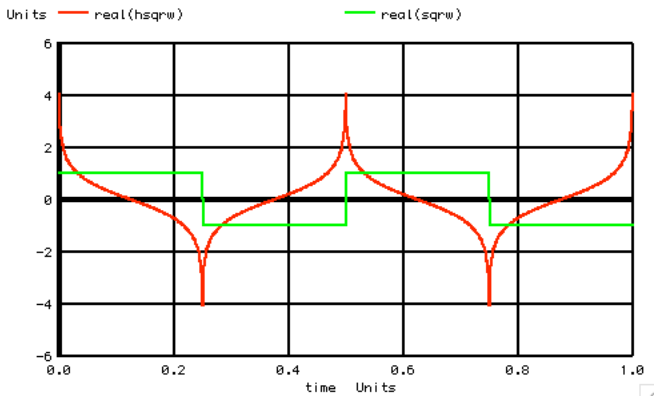
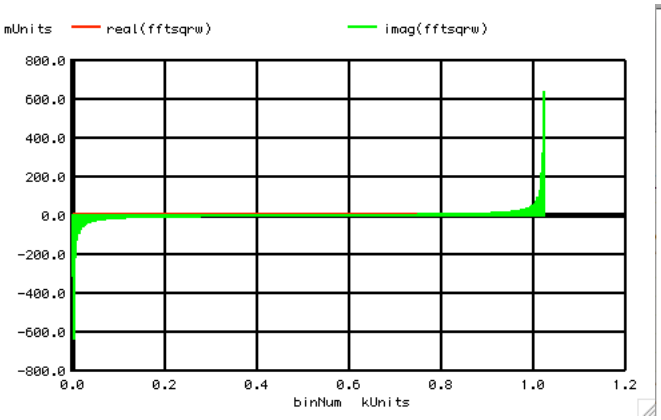
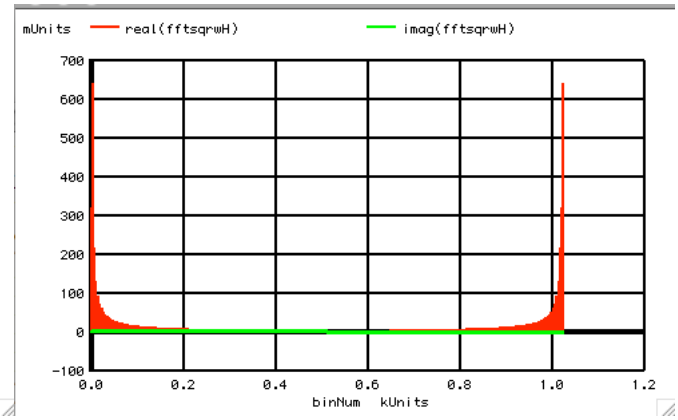
==========Hilbert_FFT_SquareWave==========



The goal is to phase shift all frequencies by 90 degrees. The output of such a filter (Hilbert) will be a little unusual.

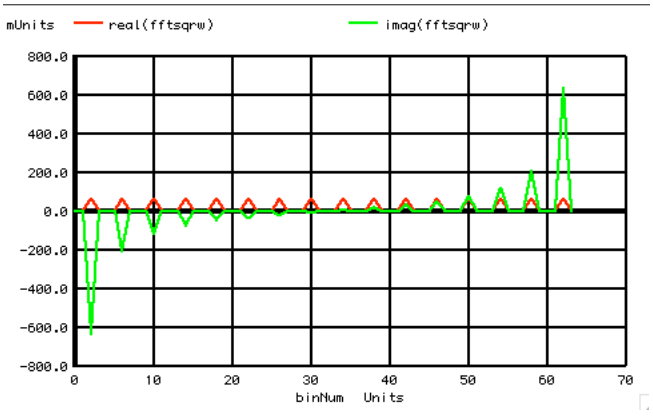
Having a FFT/IFFT feature in MacSpice makes it easy to see what such a type of filter should do.

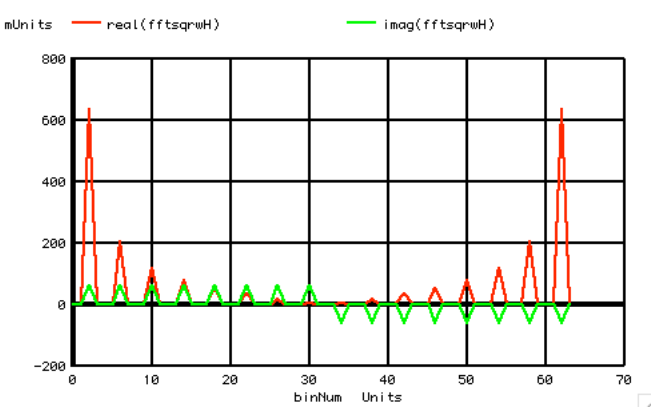




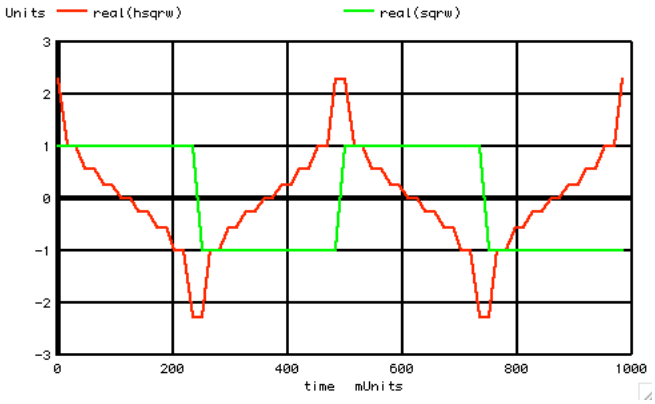
A square wave into fft will have the frequency bins shown above. It seems to be common to have low frequency bin represent the positive jw terms up to the nyquist frequency. The negative jw terms are coming from the top bin down.

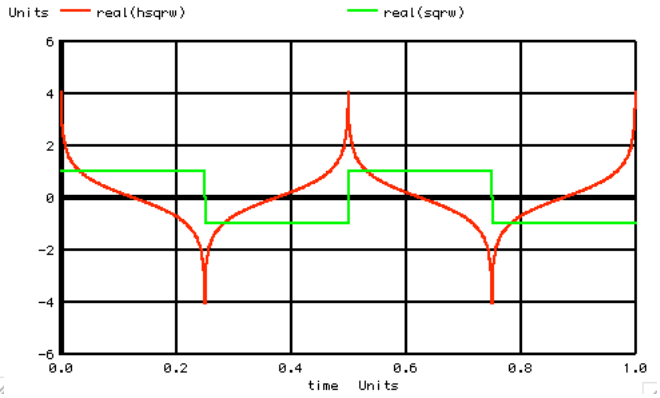
Given the FFT spectrum is complex, the higher and lower bins need to be rotated by 90 degrees by multiplying by a + or - j terms where appropriate.





Adding more data points will make things better. But economics often needs some consideration. The impact of using less data points should be tested. With less data points, the spectrum for the square wave is easier to see.





The output response should resemble that of a discreet time response compared to a analog response.

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

```
Hilbert_FFT_squareW

*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 numbq = numb/4
let numb0 = numb/8
let time  = vector(numb)
*=========Create_Complex_Vector===========
let       a = vector(numb)
let       sqrw = a+j(0)

*=========Define_Square_Wave===========
let       indx = 0
repeat    2
repeat    $&numbq
let       sqrw[indx] = 1+j(0)
let       time[indx] = indx/numb
let       indx = indx +1
end
repeat    $&numbq
let       sqrw[indx] = -1+j(0)
let       time[indx] = indx/numb
let       indx = indx +1
end
end
plot      real(sqrw)  vs time
plot      real(sqrw)  vs time ylimit -1.5 1.5

*=========FFT_Square_Wave===========
let       fftsqrw =fft(sqrw)
unlet     fftsqrwH
unlet     binNum
let       binNum = vector(numb)
let       fftsqrwH = a+j(0)
plot      real(fftsqrw) imag(fftsqrw)   vs binNum title FFT_COS
plot      real(fftsqrw) imag(fftsqrw)   vs binNum title FFT_COS
*=========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 FFT_COS
*=========Inverse_FFT===========
hsqrw =   ifft(fftsqrwH)
plot      real(hsqrw)  real(sqrw) vs time

.endc
.end
```

6.7.11_12.46PM
dsauersanjose@aol.com
Don Sauer