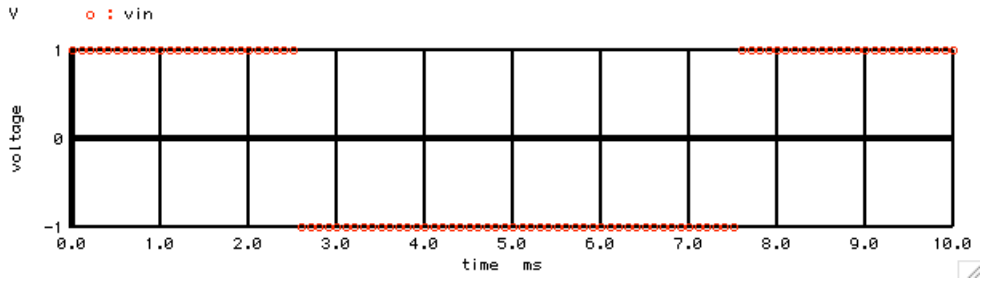
*=========TIMING_Needs2===============

Note that the linearize function is applied to make time periods consistent. The pointplot feature can show this feature working. One other feature. Notice the time period is now .9999 seconds

```
=================================================================
*V_PULSE# NODE_P NODE_N DC      VALUE   PULSE( VINIT  VPULSE TDELAY TRISE   TFALL   PWIDTH PERIOD )
V_SQR     VIN    0      DC      0       PULSE( -1     1      -2.5m  1u      1u      5m     10m    )
.control
*TRAN     TSTEP  TSTOP  TSTART TMAX
tran      .1m    .9999  0      .1m
set       pensize = 2
linearize
plot      vin  xlimit 0 10m pointplot
```



*=========Fix_the_Odd_Number_Of_Points====================

Now there are only 10,000 time points

```
=================================================================
let       numb2 = length(vin)
print     numb2

=================================================================

numb2 =   1.00000e+04
```

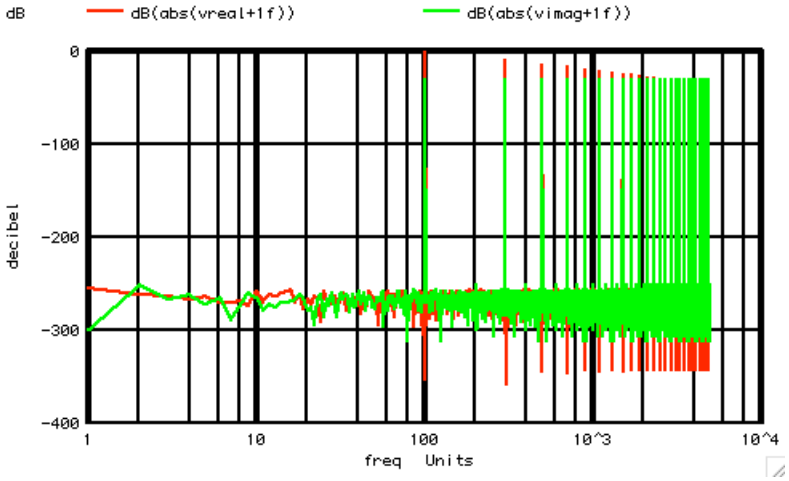
*=========Makes_Converting_to_Frequency_Cleaner====================

When it comes down to converting the FFT output to a frequency scale, dividing 10,000 time points is better than dividing 10,001 time points.

```
=================================================================
let       ac = vin +j(0)
let       ac_fft=fft(ac)

let       numb_f = (numb2)/2
compose   freq   start = 1 stop = $&numb_f  step =1
compose   vreal  start = 1 stop = $&numb_f  step =1
compose   vimag  start = 1 stop = $&numb_f  step =1
let       i = 0
repeat    $&numb_f
let       freq[i] =  freq[i]
let       vreal[i] = 1.414*real(ac_fft[i])
let       vimag[i] = 1.414*imag(ac_fft[i])
let       i = i +1
end

plot      dB(abs(vreal+1f)) dB(abs(vimag+1f)) vs freq xlog
```



So the spectrum leakage got much better. There is a -30dB amount of phase shift in the output. Still looking into ways to remove it.

***=========Using_Sortorder_To_Find_Things====================**

In cases where it is not so obvious where the largest frequency bins are, the sortorder function can be used. Note that the output of sortorder starts with the smallest values first. Inverting the data can make it display the largest values first.

```
================================================================
let       harm = sortorder(1/(mag(real(ac_fft))+1u))
foreach   ii 0 1 2 3 4  5
let k =   $ii
harmNum = harm[k]
echo k  = $&k   harmNum =  $&harmNum
end
print     ac_fft[100]
print     ac_fft[numb2-100]

================================================================
k = 0 harmNum = 100
k = 1 harmNum = 9900
k = 2 harmNum = 300
k = 3 harmNum = 9700
k = 4 harmNum = 500
k = 5 harmNum = 9500

ac_fft[100] =        ( 6.36410e-01,-2.00000e-02 )
ac_fft[numb2-100] = ( 6.36410e-01, 2.00000e-02 )
```

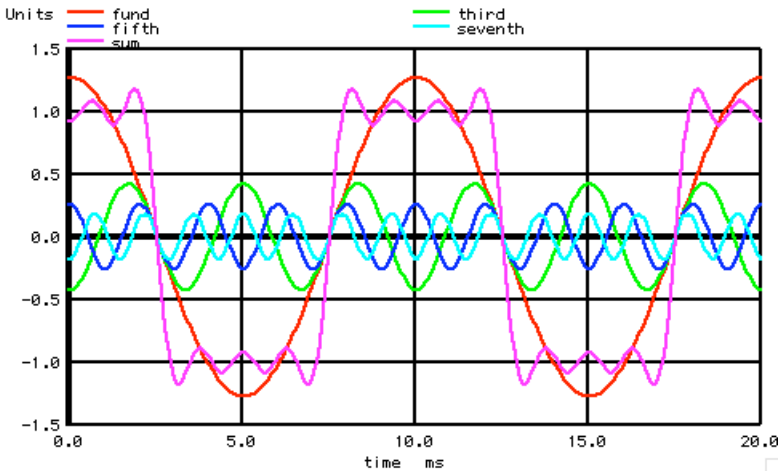
***=========Automated_Harmonic_Extraction====================**

There are ways to further automate the harmonic dissection process.

```
================================================================
let       funb                      = 100
let       unvect                    = unitvec($&numb2)
let       fundspec                  = unvect*0 +j(0)
let       fundspec[funb]            = real(ac_fft[funb])        +j(imag(ac_fft[funb] ))
let       fundspec[numb2-funb]      = real(ac_fft[numb2-funb])  +j(imag(ac_fft[numb2-funb] ))
let       fund =  ifft(fundspec)
let       thirdspec                 = unvect*0 +j(0)
let       thirdspec[3*funb]         = real(ac_fft[3*funb])        +j(imag(ac_fft[3*funb] ))
let       thirdspec[numb2-3*funb]   = real(ac_fft[numb2-3*funb])  +j(imag(ac_fft[numb2-3*funb] ))
let       third                     = ifft(thirdspec)
let       fifthspec                 = unvect*0 +j(0)
let       fifthspec[5*funb]         = real(ac_fft[5*funb])        +j(imag(ac_fft[5*funb] ))
```

```
let       fifthspec[numb2-5*funb]    = real(ac_fft[numb2-5*funb])  +j(imag(ac_fft[numb2-5*funb] ))
let       fifth                      = ifft(fifthspec)
let       seventhspec                = unvect*0 +j(0)
let       seventhspec[7*funb]        = real(ac_fft[7*funb])        +j(imag(ac_fft[7*funb] ))
let       seventhspec[numb2-7*funb]  = real(ac_fft[numb2-7*funb])  +j(imag(ac_fft[numb2-7*funb] ))
let       seventh                    = ifft(seventhspec)
let       sum  = fund + third + fifth + seventh

set       scale  time
plot      fund  third fifth seventh  sum xlimit 0 20m
```



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

```
FFT_Timing_Needs_Two
*V_PULSE# NODE_P NODE_N DC     VALUE  PULSE( VINIT  VPULSE TDELAY TRISE  TFALL  PWIDTH PERIOD )
V_SQR     VIN    0      DC     0      PULSE( -1     1      -2.5m  1u     1u     5m     10m    )

.control
*TRAN     TSTEP  TSTOP  TSTART TMAX   ?UIC?
tran     .1m     .9999  0      .1m
set       pensize = 2

linearize
plot      vin  xlimit 0 10m pointplot
let       numb2 = length(vin)
print     numb2

let       ac     = vin +j(0)
let       ac_fft = fft(ac)

let       numb_f = (numb2)/2
compose   freq   start = 1 stop = $&numb_f  step =1
compose   vreal  start = 1 stop = $&numb_f  step =1
compose   vimag  start = 1 stop = $&numb_f  step =1
let       i = 0
repeat    $&numb_f
let       freq[i]  = freq[i]
let       vreal[i] = 1.414*real(ac_fft[i])
let       vimag[i] = 1.414*imag(ac_fft[i])
let       i = i +1
end
plot      dB(abs(vreal+1f)) dB(abs(vimag+1f)) vs freq xlog

let       harm=sortorder(1/(mag(real(ac_fft))+1u))
foreach   ii 0 1 2 3 4  5
let k =   $ii
harmNum = harm[k]
echo k  = $&k   harmNum =  $&harmNum
end
print     ac_fft[100]
print     ac_fft[numb2-100]

let       funb                       = 100
let       unvect                     = unitvec($&numb2)
let       fundspec                   = unvect*0 +j(0)
let       fundspec[funb]             = real(ac_fft[funb])        +j(imag(ac_fft[funb] ))
let       fundspec[numb2-funb]       = real(ac_fft[numb2-funb])  +j(imag(ac_fft[numb2-funb] ))
let       fund =  ifft(fundspec)
let       thirdspec                  = unvect*0 +j(0)
```

```
let       thirdspec[3*funb]         = real(ac_fft[3*funb])        +j(imag(ac_fft[3*funb] ))
let       thirdspec[numb2-3*funb]   = real(ac_fft[numb2-3*funb])  +j(imag(ac_fft[numb2-3*funb] ))
let       third                     = ifft(thirdspec)
let       fifthspec                 = unvect*0 +j(0)
let       fifthspec[5*funb]         = real(ac_fft[5*funb])        +j(imag(ac_fft[5*funb] ))
let       fifthspec[numb2-5*funb]   = real(ac_fft[numb2-5*funb])  +j(imag(ac_fft[numb2-5*funb] ))
let       fifth                     = ifft(fifthspec)
let       seventhspec               = unvect*0 +j(0)
let       seventhspec[7*funb]       = real(ac_fft[7*funb])        +j(imag(ac_fft[7*funb] ))
let       seventhspec[numb2-7*funb] = real(ac_fft[numb2-7*funb])  +j(imag(ac_fft[numb2-7*funb] ))
let       seventh                   = ifft(seventhspec)
let       sum  = fund + third + fifth + seventh
set       scale  time
plot      fund  third fifth seventh  sum xlimit 0 20m

.endc
.end
```