

Hi all,

I need to refresh my memory on practical applications of FFTs and wavelet transforms. And apparently also on how to handle plotting wavelet transform results using cgImage. The code below creates a sample signal from which an FFT, Hanning filtered FFT and a Morlet transform are calculated. No big surprises with the FFTs, but the wavelet result has me stumped on several levels.

Firstly, I would expect to see three bands at 5, 20 and 200 Hz in the wavelet transform, but there are only two, and they're oddly spaced. I suspect the `wv_cwt` output needs scaled/rearranged to obtain the frequency/time plot I seek to make... but the IDL 8 documentation is not very helpful with that. Does anyone know how to properly scale this?

Secondly, I've spent several hours now trying various combinations of `cgImage`, `cgPlot`, `cgAxis` to get the axes on the wavelet plot to reflecting what's really there... and not just the axis value index number (as the `/AXES` option does). The one avenue that seems most promising is `cgPlot` with `/nodata` to set up the axes, then use `cgImage` to draw the picture data. But then I'd need the tick marks to face outside, but the `tick_direction` keyword is not supported with `plot`, or `axis` or any other commands really. And what the hell is up with `xa = axis('x'....)` not being the same as `axis`, 'x'... they're apparently fundamentally different calls that don't support the same arguments?

Any insights greatly appreciated. Here's code:

PRO question

```
!p.multi = [0, 2, 3] ; plot 1 x 3 plots on one sheet
SET_PLOT, 'PS'
DEVICE, filename='question.ps', XSIZE=21, YSIZE=25, /ENCAPSULATED
```

```
; Generate the test data
time=DINDGEN(5000)*10d/5000           ; 5000 points, 10 seconds
; test signals:
; X Hz = sin(X * 2.0d * !DPI * time)
y = sin(5d * 2.d * !DPI * time) + $
    sin(20d * 2.d * !DPI * time) + $
    ;sin(0.5*time * 2.d * !DPI * time) + $
    sin(300 * 2.d * !DPI * time)      ; this last component will alias.
```

```
; Plot 1 second's worth of the original function
CGPLOT, time[0:499], y[0:499], TITLE="Original function (1s)"
CGPLOT, time, y, TITLE="Original function"
```

```
samplingInterval = time[1] - time[0]
```

```

samplingFreq = 1.0D / samplingInterval
Nyquist = 1.0D / (2.0D * samplingInterval)
freq = DINDGEN(N_ELEMENTS(time)) * samplingFreq / (N_ELEMENTS(time)-1)
PRINT, 'sampling interval', samplingInterval
PRINT, 'Nyquist', nyquist
PRINT, 'sampling Freq', samplingFreq
timespan = time[N_ELEMENTS(time)-1]-time[0]
PRINT, 'time span [s]', timespan

dft = FFT(y)

; Now fold the transform
dft = dft[0:N_ELEMENTS(dft)/2+1] ; cut in half
dft /= N_ELEMENTS(dft) ; scale
dft[1:N_ELEMENTS(dft)-1] *= 2.0D ; compensate for the discarded half
phiSpec = ATAN(dft) ; the phase spectrum
magSpec = ABS(dft) ; magnitude spectrum
powSpec = (dft)^2 ; power spectrum
newfreq = freq[0:N_ELEMENTS(freq)/2+1] ; also scale the frequency (which really is the same
as scaling to the Nyquist freq

maxy = max(magSpec,min=miny)
CGPLOT, freq, magSpec, TITLE='FFT Magnitude with aliased peak. Timespan: ' +
STRING(FORMAT='(I0)', timespan) + 's', YRANGE=[miny,maxy], XRange=[0,nyquist],
XTITLE='Freq [Hz]', /YLOG

hann = FFT( HANNING(N_ELEMENTS(y)) * y )
; Fold this one also
hann = hann[0:N_ELEMENTS(hann)/2+1] ; cut in half
hann /= N_ELEMENTS(hann) ; scale
hann[1:N_ELEMENTS(hann)-1] *= 2.0D ; compensate for the discarded half

hannMagSpec = ABS(hann)
maxy = max(hannMagSpec,min=miny)
CGPLOT, freq, hannMagSpec, TITLE='FFT Magnitude Hanning filtered. Timespan: ' +
STRING(FORMAT='(I0)', timespan) + 's', YRANGE=[miny,maxy], XRange=[0,nyquist],
XTITLE='Freq [Hz]', /YLOG

; Compute the wavelet transform and the power.
wave = WV_CWT(y, 'morlet', 6, /PAD, SCALE=scales)
wavePower = ABS(wave)^2
; Convert scales to time units.
scales *= samplingFreq

; And plot the wavelet.
cgLOADCT, 13, RGB_TABLE=palette
cgPLOT, time-time[0], newfreq, XTITLE="Time [s]", YTITLE="Freq [Hz]", /NODATA
cgIMAGE, wavePower, PALETTE=palette, /OVERPLOT

```

DEVICE,/CLOSE

STOP

END
