

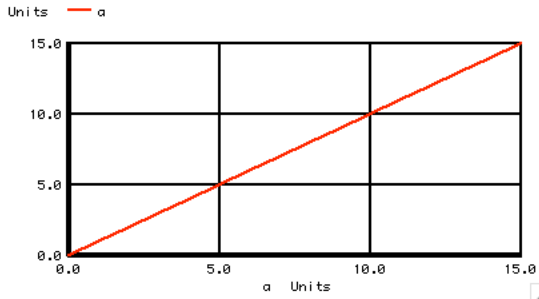
**\*\*\*\*\*FFT/IFFT\_CHECK\_OUT\*\*\*\*\***

**TRANSFORM FROM TIME TO SPECTRUM AND BACK AGAIN.  
SIGNAL CAN NOW BE FILTERED IN SPECTRUM FORMAT.  
IS IT POSSIBLE TO CREATE IMAGINARY TIME?**

This is a checkout of the new fft/iff functions.  
The **vector** function comes in handy when plotting waveforms in terms of an index.

=====

```
let    a = vector(16)
set    pensize = 2
plot   a vs a
```



**\*\*\*\*\*Need\_complex\_inputs\*\*\*\*\***

Often fft/iff functions require a complex input with the complex value set to zero. This makes sense since the same code does both the fft and ifft. This fft will accept both real and complex inputs.

The following is how to turn a real input into a complex input.

=====

```
let    ac = a+j(0)
print  a    ac
```

Constant values  
constants Tue Dec 16 22:28:46 GMT 2008

Index	a	ac
0	0.00000e+00	( 0.00000e+00, 0.00000e+00 )
1	1.00000e+00	( 1.00000e+00, 0.00000e+00 )
2	2.00000e+00	( 2.00000e+00, 0.00000e+00 )
3	3.00000e+00	( 3.00000e+00, 0.00000e+00 )
4	4.00000e+00	( 4.00000e+00, 0.00000e+00 )
5	5.00000e+00	( 5.00000e+00, 0.00000e+00 )
6	6.00000e+00	( 6.00000e+00, 0.00000e+00 )
7	7.00000e+00	( 7.00000e+00, 0.00000e+00 )
8	8.00000e+00	( 8.00000e+00, 0.00000e+00 )
9	9.00000e+00	( 9.00000e+00, 0.00000e+00 )
10	1.00000e+01	( 1.00000e+01, 0.00000e+00 )
11	1.10000e+01	( 1.10000e+01, 0.00000e+00 )
12	1.20000e+01	( 1.20000e+01, 0.00000e+00 )
13	1.30000e+01	( 1.30000e+01, 0.00000e+00 )
14	1.40000e+01	( 1.40000e+01, 0.00000e+00 )
15	1.50000e+01	( 1.50000e+01, 0.00000e+00 )

**\*\*\*\*\*The\_output\_Format\_depends\_on\_Input\_Format\*\*\*\*\***

```
print  fft(a)  fft(ac)
```

constants Tue Dec 16 22:28:46 GMT 2008

Index	fft(a)	fft(ac)
0	7.50000e+00	( 7.50000e+00, -0.00000e+00 )
1	-1.00000e+00	( -5.00000e-01, 2.51367e+00 )
2	-5.02734e+00	( -5.00000e-01, 1.20711e+00 )
3	-1.00000e+00	( -5.00000e-01, 7.48303e-01 )
4	-2.41421e+00	( -5.00000e-01, 5.00000e-01 )
5	-1.00000e+00	( -5.00000e-01, 3.34089e-01 )
6	-1.49661e+00	( -5.00000e-01, 2.07107e-01 )
7	-1.00000e+00	( -5.00000e-01, 9.94562e-02 )
8	-1.00000e+00	( -5.00000e-01, -4.89859e-16 )
9	-1.00000e+00	( -5.00000e-01, -9.94562e-02 )

```

10    -6.68179e-01 (-5.00000e-01,-2.07107e-01 )
11    -1.00000e+00 (-5.00000e-01,-3.34089e-01 )
12    -4.14214e-01 (-5.00000e-01,-5.00000e-01 )
13    -1.00000e+00 (-5.00000e-01,-7.48303e-01 )
14    -1.98912e-01 (-5.00000e-01,-1.20711e+00 )
15    -5.00000e-01 (-5.00000e-01,-2.51367e+00 )

```

If the input is real as for **a** , the output will be formatted such that the real values are stored at odd index values, and values that follow it are the imaginary parts. For instance the `fft(a)` at index = 1 is **-1**. The value at index = 2 is **-5.02734e+00**. This corresponds to the first frequency bin as having a complex value of **-1.00000e+00 -j\*5.02734e+00**. The next frequency bin is **-1.00000e+00 -j\*2.41421e+00**.

If the input is complex as for **ac**, the output is stored in a euler identity format which will be discussed later. Both `fft(a)` and `fft(ac)` contain the same data. They are just formatted differently.

**\*\*\*\*\*DC\_Works\*\*\*\*\***

```

let      b = unitvec(16)
let      bc = b+j(0)
print    fft(b)  fft(bc)

```

```

-----
Index    fft(b)          fft(bc)
-- hit return for more, ? for help --
-----
0         1.00000e+00 ( 1.00000e+00,-0.00000e+00 )
1        -7.69589e-17 (-3.84795e-17,-7.65404e-18 )
2         1.53081e-17 (-3.69570e-17,-1.53081e-17 )
3        -7.39140e-17 (-1.14551e-17,-7.65404e-18 )
4         3.06162e-17 (-3.06162e-17,-3.06162e-17 )
5        -2.29102e-17 (-5.11427e-18,-7.65404e-18 )
6         1.53081e-17 (-6.34082e-18,-1.53081e-17 )
7        -6.12323e-17 (-1.52248e-18,-7.65404e-18 )
8         6.12323e-17 ( 0.00000e+00,-6.12323e-17 )
9        -1.02285e-17 ( 1.52248e-18,-7.65404e-18 )
10        1.53081e-17 ( 6.34082e-18,-1.53081e-17 )
11       -1.26816e-17 ( 5.11427e-18,-7.65404e-18 )
12        3.06162e-17 ( 3.06162e-17,-3.06162e-17 )
13       -3.04497e-18 ( 1.14551e-17,-7.65404e-18 )
14        1.53081e-17 ( 3.69570e-17,-1.53081e-17 )
15        0.00000e+00 ( 3.84795e-17,-7.65404e-18 )

```

First to check out is the DC term. It should happen at index = 0 and be real.

Next a cosine is added.

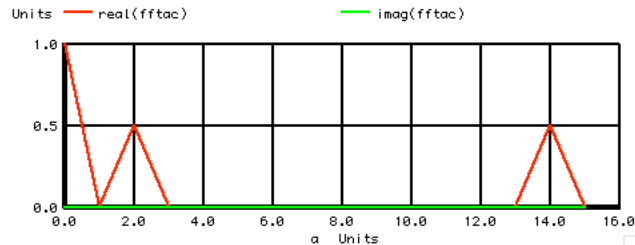
**\*\*\*\*\*DC\_Plus\_cos\_AC\*\*\*\*\***

```

let      numb = length(ac)
print    numb
let      indx = 0
repeat   $&numb
let      ac[indx] = cos(indx*360/8)+1 +j(0)
let      indx = indx +1
end

let      fftac = fft(ac)
plot     real(fftac) imag(fftac) vs a          title DC_Plus_COS

```



`numb = 1.60000e+01`

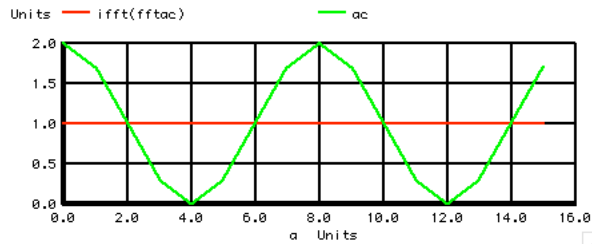
For this version of FFT, apparently the **-jw** terms are stored at the frequency bins above nyquist. Since there are 16 data points, the true bins are from 0 to 8. The values for a unit cosine comes out at 1/2. So it appears to use the following function.

$$\cos(w) = \exp(jw)/2 + \exp(-jw)/2$$

In this case  $jw$  is unit 2, and  $-jw$  is bin 16-2

\*=====DC\_Plus\_cos\_Remove\_AC=====

```
let  fftac[2]  = (0,0)
let  fftac[14] = (0,0)
let  ifftac   = ifft(fftac)
plot  ifftac   ac          vs a          title COS_REMOVED
```



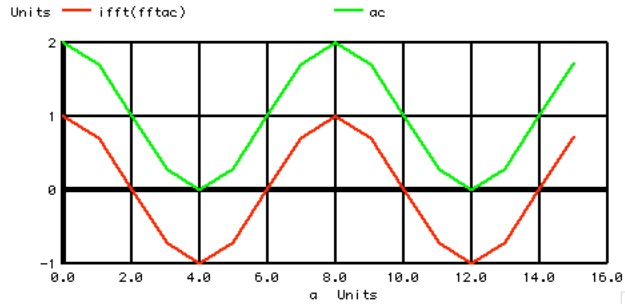
The  $jw$  terms for the cosine wave can be removed and the inverse  $fft$  or  $ifft$  should produce a pure DC wave. Both the  $jw$  and  $-jw$  need to be removed to remove the cosine wave.

\*=====DC\_Plus\_cos\_Remove\_DC=====

```
let  indx      = 0
repeat $&numb
let  ac[indx]  = cos(indx*360/8) +1 +j(0)
let  indx      = indx +1
end

let  fftac     = fft(ac)
plot  real(fftac) imag(fftac) vs a

let  fftac[0]  = (0,0)
let  ifftac    = ifft(fftac)
plot  ifftac   ac          vs a          title COS_With_DC_REMOVED
```



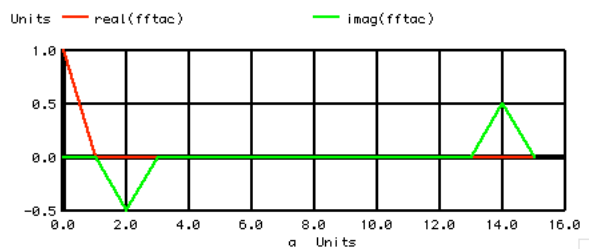
When the DC term is removed from the  $fft$  output, the inverse  $fft$  or  $ifft$  should produce a pure AC wave.

Note the imaginary part of the DC term must be zero.

\*=====DC\_Plus\_sin\_AC=====

```
let  indx      = 0
repeat $&numb
let  ac[indx]  = sin(indx*360/8) +1 +j(0)
let  indx      = indx +1
end

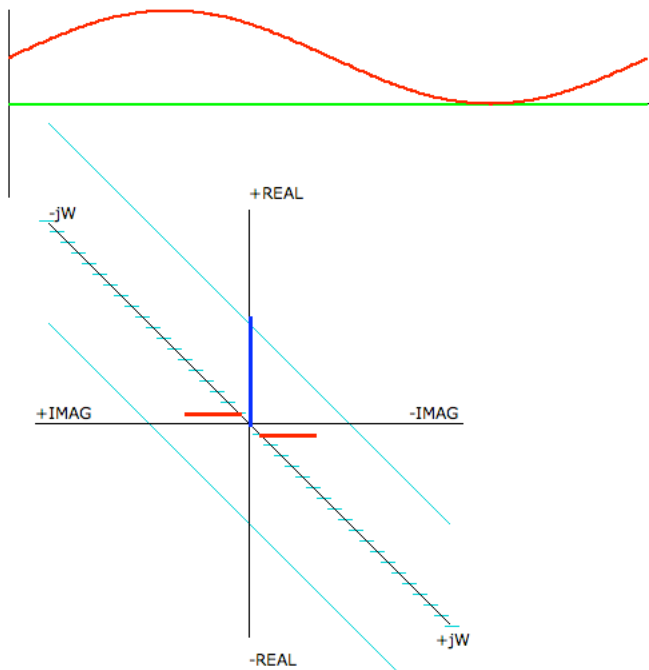
let  fftac     = fft(ac)
plot  real(fftac) imag(fftac) vs a          title DC_Plus_SIN
```



It looks like the polarities and magnitude for the  $j\omega$  terms are all Euler. The imaginary terms for a sine wave are more like a video mirror image of each other.

$$\sin(\omega) = \frac{\exp(j\omega)}{2j} + \frac{\exp(-j\omega)}{2j}$$

A 3D plot shows how the DC term and the sine wave relate.

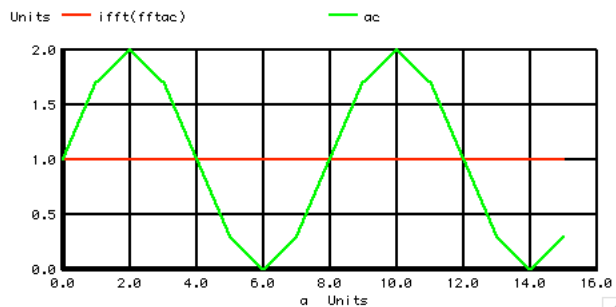


#4\_DC\_offset\_Sin Wave\_@w0 + DC\_@1

[http://www.idea2ic.com/PlayWithJavascript/3D\\_FFT8v.html](http://www.idea2ic.com/PlayWithJavascript/3D_FFT8v.html)

\*\*\*\*\*DC\_Plus\_sin\_Remove\_AC\*\*\*\*\*

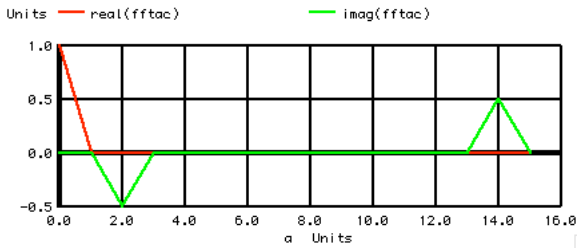
```
let fftac[2] = (0,0)
let fftac[14] = (0,0)
let ifftac = ifft(fftac)
plot ifft(fftac) vs a
```



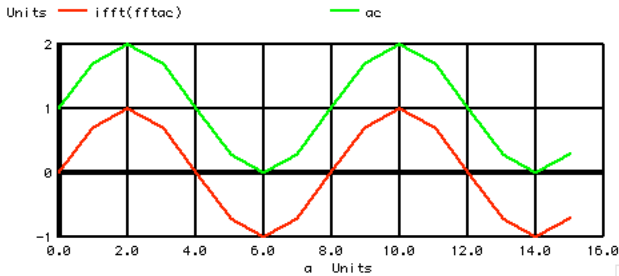
The  $j\omega$  terms for the Sine wave can be removed and the inverse  $fft$  or  $ifft$  should produce a pure DC wave. Both the  $j\omega$  and  $-j\omega$  need to be removed to remove the Sine wave.

\*\*\*\*\*DC\_Plus\_sin\_Remove\_DC\*\*\*\*\*

```
let indx = 0
repeat $numb
let ac[indx] = sin(indx*360/8)+1 +j(0)
let indx = indx +1
end
let fftac = fft(ac)
plot real(fftac) imag(fftac) vs a title DC_Plus_SIN
```



```
let fftac[0] = (0,0)
let ifftac = ifft(fftac)
plot ifft(fftac) ac vs a title SIN_With_DC_REMOVED
```

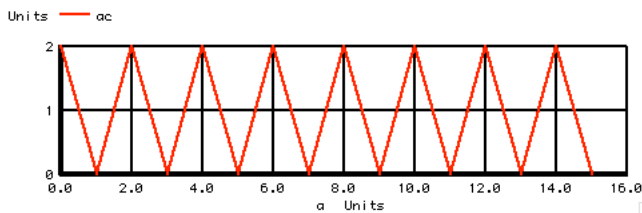


When the DC term is removed from the fft output, the inverse fft or ifft should produce a pure AC wave.

Note the imaginary part of the DC term must be zero.

```
*=====DC_Plus_COS_NYQUIST=====
let indx = 0
repeat $&numb
let ac[indx] = cos(indx*360/2)+1 +j(0)
let indx = indx +1
end
```

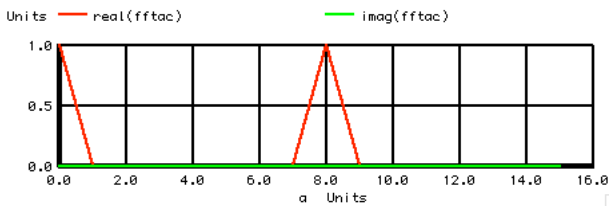
```
plot ac vs a
```



Since there are only 16 data points, maximum number of cycles that can be detected is 8 using a cosine.

```
*=====COS_NYQUIST=====
```

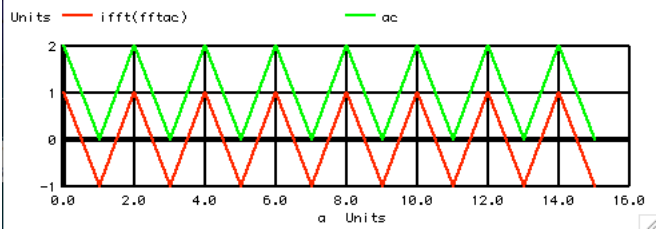
```
let fftac = fft(ac)
plot real(fftac) imag(fftac) vs a
```



Bin number eight is doing the same thing as bin zero. The two half\_gain jw terms add up to one went w goes to zero. Those two terms are right on top of each other. They appear to be doing this at bin 8 as well.

```
*=====DC_Plus_COS_NYQUIST_LESS_DC=====
```

```
let fftac[0] = (0,0)
let ifftac = ifft(fftac)
plot ifft(fftac) vs a
```



**\*=====DC\_Plus\_SIN\_NYQUIST=====**

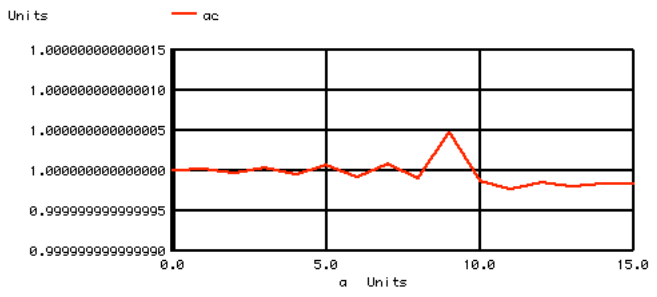
```

let      indx      = 0
repeat  $&numb
let      ac[indx]  = sin(indx*360/2)+1 +j(0)
let      indx      = indx +1
end

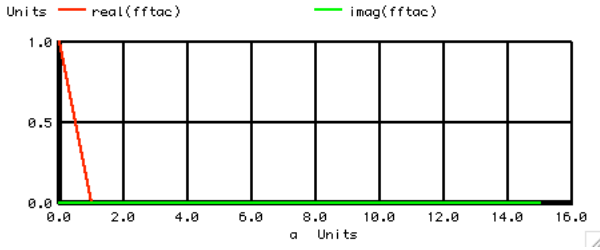
let      fftac     = fft(ac)
plot     real(fftac) imag(fftac) vs a

let      fftac[0]  = (0,0)
let      ifftac    = ifft(fftac)
plot     ifft(fftac) vs a

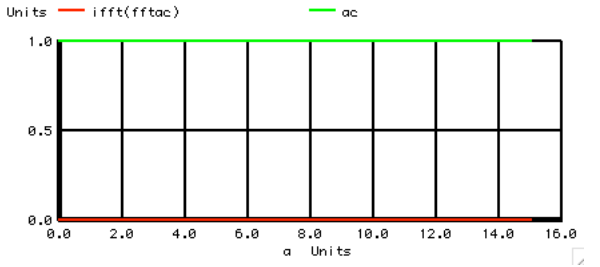
```



For a sine wave, all the 16 points are sampling when the sine wave is at zero. So the imaginary term a unit 8 must equal zero.



Like for the cosine wave, the opposite polarity **jw** terms of a sine wave stack up on top of each other at bin 0 and bin 8. So they cancel out to zero.



**\*=====REMOVE\_ONE\_JW\_Term=====**

```

let      indx = 0
repeat  $&numb
let ac[indx]= cos(indx*360/8)+1 +j(0)
let indx =  indx +1
end

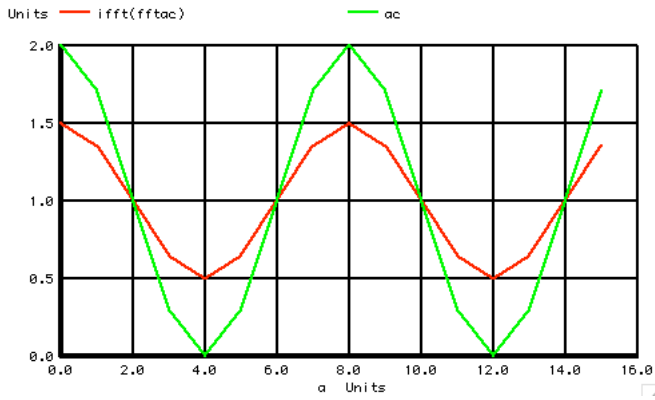
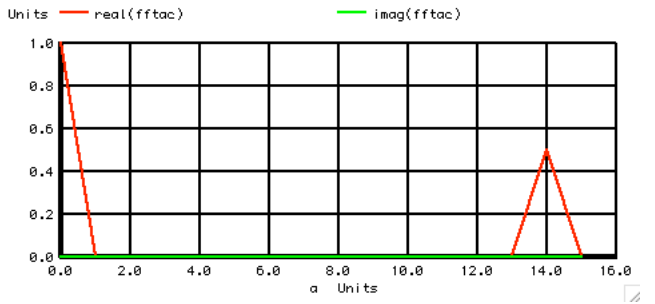
let fftac=  fft(ac)

```

```

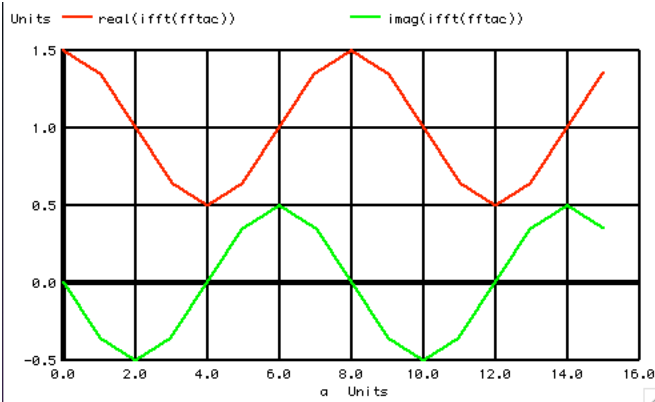
let fftac[2]= (0,0)
plot real(fftac) imag(fftac) vs a title DC_Plus_Cos
let ifftac = ifft(fftac)
plot ifft(fftac) ac vs a title ONE_BIN_REMOVED

```



Subtracting just one of the  $j\omega$  terms you would think would follow something like this equation..

$$e(j\omega) = \cos(\omega) - j\sin$$



Apparently it does...

Wonder what imaginary time could be used for?

=====Full\_Netlist\_For\_Copy\_Paste=====

```

FFT tests
.control
let a = vector(16)
*plot a vs a
set pensize = 2
*=====Need_a_complex_input=====
let ac = a+j(0)
print a ac
*plot fft(a) vs a
*plot real(fft(ac)) imag(fft(ac)) vs a
print fft(a) fft(ac)

*=====DC Works=====
let b = unitvec(16)

```

```

let bc = b+j(0)
*plot fft(b) vs a                               title DC_WORKS
print fft(b) fft(bc)

*=====DC_Plus_cos_Remove_AC=====
let   numb = length(b)
print   numb

let   indx = 0
repeat $&numb
let   ac[indx]= cos(indx*360/8)+1 +j(0)
let   indx = indx +1
end

let   fftac=fft(ac)
plot  real(fftac)      imag(fftac) vs a   title DC_Plus_COS

let   fftac[2]=(0,0)
let   fftac[14]=(0,0)
let   ifftac = ifft(fftac)
plot  ifftac ac vs a                               title COS_REMOVED

*=====DC_Plus_cos_Remove_DC=====
let   indx = 0
repeat $&numb
let   ac[indx]= cos(indx*360/8)+1 +j(0)
let   indx = indx +1
end

let   fftac=fft(ac)
let   fftac[0]=(0,0)
let   ifftac = ifft(fftac)
plot  ifftac ac vs a                               title COS_With_DC_REMOVED

*=====DC_Plus_sin_Remove_AC=====
let   indx = 0
repeat $&numb
let   ac[indx]= sin(indx*360/8)+1 +j(0)
let   indx = indx +1
end

let   fftac=fft(ac)
plot  real(fftac) imag(fftac) vs a           title DC_Plus_SIN

let   fftac[2]=(0,0)
let   fftac[14]=(0,0)
let   ifftac = ifft(fftac)
plot  ifft(fftac) ac vs a                     title SIN_REMOVED

*=====DC_Plus_sin_Remove_DC=====
let   indx = 0
repeat $&numb
let   ac[indx]= sin(indx*360/8)+1 +j(0)
let   indx = indx +1
end

let   fftac=fft(ac)
let   fftac[0]=(0,0)
let   ifftac = ifft(fftac)
plot  ifft(fftac) ac vs a                     title SIN_With_DC_REMOVED

*=====DC_Plus_cos_Nyquist_Remove_DC=====
let   indx = 0
repeat $&numb
let   ac[indx]= cos(indx*360/2)+1 +j(0)
let   indx = indx +1
end

plot  ac vs a                                   title Nyq_COS

let   fftac=fft(ac)
plot  real(fftac)  imag(fftac) vs a           title Nyq_FREQ_COS
let   fftac[0]=(0,0)
let   ifftac = ifft(fftac)
plot  ifft(fftac) ac vs a                     title COS_With_DC_REMOVED

*=====DC_Plus_sin_Nyquist_Remove_DC=====
let   indx = 0
repeat $&numb
let   ac[indx]= sin(indx*360/2)+1 +j(0)
let   indx = indx +1
end

plot  ac vs a                                   title Nyq_SIN

let   fftac=fft(ac)
plot  real(fftac)  imag(fftac) vs a           title Nyq_FREQ_SIN
let   fftac[0]=(0,0)
let   ifftac = ifft(fftac)
plot  ifft(fftac) ac vs a                     title COS_With_DC_REMOVED

*=====DC_Plus_COS_Remove_One_BIN=====
let   indx = 0
repeat $&numb
let   ac[indx]= cos(indx*360/8)+1 +j(0)
let   indx = indx +1
end

let   fftac=fft(ac)
let   fftac[2]=(0,0)
plot  real(fftac)  imag(fftac) vs a           title DC_Plus_Cos

let   ifftac = ifft(fftac)

```

```
plot ifft(fftac) ac vs a          title ONE_BIN_REMOVED
plot real(ifft(fftac)) imag(ifft(fftac)) vs a    title ONE_BIN_REMOVED
.endc
.end
```