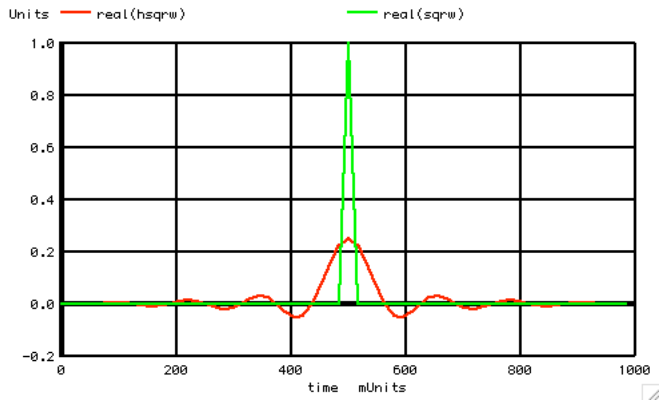
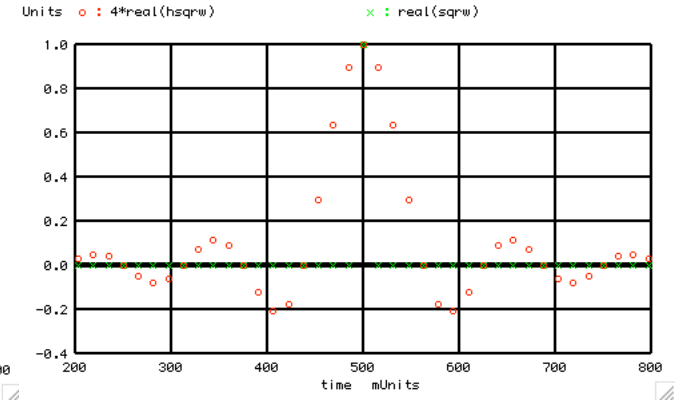
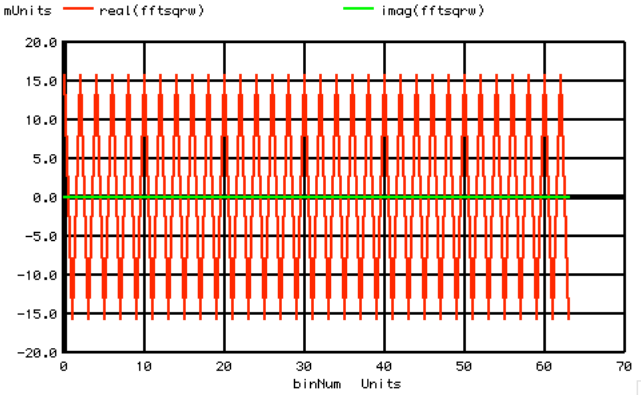
==========Brickwall_FFT_Impulse==========

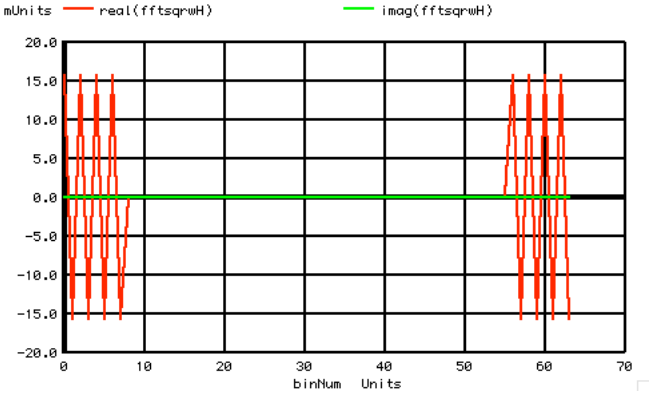




The impulse response to a brickwall filter is of interest, because open loop FIR digital filters can to a certain degree reproduce the impulse response of anything.

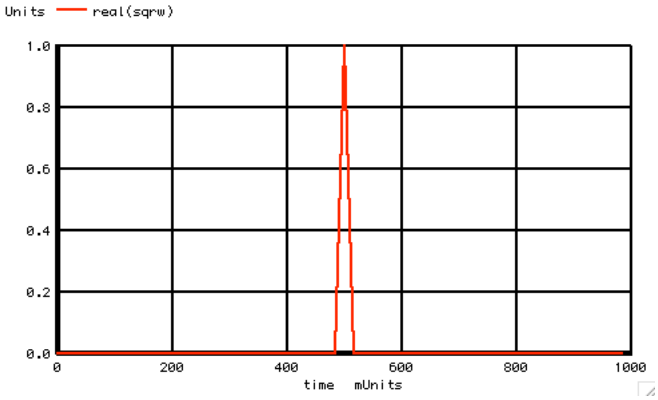
Here a single point impulse appears at the input. An FFT is taken. All but the first 8 frequency bins are removed. The inverse FFT will produce an output response that begins happening before the input signal arrives. Since only 64 data points have been taken, the impulse response can be plotted in terms of discrete values happening at discrete points in time.



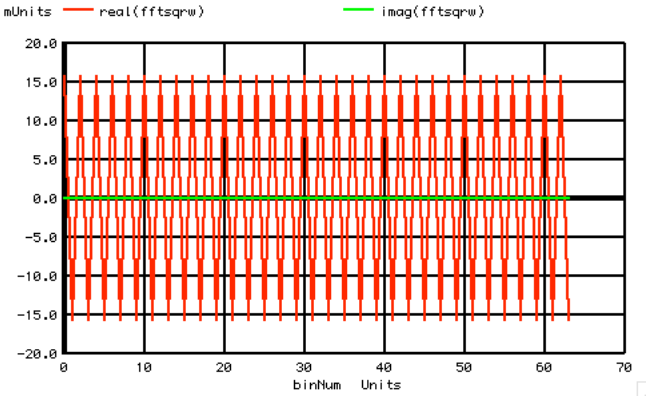



```
*==========Use_Number_Points_to_Power_Of_two===========
let numb  = 2^6
let numbh = numb/2
let numbq = numb/4
let time  = vector(numb)
let a     = vector(numb)
*==========Create_Complex_vector===========
let sqrw  = a+j(0)
*==========Define_Square_AS_Zero===========
let indx = 0
repeat $&numb
let sqrw[indx] = 0+j(0)
let time[indx] = indx/numb
let indx = indx +1
end
*==========Add_Impulse===========
let sqrw[numbh] = 1+j(0)
plot      real(sqrw)  vs time
```

The above shows how to create a complex impulse.

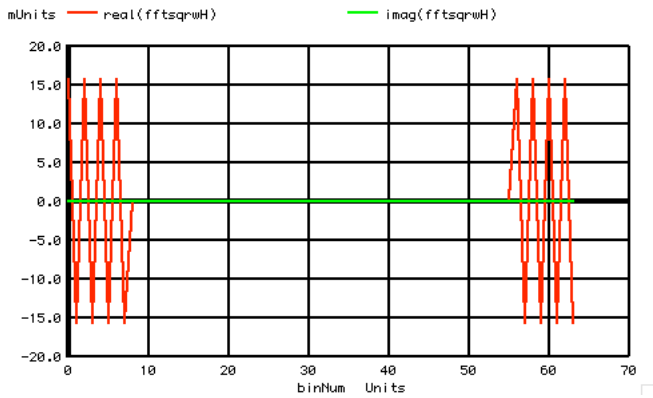



```
*=========FFT_Impulse===========
let fftsqrw =fft(sqrw)
unlet binNum
let binNum = vector(numb)
let fftsqrwH = a+j(0)
plot real(fftsqrw) imag(fftsqrw)   vs binNum title FFT COS
```



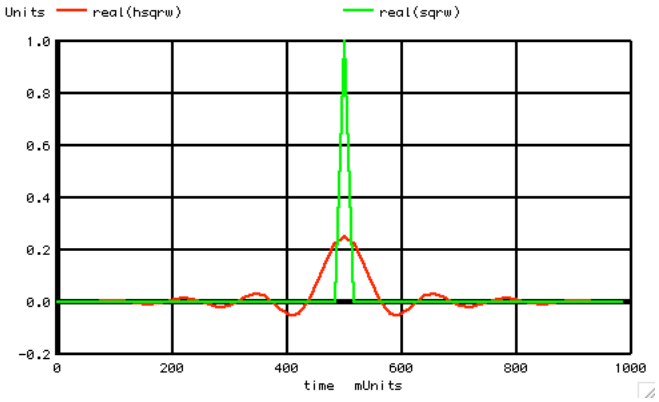
This is what the spectrum looks like for a 64 point impulse.

```
*=========Make_Filterred_Array===========
let indx = 0
repeat $&numb
let fftsqrwH[indx] = fftsqrw[indx]
let indx = indx +1
end
*=========Brickwall_the_Impulse===========
let LP = 8
let Blank=numb-2*LP
let indx = LP
repeat $&Blank
let fftsqrwH[indx] = j(0)*fftsqrw[indx]
let indx = indx +1
end
plot real(fftsqrwH) imag(fftsqrwH)   vs binNum title FFT_COS
```



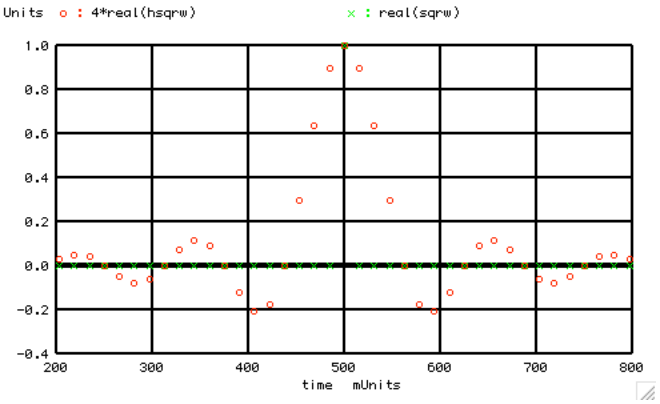
Brickwall filter the first 8 frequency bins.

```
*=========Inverse_FFT===========
hsqrw =   ifft(fftsqrwH)
plot      4*real(hsqrw)  real(sqrw) vs time  pointplot xlimit .2 .8 ylimit -.4 1
plot      real(hsqrw)    real(sqrw) vs time
```



.endc
.end

The impulse response for a brickwall lowpass is a sinc function which begins to happen before impulse arrives.



The impulse response for a 64 point a brickwall lowpass can further be displayed in a normalize point fashion.

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

```
Brickwall_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 numbq = numb/4
let time =    vector(numb)
let a =       vector(numb)
let sqrw = a+j(0)


*=========Define_Square_AS_Zero===========
let indx = 0
repeat $&numb
let sqrw[indx] = 0+j(0)
let time[indx] = indx/numb
let indx = indx +1
end
*=========Add_Impulse===========
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 FFT_COS
*=========Make_Filterred_Array===========
let indx = 0
repeat $&numb
let fftsqrwH[indx] = fftsqrw[indx]
let indx = indx +1
end
```

```
*=========Brickwall_the_Impulse===========
let LP = 8
let Blank=numb-2*LP
let indx = LP
repeat $&Blank
let fftsqrwH[indx] = j(0)*fftsqrw[indx]
let indx = indx +1
end
plot real(fftsqrwH) imag(fftsqrwH)   vs binNum title FFT_COS
*=========Inverse_FFT===========
hsqrw =   ifft(fftsqrwH)
plot      4*real(hsqrw)  real(sqrw) vs time  pointplot xlimit .2 .8 ylimit -.4 1
plot      real(hsqrw)    real(sqrw) vs time

.endc
.end
```

6.7.11_12.01PM
dsauersanjose@aol.com
Don Sauer