FIR filter design with Python and SciPy

Matti Pastell
http://mpastell.com

15th April 2013

1 Introduction

This an example of a document that can be published using Pweave. Text is written using LaTeX and code between <<>> and @ is executed and results are included in the resulting document.

You can define various options for code chunks to control code execution and formatting (see Pweave docs).

2 FIR Filter Design

We'll implement lowpass, highpass and ' bandpass FIR filters. If you want to read more about DSP I highly recommend The Scientist and Engineer's Guide to Digital Signal Processing which is freely available online.

2.1 Functions for frequency, phase, impulse and step response

Let's first define functions to plot filter properties.

```
from pylab import *
import scipy.signal as signal

#Plot frequency and phase response
def mfreqz(b,a=1):
    w,h = signal.freqz(b,a)
    h_dB = 20 * log10 (abs(h))
    subplot(211)
    plot(w/max(w),h_dB)
    ylim(-150, 5)
    ylabel('Magnitude (db)')
    xlabel(r'Normalized Frequency (x$\pi$rad/sample)')
    title(r'Frequency response')
    subplot(212)
    h_Phase = unwrap(arctan2(imag(h),real(h)))
    plot(w/max(w),h_Phase)
    ylabel('Phase (radians)')
    xlabel(r'Normalized Frequency (x$\pi$rad/sample)')
    title(r'Phase response')
    subplots_adjust(hspace=0.5)

#Plot step and impulse response
```

```
def impz(b,a=1):
    l = len(b)
    impulse = repeat(0.,l); impulse[0] =1.
    x = arange(0,l)
    response = signal.lfilter(b,a,impulse)
    subplot(211)
    stem(x, response)
    ylabel('Amplitude')
    xlabel(r'n (samples)')
    title(r'Impulse response')
    subplot(212)
    step = cumsum(response)
    stem(x, step)
    ylabel('Amplitude')
    xlabel(r'n (samples)')
    title(r'Step response')
    subplots_adjust(hspace=0.5)
```

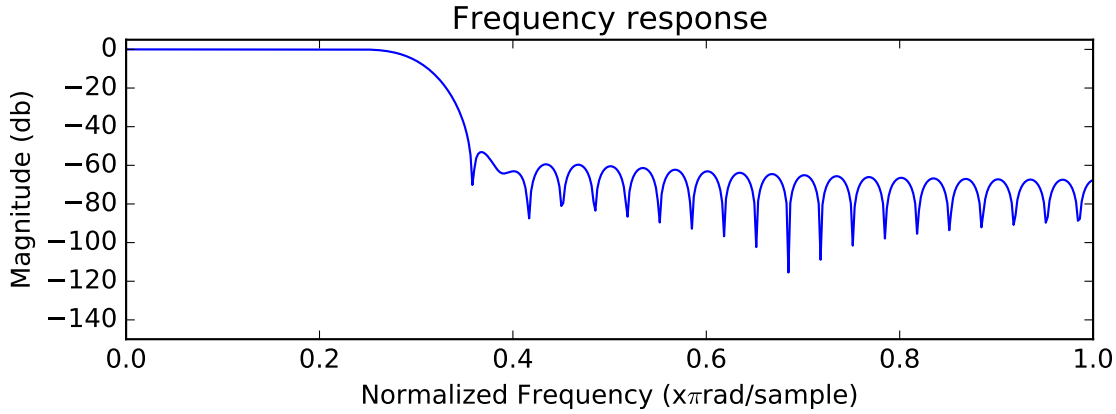
2.2 Lowpass FIR filter

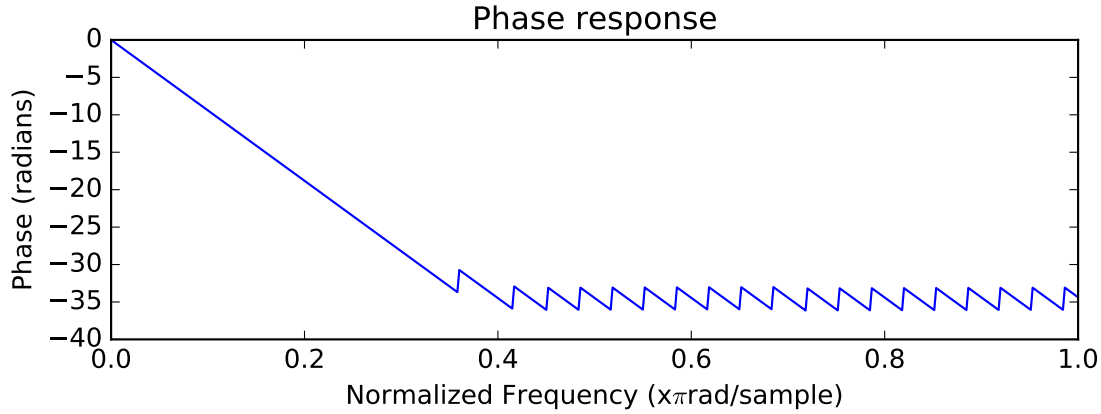
Designing a lowpass FIR filter is very simple to do with SciPy, all you need to do is to define the window length, cut off frequency and the window.

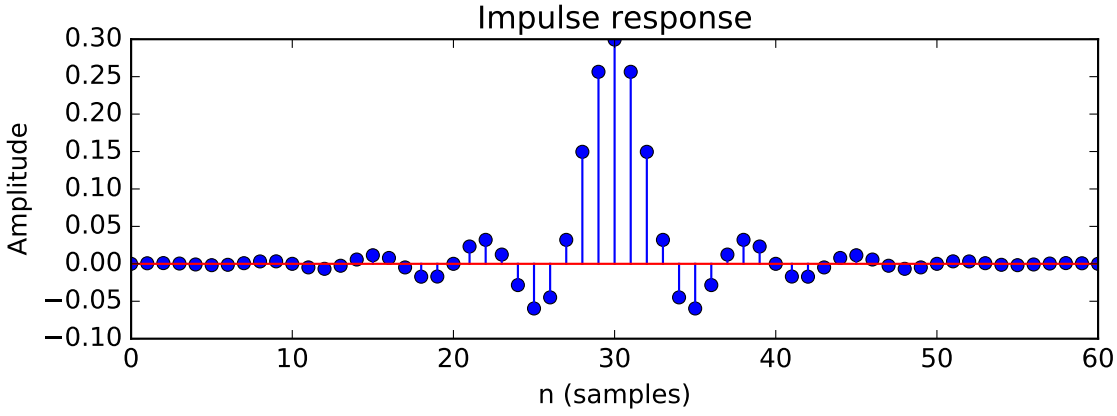
The Hamming window is defined as: $w(n) = \alpha - \beta \cos \frac{2\pi n}{N-1}$, where $\alpha = 0.54$ and $\beta = 0.46$

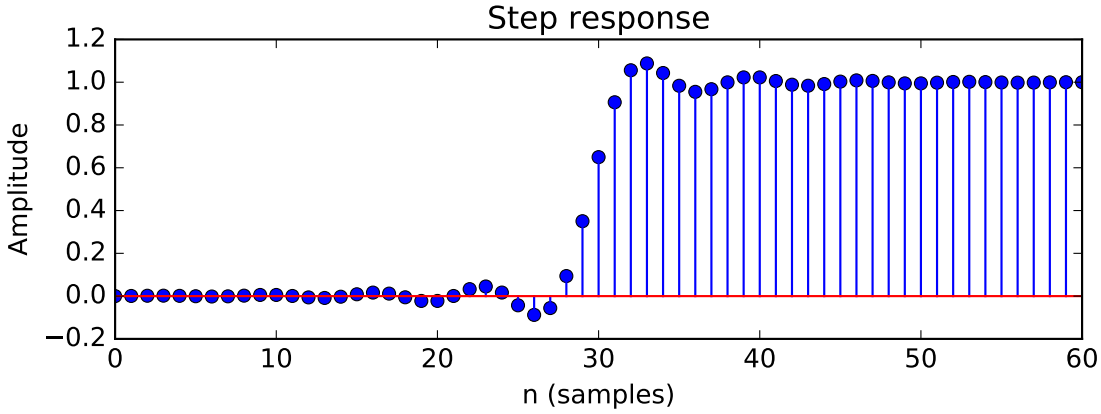
The next code chunk is executed in term mode, see the source document for syntax. Notice also that Pweave can now catch multiple figures/code chunk.

```
In [1]: n = 61

In [2]: a = signal.firwin(n, cutoff = 0.3, window = "hamming")

In [3]: #Frequency and phase response

In [4]: mfreqz(a)

In [5]: show()

In [6]: #Impulse and step response

In [7]: figure(2)
Out[7]: <matplotlib.figure.Figure object at 0x7f04282c3210>
In [8]: impz(a)

In [9]: show()
```


Frequency response
Magnitude (db)
0
−20
−40
−60
−80
−100
−120
−140
0.0
0.2
0.4
0.6
0.8
1.0
Normalized Frequency (xπrad/sample)


Phase response
Phase (radians)
0
−5
−10
−15
−20
−25
−30
−35
−40
0.0
0.2
0.4
0.6
0.8
1.0
Normalized Frequency (xπrad/sample)

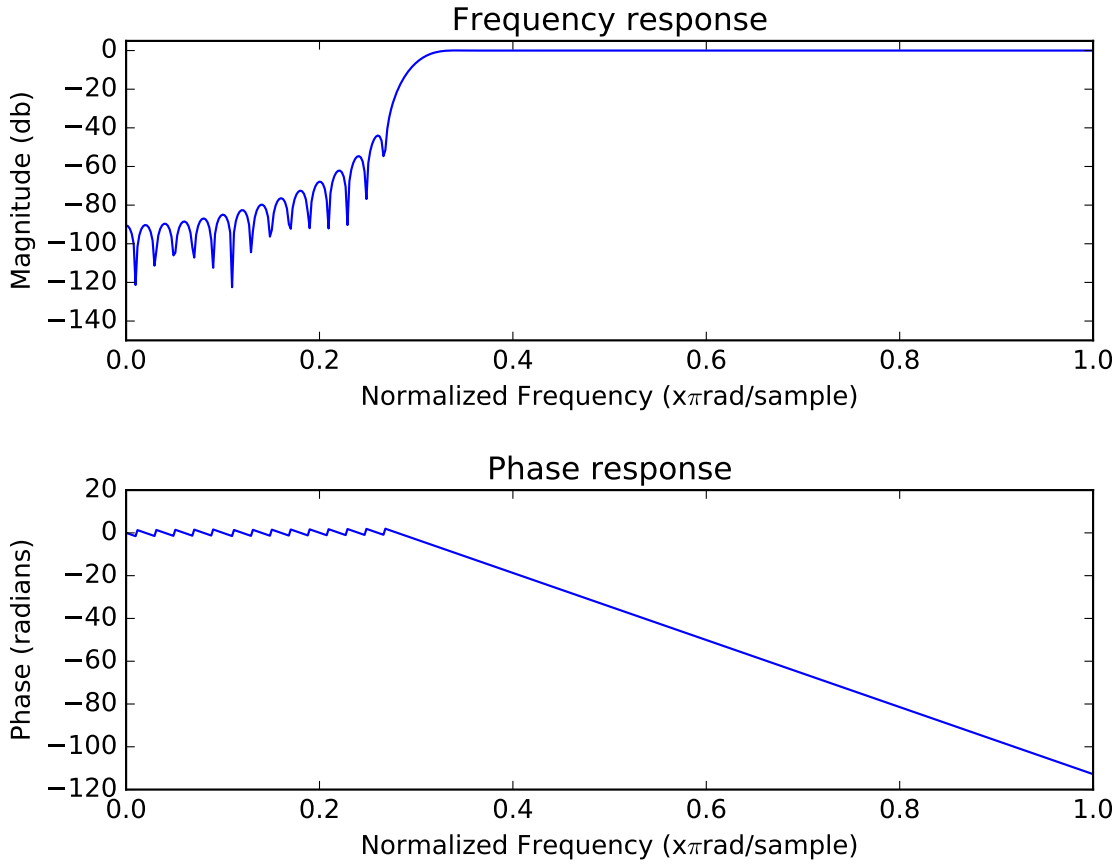

Impulse response
Amplitude
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
0
10
20
30
40
50
60
n (samples)


Step response
Amplitude
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
10
20
30
40
50
60
n (samples)

2.3 Highpass FIR Filter

Let's define a highpass FIR filter:

```
n = 101
a = signal.firwin(n, cutoff = 0.3, window = "hanning",
pass_zero=False)
mfreqz(a)
show()
```



2.4 Bandpass FIR filter

Notice that the plot has a caption defined in code chunk options.

```
n = 1001
a = signal.firwin(n, cutoff = [0.2, 0.5], window = 'blackmanharris',
pass_zero = False)
mfreqz(a)
show()
```

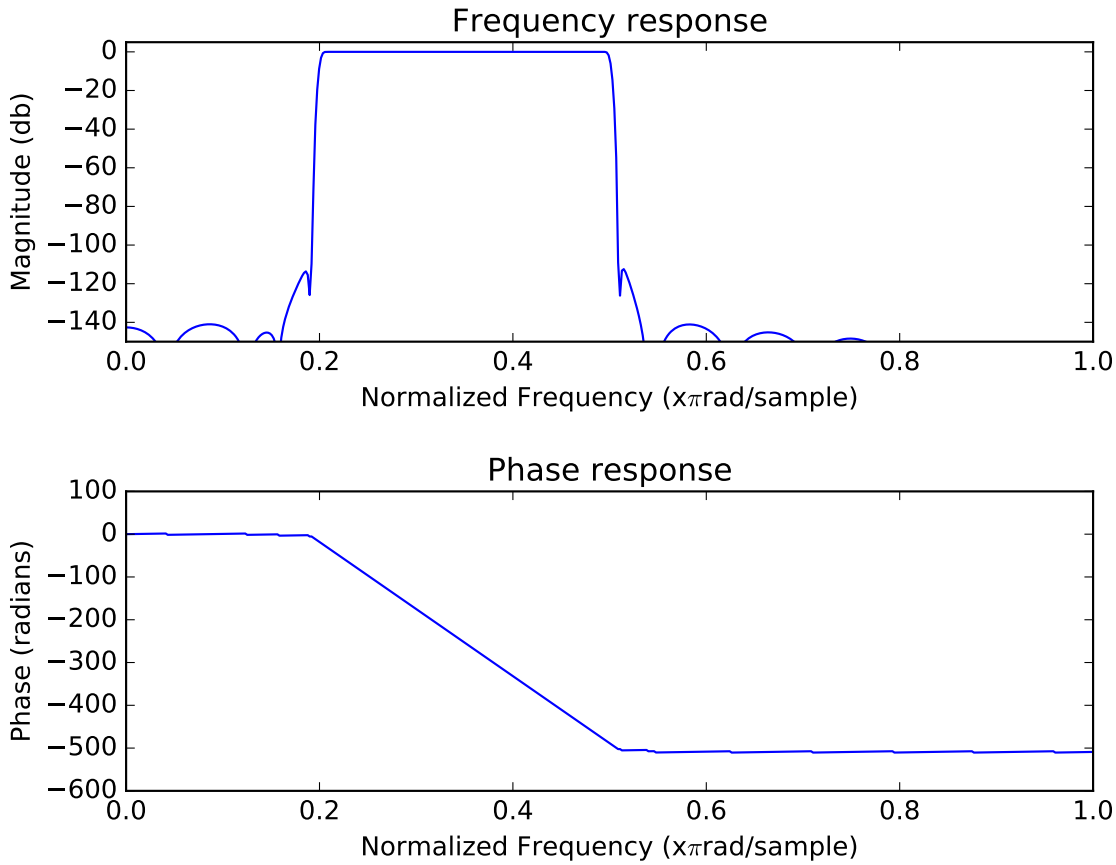


Figure 1: Bandpass FIR filter.