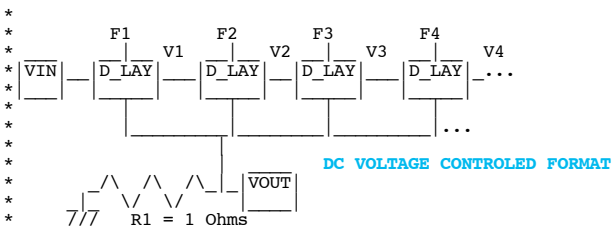
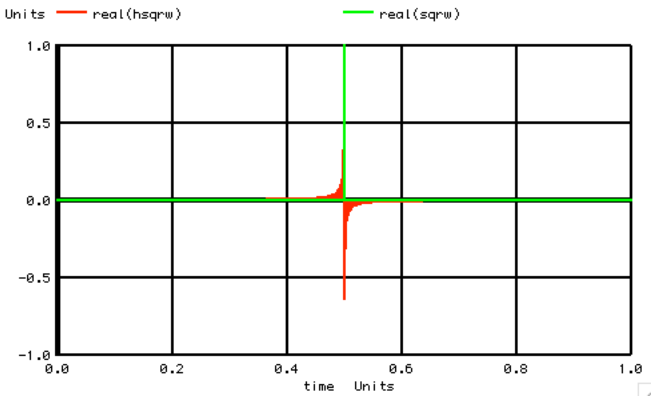
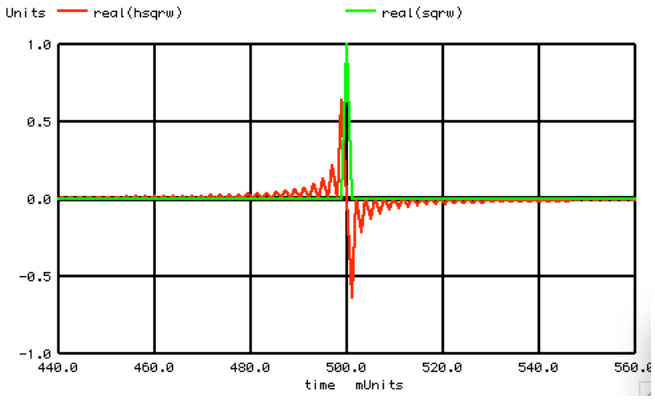
==========Hilbert_FFT_Impulse==========

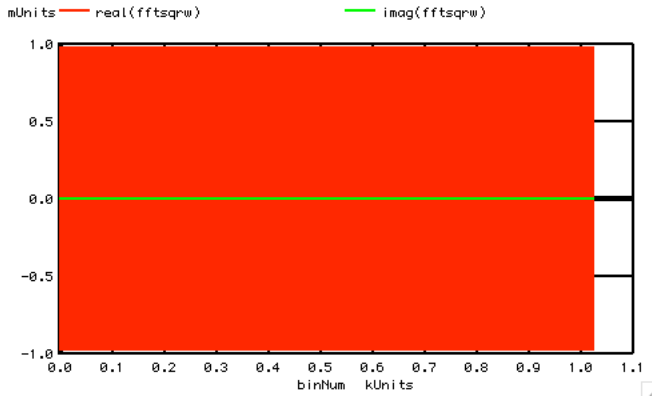


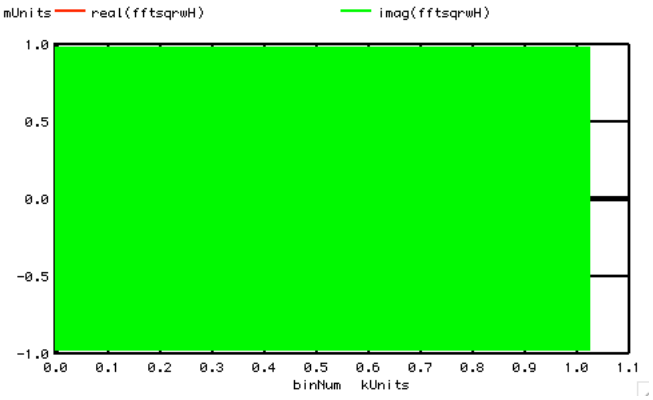
So if it is possible to build a brick wall filter by creating an array of signal delay elements which match an impulse response, why not do the same for a Hilbert filter?



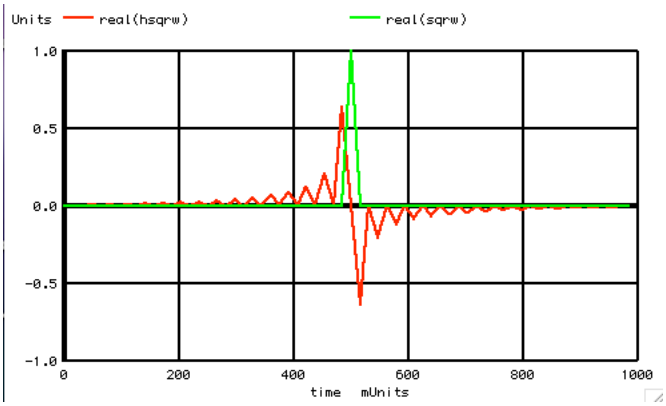


FFT an impulse, rotate its harmonic phases, and IFFT back. When doing an impulse response for 1024 data points, it gets a little hard to see the details as to what is going on.



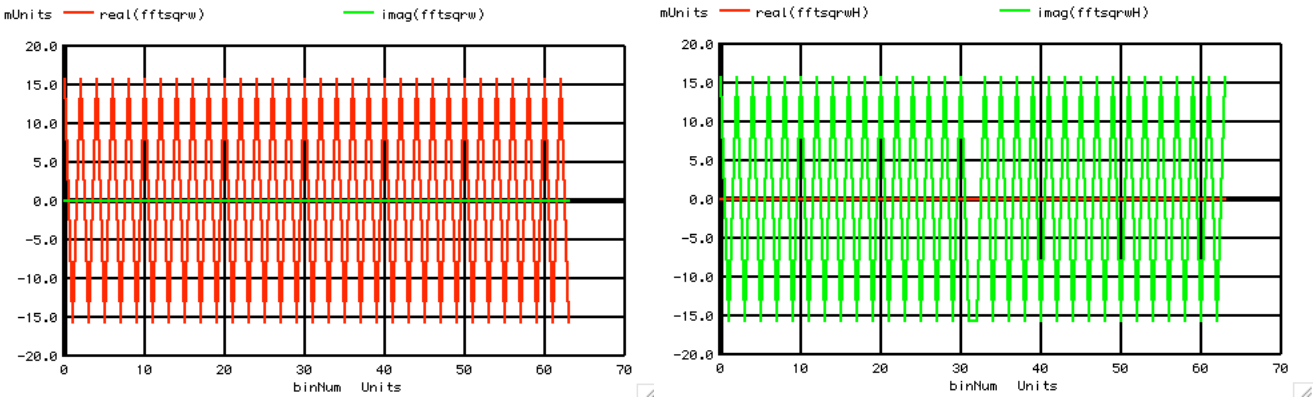


An impulse creates a string of alternating harmonics. Hard to see things a 1024 data points. The rotation spectrum is displayed on the right.



Using 64 points makes it easier to see the impulse response. The response

should be something like -1/n*Pi where n is the number of samples after the impulse. And all even number values for n are at zero.



The 64 frequency bins need to be rotated two different ways depending on whether a +jw term or -jw term is to be represented. In this FFT output format, the +jw terms are from 0 to nyquist. The negative jw terms start at the maximum frequency bin and come down.

==========Full_Netlist_For_Copy_Paste==========

Hilbert_FFT_Impulse

```
*V_SIN#    NODE_P NODE_N DC      VALUE  SIN(    V_DC    AC_MAG FREQ    DELAY  FDamp)
Vsin       SIN    0      DC      0      SIN(    0       1      1k      )

.control
set pensize = 2


unlet a
unlet time
unlet sqrw
unlet fftsqrw
unlet fftsqrwH
unlet hsqrw
let numb  =2^6
let numbh = numb/2
let time =    vector(numb)
*==========Create_Complex_Vector===========
let a =       vector(numb)
let sqrw = a+j(0)
*==========Define_Impulse===========
let indx = 0
repeat $&numb
let sqrw[indx] = 0+j(0)
let time[indx] = indx/numb
let indx = indx +1
end
let sqrw[numbh] = 1+j(0)
plot      real(sqrw)  vs time

*==========FFT_Impulse===========
let fftsqrw =fft(sqrw)
unlet binNum
let binNum = vector(numb)
let fftsqrwH = a+j(0)

plot real(fftsqrw) imag(fftsqrw)    vs binNum title ImpluseInput
*==========Phase_Shift_Harmonics===========
let indx = 0
repeat $&numbh
let fftsqrwH[indx] = j(1)*fftsqrw[indx]
let indx = indx +1
end

repeat $&numbh
let fftsqrwH[indx] = -1*j(1)*fftsqrw[indx]
let indx = indx +1
end

plot real(fftsqrwH) imag(fftsqrwH)    vs binNum title PhaseSiftHarmonics

*==========Inverse_FFT===========
hsqrw = ifft(fftsqrwH)
plot      real(hsqrw)  real(sqrw) vs time  xlimit .45 .55
plot      real(hsqrw)  real(sqrw) vs time

.endc
.end
```

6.7.11_12.54PM
dsauersanjose@aol.com
Don Sauer