Sample Online Lab Notebook

Tuesday, August 27, 2013, 09:36

I was having trouble getting reasonable signals from the LabJack U3-HV, which were incredibly noisy and not at all like the pure sine wave I was sending in from the Agilent 33220A function generator. Ultimately, I tracked the problem down to a faulty pair of clipleads, so I threw them away. However, I would like to explore the resolution of the LabJack and plan to look at the noise level on digitized sine waves for various frequencies and settings of the LabJack.

I will modify the streamU3.py program to use a single channel and to record a number of samples at steady rate. How many? I'd like to go for a fairly large number to get good statistics. Let's try N = 1024.

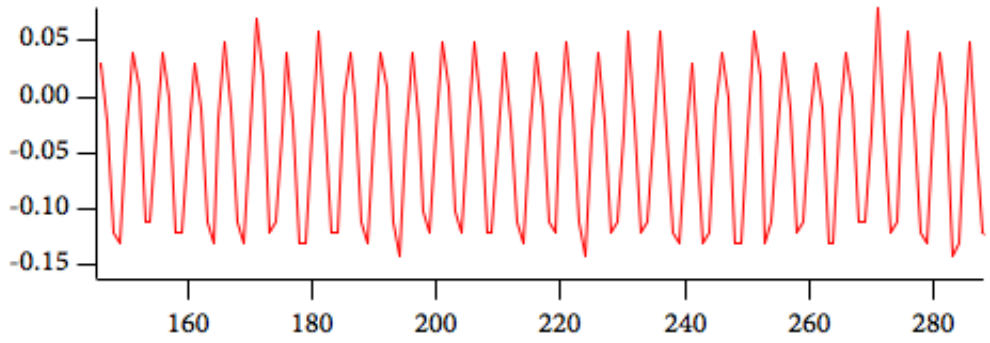
Tuesday, August 27, 2013, 10:01

After plenty of stupidity, I have a working program!

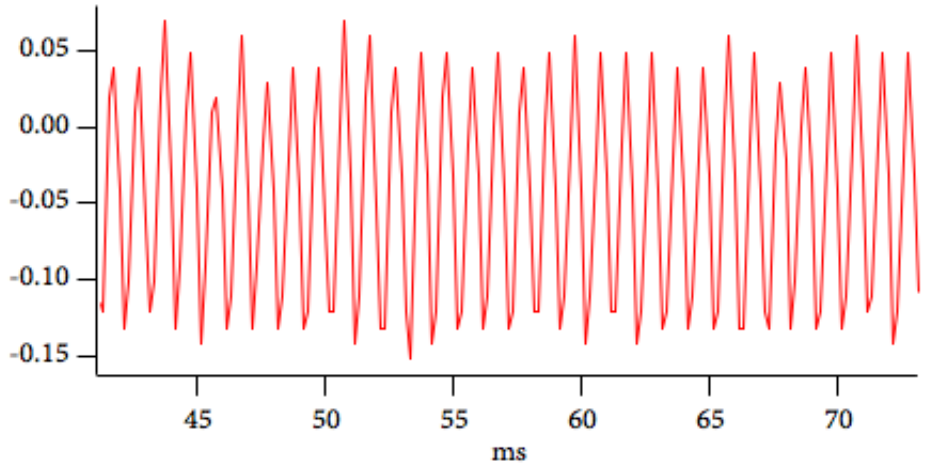
```
# streamU3.py
# Peter N. Saeta, 2011 July 2
# modified 2013 August 27
# Using the LabJack U3, sample four voltages at a fixed sampling rate
# and log the results to a text file. This file can then be used by
# Igor to manage further analysis

import u3
from time import sleep
from math import sqrt

numSamples = 1024

# Prepare the u3 interface for streaming

d = u3.U3()          # initialize the interface; assumes a single U3 is plugged in
to a USB port
d.configU3()    # set default configuration
d.configIO( FIOAnalog = 1 )         # ask for analog inputs

# The following requests 4 analog input channels to be streamed,
# with the positive terminals being 0-3 and the negative terminals
# all set to 31. The sampling frequency is 5000 samples (of each channel)
# per second. The Resolution parameter sets the effective quality of the
# samples. See http://labjack.com/support/u3/users-guide/3.2
d.streamConfig( NumChannels = 1,
     PChannels = [ 0,],
     NChannels = [ 31,],
     Resolution = 3,
     SampleFrequency = 5000 )

#d.packetsPerRequest = 1000

# Try to measure a data set.
def measure():
     vals = []
```

```
	try:
		d.streamStart()
		print( "starting to stream")
		for r in d.streamData():
			if r is not None:
				if r['errors'] or r['numPackets'] != d.packetsPerRequest or
r['missed']:
					print "error"
					break
				# append these values
				v = r['AIN0']
				vals.extend( v )
				if len(vals) >= 1024:
					break
	finally:
		d.streamStop()
	return vals

# Write a set of data to a file "myGloriousData.txt"
def writeData( vals ):
	f = open( 'myGloriousData.txt', 'w' )

	for i in range(0, len(vals)-1):
		f.write( '{0:.6f}\n'.format( vals[i]) )
	f.close()

writeData( measure() )
```

Now I need to read those values into Igor.



Hmm, that doesn't look too promising. I think I either need to sample faster or slow down the sine wave. It would be nice if the Python program would tell Igor its sampling rate. I could write that in the first line embedded in the wavename.



Tuesday, August 27, 2013, 10:07

Now I want to automate the read operation and combine it with setting the sample rate in Igor. From the history area, I see that the command I need is

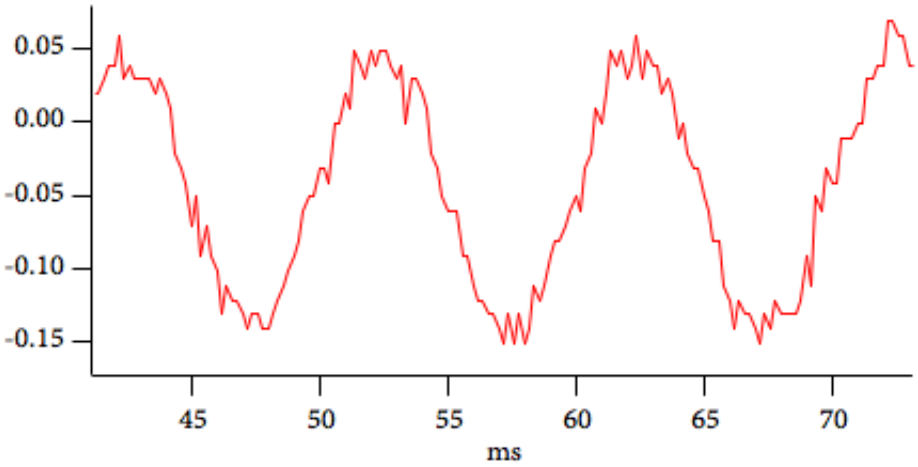
setscale/P x 0,(1/5000),"s",Freq5000

I'll write a little function to do this.

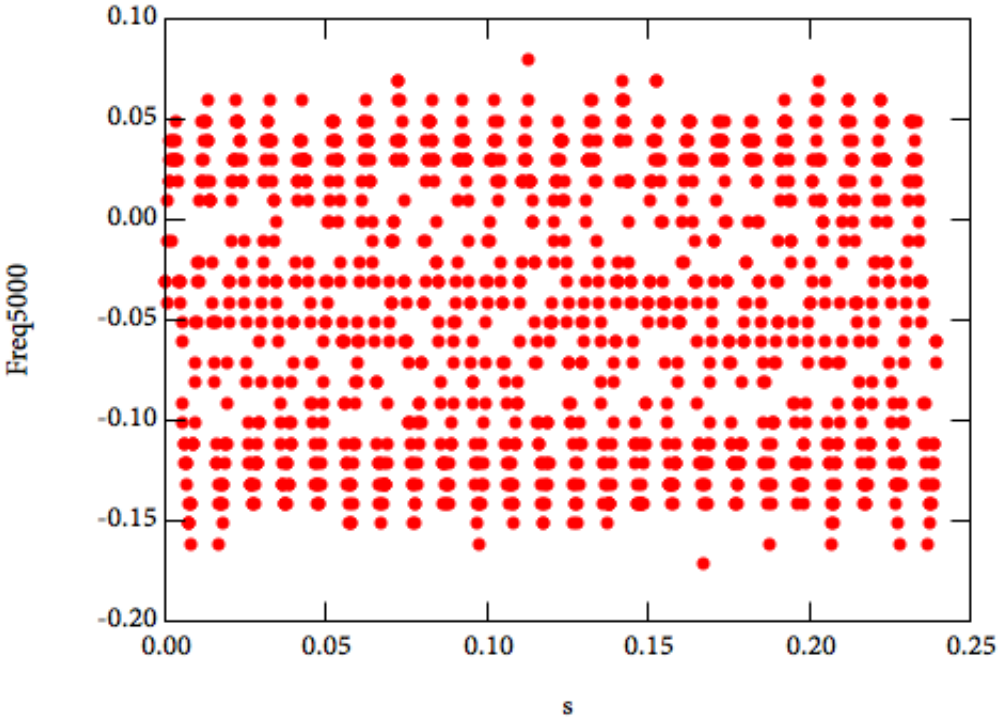
Tuesday, August 27, 2013, 10:13

Here's the load() function:

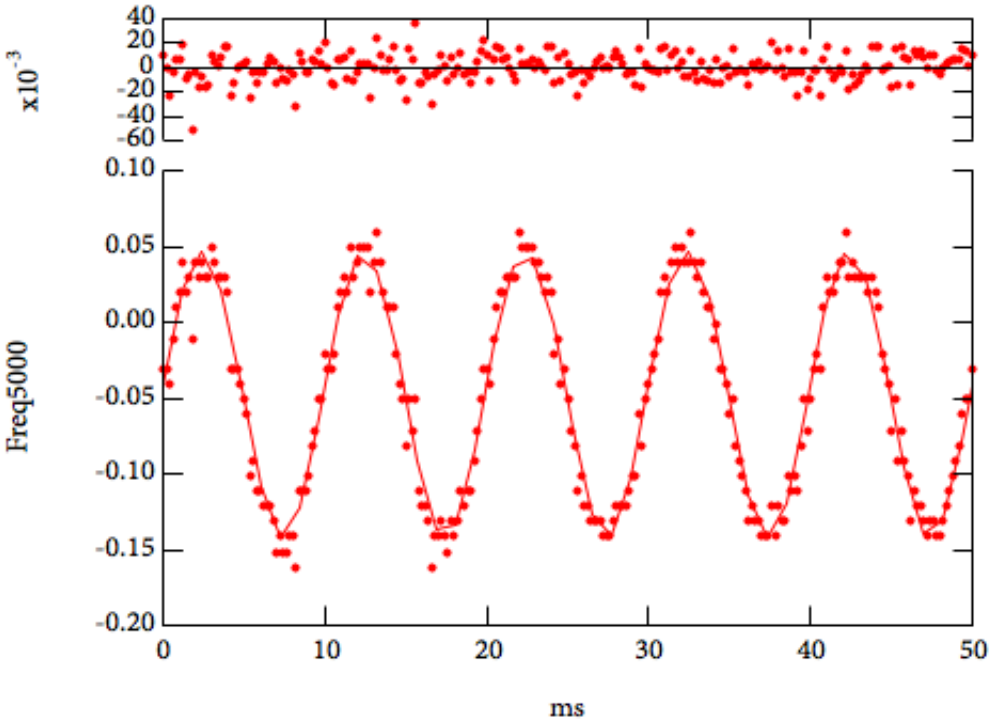
```
Function load()
     //First load the data in myGloriousData.txt
     LoadWave/A/O/J/D/W/K=0
"HD8:Users:saeta:Documents:Courses:cl57:2013:myGloriousData.txt"
     //   According to the documentation for LoadWave, the name of the wave(s)
that are loaded are stored
     //   in S_waveNames. We should use that to set the scale for the wave. I'll
use the stringFromList
     //   function in case LoadWave appends a semicolon to the name
     String wname = stringFromList( 0, S_waveNames )
     //   The form of wname should be Freqnnnn, where nnnn is the frequency.
Let's extract it
     Variable freq = Str2Num( wname[4,inf] )
     setscale/P x 0,(1/freq),"s",$wname
     setscale d 0,0,"V",$wname
End
```



Wow, that looks a bit noisy. I guess I should be plotting dots and then run a fit through the data to look at the residuals.



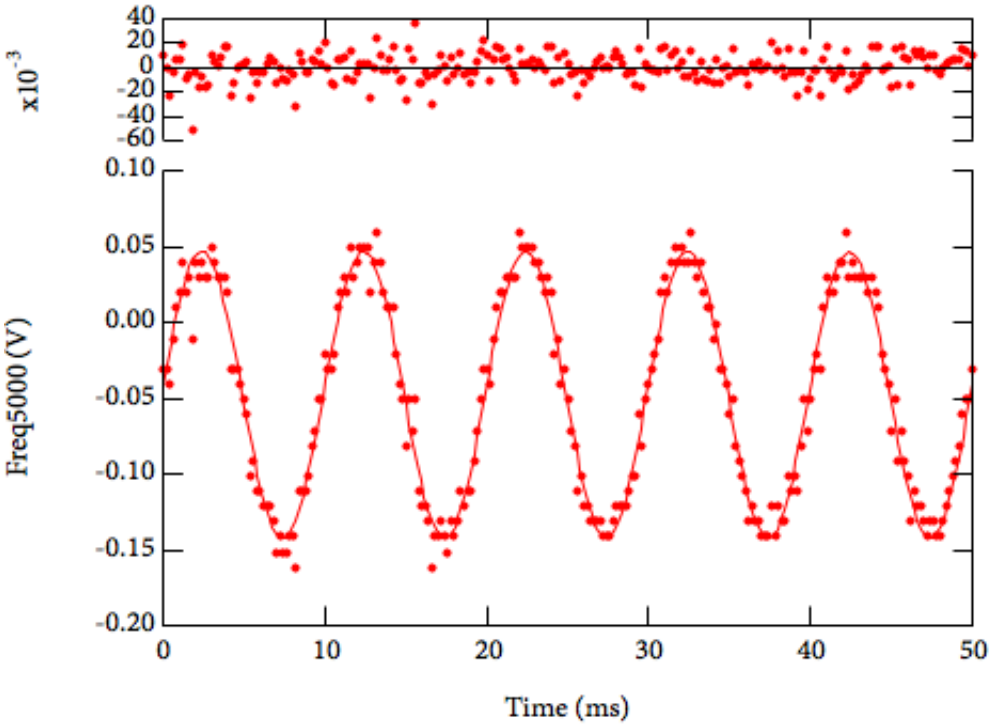
I think I need to zoom in a bit more.



Function: y0+A*sin(f*x+phi)
Coefficient values (**no uncertainties used**)

y0	=-0.047784
A	=0.094455
f	=628.39
phi	=0.064966

I think the jagged lines in the fit are an artifact of plotting. If I override the "auto" setting on the length of the fit curve, I get



Function: sin

Coefficient values ± one standard deviation

y0	=-0.047784 ± 0.000333
A	=0.094455 ± 0.000471
f	=628.39 ± 0.0721
phi	=0.064966 ± 0.00996

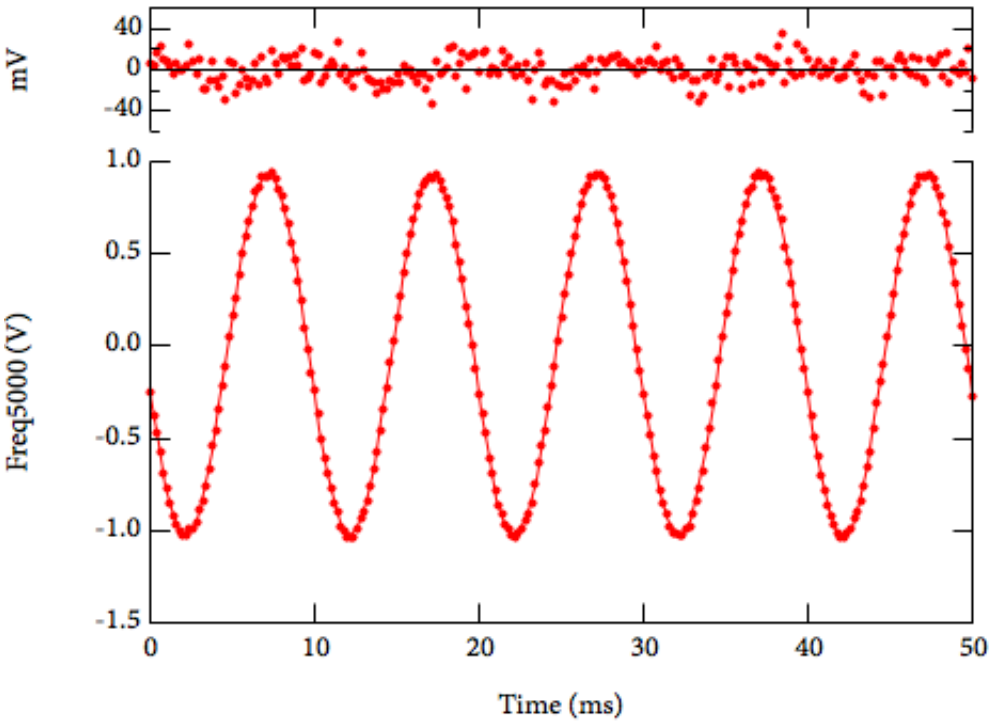
Huh. Why is there a y0 offset? The offset on the function generator says 0, and the amplitude is supposed to be 100 mV, peak to peak. Seems like somebody's a little off. The frequency looks okay, since 628.39 / 2π is 100.011 Hz.

Tuesday, August 27, 2013, 10:18

So now, how do I figure out the noise? I think I can just take statistics on the residuals.

```
WaveStats Res_Freq5000
  V_npnts= 1199; V_numNaNs= 0; V_numINFs= 0; V_avg= 7.98942e-13;
  V_Sum= 9.57931e-10; V_sdev= 0.0115112; V_sem= 0.000332438;
  V_rms= 0.0115064; V_adev= 0.00907195; V_skew= -0.0994873;
  V_kurt= 0.353478; V_minloc= 0.0018; V_maxloc= 0.0738;
  V_min= -0.0509646; V_max= 0.037533; V_minRowLoc= 9;
  V_maxRowLoc= 369; V_startRow= 0; V_endRow= 1198;
```

So the rms noise appears to be 0.012 V. Let's try a larger signal amplitude.



Function: sin
Coefficient values ± one standard deviation

y0	=-0.047349 ± 0.000372
A	=0.98363 ± 0.000527
f	=628.46 ± 0.00774
phi	=-2.9263 ± 0.00107

Now the rms voltage of the residuals is 0.013 V. Pretty similar. So is the offset. I'd like to automate a bit more. Let's modify load() so it does the fit and computes the rms noise.

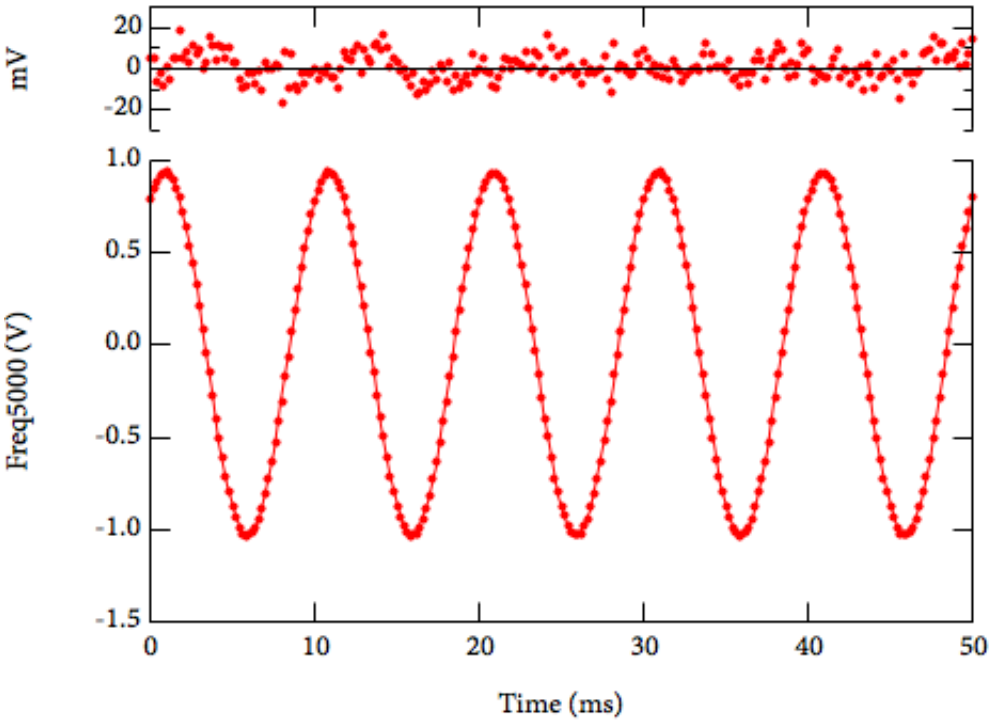
Tuesday, August 27, 2013, 10:30

Now the function load() looks as follows:

```
Function load()
     //First load the data in myGloriousData.txt
     LoadWave/A/O/J/D/W/K=0
"HD8:Users:saeta:Documents:Courses:cl57:2013:myGloriousData.txt"
     //   According to the documentation for LoadWave, the name of the wave(s)
that are loaded are stored
     //   in S_waveNames. We should use that to set the scale for the wave. I'll
use the stringFromList
     //   function in case LoadWave appends a semicolon to the name
     String wname = stringFromList( 0, S_waveNames )
     //   Make a reference to the data
     WAVE w = $wname
     //   The form of wname should be Freqnnnn, where nnnn is the frequency.
Let's extract it
     Variable freq = Str2Num( wname[4,inf] )
     setscale/P x 0,(1/freq),"s",w
```

```
      setscale d 0,0,"V",w
      //   Run the curvee fit
      CurveFit/Q/B=20 /L=512 /NTHR=0/TBOX=784 sin  w /D /R
      //   Identify the residual wave and compute statistics
      WAVE res = $("Res_" + wname)
      WaveStats/Q res
      //   Now print out the rms voltage
      printf "V_rms = %0.1f mV\r", V_rms * 1000
End
```

I get rms noise of 12.9 mV. I wonder what happens if I change the Resolution number in the Python program, which is 3 now. I'll try 1.

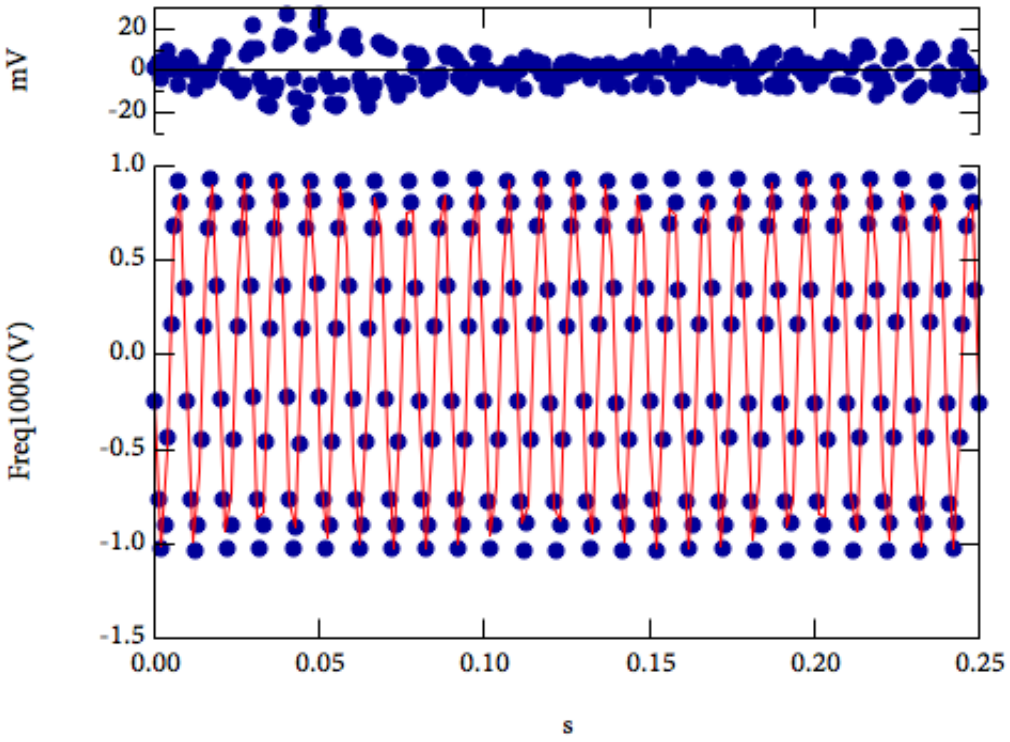


Function: sin
Coefficient values ± one standard deviation

y0	=-0.045268 ± 0.00018
A	=0.9848 ± 0.000255
f	=628.35 ± 0.00374
phi	=0.99868 ± 0.00052

V_rms is 6.2 mV. Wow, that's a bit better. Let's try 0.

error! I guess at Resolution 0 it can't keep up with the 5000 Hz sample rate.



Function: sin
Coefficient values ± one standard deviation

y0	=-0.046115 ± 0.000193
A	=0.9819 ± 0.000273
f	=628.33 ± 0.000481
phi	=-2.9342 ± 0.000555

V_rms is 8.6 mV.

Tuesday, August 27, 2013, 10:38

So I guess there is a tradeoff between sample rate and accuracy. Not surprising. Summarizing:

Resolution	rms error	Sample rate
0	8.6	1000 samples/s
1	6.2	5000 S/s
3	12.9	5000 S/s

I should look up the expected values on the LabJack website.