

How to Obtain Noise Spectra and Obtain Faster Data using SignalScope2

SignalScope2 is a program for MACs which enables us to acquire signals at a higher rate than is possible with the Host computer and USB interface. The purpose of this program is to allow us to record two independent signals at 44 kHz each. The signals are fed into the digital stereo audio input of the Host computer. By using this program in "FFT mode", we can display frequency spectra in real time. By using it in "Oscilloscope mode", it is possible to acquire fast waveforms at 44 kHz and record them in text output files for later analysis. This program is available from Faber Acoustical at the web page

<http://www.faberacoustical.com/products/signalscope/>.

We use this program:

- To make spectral measurements that help locate sources of electrical interference
- To obtain force calibration and trap stiffness from power spectra of thermal noise.
- To take high speed signal traces to observe fast molecular transitions.

How to install SignalScope2 .

This program can be downloaded from Faber Acoustical on a trial basis, but it is necessary to buy a license for permanent use. Such a license typically costs \$99 for schools.

You must connect at : <http://www.faberacoustical.com/> --> PRODUCTS --> SIGNALSCOPE -->DOWNLOAD.

Directly the file SignalScope.dmg will be downloaded and if you agree, the program will be installed in your computer. You can try a free version for 30 days however while you do not have the number of the license, you can not record data. To test that the program is measuring correctly, you can open the window of FFT analyzer and watch as the graph responds to any kind of noise, for example at your voice. These signals now enter to the host computer by the built-in Microphone. If you use a MacMini computer the signals enter only by the external input jack because this computer does not have a microphone.

In the electronic tower there are four preamp boards, each one with a stereo phono jack, whose outputs work in a frequency range between 0-100kHz. Each preamp board collects the signal from a different photodetector: PSD trap A, PSD trap B, Lightlever trap A and Lightlever trap B. In the next picture (See fig.1) you can see the preamp boards corresponding to the trapB and the wires coming from the photodetectors. Next to these boards (but on the ADC board) are placed the stereo phono jacks. Each jack collects two signals, the components x and y of each PSD. In the fig 2 it is possible to see a view of the 4 stereo jack and the signals that we can obtain from them. We are interested in collecting the data from the stereo phono jack and

sending the signals to the host computer's audio input for analysis by the SignalScope program. This task can be done at the same time that the Host computer is running the miniTweezers GUI program and collecting its own data through the USB port, albeit at a slower rate.

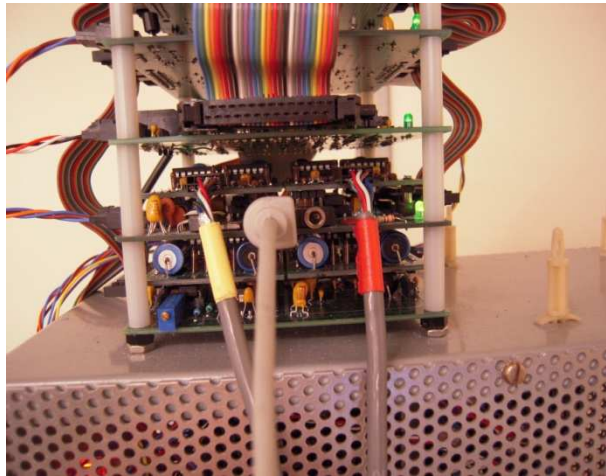
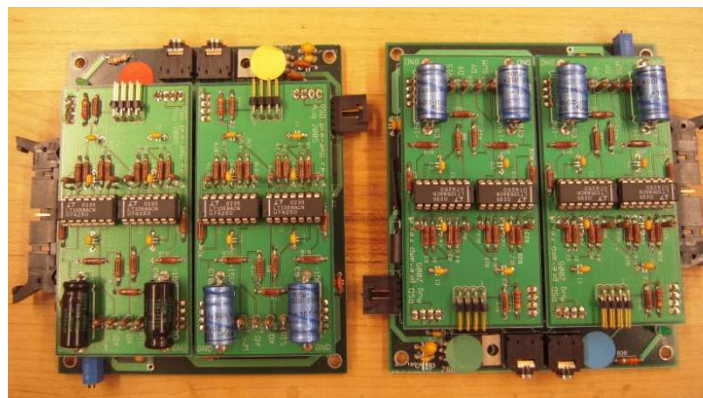


fig.1

The signals that we obtain are:



Jack 1 from GREEN input: $xForceA + yForceA$

Jack 2 from BLUE input: $xDistanceA + yDistanceA$

Jack 3 from YELLOW input: $xForceB + yForceB$

Jack 4 from RED input: $xDistanceB + yDistanceB$

How to connect the preamp boards with the host computer

To carry out these measurements it is necessary to connect the preamp boards with the host computer. This setup is done by using three stereo plugs (for instance Digikey CP-2207-ND) (See fig.3). If you wish to switch between X and Y signals, a toggle switch (DPDT) is connected with the three stereo jacks, with two input stereo jacks (4signals+2Grounds) and one output stereo jack (2 signals+1Ground). The input stereo jacks are plugged in the preamp boards and the output stereo jack is connected to the host computer(See fig.4).

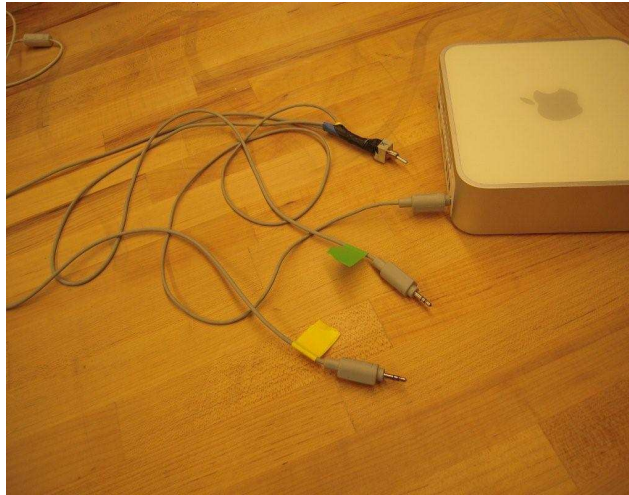


Fig.3

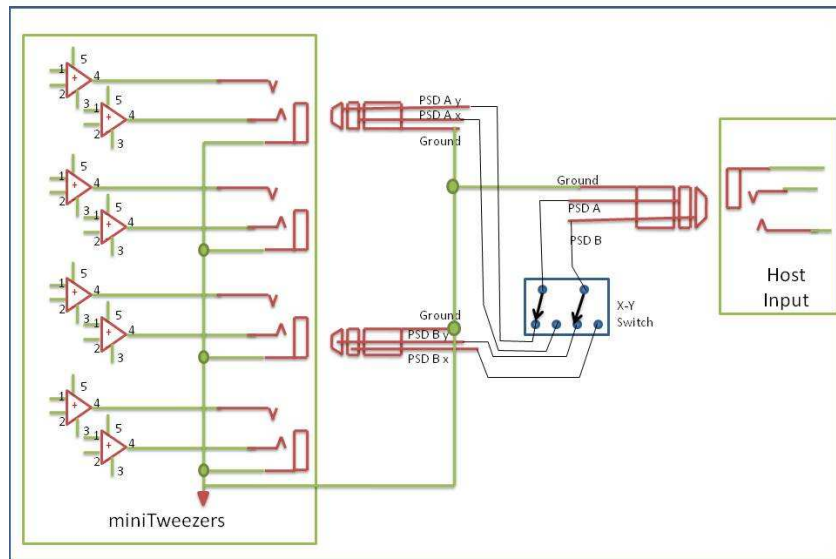

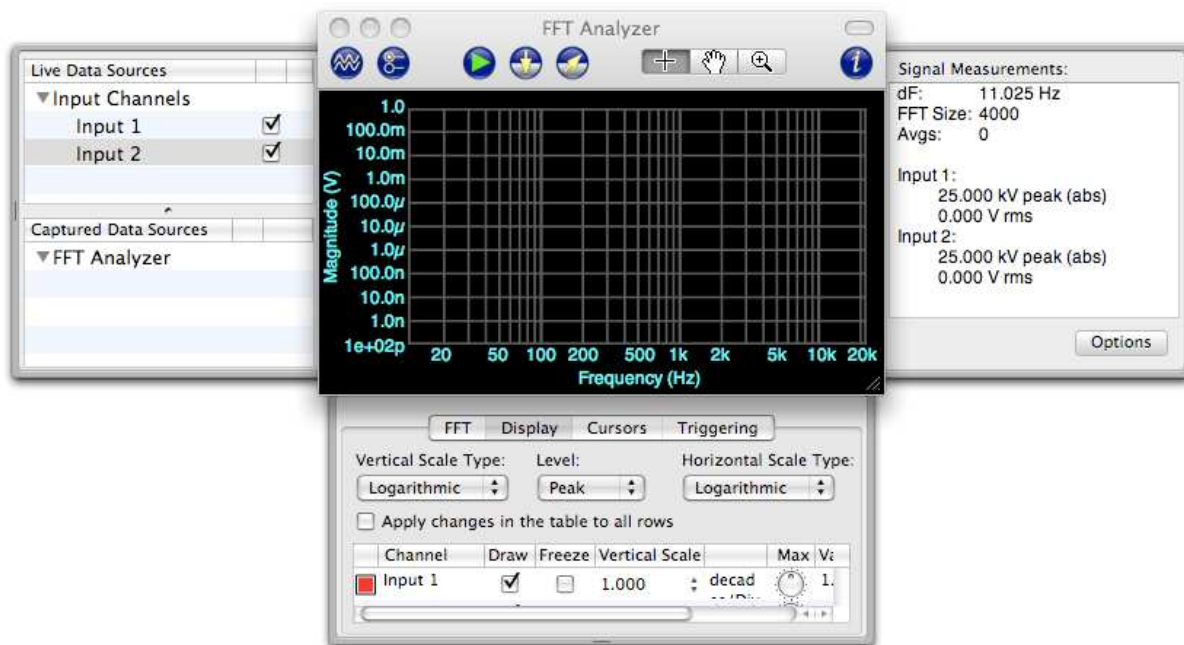


Fig.4

To calibrate the force, we need to obtain the signals of force PSD for both traps. For this reason the stereo jacks will be connected in the GREEN and YELLOW.

When we have made the electrical connections between the miniTweezers and the host computer, we can run the SignalScope2. The main window will appear. Under “Input Device”. select “Built-In Input” .

There are 4 function buttons to select from. To obtain the trap stiffness or the force calibration by thermal fluctuations, use the “FFT Analyzer” button.  That will get us the FFT window:



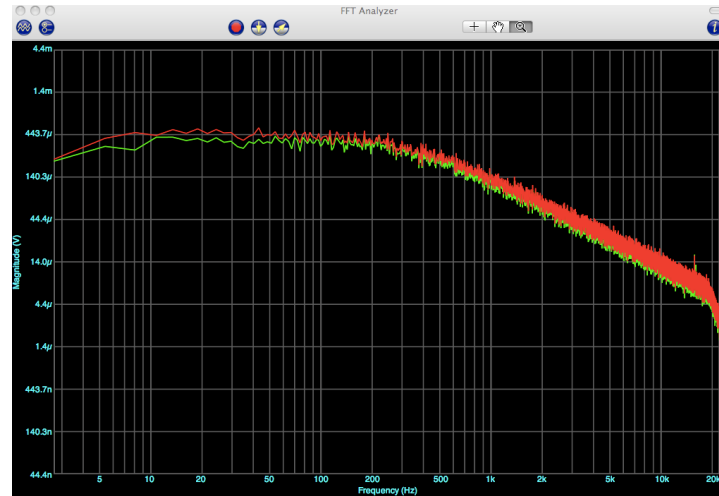
In first place, we must select two line input channels, whereas the default selection is for only one input. Check “Input 1” and “Input 2”.

Next, it is possible to select between different numbers of “spectral lines”, a good option is to put 8192 lines. You can obtain more information about this subject in: http://www.fabracoustical.com/products/signalscope/fft_analyzer/fft_lengths/.

To obtain a good looking power spectrum, it is necessary to select the vertical and the horizontal axis in logarithmic scale. You can make this choice in the Display window. You can also change the scales of both axes.

You must start the measurement by pushing the green arrow (which turns to a red dot). After a stable spectrum has been obtained on the screen, use the “Capture” button (next to the red dot) to record the data showed in the display. Notice that a pair of file names appears in the left side-bar window, one for each input. You can change the name of these files by clicking twice on the file names. Use the capture button again and again, each time you want to record

a spectrum. Thus you will make a long list of file names. Finally use the “Export” button and the captured files will be saved in .txt or .mat files in a folder. Alternately you could capture the screen with the GRAB utility. The graphs below show us some spectral measurements obtained in different experiments.



How to analyze the Power Spectrum

In the appendix below, you can find an analysis program called test.py that allows us to analyze the spectra obtained with SignalScope. This program has been made in Python, a high-level programming language that is free for Macintosh (<http://www.python.org/>). If the computer is Macintosh the program is already installed in *home/usr/bin/python*. However, the analysis program requires a pack of libraries that are available online. Go to www.scipy.org/Download --> *Scipy* --> *Source Forge download site for Scipy* and download “Superpack_2009.06.22sh”

To use the analysis program, it is necessary to open a terminal window by going to Applications -> Utilities->Terminal. Then use the command *cd* to place the terminal program in the same folder where you have the analysis program. For instance, if you have the analysis program in *Desktop* and your terminal is in */home*, you must write *cd /home/Desktop*. It is useful to know that the command *ls* lists the files that reside in your current directory. Now, you can execute the program using the commands *python test.py*. Note the spectrum TXT files must be placed in the same directory/folder as the program.

The program asks you for:

→Enter the name of the file (Trap A):

→Enter the name of the file (Trap B):

At these questions, you must write the name of the file (without .txt) that you have obtained with the SignalScope program.

→How many pN are 1 Volt ?:

To answer this question is necessary to know the calibration factor between pN and Volts. This parameter depends on the miniTweezers machine that you are using. Before using this program, you must first make a measurement with a voltmeter to see what voltage comes out of the signal jacks when a given force is sensed by the trap. Here you would place a bead on a pipette, and place both traps within the bead, and call for *Constant Force* mode *CF1*, and specify a force of 0 (zero) picoNewtons. Then record the Y-axis voltage from both trapA and trapB. Then call for *Constant Force* mode *CF2*, and specify a force of 20 pN. Again note the voltages from the two Y-axis outputs. Compute the voltage difference for each output between the two forces. That voltage difference corresponds to a force change of 10 pN in each trap. Now divide by 10 to get the volts per one pN change, and invert that number to get the picoNewtons per volt. That number will remain fixed for that particular miniTweezers machine.

The program gives us some numerical values such as the Value of the Plateau, Value of the corner frequency, Value of the area under curve. Also the value of the stiffness and the value of the radius of the bead are obtained. Note, the viscosity of water must be corrected in the program to get an accurate value.

Appendix: a Python program to fit the Lorentzian function to noise spectrum from a text file

```
#!/usr/bin/python
import os, sys

#The package of the libraries can be downloaded of the web page:
#www.scipy.org/Download/scipy superpack for Python 2.5

import matplotlib.pyplot as plt
import scipy
import pylab
from scipy.optimize import leastsq
from scipy.integrate import inf
from math import sqrt

#Definition of variables
##TRAP A
xaxisa = []
yaxisa = []
Lorentziana = []
pnamea = []
pa = []
p0a = []
```

```

xaxis2a = []
la = []
timea = []
forcea = []
force2a = []

##TRAP B
xaxisb = []
yaxisb = []
Lorentzianb = []
pnameb = []
pb = []
p0b = []
xaxis2b = []
lb = []
timeb = []
forceb = []
force2b = []

pi = 3.1416

#Reading the data of the Signal Scope
##TRAP A
ext = ['.txt']
filename = raw_input('Enter the name of the file (Trap A):')
i=0
f = open(filename + ext[0], "r")

for line in f.readlines():

    i=i+1
    if i>29:
        templine = line.split()
        if len(templine) is 0:
            break
        xaxisa.append(float (templine[0]) )
        yaxisa.append(float (templine[1]) * float (templine[1]) )
        xaxis2a.append(float (templine[0]) * float (templine[0]) )

#Fitting to a Lorentzian function
def residuals(pa, yaxisa, xaxisa):
    erra = yaxisa-Lorentziana(xaxis2a,pa)
    return erra

def Lorentziana(xaxis2a , pa):
    return pa[0] / (pa[1]+ xaxis2a)

a_0a = 0.225
b_0a = 40000
#a_0 = float(inout("Value of a:"))
#b_0 = float(input("Value of b: "))

#Calculating the fitting parameters
pnamea = (['aa','ba'])
p0a = ([a_0a , b_0a])
plsqa = leastsq(residuals, p0a, args=(yaxisa, xaxisa))

```

```

#Adjusting the string: plsqa[0] in an array:1

s = plsqa[0]
la = list(s)

##TRAP B
#Reading the data of the file

ext = ['.txt']
filename2 = raw_input('Enter the name of the file (Trap B):')
j=0
g = open(filename2 + ext[0], "r")
for line in g.readlines():
    if len(line) is 0:
        break
    j=j+1
    if j>29:

        templine = line.split()
        if len(templine) is 0:
            break
        xaxisb.append(float (templine[0]) )
        yaxisb.append(float (templine[1]) * float (templine[1] ) )
        xaxis2b.append(float (templine[0]) * float (templine[0]) )

#Fitting to a Lorentzian function

def residuals(pb, yaxisb, xaxisb):
    errb = yaxisb-Lorentzianb(xaxis2b,pb)
    return errb

def Lorentzianb(xaxis2b , pb):
    return pb[0] / (pb[1]+ xaxis2b)

a_0b = 0.225
b_0b = 40000

#Calculating the fitting parameters
pnameb = (['ab','bb'])
p0b = ([a_0b , b_0b])
plsqb = leastsq(residuals, p0b, args=(yaxisb, xaxisb))

r = plsqb[0]
lb = list(r)

cfactor = raw_input('How many pN are 1 Volt?')

#Printing the value of the plateaus and th corner frequencies (cf) for each
trap.
plateaua= float(la[0]/la[1])
plateaub= float(lb[0]/lb[1])
cfa=sqrt(float(la[1]))
cfb=sqrt(float(lb[1]))
mcf=float ((cfa+cfb)/2)
print '*****'
print 'Value of the plateau (Trap A):' , '%.8f'%plateaua, 'V^2/Hz'

```



```

print 'Value of the plateau (Trap B):' , '%.8f'%plateaub, 'V^2/Hz'
print '*****'
print 'Value of the corner frequency (Trap A)' , '%.1f'%cfa, 'Hz'
print 'Value of the corner frequency (Trap B)' , '%.1f'%cfb, 'Hz'
print '*****'
print 'Value of the averaged corner frequency' , '%.1f'%mcf, 'Hz'
print '*****'

#Conversion factor between pN and V (1V=25pN)
plateaua_pn = plateaua*float(cfactor)
plateaub_pn = plateaub*float(cfactor)
mplateaul= sqrt(float(plateaua_pn))
mplateau2= sqrt(float(plateaub_pn))
plateau = float((mplateaul + mplateau2) * (mplateaul + mplateau2))
print 'Value of the plateau (TrapA):', '%.8f'%plateaua_pn, 'pN^2/Hz'
print 'Value of the plateau (TrapB):', '%.8f'%plateaub_pn, 'pN^2/Hz'
print 'Value of the plateau:', '%.8f'%plateau, 'pN^2/Hz'
print '*****'

#Calculate of the area of the Lorentzian fitting.
#The solution of the integral is given by plateau.pi.cf/2
areaa = float(plateaua_pn*pi*cfa/2)
areab = float(plateaub_pn*pi*cfb/2)
area = float (plateau*pi*mcf/2)
print 'Value of the area (Trap A):', '%.6f' % areaa, 'pN^2'
print 'Value of the area (Trap B):', '%.6f' % areab, 'pN^2'
print 'Value of the area:', '%.6f' % area, 'pN^2'
print '*****'

#Calculating the stiffness of the trap as area_under_Lorenzian =
kBT*stiffness
stiffnessa = float(areaa/4.11)
stiffnessb = float(areab/4.11)
stiffness = float (area/4.11)
print 'Value of the stiffness (Trap A):', '%.5f'% stiffnessa, 'pN/nm'
print 'Value of the stiffness (Trap B):', '%.5f'% stiffnessb, 'pN/nm'
print 'Value of the stiffness:', '%.5f'% stiffness, 'pN/nm'
print '*****'

# Considering the viscosity of the water (0.001N.s/m^2)*1000=0.001pN.s/um^2
radiusa = float(plateaua_pn/(4*4.11*0.001*6*pi*0.001))
radiusb = float(plateaub_pn/(4*4.11*0.001*6*pi*0.001))
radius = float(plateau/(4*4.11*6*0.001*pi*0.001))
print 'Value of the radius(Trap A):', '%.5f'% radiusa, 'um'
print 'Value of the radius(Trap B):', '%.5f'% radiusb, 'um'
print 'Value of the radius:', '%.5f'% radius, 'um'
print '*****'

print '>>>>To continue,you must close the window of the figure.'

#Plotting the graph Trap A
plt.loglog( xaxisa, yaxisa, '.' ,xaxisa , Lorentziana (xaxis2a,plsqa[0]),
'r')
plt.xlabel('Frequencies (Hz)')

```

```

plt.ylabel('Power Spectral Density (V^2)')
plt.axis(xmin=10, xmax=25000, ymin=0.0000000001, ymax=0.00001)
plt.grid(True)
#plt.title('Power Spectrum Trap A')
#plt.show()

plt.title('$\mathrm{Power\ Spectrum\ Trap A}$')

#Plotting the graph Trap B
plt.loglog( xaxisb, yaxisb, '.')
plt.loglog( xaxisb , Lorentzianb (xaxis2b,plsqb[0]), 'orange')
plt.xlabel('Frequencies (Hz)')
plt.ylabel('Power Spectral Density (Volts^2)')
plt.axis(xmin=10, xmax=30000, ymin=0.0000000001, ymax=0.00001)
plt.grid(True)
plt.title('$\mathrm{Power\ Spectrum\ Trap A+B}$ $')

plt.text(15, 0.00000002, 'cornerF(A)=''%1f'%float(cfa), color='r')
plt.text(15, 0.00000001, 'stiffness(A)=''%3f'%float(stiffnessa), color='r')
plt.text(15, 0.000000005, 'radius/bead(A)=''%3f'%float(radiusa), color='r')
plt.text(15, 0.000000002, 'cornerF(B)=''%1f'%float(cfb), color='orange')
plt.text(15, 0.000000001, 'stiffness(B)=''%3f'%float(stiffnessb),
color='orange')
plt.text(15, 0.0000000005, 'radius/bead(B)=''%3f'%float(radiusb),
color='orange')

ext=['.jpg', '.eps', '.ps', '.svg ']
r = filename

name_fich= r + ext[0]
plt.savefig(name_fich)

ext=['.jpg', '.eps', '.ps', '.svg ']
s = filename2
name_fich = s + ext[0]
plt.savefig(name_fich)

plt.show()

```