AST2210 - Lab exercise FFT

1 Goal

This lab will illustrate the use of Fast Fourier Transform (FFT) techniques in signal analysis and image processing. IDL is used throughout as programming language.

2 FFT in IDL

IDL has a built in function to calculate the Fourier transform of a function. The syntax is:

```
IDL> fy=fft(y)
IDL> y=fft(fy,/inverse)
```

where the first form calculates the discrete FFT of the function `y` and stores the result in the variable `fy`. The second form is the inverse transform going from the Fourier domain to the real domain.

`fy` is a complex array with the same number of elements as the function `y`. The result of the inverse transform is also complex. The Fourier transform contains both positive and negative frequencies. For the transform of a real valued function `y`, the negative frequencies are the complex conjugate of the positive frequencies. In IDL (and most implementations of discrete Fourier transforms) the zero element corresponds to $n = 0$, positive frequencies $0 < \nu < \nu_c$ correspond to indices $1 \leq n \leq N/2-1$ while negative frequencies $-\nu_c < \nu < 0$ correspond to $N/2-1 \leq n \leq N-1$ where $\nu_n \equiv \frac{n}{N\Delta}$ and ν_c is the Nyquist frequency.

Exercises 1-5 are from the lecture notes on FFT

Exercise 1: Compute the amplitude of the transform of $f = sinx$. In plotting this amplitude, what should be used for the x-axis?

Exercise 2: What happens if you set the edges of f to zero: e.g. multiply f by a stepfunction s such as

```
IDL> s=fltarr(2000)
IDL> s[200:1800]=1.0
```

Overplot the amplitude of the tranform as you narrow the region of s that is equal to one.

Exercise 3: Compute the transform of step-functions of various widths.

Exercise 4: Add sinusoidal functions to f with different frequencies and check the resulting transforms. What happens to the transform if you add a constant to f?

Exercise 5: Consider a function g given by the sum of three sinusoidals of differing frequency. Construct such a g and remove one of the frequencies from g using forward and inverse transforms FFT's in IDL.

Exercise 6: Power spectrum

A common use of Fourier analysis is to find dominant periods in time-dependent signals. Read in an example function:

```
IDL> restore,'~matsc/kurs/ast2210/fft_example1.idl',/verbose
```

You need to be logged into one of the astrophysics machines for this to work (e.g. nekkar). The restore command reads in two arrays, `t` and `v` into IDL. The example is the photospheric velocity (in m/s) in a sunspot as function of time (in seconds). Plot the power as function of frequency:

```
IDL> n=n_elements(t)
IDL> nu=findgen(n)/n*1000./(t[1]=t[0])
IDL> plot,nu,abs(fft(v))^2,/ylog,xtitle='Frequency [mHz]'
```

Note that you now see both positive and negative frequencies mirrored around the Nyquist frequency of 50 mHz. Since we have a real-valued function we are not interested in the negative frequencies and should therefore never plot frequencies above the Nyquist frequency. Restrict the plotting range with the `xrange` keyword and plot with linear y-scale

```
IDL> plot,nu,abs(fft(v))^2,xtitle='Frequency [mHz]',xrange=[0,20]
```

At what frequency is the maxmimum power? What period does this correspond to?

In the zero index of the Fourier transform you have the mean of the function. Check this.

Exercise 7: Low-pass Fourier filtering

It seems the time-series contains a dominant period with white noise on top of the signal. If we assume that all frequencies above a given frequency is noise we can filter that out by zeroing out that part of the Fourier transform before transforming back. It is important to filter both the positive and negative frequencies identically. If we assume that all frequencies above 4.5 mHz only contain noise we can do this filtering with the following procedure (in the example below you don't have to type in the comments (starting with ;))

```
IDL> iw=where(nu lt 4.5,count)  ; find part to keep
IDL> iw=[iw,reverse(n-iw[1:*])] ; extend with corresponding negative frequencies
IDL> filter=fltarr(n)           ; filter function initialized to zero
IDL> filter[iw]=1.0             ; keep low-frequency part
IDL> fv=fft(v)                  ; Fourier transform
IDL> fv2=fv*filter              ; filtered transform
IDL> v2=fft(fv2,/inverse)       ; transform back
IDL> v2=float(v2)               ; convert from complex to float
```

Compare the unfiltered sequence with the filtered; both the velocities and the power spectra. Since we keep the low frequencies this type of filtering is called low-pass filtering. For real applications one normally uses a smoother transition from one to zero in the filter.

Exercise 8: Wiener-filtering

It is possible to show there is an optimum filter to get rid of noise if one has a good model of the noise frequency statistics. This filter is called Wiener filter and the filter value is $P(S)/(P(S)+P(N))$ where $P(S)$ is the power of the signal and $P(N)$ is the power of the noise. Construct such a filter by assuming white noise (power constant with frequency) for the example above and compare the result with the low-pass filtering.