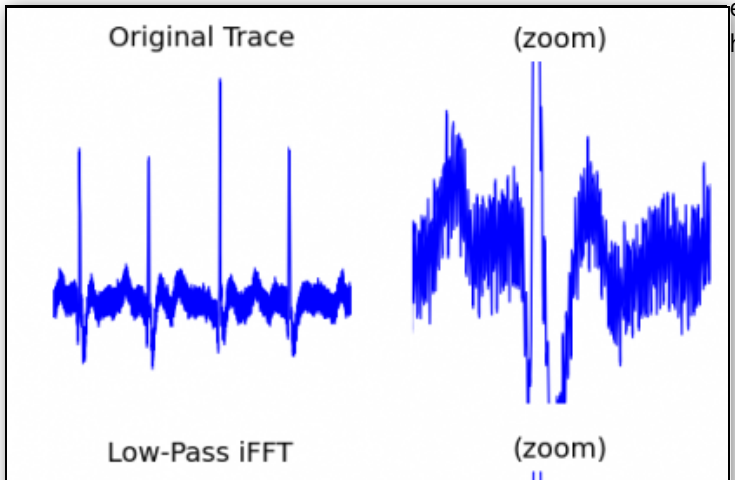
# Signal Filtering with Python

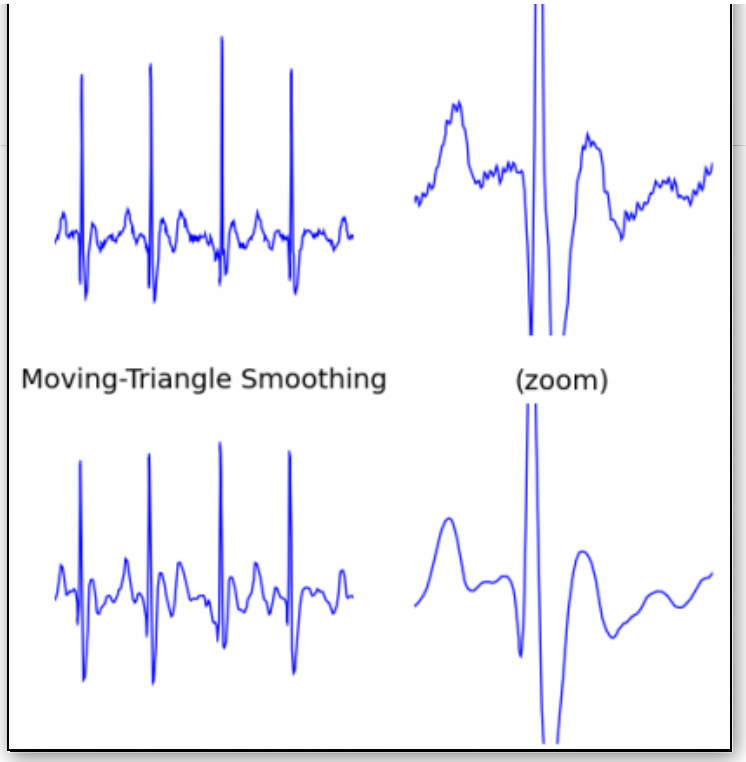
January 21, 2009 by Scott | DIY ECG, General, Python | Leave a comment

> ***UPDATE:*** *An improved ECG design was posted in August, 2016.*
> *Check out: https://www.swharden.com/wp/2016-08-08-diy-ecg-with-1-op-amp/*

**It's time for a lecture.** I've been spending a lot of time creating a DIY dlectrocardiogram and it produces fairly noisy signals. I've spent some time and effort researching the best ways to clean-up these signals, and the results are incredibly useful! Therefore, I've decided to lightly document these results in a blog entry.

**Here's an example of my magic!** I take a noisy recording and turn it into a beautiful trace. See the example figure with the blue traces. How is this possible? Well I'll explain it for you. Mostly, it boils down to eliminating excess high-






frequency sine waves which are in the original recording due to electromagnetic noise. A major source of noise can be from the alternating current passing through wires traveling through the walls of your house or building. My original ECG circuit was highly susceptible to this kind of interference, but my improved ECG circuit eliminates most of this noise. However, noise is still in the trace (see the figure to the left), and it needed to be removed.

**The key is the FFT (Fast Fourier Transformation) algorithm** which can get pretty intimidating to research at first! I'll simplify this process. Let's say you have a trace with repeating sine-wave-shaped noise. The output of the FFT transformation of the signal is the breakdown of the signal by frequency. Check out this FFT trace of a noisy signal from a few posts ago (top graph). High peaks represent frequencies which are common. See the enormous peak around 60 Hz? (Hz means "per second" by the way, so 60 Hz is a sine wave that repeats 60 times a second) That's from my AC noise. Other peaks (shown in colored bands) are other electromagnetic noise sources, such as wireless networks, TVs, telephones, and your computer processor. The heart produces changes in electricity that are very slow (the heartbeat is about 1 Hz, or 1 beat per second), so *if we can eliminate all of the sine waves with frequencies higher than what we want to isolate* we can get a pretty clear trace. This is called a band-stop filter (we block-out certain bands of frequencies) A band-pass filter is the opposite, where we only allow frequencies which are below (low-pass) or above (high-pass) a given frequency. By eliminating each of the peaks in the colored regions (setting each value to 0), then performing an inverse fast Fourier transformation (going backwards from frequency back to time),

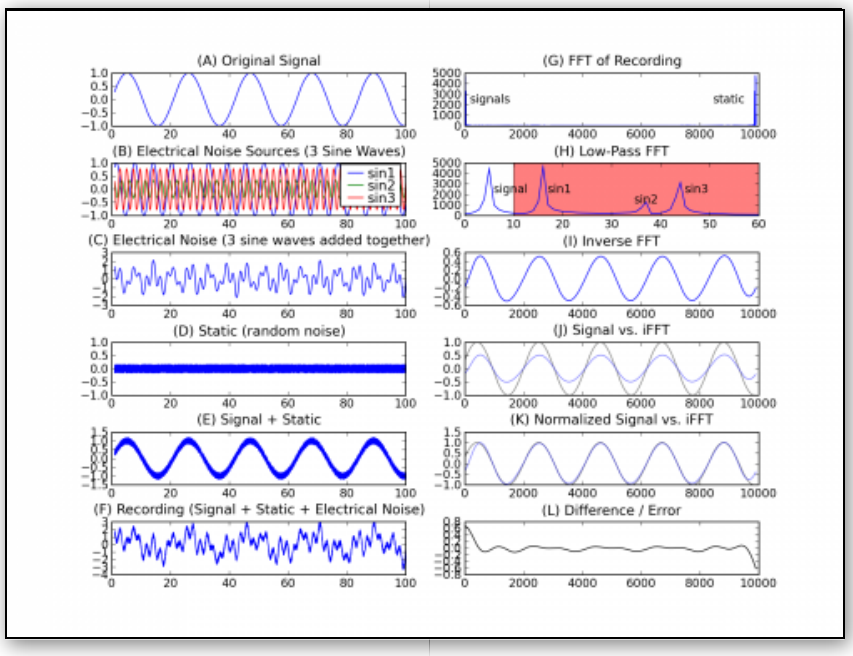


## Categories

C/C++ Circuitry DIY ECG Electronics General GitHub HAB (high altitude balloon) Linux Microcontrollers Molecular Biology My Website PHP Prime Numbers Programming Python QRSS / MEPT

the result is the signal trace (seen as light gray on the bottom graph) with those high-frequency sine waves removed! (the gray trace on the bottom graph). A little touch-up smoothing makes a great trace (black trace on the bottom graph).

**Here's some Python code to get you started in cleaning-up your noisy signals!** The image below is the output of the Python code at the bottom of this entry. This python file requires that test.wav (~700kb) (an actual ECG recording of my heartbeat) be saved in the same folder. Brief descriptions of each portion of the graph will follow.




**(A)** The original signal we want to isolate. (IE: our actual heart signal)

**(B)** Some electrical noise. (3 sine waves of different amplitudes and periods)

**(C)** Electrical noise (what happens when you add those 3 sine waves together)

**(D)** Static (random noise generated by a random number generator)

**(E)** Signal (A) plus static (D)

**(F)** Signal (A) plus static (D) plus electrical noise (C)

**(G)** Total FFT trace of (F). Note the low frequency peak due to the signal and electrical noise (near 0) and the high frequency peak due to static (near 10,000)

(manned experimental propagation transmistter)
RF (Radio Frequency)
Thermoregulation
UCF Lab
Uncategorized

## Categories

C/C++ (46)
Circuitry (103)
DIY ECG (16)
Electronics (15)
General (196)
GitHub (16)
HAB (high altitude balloon) (4)
Linux (15)
Microcontrollers (77)
Molecular Biology (3)
My Website (10)
PHP (4)
Prime Numbers (16)
Programming (4)
Python (69)
QRSS / MEPT (manned experimental propagation transmistter) (4)
RF (Radio Frequency) (84)
Thermoregulation (3)
UCF Lab (4)
Uncategorized (2)

## Archives

**(H)** This is a zoomed-in region of (F) showing 4 peaks (one for the original signal and 3 for high frequency noise). By blocking-out (set it to 0) everything above 10Hz (red), we isolate the peak we want (signal). This is a low-pass filter.

**(I)** Performing an inverse FFT (iFFT) on the low-pass iFFT, we get a nice trace which is our original signal!

**(J)** Comparison of our iFFT with our original signal shows that the amplitude is kinda messed up. If we normalize each of these (set minimum to 0, maximum to 1) they line up. Awesome!

**(K)** How close were we? Graphing the difference of iFFT and the original signal shows that usually we're not far off. The ends are a problem though, but if our data analysis trims off these ends then our center looks great.

**Here's the code I used to make the image:**

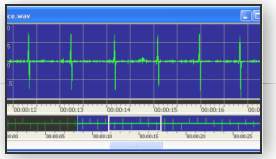
```
import numpy, scipy, pylab, random

# This script demonstrates how to use band-pass (low-pass)
# filtering to eliminate electrical noise and static
# from signal data!

###################
### PROCESSING ###
###################

xs=numpy.arange(1,100,.01) #generate Xs (0.00,0.01,0.02,0.03,...,100.0)
signal = sin1=numpy.sin(xs*.3) #(A)
sin1=numpy.sin(xs) # (B) sin1
sin2=numpy.sin(xs*2.33)*.333 # (B) sin2
sin3=numpy.sin(xs*2.77)*.777 # (B) sin3
noise=sin1+sin2+sin3 # (C)
static = (numpy.random.random_sample((len(xs)))-.5)*.2 # (D)
sigstat=static+signal # (E)
rawsignal=sigstat+noise # (F)
fft=scipy.fft(rawsignal) # (G) and (H)
bp=fft[:]
for i in range(len(bp)): # (H-red)
   if i>=10:bp[i]=0
ibp=scipy.ifft(bp) # (I), (J), (K) and (L)

#################
### GRAPHING ###
#################

h,w=6,2
pylab.figure(figsize=(12,9))
pylab.subplots_adjust(hspace=.7)

pylab.subplot(h,w,1);pylab.title("(A) Original Signal")
```

December 2017 (1)
September 2017 (2)
August 2017 (3)
June 2017 (1)
April 2017 (1)
February 2017 (2)
October 2016 (1)
September 2016 (4)
August 2016 (4)
July 2016 (6)
December 2015 (1)
April 2014 (1)
February 2014 (1)
June 2013 (5)
May 2013 (4)
April 2013 (3)
January 2013 (1)
December 2012 (2)
August 2012 (1)
June 2012 (3)
August 2011 (5)
July 2011 (7)
June 2011 (3)
March 2011 (4)
February 2011 (6)
January 2011 (4)
December 2010 (1)
November 2010 (5)
September 2010 (5)
August 2010 (6)
July 2010 (3)
June 2010 (19)
May 2010 (14)
April 2010 (2)
March 2010 (13)
February 2010 (2)

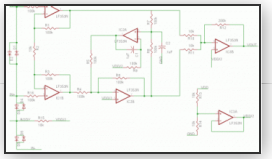
```
pylab.plot(xs,signal)

pylab.subplot(h,w,3);pylab.title("(B) Electrical Noise Sources (3 Sine Waves)")
pylab.plot(xs,sin1,label="sin1")
pylab.plot(xs,sin2,label="sin2")
pylab.plot(xs,sin3,label="sin3")
pylab.legend()

pylab.subplot(h,w,5);pylab.title("(C) Electrical Noise (3 sine waves added together)")
pylab.plot(xs,noise)

pylab.subplot(h,w,7);pylab.title("(D) Static (random noise)")
pylab.plot(xs,static)
pylab.axis([None,None,-1,1])

pylab.subplot(h,w,9);pylab.title("(E) Signal + Static")
pylab.plot(xs,sigstat)

pylab.subplot(h,w,11);pylab.title("(F) Recording (Signal + Static + Electrical Noise)")
pylab.plot(xs,rawsignal)

pylab.subplot(h,w,2);pylab.title("(G) FFT of Recording")
fft=scipy.fft(rawsignal)
pylab.plot(abs(fft))
pylab.text(200,3000,"signals",verticalalignment='top')
pylab.text(9500,3000,"static",verticalalignment='top',
        horizontalalignment='right')

pylab.subplot(h,w,4);pylab.title("(H) Low-Pass FFT")
pylab.plot(abs(fft))
pylab.text(17,3000,"sin1",verticalalignment='top',horizontalalignment='left')
pylab.text(37,2000,"sin2",verticalalignment='top',horizontalalignment='center')
pylab.text(45,3000,"sin3",verticalalignment='top',horizontalalignment='left')
pylab.text(6,3000,"signal",verticalalignment='top',horizontalalignment='left')
pylab.axvspan(10,10000,fc='r',alpha='.5')
pylab.axis([0,60,None,None])

pylab.subplot(h,w,6);pylab.title("(I) Inverse FFT")
pylab.plot(ibp)

pylab.subplot(h,w,8);pylab.title("(J) Signal vs. iFFT")
pylab.plot(signal,'k',label="signal",alpha=.5)
pylab.plot(ibp,'b',label="ifft",alpha=.5)

pylab.subplot(h,w,10);pylab.title("(K) Normalized Signal vs. iFFT")
pylab.plot(signal/max(signal),'k',label="signal",alpha=.5)
pylab.plot(ibp/max(ibp),'b',label="ifft",alpha=.5)

pylab.subplot(h,w,12);pylab.title("(L) Difference / Error")
pylab.plot(signal/max(signal)-ibp/max(ibp),'k')

pylab.savefig("SIG.png",dpi=200)
pylab.show()
```

January 2010 (2)
December 2009 (2)
September 2009 (1)
August 2009 (7)
July 2009 (4)
June 2009 (18)
May 2009 (11)
April 2009 (10)
March 2009 (2)
February 2009 (4)
January 2009 (18)
November 2008 (5)

**Share this:**



ECG Success!
January 14, 2009
In "Circuitry"



DIY ECG?
January 13, 2009
In "Circuitry"

Defibrillating My DIY ECG Project
I've done a lot of random things the last few months, but few things were as random, cool, or
August 6, 2009
In "Circuitry"

DIY ECG, General, Python | Leave a comment

‹ DIY ECG Detected an Irregular Heartbeat

› Using PHP to Create Apache-Style Access.log

You must log in to post a comment.

## About Scott

Scott Harden lives in Gainesville, Florida and works at the University of Florida as a biological research scientist studying cellular neurophysiology. Scott has lifelong passion for computer programming and electrical engineering, and in his spare time enjoys building small electrical devices and writing cross-platform open-source software. more →

## Meta

Log in

Entries RSS

Comments RSS

WordPress.org

## Contact

SWHarden@gmail.com

## Subscribe via Email

Enter your email address to subscribe to this website and receive notifications of new posts by email.

Email Address

Subscribe