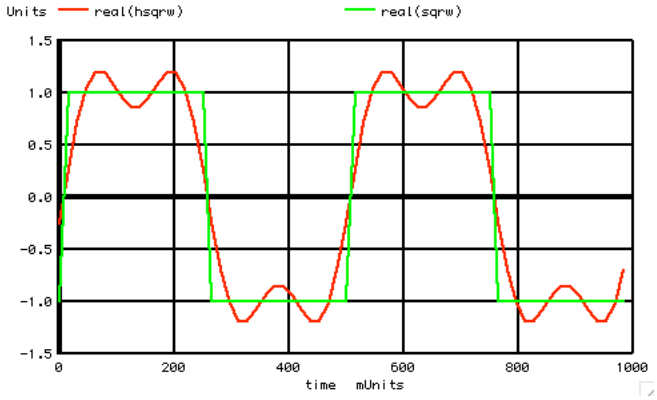
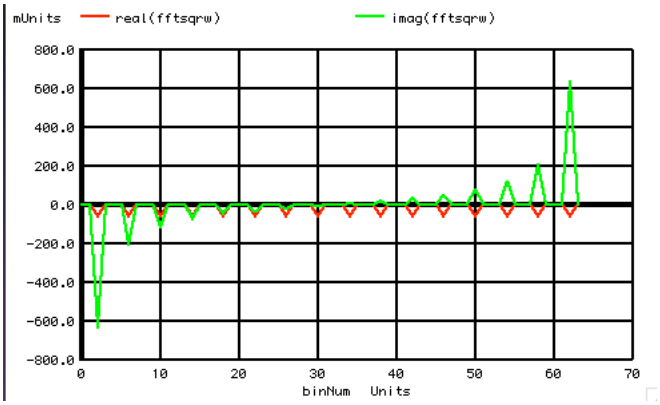
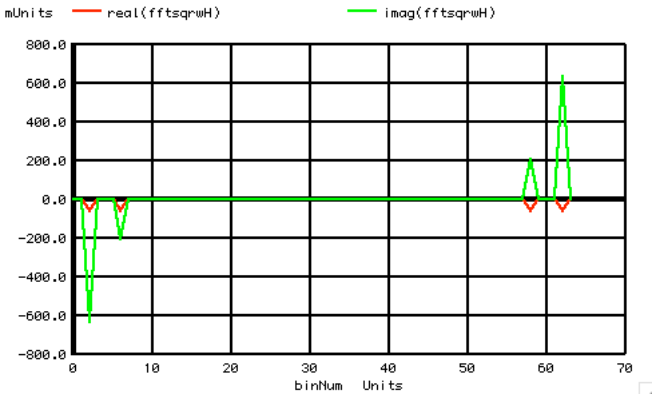
==========Brickwall_Using_FFT==========



The FFT/IFFT function allows the simulation of a perfect "brickWall" lowpass filter to see how it behaves. The square wave response will be that of the fundamental plus any harmonics left over with no timing delays or phase shifts.

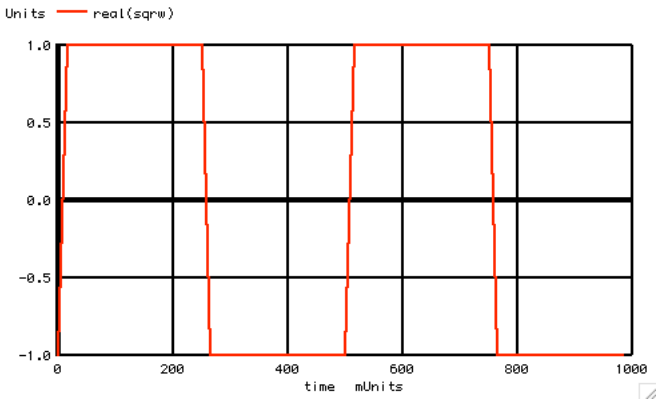




For MacSpice's FFT, the positive jw terms are stored for bin numbers below nysquist. If 64 samples are taken, nyquist is at 32. The -jw are stored as a mirror image from say bin 64 and down.

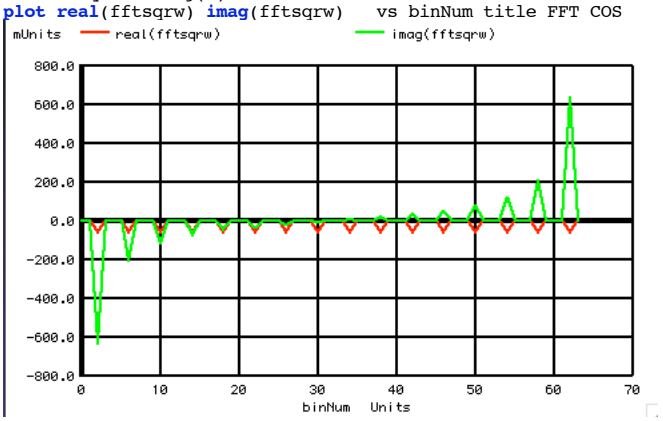
One only needs to blank out all harmonics above the third from both directions. Take an ifft of the spectrum to see what a perfect brickwall filter should do.

```
let numb  = 2^6
let numbh = numb/2
let numbq = numb/4
let time  = vector(numb)
*==========Make_A_Complex_Vector===========
let a     = vector(numb)
let sqrw =   a+j(0)
*==========Define_SquareWave===========
let indx = 0
repeat $&numb
let sqrw[indx] = 2*pos(sin( 360*2*indx/numb ))-1  +j(0)
let time[indx] = indx/numb
let indx = indx +1
end
plot      real(sqrw) vs time
```



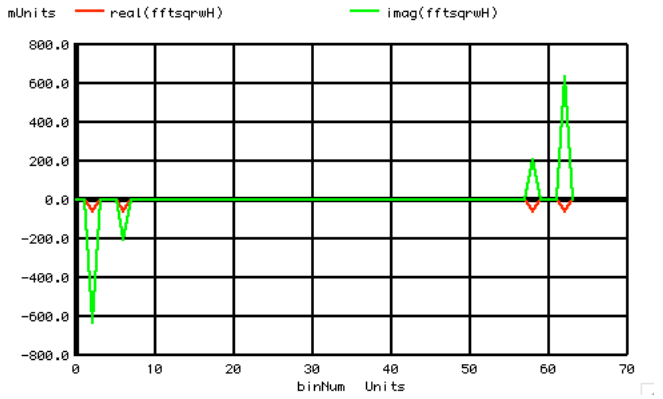
This simulation is best done using math and arrays.
Note that the arrays need to be complex.

```
*=========FFT_SquareWave===========
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
```



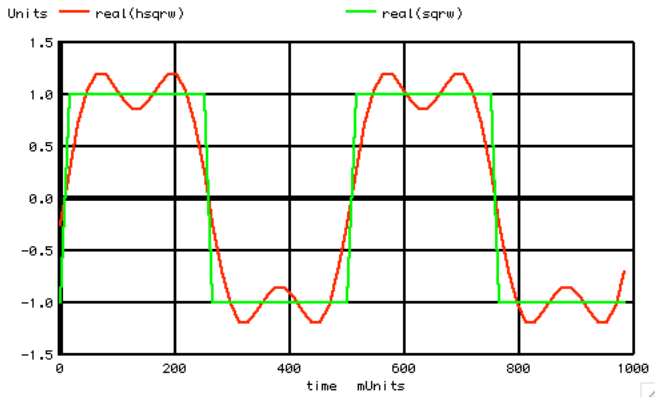
The input square wave FFT is an array of signwave harmonics.

```
*=========Make_Filterred_FFT_Array===========
let indx = 0
repeat $&numb
let fftsqrwH[indx] = fftsqrw[indx]
let indx = indx +1
end
*=========Remove_Harmonics===========
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
```



Easy enough to remove the higher harmonics.

```
*=========Inverse_FFT===========
hsqrw = ifft(fftsqrwH)
plot      real(hsqrw)  real(sqrw) vs time
```




```
.endc
.end
```

Doing and plotting an IFFT is easy enough.

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

```
Brickwall_FFT_Square
*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_SquareWave===========
let indx = 0
repeat $&numb
let sqrw[indx] = 2*pos(sin( 360*2*indx/numb ))-1  +j(0)
let time[indx] = indx/numb
let indx = indx +1
end
plot      real(sqrw) vs time
*==========FFT_SquareWave===========
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_FFT_Array===========
let indx = 0
repeat $&numb
let fftsqrwH[indx] = fftsqrw[indx]
let indx = indx +1
end
*==========Remove_Harmonics===========
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      real(hsqrw)  real(sqrw) vs time

.endc
.end
```