UNIVERSITY OF EDINBURGH

DEPARTMENT OF GEOLOGY AND GEOPHYSICS

Mathematical Modelling of Geophysical Systems

Time Series Analysis

Ian Bastow 2001

Mathematical Modelling of Geophysical Systems

Time Series Analysis

---

1. Introduction

The aim of this exercise is to generate a time series for a simple monochromatic cosine wave and investigate its power spectrum using a range of pre-processing filters. The need for time series analysis in geophysical systems and the advantages and disadvantages of a range of pre-processing techniques will be discussed in this report.

The time series to be analysed is represented by a cosine wave (1):

$$X(t) = A\cos(2pft + f) \qquad \textbf{(1)}$$

Where, A is the amplitude, f is the frequency, t is time and ϕ is the phase of the cosine wave X(t).

2. Theory and Background Information

2.1 Time Versus Frequency Domain Analysis

Many geophysical data are ordered systematically either by time or position or both. Our initial measurements are generally in the time domain and we can describe these as time series. However, in many cases it is more diagnostic to consider data in the frequency domain. In order to achieve this we use the Discrete Fourier Transform (DFT) (2) to convert between the time and frequency domains.

$$F(x(t)) = X(f) = \int_{-\infty}^{+\infty} x(t)\exp(i2pft)dt \qquad \textbf{(2)}$$

Where, x(t) describes the function or data in the time domain, f is frequency and t is time.

As we will see in section 3, any noise component in a time series can disguise it's diagnostic frequency domain characteristics considerably. Thus Fourier methods are necessary.

A principal disadvantage of using the DFT to work in the frequency domain is that all knowledge of the temporal distribution of the information is lost. Although Complex demodulation and wavelet transforms are recognised methods of obtaining optimal time and frequency domain information simultaneously, this report will focus principally on the frequency spectrum.

2.2 Power Spectra in the Time and Frequency Domains

The total power is the variance of the time series and is the same in the time and frequency domains. The difference is whether we are integrating the squared samples over time or the squared Fourier transforms over frequency. We thus define Parseval's Theorem (3):

$$TotalPower = \int_{-\infty}^{+\infty} |\, x(t)\,|^2 dt = \int_{-\infty}^{+\infty} |\, X(f)\,|^2 df \qquad \textbf{(3)}$$

The simplest method of computing the power per unit frequency bandwidth – "*power spectrum*" for short follows from Parseval's theorem:

$$\hat{S} = |\, DFT(x(t))\,|^2 \qquad \textbf{(4)}$$

This estimate is called the raw power spectrum or ***periodogram*** and because the original data are real, the spectrum is symmetric about zero frequency. Spectral estimates are distributed at arithmetic intervals separated by:

$$df = \frac{1}{Ndt} \qquad \textbf{(5)}$$

This approach to the computation of power spectra can cause problems. Time series in reality are finite in length so a practical data set will be restricted to a finite range, say from –T/2 to +T/2. This means that we are actually computing (6) instead of (4).

$$\hat{S} = |\, DFT(x(t)d(t))\,|^2 \qquad \textbf{(6)}$$

Where, d(t) (a so called ***boxcar***) is the data window (1 from –T/2 to T/2, zero everywhere else). This means that the spectrum we obtain is the convolution of $|X(f)|^2$ with the spectrum of d(t). The appendix at the back of this report shows a proof of the Fourier transform W(f) of d(t). The result is a sinc function (6):

$$W(f) = T\left(\frac{\sin pfT}{pfT}\right) \qquad \textbf{(7)}$$

Where, T is the width of the data window.

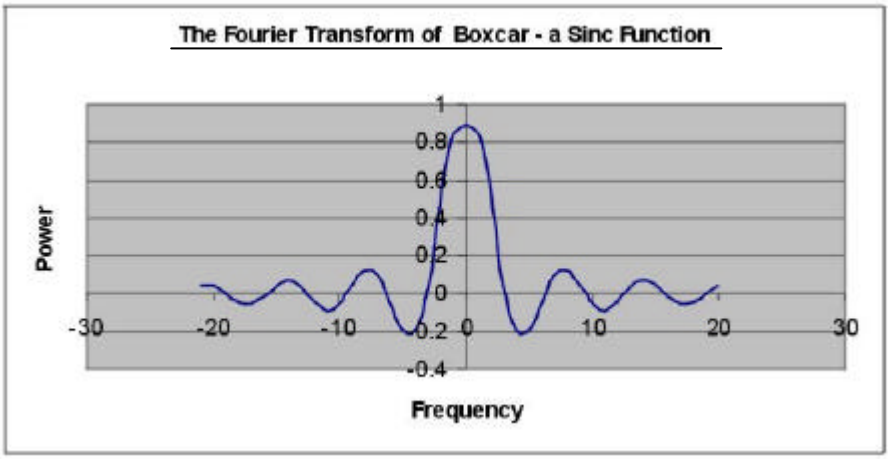

Fig. 1

When the sinc function is convolved with $|X(f)|^2$ the spectrum is given substantial side lobes (see fig. 1) which have the undesirable effect of transferring the power from the correct location to adjacent frequencies, thus biasing the spectra; estimate. This problem is known as ***leakage,*** which biases the spectral estimate.

2.3 Overcoming the Problem of Leakage

One method of overcoming the problem of leakage is to soften the sharp corners of the boxcar function by using a smooth cosine bell. This means that some of the data is effectively discarded and an undesirable effect is an increase in the variance of the periodogram. It is possible to overcome this problem by using filters which look like boxcars in the middle, but which are smooth at the edges. Examples which will be investigated in this project include:

- A Cosine Bell
- A Boxcar Tapered with a Cosine Bell
- A Welch Window

2.4 The Computation of Fourier Transforms - The Fast Fourier Transform (FFT)

The conventional discrete Fourier transform (2) proves to be computationally expensive for more than a few hundred data points since the number of operations increases with N^2, the square of the number of data points. The Fast Fourier transform, however, only requires $N\log_2 N$ operations and is therefore, more efficient for larger data sets. The FFT relies on being able to factorise the number of data points, maximum efficiency occurring for $N = 2^n$, where n is an integer. Data sets are processed best by the FFT if they are padded out with zeros to the nearest 2^n.

1. Method

The time series to be analysed in this project is represented by (1). The FORTRAN program coswave.f90 (see Appendix) was used to generate the time series. coswave.f90 reads in values for the amplitude, phase and frequency of the cosine wave. The time series is then output to the file output.dat in the form of a single data column.

coswave.f90 was compiled and run for the following input parameters:

- Amplitude (A) – 20
- Phase (ϕ) – 0
- Period (t) – 1 day
- Number of data points (N) – 8784

Once the time series was generated, the program testdata.f (from Dr R J Banks' library of FORTRAN programs at ***/home/rbanks/MRES/bin/testdata)*** was used to regenerate the time series, but this time with a random noise component with a standard deviation of 20.

SPEC1.f (again from Dr R J Banks' libraries) was then used to compute the power spectrum of the time series using the Fast Fourier transform method described in section 2.4. SPEC1.f allows pre-processing of the data using the filters outlines in section 2.3.

The output from these FORTRAN programs is discussed in the next section.

2. Discussion

The program coswave.f90 yielded a time series as shown in fig. 2.

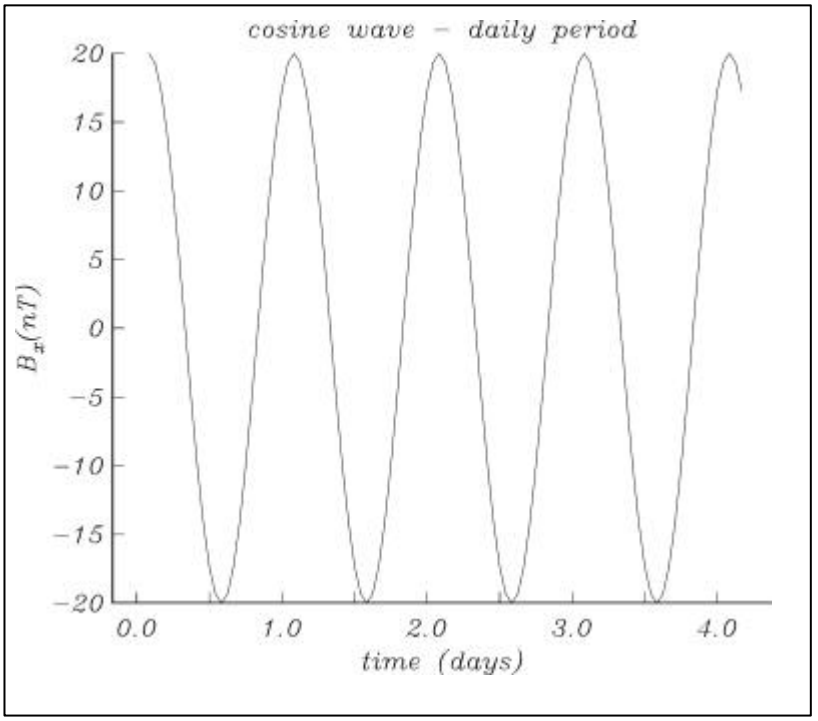

Fig. 2

We now wish to compute the power spectra of the data using a variety of pre-processing techniques as described in section 2.3. We would intuitively expect that the periodogram for a perfect cosine wave such as the one in fig. 2 would be a single spike. Any deviations can be attributed to inefficiencies in the computation of the spectra. Fig. 3 shows the power spectra of the time series, computed after pre-processing with a boxcar function.

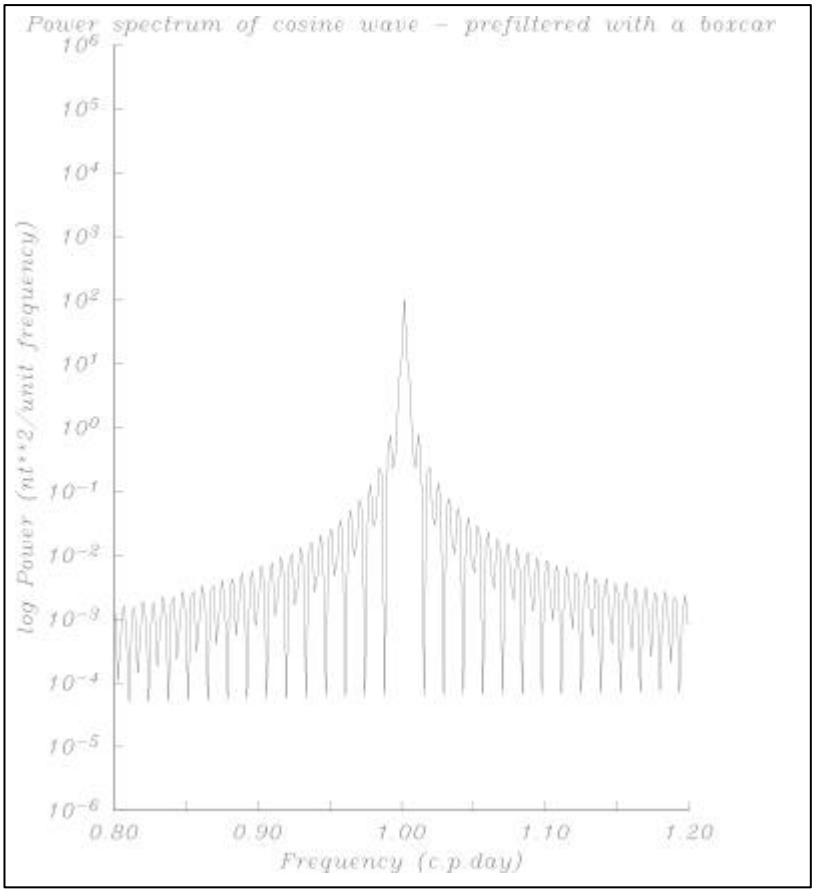

Fig. 3

Here we clearly see the undesirable effect of the convolution of $|X(f)|^2$ with the sinc function from the Fourier transform of the boxcar data window (as described in section 2.2. The side lobes to the right of f = 1000 can therefore, be attributed to leakage.

We now consider the effect of using a cosine bell filter (fig. 4), a boxcar tapered with a cosine bell (fig. 5), and a Welch filter (fig. 6). The boxcar tapered with the cosine bell (fig. 5) has a fractional taper length of 0.05 (5%). In each case, before spec1 was run to compute the power spectra, the data set was zero padded to 12288 data points. This value was chosen because is is easily factorisible ($2^{12}\times3$) and therefore more efficient for the FFT computation. Another reason for choosing N=12288 is that it is easily factorised by 24 hours – the period of the cosine wave which we expect to have maximum power.

Use of a cosine bell filter (fig. 4) results in a much more rapid decay of power in the frequencies adjacent to one cycle per day. Unfortunately, the power of the central peak is lower than that found using the boxcar filter.

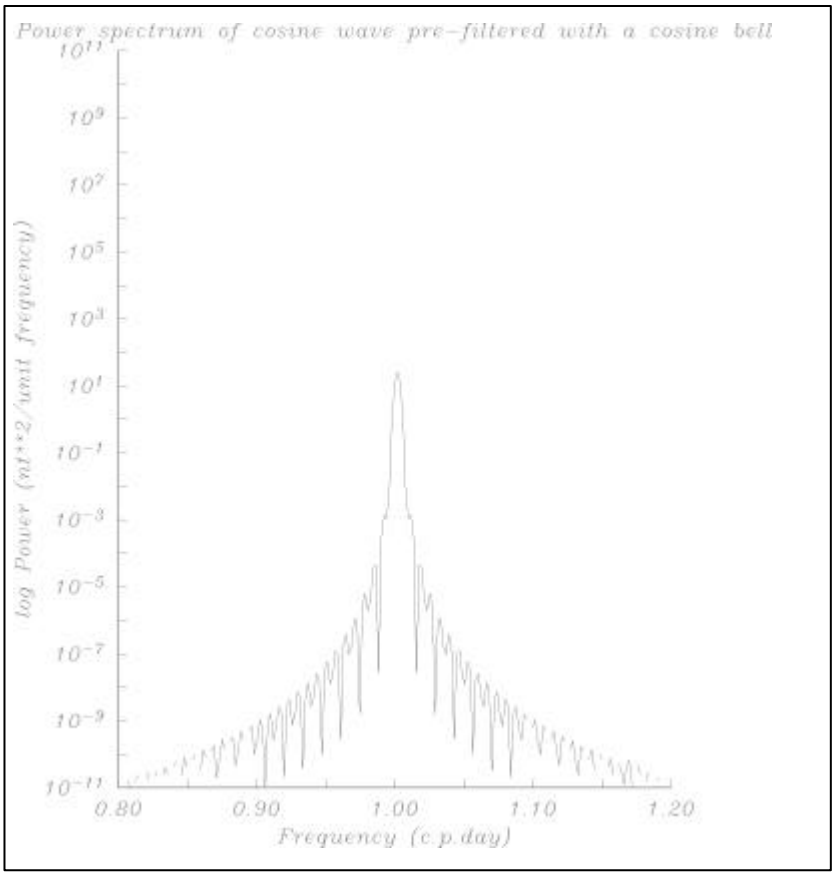
Power spectrum of cosine wave pre-filtered with a cosine bell
log Power (nt**2/unit frequency)
10^11
10^9
10^7
10^5
10^3
10^1
10^-1
10^-3
10^-5
10^-7
10^-9
10^-11
0.80
0.90
1.00
1.10
1.20
Frequency (c.p.day)

Fig. 4

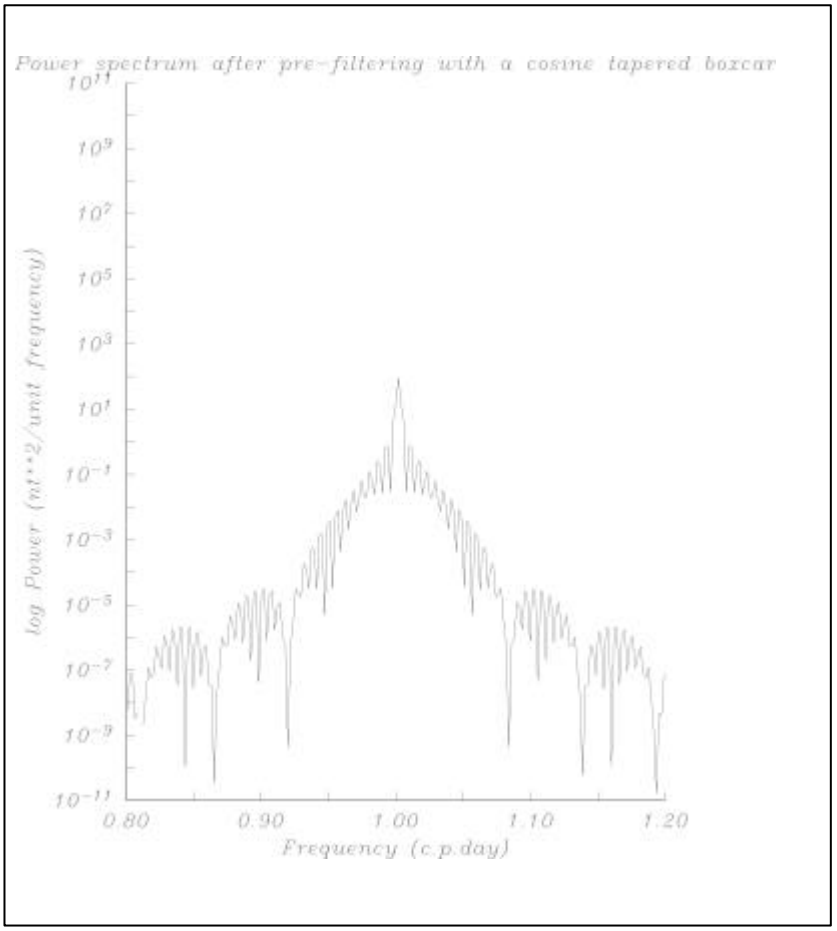
Power spectrum after pre-filtering with a cosine tapered boxcar
log Power (nt**2/unit frequency)
10^11
10^9
10^7
10^5
10^3
10^1
10^-1
10^-3
10^-5
10^-7
10^-9
10^-11
0.80
0.90
1.00
1.10
1.20
Frequency (c.p.day)

Fig. 5

The use of a boxcar tapered with a cosine bell (fig. 4) shows more pronounced side lobes than fig. 4 but the amplitude of the central peak is higher at $10^2 nT^2$/unit frequency. The side lobes are clearly due to the boxcar component of the filter and the higher amplitude occurs because less data is being discarded. The use of a Welch filter produces similar results to the cosine bell of fig. 4.

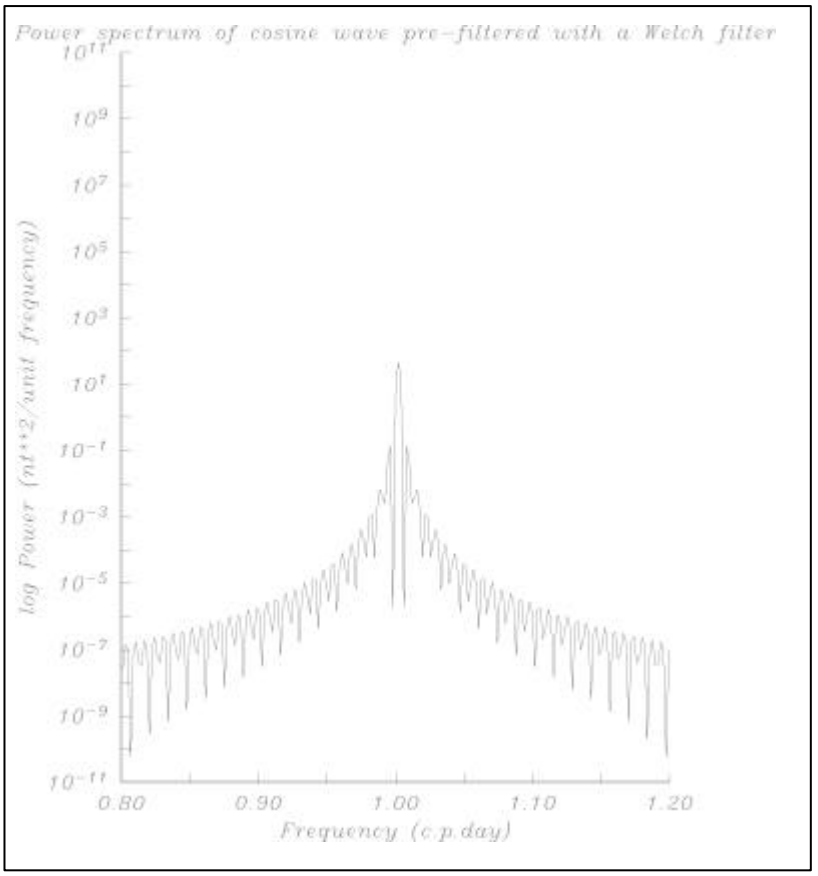

Fig. 6

So far, I have looked only at time series analysis of a perfect cosine function. In reality, most geophysical signals are not perfect cosines at all and they are often noisy. So, the results so far can be compared with the output from testdata.f (see fig. 7). This program was used to generate the same time series as in fig. 2, except this time with an added random noise component with a standard deviation of 20.

Clearly, the introduction of a noise component into the data distorts the appearance of the data in the time domain. The presence of a cosine pattern in the data is evident but unclear nevertheless. Realistic data not generated from ideal mathematical functions are generally noisy and so their time series are complicated.

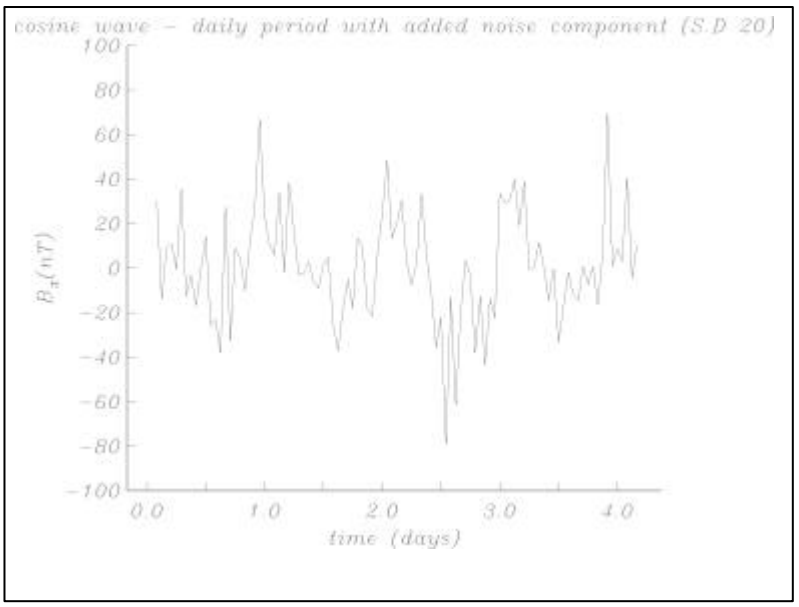

Fig. 7

It is clear, therefore, that it is more diagnostic to consider data in the frequency domain. Fig. 8 shows the power spectra for the noisy data set.

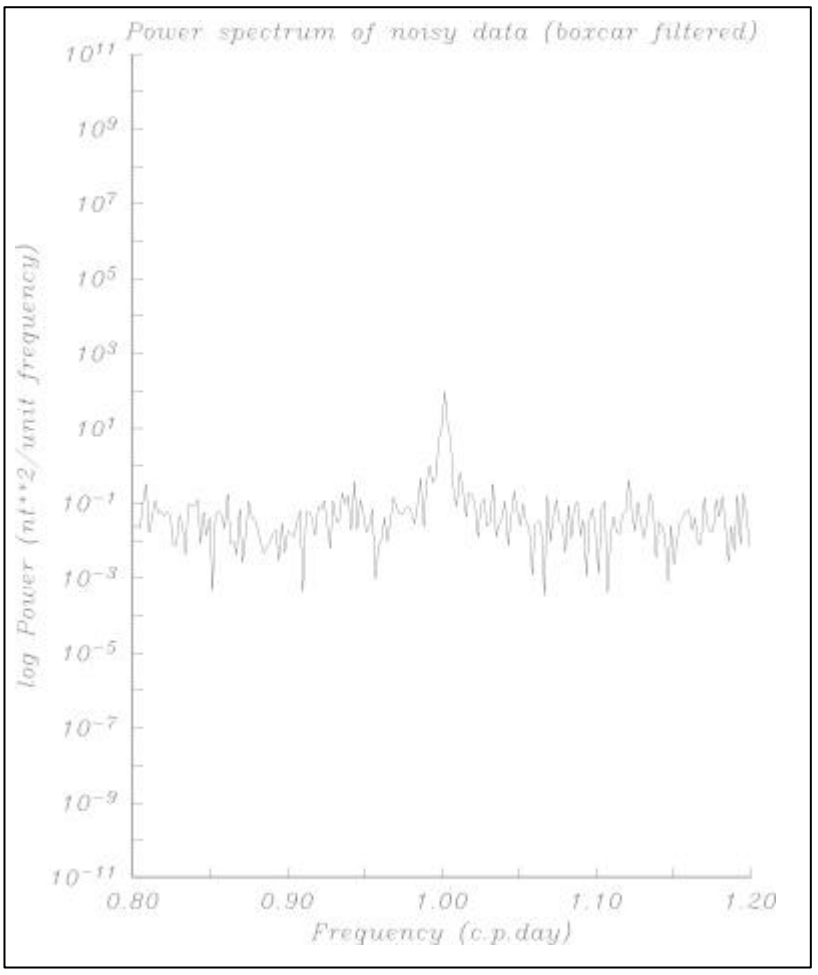

Fig. 8

The central peak at one cycle per day is still recognisable but it is unclear whether other frequencies may exist in the data since the decay of power either side of one cycle per day is not smooth as in fig. 3

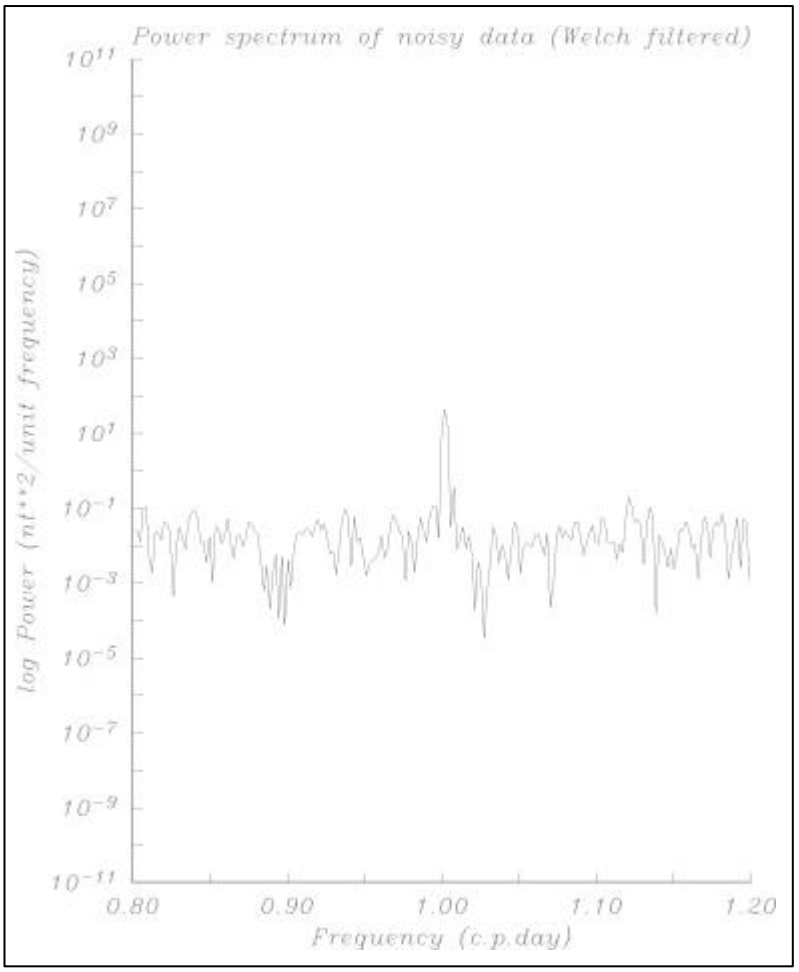

Fig. 9

The use of a Welch filter reduces the side lobe leakage. The central peak, however, is narrower and thus more readily identified. Analysis of the time series with added noise component is clear evidence that careful pre-filtering of raw data is essential.

Conclusions

A good understanding of pre-processing filters is essential to successful spectral analysis of time series. Although the boxcar filter is seemingly the ideal way of focusing on a specific section of data, side lobes in the periodogram occur due to the problem of leakage. Thus it is necessary to smooth the sharp corners of the boxcar in order to reduce the problem.

The use of cosine bell, cosine tapered boxcar and Welch filters at the pre-processing stage all yield better results as they reduce the aliasing problem considerably. However, these methods do cause the problem of a lower power central peak in the power spectrum. This occurs because some of the original data points are discarded in the filtering process. The ideal filter has a boxcar centre with smoothed edges. With this in mind, there is little difference in results obtained using a cosine bell or a Welch filter.

The introduction of noise into the initial data set confirms the need for frequency domain analysis in geophysical time series analysis. Noise can make the time series look inherently complicated and in such cases it is clearly only diagnostic to consider data in the frequency domain. Once again, the use of the Welch filter provides better results than the boxcar.

APPENDIX A – FORTRAN PROGRAMS

A1 – The program coswave.f used to generate the time series.

```
PROGRAM COSWAVE
    IMPLICIT NONE
    CHARACTER*32 :: output
        REAL :: A, f, t, phi, deltaT, s, F, X, n, pi, c
     INTEGER :: P, i
PRINT*, "********************************************************************* "
       PRINT*, "This program is designed to generate a time seies for &
            & The monochromatic cosine wave X(t)=Acos(2pift+phi)"
PRINT*, " ********************************************************************* "
         PRINT*, "Enter a value for the wave amplitude (A)"
        READ*, A
         PRINT*, "Enter a value for the phase (phi) in degrees"
        READ*, phi
         PRINT*, "Enter a value for the period (t) in days"
        READ*, f
          PRINT*, "How long should the data series be?"
        READ*, P
          PRINT*, "What sampling interval would you like to use? (samples &
                    & per day (c)"
      READ*, c
           s=2/c
           F = 1/t
           deltaT=1/Fn
           pi = 4*atan(1.0)
       OPEN (unit=10,FILE='output.dat', STATUS='NEW', ACTION='WRITE')
      DO i = 1, P
      X = A*cos((pi*i*Fn*s) + phi)
        WRITE (unit=10,*) X
      END DO
END PROGRAM COSWAVE
```

A3 – The program used to generate the time series and add a noise component to it. From Dr Roger Banks' Library

```
      PROGRAM testdata
C
C       GENERATES SIMPLE TIME SERIES FOR TESTING ANALYSIS
PROGRAMS.
C-----------------------------------------------------------------------
C       ALL LINEAR COMBINATIONS OF THE FOLLOWING FUNCTIONS ARE
ALLOWED
C       (1)  COSINE WAVE :   X(T) = A*COS(2*PI*F*T + PHI)
C       (2)  LINEAR TREND OR MEAN VALUE
C       (3)  GAUSSIAN RANDOM DEVIATE
C-----------------------------------------------------------------------
C
      PARAMETER (NMAX=10000)
      CHARACTER*20 OUTFIL
      DIMENSION X(10000)
      DATA IOCH/8/, PI/3.1415927/
C
      WRITE (*,2000)
 2000 FORMAT  (///,1X,'GENERATION OF SIMPLE TIME SERIES'/)
C       OPEN CHANNEL FOR OUTPUT OF TIME SERIES
      WRITE (*,2001)
 2001 FORMAT  (/,1X,'SUPPLY NAME OF OUTPUT FILE (MAX.=20 CHARS)')
      READ  (*,3000) OUTFIL
 3000 FORMAT  (A20)
      OPEN (IOCH,FILE=OUTFIL,STATUS='NEW')
C
C       INPUT SERIES LENGTH AND INITIALISE ARRAY
      WRITE (*,2010) NMAX
 2010 FORMAT  (/,1X,'LENGTH OF TIME SERIES (MAX.=',I5,')')
      READ  (*,*) NPTS
      DO 10 N=1,NPTS
   10 X(N)=0.
C
C-----------------------------------------------------------------------
C       CHOOSE WHICH DATA TYPE TO ADD TO THE EXISTING SERIES
   20 WRITE (*,2020)
 2020 FORMAT  (//,1X,'************************************************'/
     1       1X,'*  CHOOSE FUNCTION TO ADD TO EXISTING SERIES   *'/
     2       1X,'*       (WHICH IS ZERO ON FIRST PASS)        *'/
     3       1X,'*  1.  COSINE WAVE : x(t) = A cos(2PIft + phi) *'/
     4       1X,'*  2.  GAUSSIAN RANDOM DEVIATE                *'/
     5       1X,'*  3.  LINEAR TREND OR MEAN LEVEL             *'/
     6       1X,'*  4.  COMPUTATION COMPLETE - OUTPUT SERIES    *'/
     7       1X,'************************************************'/)
      READ  (*,*) ITYPE
      IF ((ITYPE.LT.1).OR.(ITYPE.GT.4)) GO TO 20
      GO TO (30,40,50,200),ITYPE
C
C-----------------------------------------------------------------------
C       COSINE WAVE
   30 WRITE (*,2030)
```

```
 2030 FORMAT  (//,1X,'ADDS COSINE WAVE TO EXISTING SERIES')
      WRITE (*,2031)
 2031 FORMAT  (/,1X,'INPUT 1: AMPLITUDE')
      READ  (*,*) A
      WRITE (*,2032)
 2032 FORMAT  (/,1X,'INPUT 2: FREQUENCY (AS A FRACTION OF THE
NYQUIST)')
      READ  (*,*) F
      PIF=PI*F
      WRITE (*,2033)
 2033 FORMAT  (/,1X,'INPUT 3: PHASE (DEGREES)')
      READ  (*,*) PHI
      PHI=PHI*PI/180.
      DO 35 N=1,NPTS
      PIFN=PIF*FLOAT(N-1)
   35 X(N)=X(N)+A*COS(PIFN+PHI)
      GO TO 100
C
C       GAUSSIAN RANDOM DEVIATES.
C         FOR DETAILS OF THE PROCEDURE USED TO GENERATE THE
C         NOISE, SEE CHAPTER 7 OF "NUMERICAL RECIPES".
   40 WRITE (*,2040)
 2040 FORMAT  (//,1X,'ADDS GAUSSIAN NOISE TO EXISTING SERIES')
      WRITE (*,2041)
 2041 FORMAT  (/,1X,'INPUT 1: STANDARD DEVIATION OF NOISE')
      READ  (*,*) STDEV
      WRITE (*,2042)
 2042 FORMAT  (/,1X,'INPUT SEED FOR RANDOM NUMBER GENERATOR')
      WRITE (*,2043)
 2043 FORMAT  (1X,'(ANY NEGATIVE INTEGER)')
      READ  (*,*)
      DO 45 N=1,NPTS
   45 X(N)=X(N)+STDEV*GASDEV(ISEED)
      GO TO 100
C
C       TREND OR MEAN LEVEL
   50 WRITE (*,2050)
 2050 FORMAT  (//,1X,'ADDS TREND OR MEAN LEVEL TO EXISTING SERIES')
      WRITE (*,2051)
 2051 FORMAT  (/,1X,'INPUT 1: DX/DT - CHANGE IN X IN UNIT TIME'/
     1        1X,'        ENTER 0 IF ONLY MEAN LEVEL REQUIRED')
      READ  (*,*) DXDT
      WRITE (*,2052)
 2052 FORMAT  (/,1X,'INPUT 2: VALUE AT T = 0'/
     1        1X,'        (MEAN LEVEL IF DX/DT = 0)')
      READ  (*,*) B
      DO 55 N=1,NPTS
   55 X(N)=X(N)+B+DXDT*FLOAT(N-1)
C
  100 GO TO 20
C
C-----------------------------------------------------------------------
C       COMPUTATION COMPLETED - WRITE RESULTS TO FILE IN RWTEMP
C       FORMAT
  200 WRITE (IOCH,2200) NPTS
```

```
 2200 FORMAT (I5,/)
      DO 201 N=1,NPTS
  201  WRITE (IOCH,2201) X(N)
 2201 FORMAT (E12.6)
C
      STOP
      END
C*********************************************************************
      FUNCTION GASDEV(IDUM)
      DATA ISET/0/
      IF (ISET.EQ.0) THEN
1        V1=2.*RAN1(IDUM)-1.
         V2=2.*RAN1(IDUM)-1.
         R=V1**2+V2**2
         IF(R.GE.1.)GO TO 1
         FAC=SQRT(-2.*LOG(R)/R)
         GSET=V1*FAC
         GASDEV=V2*FAC
         ISET=1
      ELSE
        GASDEV=GSET
        ISET=0
      ENDIF
      RETURN
      END
C*********************************************************************
      FUNCTION RAN1(IDUM)
      DIMENSION R(97)
      PARAMETER (M1=259200,IA1=7141,IC1=54773,RM1=3.8580247E-6)
      PARAMETER (M2=134456,IA2=8121,IC2=28411,RM2=7.4373773E-6)
      PARAMETER (M3=243000,IA3=4561,IC3=51349)
       DATA IFF /0/
       IF (IDUM.LT.0.OR.IFF.EQ.0) THEN
        IFF=1
         IX1=MOD(IC1-IDUM,M1)
         IX1=MOD(IA1*IX1+IC1,M1)
         IX2=MOD(IX1,M2)
         IX1=MOD(IA1*IX1+IC1,M1)
         IX3=MOD(IX1,M3)
       DO 11 J=1,97
        IX1=MOD(IA1*IX1+IC1,M1)
        IX2=MOD(IA2*IX2+IC2,M2)
        R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
11     CONTINUE
        IDUM=1
      ENDIF
      IX1=MOD(IA1*IX1+IC1,M1)
      IX2=MOD(IA2*IX2+IC2,M2)
      IX3=MOD(IA3*IX3+IC3,M3)
      J=1+(97*IX3)/M3
      IF(J.GT.97.OR.J.LT.1)PAUSE
      RAN1=R(J)
      R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
      RETURN
      END
```