequency domain. The increasing arrows represent the increasing magnitude of the pulse as it rises during the time (5 ns).





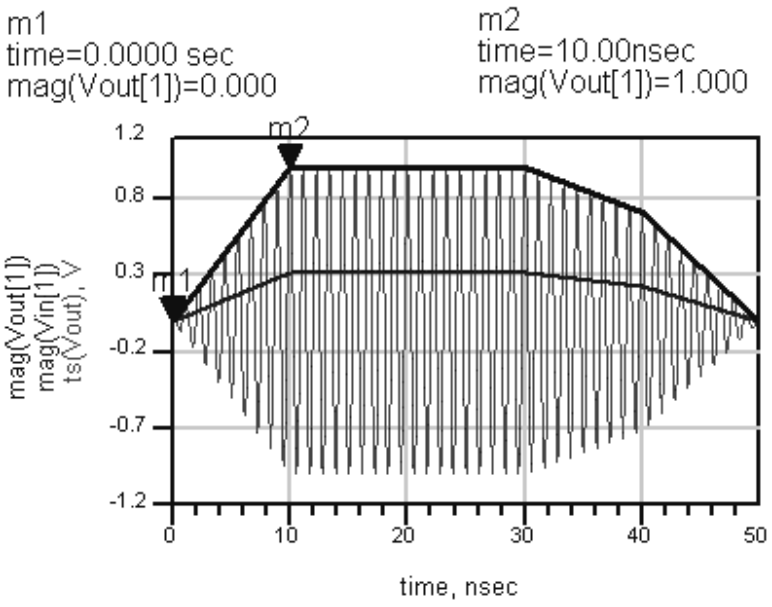
f. In the data display, insert a **List**. When the dialog box appears, click the **Advanced** button (shown here) and type in the expression: **what (Vout)**. Click OK and you will see what dependencies there are for Vout. The purpose of this is to show how the **what** function works and to show that both time and frequency exist in the circuit envelope data. There are 51 time points of the two frequencies: 0 (dc) and 900 MHz. The Matrix Size refers to the 1x1 matrix (ADS calls it scalar) and the data is complex (mag and phase of the 900 MHz).



## 4. Set the envelope controller time step to 10 ns and simulate

Watch the results. The only thing you changed was the sampling rate but the envelope now looks different because it was not sampled with enough resolution. Simply because the time step is greater than the rise time. On the plot, you see the X axis has been increased and the markers are on the first two time points: 0 and10 nsec.

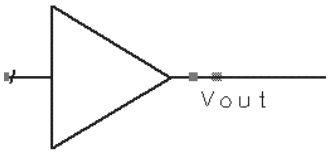



**Note on Circuit Envelope sampling:** In practice, the time step must be based on the rise time or the modulation bandwidth. For rise time, the time step can be: (rise time / 5) or less. Based on BW (modulation bandwidth) use 1 / (5x BW) to include distortion effects or side-bands. You could use 10x but this would take more computation time. Sometimes, it may be necessary. Now, the next steps will test this theory.

## 5. Add distortion to the behavioral amplifier

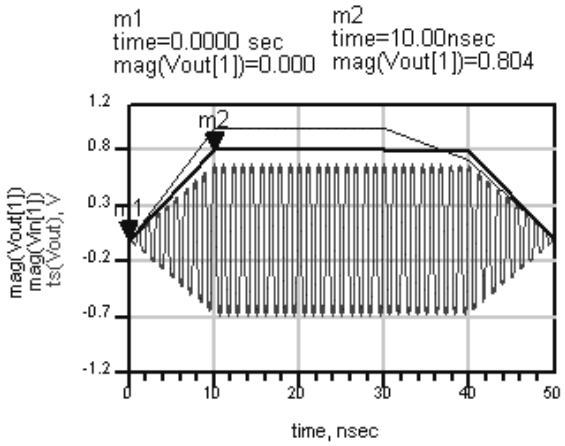
a. Edit the Amplifier by setting: Gain Compression Power = 5 (dbm is the default) and Gain Compression = 1dB. Of course, these are not realistic values but it illustrates the point. Be sure to **display** these settings (good practice).

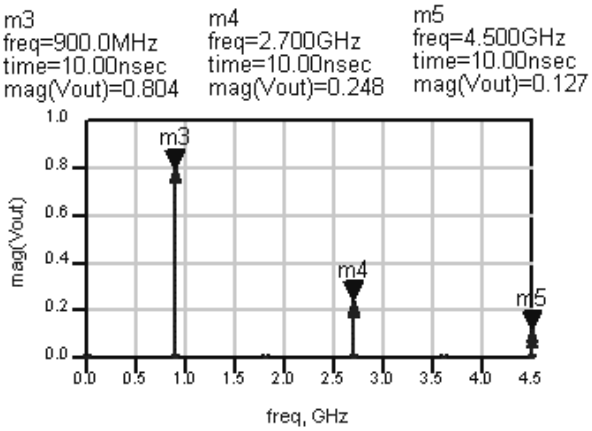
b. Set the controller **order = 5** and keep the time step at 10 ns. Also, set the source input power to 10 dbm: dbmtow (**10)**.



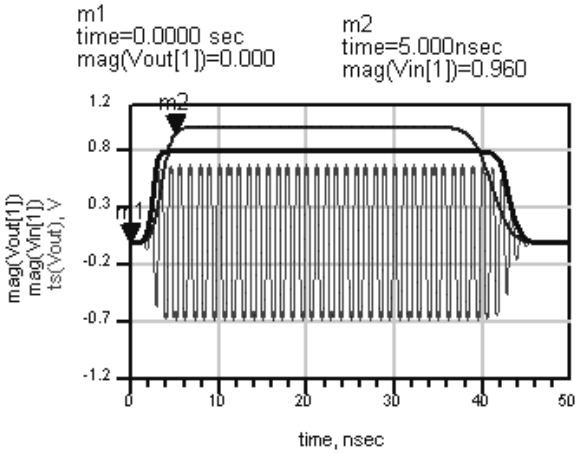

Amplifier
AMP1
S21=dbpolar(10,0)
S11=dbpolar(-50,0)
S22=dbpolar(-50,0)
S12=0.
GainCompPower=5
GainComp=1 dB

c. **Simulate and view the data**. Put the frequency domain plot X-axis back to Auto Scale and try to place the markers as shown. As you can see, there are strong odd harmonics due to the amplifier distortion, which are summing out-of-phase. This results in the envelope amplitude being smaller than the magnitude of the Vin or Vout magnitude. Also, the envelope shape is not accurate because the sampling rate is too coarse for the rise time of the Vin signal:




d. Set the time **step to 1 ns** and **Simulate** again. After the simulation, the data will update and you will get an accurate representation of the envelope. Also, the magnitude of fundamental of Vin and Vout are still greater than the envelope magnitude, due to amplifier compression.




e. Insert a **List** of **Vout**. Scroll down to the 5 nanosecond data. Now, you can see that the third hamonic is 180 degrees out-of-phase, making the envelope smaller than the magnitude of the fundamental.

| freq | Vout |
|---|---|
| time=5.000nsec | |
| 0.0000 Hz | 0.000 / 0.000 |
| 900.0MHz | 0.803 / 2.624E-15 |
| 1.800GHz | 0.000 / 0.000 |
| 2.700GHz | 0.246 / -180.000 |
| 3.600GHz | 0.000 / 0.000 |
| 4.500GHz | 0.124 / -9.696E-15 |

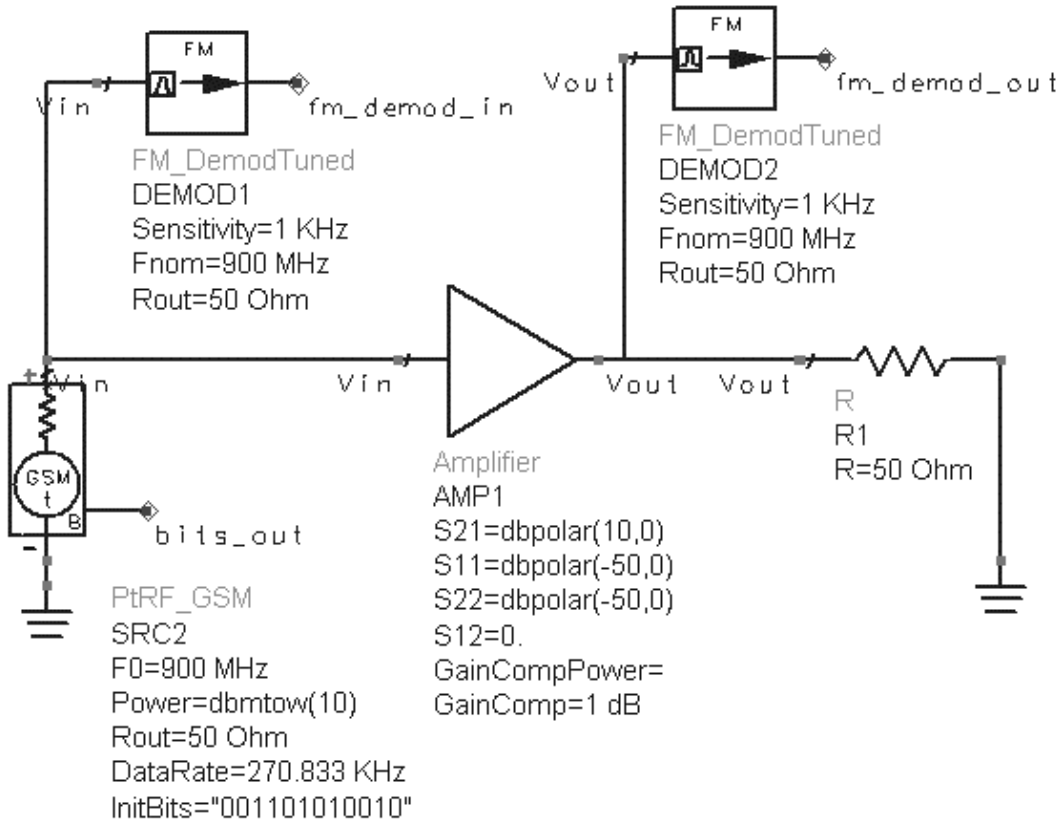
## 6. Set up the amp circuit: demodulators and a GSM source

Note on GSM: This is a phase modulation of the carrier where the phase variation represents 1 or 0.

a. From the modulated sources palette, insert the **GSM** source and put a named node at the B output as shown (node name: bits_out). It looks like a non-connected pin but it is OK. Set the source power to **10 dBm.** Also, remove the amplifier distortion: GainComp = (blank).

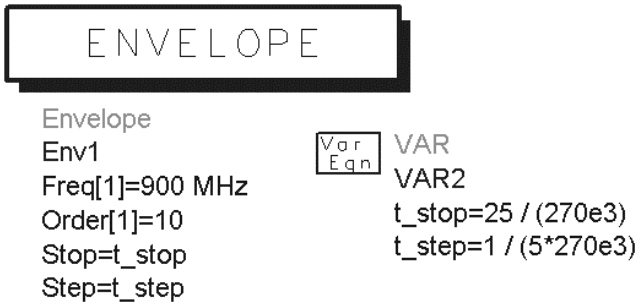
b. Go to the System-Mod/Demod palette and insert two demodulators: FM_DemodTuned components and insert them as shown. Set the value of Fnom on the two demodulators as shown: **900 MHz**. Also, insert a node name at each output: **fm_demod_in** and **fm_demod_out** (or similar names). These will be used to look at the GSM signal.

**System Design Note:** Although this may look like a system level circuit, only the amplifier is a system level component and it represents any circuit where you inject a modulation signal and want to look at the input and output. Also, you could use phase-demodulators but the FM demodulators are easier to use for this course.



## 7. Set up the Envelope Simulation

a. Insert a variable equation (VAR) and set up the Stop and Step times for the GSM signal as show: 270 kHz modulation BW. The variable: t_stop is set to cover approximately 100 us and t_step is 5 times the BW. Also, note that the default ADS Envelope time units (seconds) does not have to be specified.

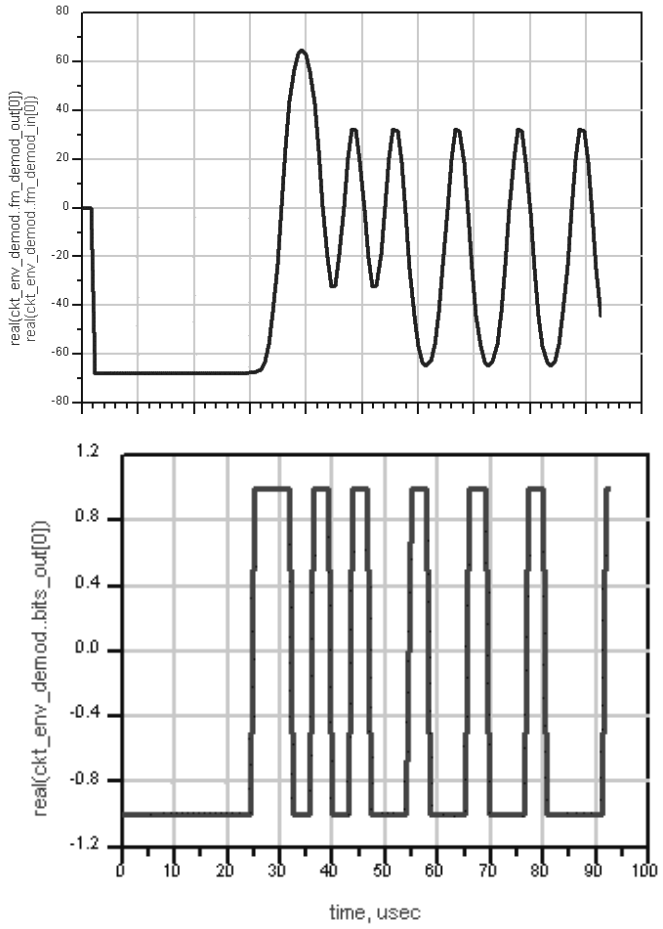



## 8. Simulate (dataset name: ckt_env_demod) and plot the results

a. Simulate with the dataset name: **ckt_env_demod**.

b. Your previous plots are not set up to display this data so use a new dataset name to keep the data in separate plots. So, **plot the two FM nodes** as Baseband signal in the time domain. These traces will be the real part indexed to [0]. The demodulator only outputs a signal at baseband (similar to the dc component).

c. In a separate graph, plot the real part of **bits_out**. Except for some delay, you should see the 001101010010 pattern.

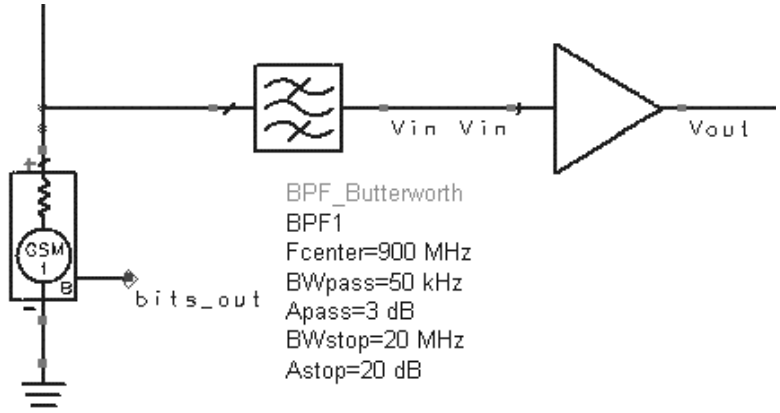


## 9. Create distortion and look at the difference

a. On the amplifier, set the GainCompPower on the amplifier to **5** (this is 5 dbm at the amp output) and set the GainComp to **1 dB**.

b. Be sure the GSM source power is set to 10 dBm.

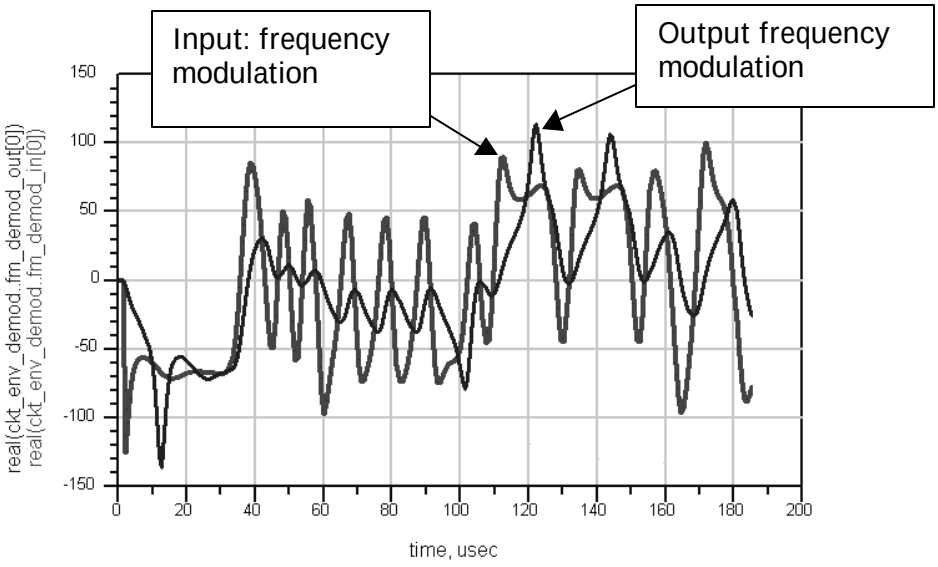
c. Insert a **Butterworth** filter (Filters-Bandpass) between the amplifier and the source and set it as shown. This will create some distortion as only the narrower bandwidth passes to the amplifier and the full signal goes to the first demodulator.




d. Change the **tstop** numerator to the number **50** to get 200 us.

## 10. Simulate and look at the response

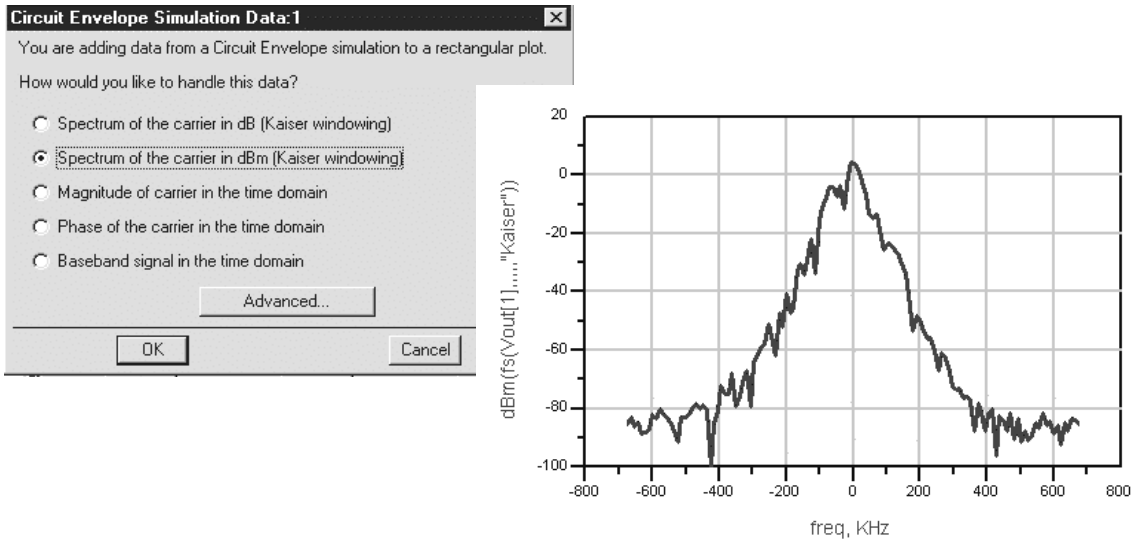
Your plot should show the distortion and delay from the input to the output similar to the one shown here.



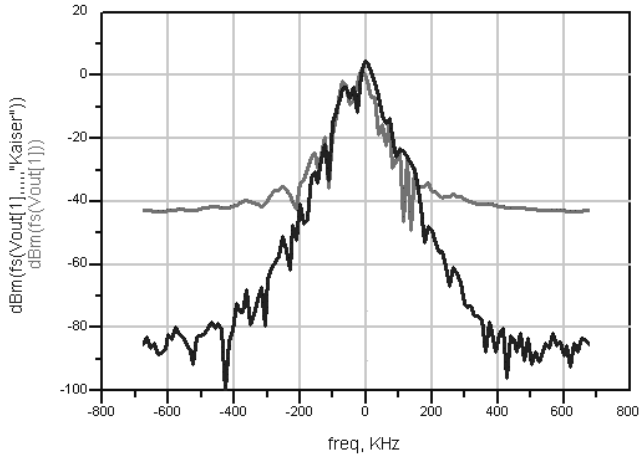
## 11. Plot the spectrum of Vout

a. Insert a new plot of the Spectrum of the carrier in dBm with a Kaiser window. This is the output spectrum around the fundamental frequency. The window helps ensure that the first and last time data points equal zero. This improves the dynamic range of the computed spectrum.

b. Add a second Spectrum trace of Vout (same trace) but remove the




window argument by editing the new trace: remove the window argument from the expression:

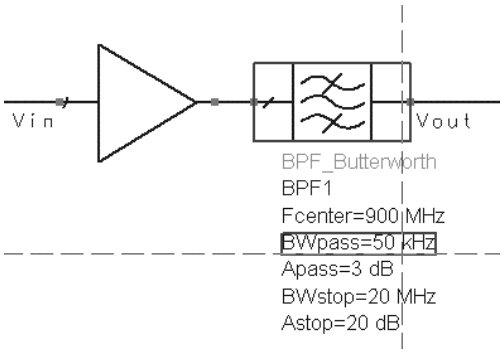




Removing the Kaiser window argument from the fs function (Fourier transform) results in a much higher noise floor. This occurs when the signal is not exactly periodic and results in an incorrect transform. The window removes the first and last time points and the result is a more accurate spectrum.

## 12. Move the filter to the amplifier output and set up tuning

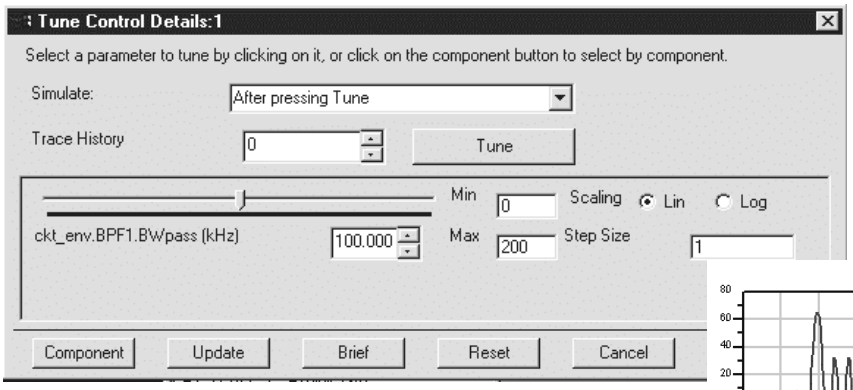
This step demonstrates how to select a specific parameter and tune it with precision so that only one trace appears updated each time you tune. For Transient or Envelope simulations this is often desirable.

a. Disconnect the filter and reconnect it to the other side of the amplifier (practice). Use the Edit > Component > Break Connections command or use your keyboard Hot Key if it is set for this command.

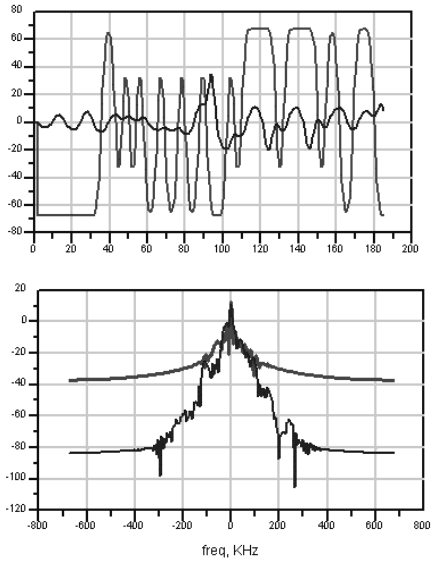



b. Start the Tune Mode and position the cursor on the 50 kHz value of the BWpass parameter and select it. You should see it appear on the Tune controller.

c. Click the **Details** button and set: Simulate: After pressing Tune, Trace History: 0, Min = 0 and Max = 200, and Step Size = 1 as shown here.




d. Position the Data Display so you can see the two plots and then do the following:




Move the slider to a position such as 100 which is a precise value (still in kHz), and then press the Tune button and watch the plot update after the simulation. Try this several times and compare it to tuning while the slider moves. This method of controlling the tuner is more precise and can be more efficient for time domain simulations such as Circuit Envelope**.**

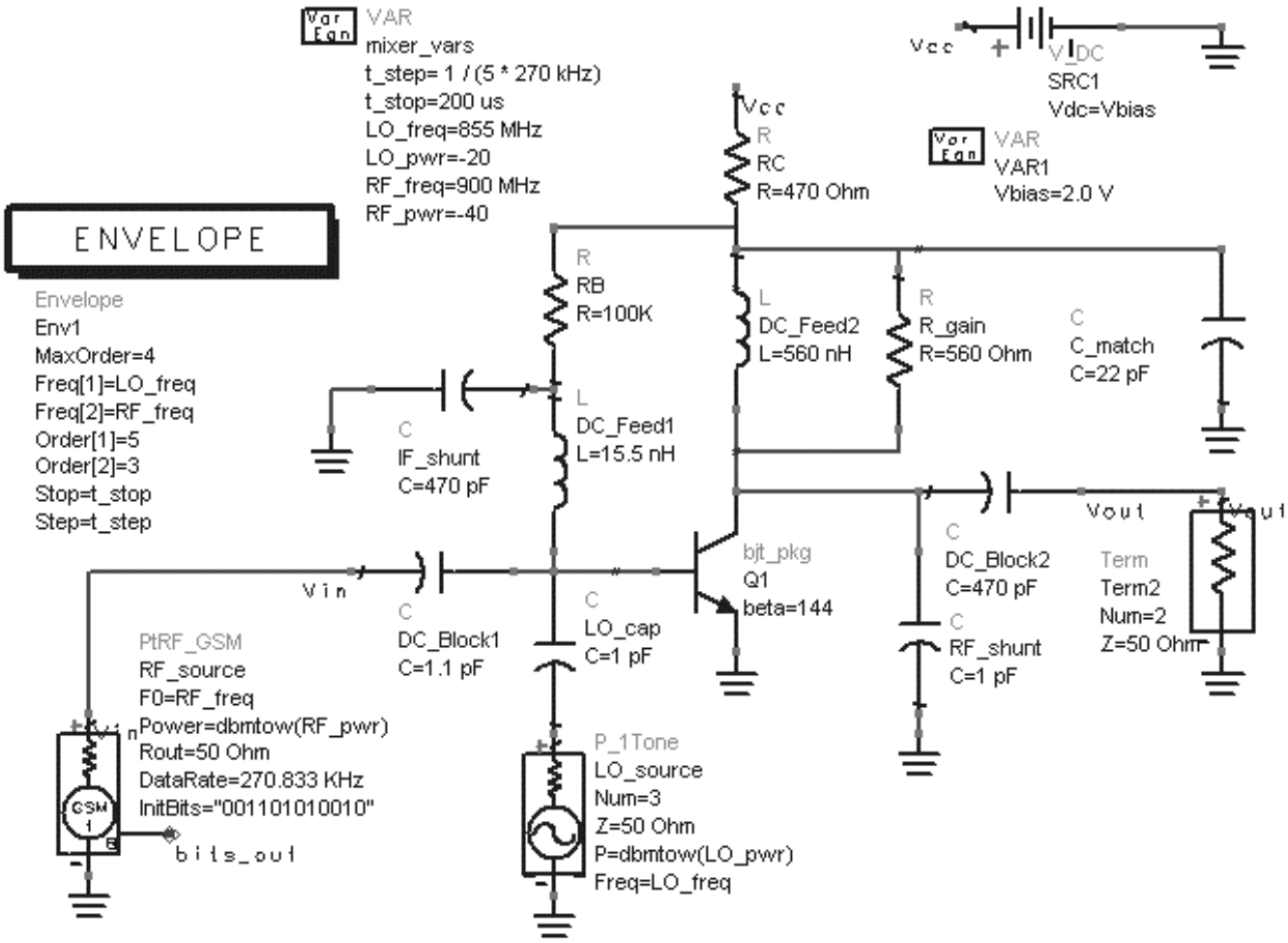
e. As always, save the schematic and data display.

# PROCEDURE Part B: mixer CE with GSM and CDMA

## 13. Set up the mixer with a GSM source

a. Copy the mixer design in a new schematic saved as: **ckt_env_mixer**.

b. Create the circuit and setup shown here: insert an **Envelope controller**, a GSM source for the RF, and a **VarEqn**. Be sure the sources and controller have the variables setup as shown:



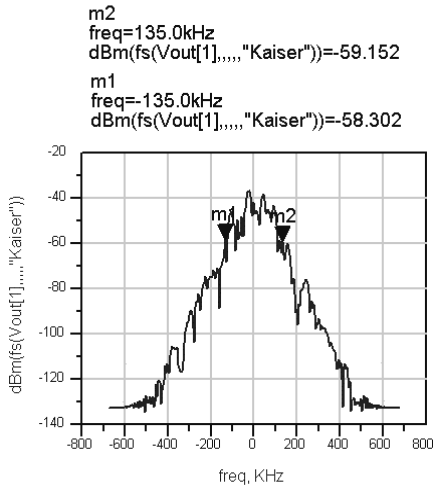

c. Check your set up, sources, and variables to be sure they match.

d. **Simulate** and watch the status window. When the simulation has finished open a new **Data Display** and save it as: **ckt_env_mixer**. The next steps will show how to post-process the data in a unique manner.

e. Plot the Vout data as: Spectrum of the carrier in dBm with a Kaiser window. Then insert two markers across the GSM bandwidth (about

270 kHz). As you can see, this spectrum analyzer-like plot shows the mixer Vout spectrum when the LO = –20 dBm. However, to verify the carrier frequency, you should verify the index value of [1].

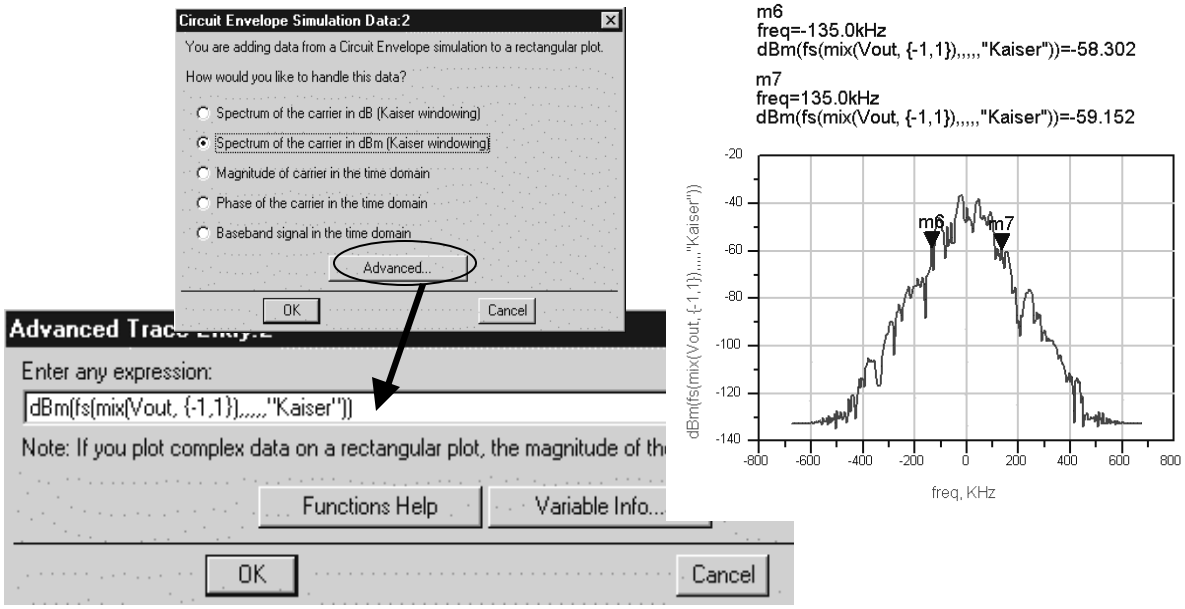
f. So, insert a list of the same Vout data and verify that the carrier is 45 MHz at any time point. This is true because it is indexed in the expression: dBm(fs(Vout[1],,,,,"Kaiser")) where [1] is 45 MHz.




| freq | Vout |
|---|---|
| time=0.0000 sec | |
| [1] 0.0000 Hz | 0.000 / 0.000 |
| 45.00MHz | 0.011 / 72.505 |
| 90.00MHz | 6.287E-4 / -27.632 |
| 810.0MHz | 0.009 / -5.191 |
| 855.0MHz | 0.084 / -108.012 |
| 900.0MHz | 0.018 / 136.332 |
| 945.0MHz | 0.001 / 17.376 |
| 1.665GHz | 0.004 / 45.578 |
| 1.710GHz | 0.047 / -38.052 |
| 1.755GHz | 0.013 / -156.484 |
| 1.800GHz | 0.001 / 78.308 |
| 1.845GHz | 3.282E-5 / -80.445 |
| 2.565GHz | 0.019 / 27.730 |

g. Insert another plot of Vout (selecting the same data type) and click the **Advanced** button. When Advanced Trace dialog appears, edit the expression using the mix function on the Mix data shown here: **dBm(fs (mix (Vout,{-1,1}),,,,,"Kaiser")** and you will get the same data as selecting the Spectrum of the carrier at [1]. Delete this plot when done.

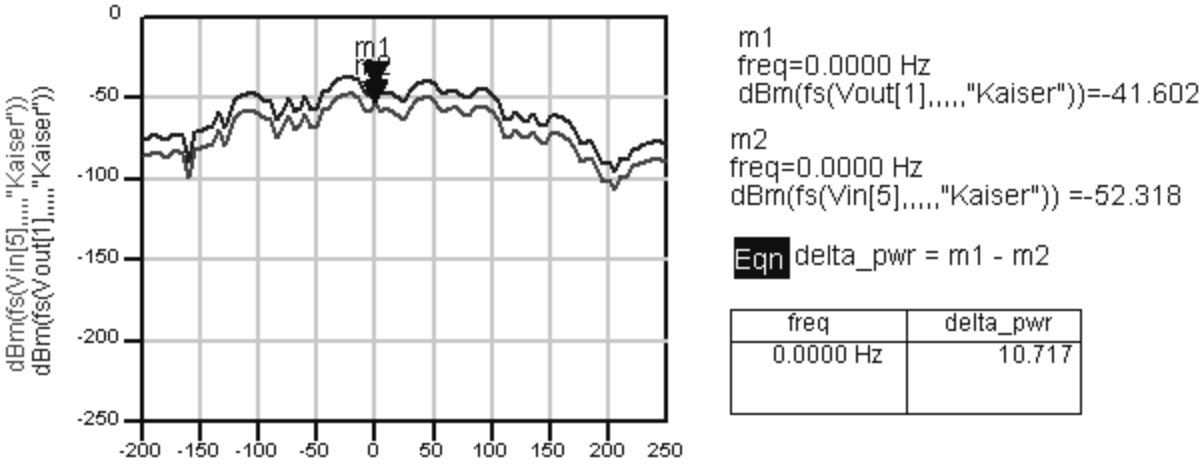
**Note on the Trace Expression:** The Kaiser window is automatically used for




Circuit Envelope spectral data as part of the fs function argument. It is also

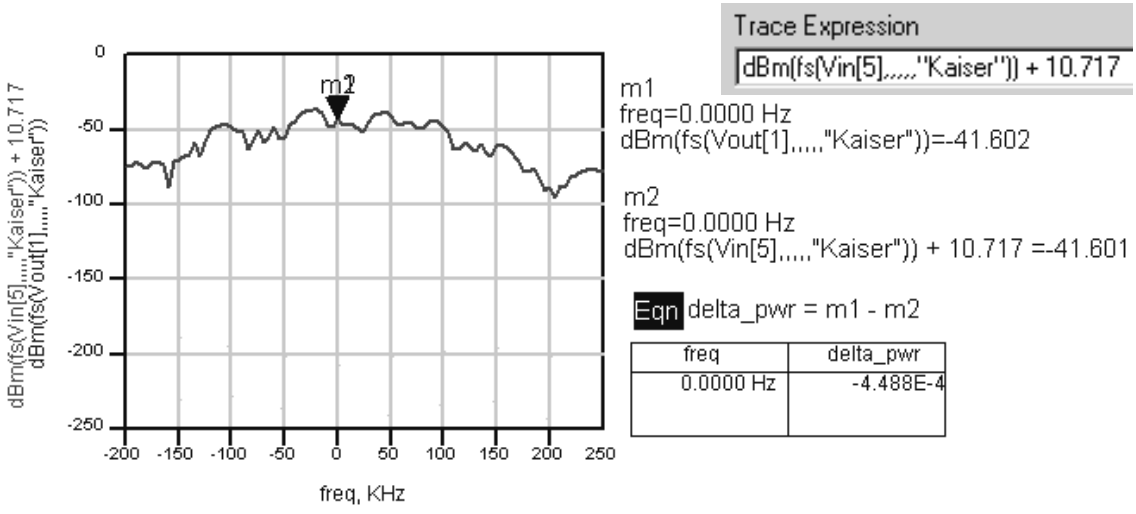
assumed that the carrier is the first frequency point [1]. However, if you were up-converting the mixer signal, you would change the index [8] which is 1710 MHz by editing the trace or you could write your own expression using: mix(Vout,{1,1}).

h. On the first Vout plot, insert **Vin** (same data format type) and edit the Trace Expression to index the 5th value **[5]** which is the RF signal at 900 MHz (from the list). You will see that the signal shape looks like Vout but with less power. Put markers on the two traces at 0 Hz each.



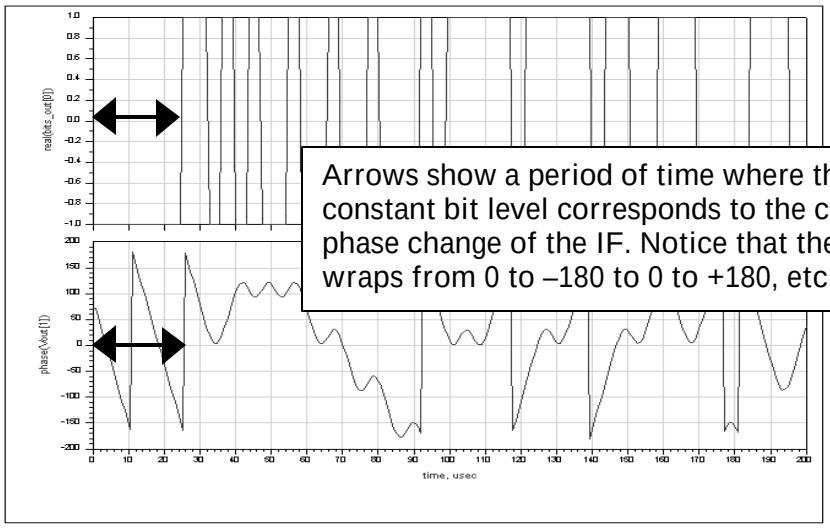

i. Write an equation (shown above) to calculate the difference in power between Vin and Vout using the marker values. Then list the equation and you will see the difference is equal to the conversion gain spec.

j. Edit the plot and add the value (10.717 or whatever you get) to the Vin trace to verify that the mixer has little or no AM to PM degradation.



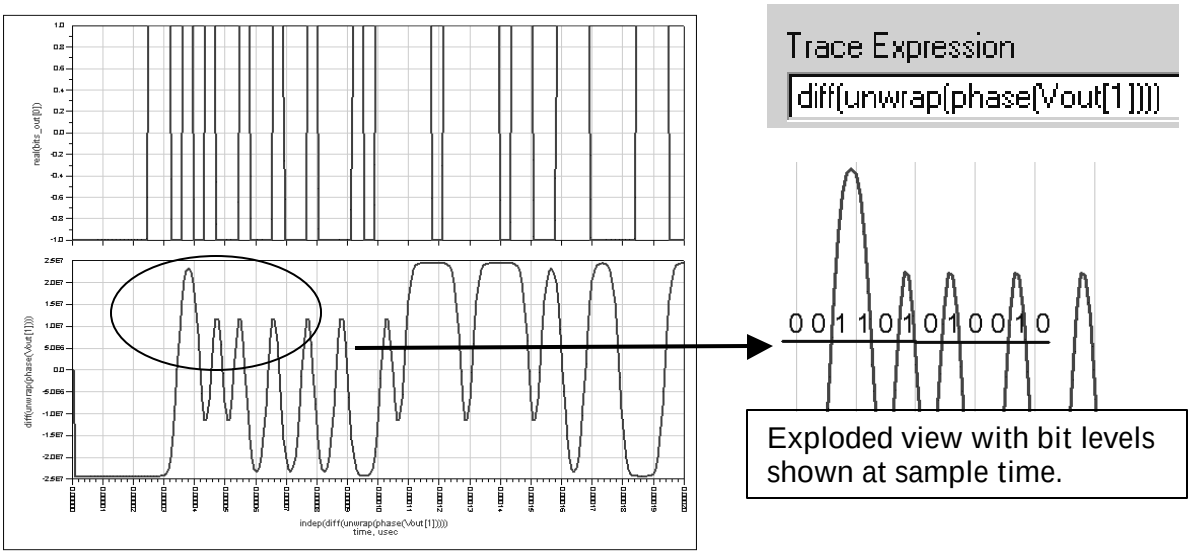
## 14. Compare the modulating bits to the output (time domain)

a. Insert a stacked rectangular plot of **bits_out** selecting the Baseband signal in the time domain and insert **Vout**, selecting Phase of the carrier (45 MHz ) in the time domain.

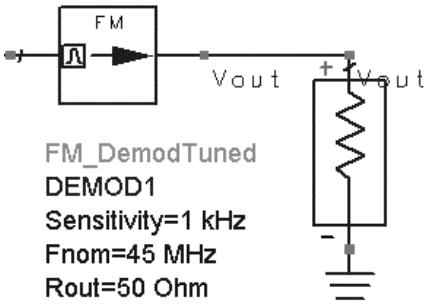



**Note on comparison of phase plots:** The phase of the carrier is difficult to compare to the bits. However, if you compare the constant phase deviation of the 45 MHz IF signal to the constant bit level during the first 20u seconds (approximately), there is correspondence. However, the GSM source has some delay due to internal filtering and further IF phase changes are difficult to relate to bit levels. But if you could unwrap (straighten) the phase trace and take the derivative of the slope of phase change, you would see the comparison more clearly, similar to demodulating the IF signal (next step).

b. Edit the Vout trace using the diff and unwrap functions: **diff(unwrap(phase(Vout[1])))** to compare bits to Vout as shown:



## 15. Demodulate the IF signal

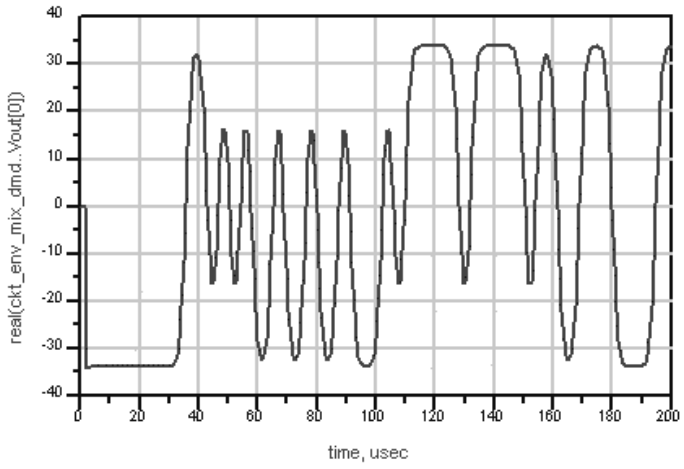



a. Insert a demodulator at the Vout node of the mixer as shown: type **FM_DemodTuned** to attach the component to your cursor and insert it as shown here. This is the same system level demodulator you used on amplifier earlier.

b. Set the Fnom = 45 MHz.

c. Setup a new dataset name, **ckt_env_mix_dmd**.

d. Simulate and plot the Vout as the Baseband signal in the time domain (remember that dataset is now called: ckt_env_mix_dmd). Here, you can see the demodulated component, index value [0]. This compares to the bits in the same way as in the previous step but was easier to do because you used a demodulator instead of demodulating the signal with ADS mathematical functions. But in either case, you get the same data. Now you can see the Circuit Envelope data is both time and frequency dependent.

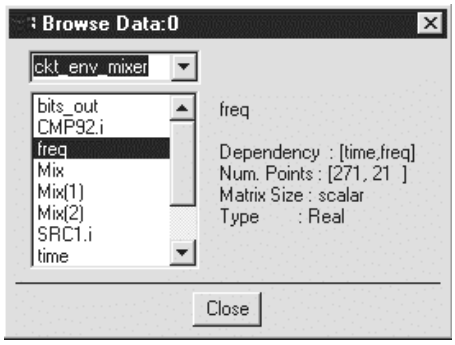



Demodulated output in the time domain is assumed to be the Baseband for CE data. If not using a demodulator, use ADS functions (*diff* and *unwrap*) to demodulate the signal.

## PASSING MARKER VALUES to other PLOTS

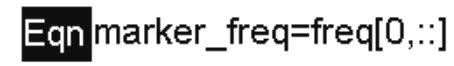
The next several steps will show you how to use the powerful expressions to pass a marker value to a function and plot the time and frequency data whenever the marker is moved. You will be using the data from the last simulation so be sure the data display shows that dataset as the active one.

**16. Write an equation equal to all frequency points in the dataset.**

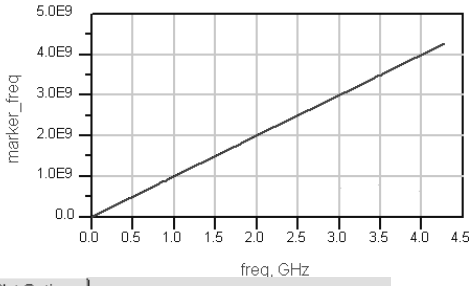



a. Insert and click the Variable Information button. You will see that **freq** is dependent on time.

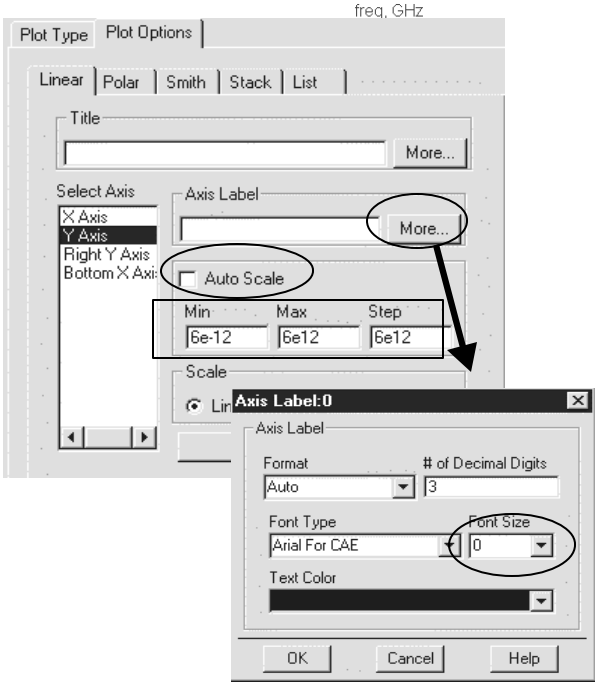
b. Insert an equation called **marker_freq** to access all the frequency points at one point in time. You can use any time point because the number of calculated frequencies are the same at any time based on the order and max order you set in the envelope simulation controller. Use zero as shown:

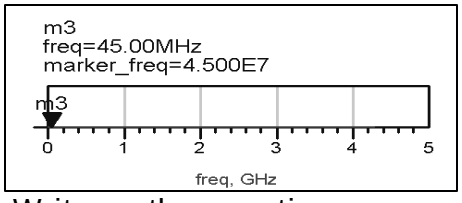



c. Insert a plot of the marker_freq equation and marker_freq is plotted against the independent variable: freq. But this can be made to look better




d. Edit the plot. Remove the Auto Scale for the Y axis. Set the Y axis Min, Max and Step to: 6e-12, 6e12, and 6e12 as shown. Then click the More button and set the Y axis font size to zero. Click OK and then put a marker on your new plot which looks like a slider.




e. Write another equation called **freq_index** using the

**find_index** function. Here, the marker value is passed into the argument along with the marker_freq data. This means you can move the marker to any point on the line of frequencies and its index value will be evaluated or assigned to freq_index. Next, you will pass freq_index as the look-up value for the Vout data you want to plot.

f. Write another equation, **marker_spectrum**, to show the spectrum

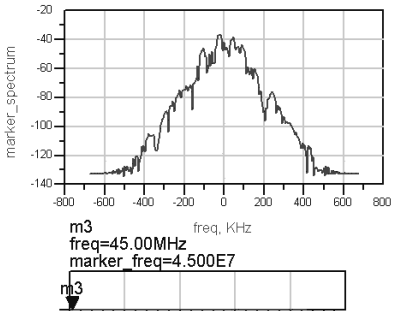
```
Eqn freq_index=find_index (marker_freq,m3)
```

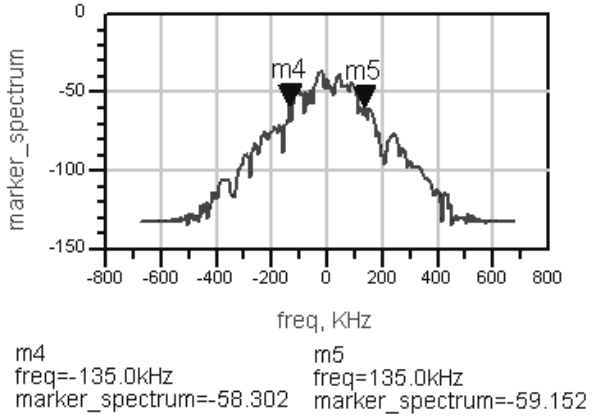
around any marker frequency point. Here, the fs function transforms the envelope time data where the two colons represent all the points in time and freq_index is the index value of the marker frequency. Note that the Kaiser window is used and it requires you to put 5 commas after the bracket as part of the fs function. In all ADS functions, you can choose disregard any argument by using the comma.

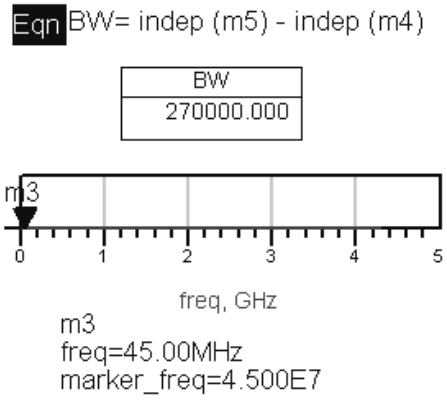
g. Now plot the marker_spectrum equation and move the marker. You will see the plot update the spectrum:

```
Eqn marker_spectrum=dbm(fs(Vout[::,freq_index],,,,,"Kaiser"))
```

h. Put two markers on the spectrum, 270 KHz apart as shown and write an equation BW using the independent variable of the marker (freq) which is the x-axis value. Insert a list of the BW equation and then move the marker to get BW and spectrum – BW should stay the same.




i. Save the data display and the schematic.

## 17. Use a CDMA source on the mixer

**Note on CDMA**: Unlike GSM, the CDMA modulation is more complex because it uses both phase and amplitude modulation. CDMA covers a wider bandwidth (bit rate) and uses multiple codes layered across the entire band.

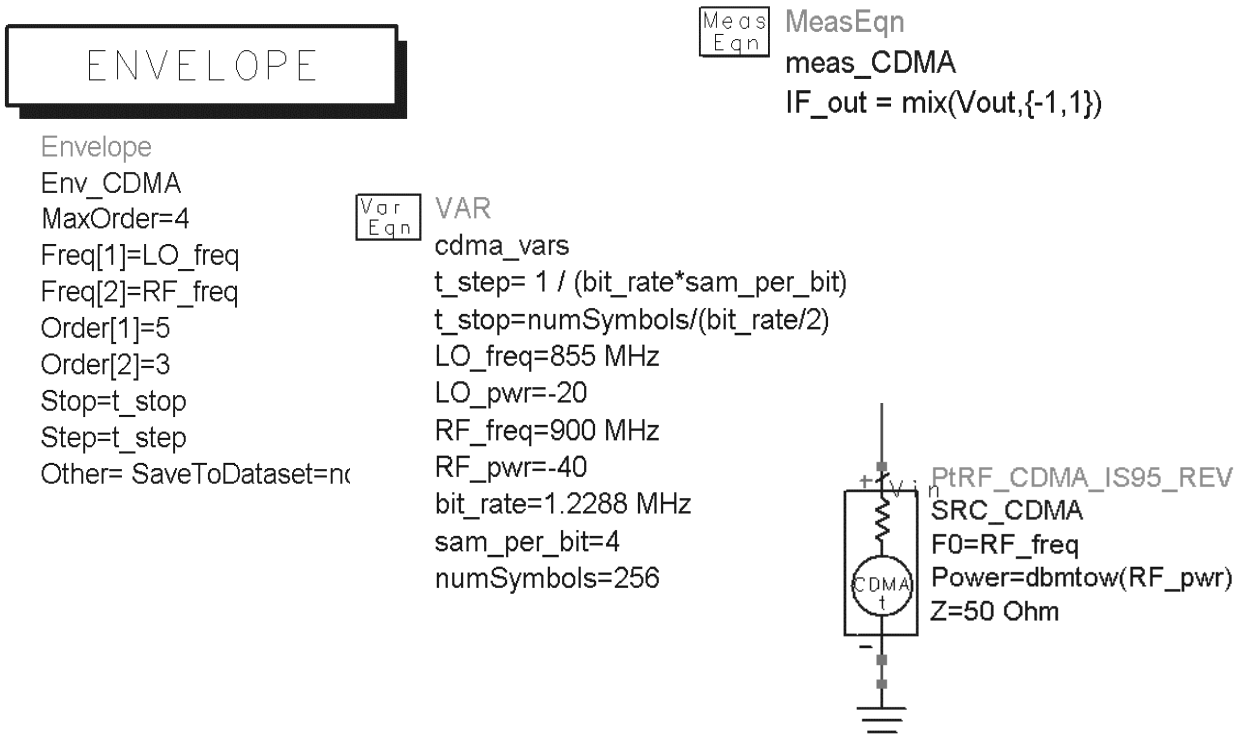
a. If you have saved the last schematic, you can save it with a new name (**ckt_env_mix_CDMA**). Then modify the controller, source, variables and add a MeasEqn as outlined in the following steps.

b. Set up the Envelope simulation controller as shown. The t_step and t_stop values are assigned in the variable block. Also, the last setting (Other = SaveToDataset = no ) means that no data will be written to a dataset except for the measurement equation data – this saves a lot of memory because CE datasets are very large compared to s-parameter.

c. Insert a measurement equation (MeasEqn) and write it as shown were you are using the mix function to get the 45 MHz envelope at Vout.

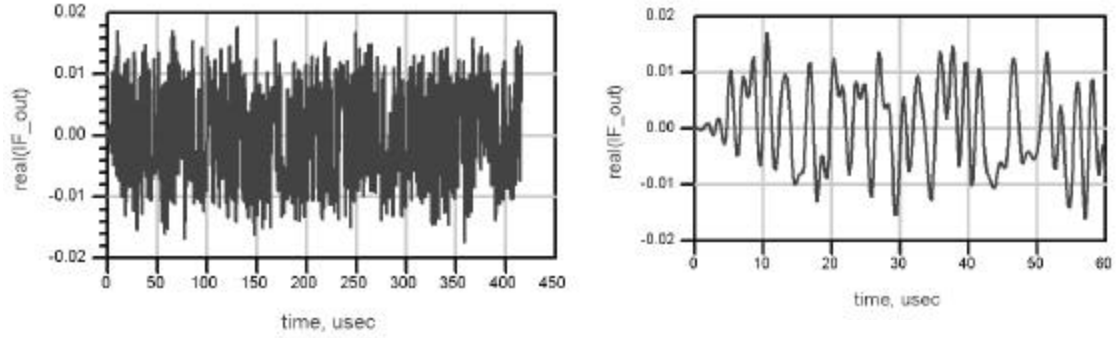
d. Set up the variables (VarEqn) as shown. The step and stop times are using the bit rate, samples per bit, and the number of symbols to accurately sample the CDMA signal.

e. Replace the GSM source with a PtRF_CDMA_ESG_REV modulated source. ESG (electronic signal generator) is one Agilent model.

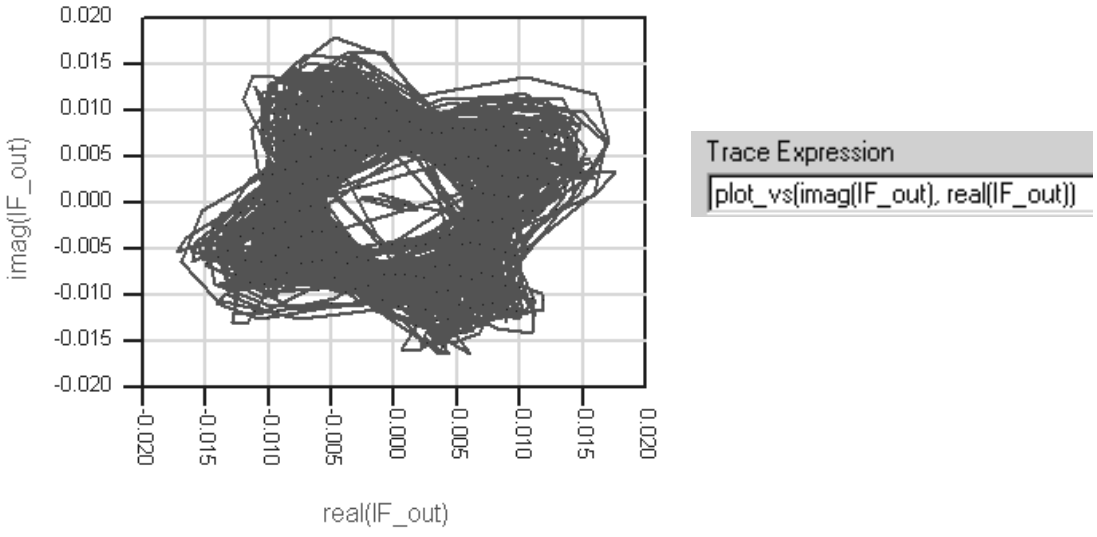


f. Use the dataset name: **ckt_env_mix_CDMA** and **simulate**.

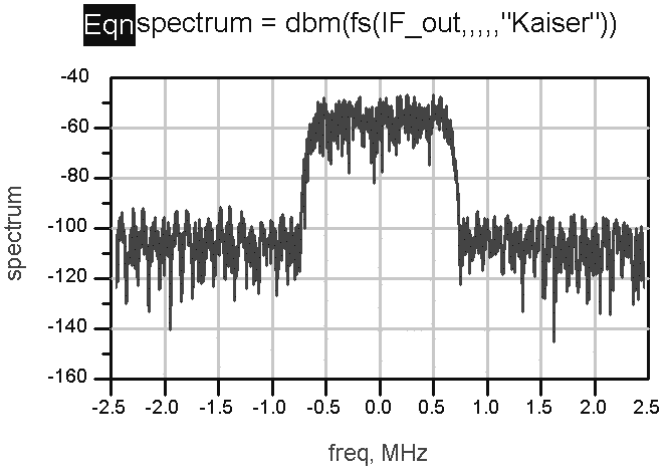
g. Open new data display where the default dataset is the one you just simulated. Insert a plot of your measurement equation: real (IF_out). It should be the only data written into the dataset. If you zoom into the first 50 usec of the plot you will see the deviation from zero, indicating that the real or I data is varying. But to see this more clearly, a trajectory diagram would be better (next step).




h. Insert a plot of the imaginary vs real part of the signal. Now you can see that the four symbol states are separated as two I and Q (real and imaginary) related to the magnitude and phase of the modulation. However, the diagram is rotated slightly and this is due to filtering in the CDMA source causing some delay. Example files of other modulation types such as Pi4QPSK and others are available with equations to calculate circuit delay and rotation.



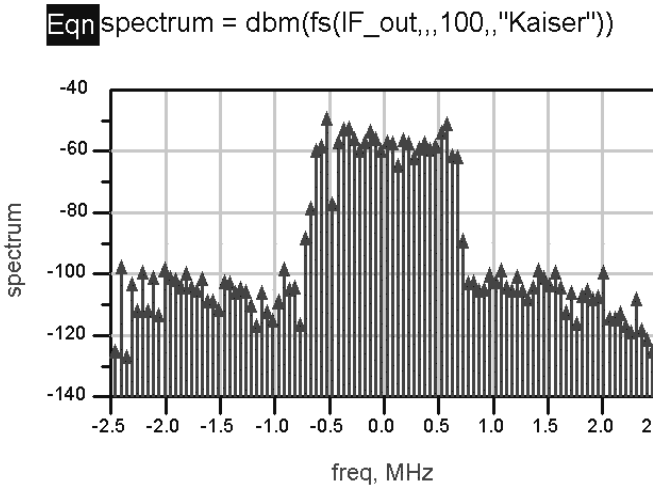
i. Write another equation to plot the spectrum of the IF_out signal and insert the plot as shown. This is centered around the IF_out frequency of 45 MHz which was indexed in your IF_out equation: mix(Vout,{-1,1}).

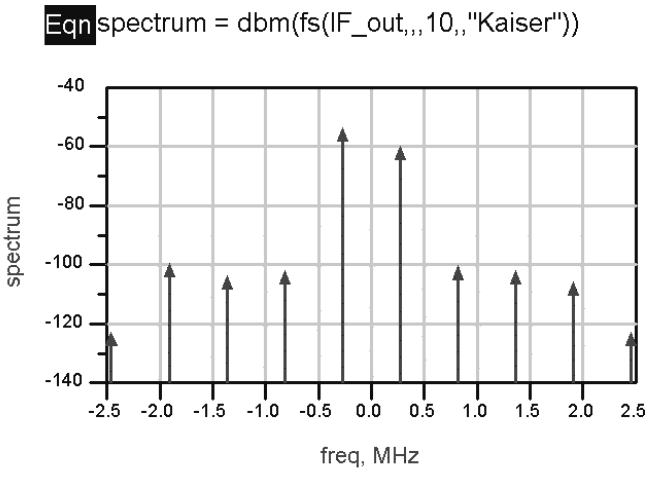



Spectral density of the Circuit Envelope data: IF signal with modulation where the BW is the CDMA BW of about 1.23 MHz..

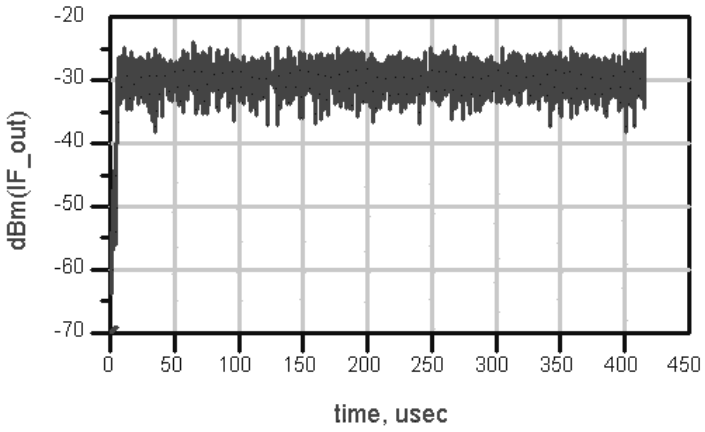
**Note on circuit envelope data and this plot**: The trace is in linear format and shows the spectral density of the circuit envelope data using all of the points sampled. For this reason, the power level is not near the –30 dBm level but must be integrated to give the actual power level.

j. To visually see the effects of how the number of envelope data points is used with the fs function, edit the equation by putting the number **100** after the 3rd comma. Also, you must change the Trace Type to **Spectral**. Afterward, try 300 or 10. The fs function and its arguments can be viewed using Function Help for the fs function and it specifies the number you can refer to it.



k. Plot dBm of IF_out and you will see the IF signal power over the simulation time. As you can see, the power is near the –30 dBm expected after settling. However, the exact power in the spectrum must be calculated using an equation (next step).




l. To accurately calculate the power in the spectrum, use the **channel_power** function shown here and write two equations. The first equation, limits, is the CDMA BW, and is used in the power calculation function. The vr argument means that it uses voltage instead of current . Go ahead and write the equations and list the channel power which should be very close to –30 dBm which is the input power (RF_in) amplified by the conversion gain. This is also the type of calculation performed on amplifiers but it also shows the mixer power integrated over the envelop spectrum.

**channel_power_vr**

**Purpose**

Computes the power (in watts) in an arbitrary frequency channel following a Circuit Envelope simulation

**Synopsis**

*Channel_power=channel_power_vr(voltage, resistance, mainCh, winType, winConst)*

Eqn limits={-(1.2288 MHz/2),(1.2288 MHz/2)}

Eqn channel_pwr=10*log (channel_power_vr(IF_out,50,limits,"Kaiser")) +30

| channel_pwr |
|---|
| -29.184 |

## *EXTRA EXERCISES:*

1. Sweep LO power and watch the change in the output.

2. Use the demodulator on the output and re-run the simulations.

3. Go to the example file: examples\Tutorial\ModSources_prj\Pi4DQPSK and copy the source and data display into your directory and try that source on the mixer, using the data display as a reference to guide you.

4. Put the filter the output and re-run the simulations.

THIS PAGE LEFT INTENTIONALLY BLANK