13.8 Spectral Analysis of Unevenly Sampled Data

Thus far, we have been dealing exclusively with evenly sampled data,

\[ h_n = h(n\Delta) \quad n = \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots \]  (13.8.1)

where \( \Delta \) is the sampling interval, whose reciprocal is the sampling rate. Recall also (§12.1) the significance of the Nyquist critical frequency

\[ f_c \equiv \frac{1}{2\Delta} \]  (13.8.2)

as codified by the sampling theorem: A sampled data set like equation (13.8.1) contains complete information about all spectral components in a signal \( h(t) \) up to the Nyquist frequency, and scrambled or aliased information about any signal components at frequencies larger than the Nyquist frequency. The sampling theorem thus defines both the attractiveness, and the limitation, of any analysis of an evenly spaced data set.

There are situations, however, where evenly spaced data cannot be obtained. A common case is where instrumental drop-outs occur, so that data is obtained only on a (not consecutive integer) subset of equation (13.8.1), the so-called missing data problem. Another case, common in observational sciences like astronomy, is that the observer cannot completely control the time of the observations, but must simply accept a certain dictated set of \( t \)'s.

There are some obvious ways to get from unevenly spaced \( t \)'s to evenly spaced ones, as in equation (13.8.1). Interpolation is one way: lay down a grid of evenly spaced times on your data and interpolate values onto that grid; then use FFT methods. In the missing data problem, you only have to interpolate on missing data points. If a lot of consecutive points are missing, you might as well just set them to zero, or perhaps "clamp" the value at the last measured point. However, the experience of practitioners of such interpolation techniques is not reassuring. Generally speaking, such techniques perform poorly. Long gaps in the data, for example, often produce a spurious bulge of power at low frequencies (wavelengths comparable to gaps). A completely different method of spectral analysis for unevenly sampled data, one that mitigates these difficulties and has some very desirable properties, was developed by Lomb [1], based in part on earlier work by Barning [2] and Vaněček [3], and additionally elaborated by Scargle [4]. The Lomb method (as we will call it) evaluates data, and sines and cosines, only at times \( t_i \) that are actually measured. Suppose that there are \( N \) data points \( h_i \equiv h(t_i) \), \( i = 1, \ldots, N \). Then first find the mean and variance of the data by the usual formulas,

\[ \bar{h} \equiv \frac{1}{N} \sum_{i=1}^{N} h_i \quad \sigma^2 \equiv \frac{1}{N-1} \sum_{i=1}^{N} (h_i - \bar{h})^2 \]  (13.8.3)

Now, the Lomb normalized periodogram (spectral power as a function of angular frequency \( \omega \equiv 2\pi f > 0 \)) is defined by

\[ P_N(\omega) \equiv \frac{1}{2\sigma^2} \left\{ \frac{\sum_j (h_j - \bar{h}) \cos \omega(t_j - \tau)}{\sum_j \cos^2 \omega(t_j - \tau)} \right\}^2 + \left\{ \frac{\sum_j (h_j - \bar{h}) \sin \omega(t_j - \tau)}{\sum_j \sin^2 \omega(t_j - \tau)} \right\}^2 \]  (13.8.4)
The constant $\tau$ is a kind of offset that makes $P_N(\omega)$ completely independent of shifting all the $t_i$’s by any constant. Lomb shows that this particular choice of offset has another, deeper, effect: It makes equation (13.8.4) identical to the equation that one would obtain if one estimated the harmonic content of a data set, at a given frequency $\omega$, by linear least-squares fitting to the model

$$h(t) = A \cos \omega t + B \sin \omega t$$

(13.8.6)

This fact gives some insight into why the method can result superior to FFT methods: It weights the data on a “per point” basis instead of on a “per time interval” basis, when uneven sampling can render the latter seriously in error.

A very common occurrence is that the measured data points $h_i$ are the sum of a periodic signal and independent (white) Gaussian noise. If we are trying to determine the presence or absence of such a periodic signal, we want to be able to give a quantitative answer to the question, “How significant is a peak in the spectrum $P_N(\omega)$?” In this question, the null hypothesis is that the data values are independent Gaussian random values. A very nice property of the Lomb normalized periodogram is that the viability of the null hypothesis can be tested fairly rigorously, as we now discuss.

The word “normalized” refers to the factor $\sigma^2$ in the denominator of equation (13.8.4). Scargle [4] shows that with this normalization, at any particular $\omega$ and in the case of the null hypothesis, $P_N(\omega)$ has an exponential probability distribution function mean. In other words, the probability that $P_N(\omega)$ will be between some positive $z$ and $z + dz$ is $\exp(-z)dz$. It readily follows that, if we scan some $M$ independent frequencies, the probability that none give values larger than $z$ is $(1 - e^{-z})^M$. So

$$P(>z) \equiv 1 - (1 - e^{-z})^M$$

(13.8.7)

is the false-alarm probability of the null hypothesis, that is, the significance level of any peak in $P_N(\omega)$ that we do see. A small value for the false-alarm probability indicates a highly significant periodic signal.

To evaluate this significance, we need to know $M$. After all, the more frequencies we look at, the less significant is some one modest bump in the spectrum. (Look long enough, find anything!) A typical procedure will be to plot $P_N(\omega)$ as a function of many closely spaced frequencies in some large frequency range. How many of these are independent?

Before answering, let us first see how accurately we need to know $M$. The interesting region is where the significance is a small (significant) number, $\ll 1$. There, equation (13.8.7) can be series expanded to give

$$P(>z) \approx Me^{-z}$$

(13.8.8)

We see that the significance scales linearly with $M$. Practical significance levels are numbers like 0.05, 0.01, 0.001, etc. An error of even $\pm 50\%$ in the estimated significance is often tolerable, since quoted significance levels are typically spaced apart by factors of 5 or 10. So our estimate of $M$ need not be very accurate.

Horne and Baliunas [5] give results from extensive Monte Carlo experiments for determining $M$ in various cases. In general $M$ depends on the number of frequencies sampled, the number of data points $N$, and their detailed spacing. It turns out that $M$ is very nearly equal to $N$ when the data points are approximately equally spaced, and when the sampled frequencies “fill” (oversample) the frequency range from 0 to the Nyquist frequency $f_c$ (equation 13.8.2). Further, the value of $M$ is not importantly different for random spacing of the data points than for equal spacing. When a larger frequency range than the Nyquist range is sampled, $M$ increases proportionally. About the only case where $M$ differs significantly from the case of evenly spaced points is when the points are closely clumped, say into groups of 3; then (as one would expect) the number of independent frequencies is reduced by a factor of about 3.
The program `period`, below, calculates an effective value for $M$ based on the above rough-and-ready rules and assumes that there is no important clumping. This will be adequate for most purposes. In any particular case, if it really matters, it is not too difficult to compute a better value of $M$ by simple Monte Carlo: Holding fixed the number of data points and their locations $t_i$, generate synthetic data sets of Gaussian (normal) deviates, find the largest values of $P_N(\omega)$ for each such data set (using the accompanying program), and fit the resulting distribution for $M$ in equation (13.8.7).

Figure 13.8.1 shows the results of applying the method as discussed so far. In the upper figure, the data points are plotted against time. Their number is $N = 100$, and their distribution in $t$ is Poisson random. There is certainly no sinusoidal signal evident to the eye. The lower figure plots $P_N(\omega)$ against frequency $f = \omega/2\pi$. The Nyquist critical frequency that would obtain if the points were evenly spaced is at $f = f_c = 0.5$. Since we have searched up to about twice that frequency, and oversampled the $f$’s to the point where successive values of $P_N(\omega)$ vary smoothly, we take $M = 2N$. The horizontal dashed and dotted lines are (respectively from bottom to top) significance levels 0.5, 0.1, 0.05, 0.01, 0.005, and 0.001. One sees a highly significant peak at a frequency of 0.81. That is in fact the frequency of the sine wave that is present in the data. (You will have to take our word for this!)

Note that two other peaks approach, but do not exceed the 50% significance level; that is about what one might expect by chance. It is also worth commenting on the fact that the significant peak was found (correctly) above the Nyquist frequency and without any significant aliasing down into the Nyquist interval! That would not be possible for evenly spaced data. It is possible here because the randomly spaced data has some points spaced much closer than...
the "average" sampling rate, and these remove ambiguity from any aliasing.

Implementation of the normalized periodogram in code is straightforward, with, however, a few points to be kept in mind. We are dealing with a slow algorithm. Typically, for \( N \) data points, we may wish to examine on the order of \( 2N \) or \( 4N \) frequencies. Each combination of frequency and data point has, in equations (13.8.4) and (13.8.5), not just a few adds or multiplies, but four calls to trigonometric functions; the operations count can easily reach several hundred times \( N^2 \). It is highly desirable — in fact results in a factor 4 speedup — to replace these trigonometric calls by recurrences. That is possible only if the sequence of frequencies examined is a linear sequence. Since that sequence is probably what most users would want anyway, we have built this into the implementation.

At the end of this section we describe a way to evaluate equations (13.8.4) and (13.8.5) — approximately, but to any desired degree of approximation — by a fast method [6] whose operation count goes only as \( N \log N \). This faster method should be used for long data sets.

The lowest independent frequency \( f \) to be examined is the inverse of the span of the input data, \( \max_i(t_i) - \min_i(t_i) \equiv T \). This is the frequency such that the data can include one complete cycle. In subtracting off the data's mean, equation (13.8.4) already assumed that you are not interested in the data's zero-frequency piece — which is just that mean value. In an FFT method, higher independent frequencies would be integer multiples of \( 1/T \). Because we are interested in the statistical significance of any peak that may occur, however, we had better (over-) sample more finely than at interval \( 1/T \), so that sample points lie close to the top of any peak. Thus, the accompanying program includes an oversampling parameter, called \( \text{ofac} \); a value \( \text{ofac} \lesssim 4 \) might be typical in use. We also want to specify how high in frequency to go, say \( f_{hi} \). One group to choose \( f_{hi} \) is to compare it with the Nyquist frequency \( f_c \) which would obtain if the \( N \) data points were evenly spaced over the same span \( T \), that is \( f_c = N/(2T) \). The accompanying program includes an input parameter \( \text{hifac} \), defined as \( f_{hi}/f_c \). The number of different frequencies \( N_P \) returned by the program is then given by

\[
N_P = \frac{\text{ofac} \times \text{hifac}}{2} \quad (13.8.9)
\]

(You have to remember to dimension the output arrays to at least this size.)

The code does the trigonometric recurrences in double precision and embodies a few tricks with trigonometric identities, to decrease roundoff errors. If you are an aficionado of such things you can puzzle it out. A final detail is that equation (13.8.7) will fail because of roundoff error if \( z \) is too large; but equation (13.8.8) is fine in this regime.

```
SUBROUTINE period(x,y,n,ofac,hifac,px,py,np,nout,jmax,prob)
INTEGER jmax,n,nout,np,NMAX
REAL hifac,ofac,prob,px(np),py(np),x(n),y(n)
PARAMETER (NMAX=2000) Maximum expected value of n.

C USES avevar
Given n data points with abscissas x(1:n) (which need not be equally spaced) and ordinates y(1:n), and given a desired oversampling factor ofac (a typical value being 4 or larger), this routine fills array px with an increasing sequence of frequencies (not angular frequencies) up to hifac times the "average" Nyquist frequency, and fills array py with the values of the Lomb normalized periodogram at those frequencies. The arrays x and y are not altered. np, the dimension of px and py, must be large enough to contain the output, or an error (pause) results. The routine also returns jmax such that \( py(jmax) \) is the maximum element in px, and prob, an estimate of the significance of that maximum against the hypothesis of random noise. A small value of prob indicates that a significant periodic signal is present.

INTEGER i,j
REAL ave,c,cc,cwtau,effm,exp,y,now,pymax,s,sx,sy,sumc,sumcy,
     sums,sumsh,sumy,cwtau,ivar,vsnow,xave,xdif,xmax,xmin,yy
DOUBLE PRECISION arg,wt,wi(NMAX),wpi(NMAX),
     wpr(NMAX),wtemp,wi(NMAX),TWOPID
PARAMETER (TWOPID=6.2831853071795865D0)

nout=0.5*ofac*hifac*n
if(nout.gt.np) pause 'output arrays too short in period'
call avevar(y,n,ave,vars) Get mean and variance of the input data.

xmin=x(1) xmax=x(1)
Go through data to get the range of abscissas.
```
do 11 j=1,n
  if(x(j).gt.xmax)xmax=x(j)
  if(x(j).lt.xmin)xmin=x(j)
enddo
xdif=xmax-xmin
xave=0.5*(xmax+xmin)
pymax=0.
pnow=1./(xdif*ofac)

11 do j=1,n
  arg=TWOPID*((x(j)-xave)*pnow)
  wpr(j)=-2.d0*sin(0.5d0*arg)**2
  wpi(j)=sin(arg)
  vr(j)=cos(arg)
  wi(j)=wpi(j)
enddo

do 12 i=1,nout
  px(i)=pnow
  snumsh=0.
  snumc=0.
  do 13 j=1,n
    c=wr(j)
    s=wi(j)
    snumsh=snumsh+s*c
    snumc=snumc+(c-s)*(c+s)
  enddo
  wtau=0.5*atan2(2.*snumsh,snumc)
  ctau=cos(wtau)
  stau=sin(wtau)
  sums=0.
  sumc=0.
  sumsy=0.
  sumcy=0.
  do 14 j=1,n
    s=wi(j)
    c=wr(j)
    ss=s*ctau-c*stau
    cc=c*ctau+s*stau
    sums=sums+ss**2
    sumc=sumc+cc**2
    yy=y(j)-ave
    sumsy=sumsy+yy*ss
    sumcy=sumcy+yy*cc
  enddo
  wtemp=wr(j)
  wr(j)=(wr(j)*wpr(j)-wi(j)*wpi(j))+wr(j)
  wi(j)=(wi(j)*wpr(j)+wtemp*wpi(j))+wi(j)
  wr(j)=wpr(j)*wr(j)
  wi(j)=wpi(j)*wpr(j)
  if(py(i).ge.pymax) then
    pymax=py(i)
    jmax=i
  endif
  pnow=pnow+1./(ofac*xdif)
enddo

12 do i=1,nout
  py(i)=0.5*(sumcy**2/sumc+sumsy**2/sums)/var
  if (py(i).ge.pymax) then
    pymax=py(i)
    jmax=i
  endif
enddo

pymax=exp(-pymax)

13.8 Spectral Analysis of Unevenly Sampled Data

Evaluate statistical significance of the maximum.
effm=2.*nout/ofac
prob=effm*expy
if(prob.gt.0.01)prob=1.-expy**effm
return
END
Fast Computation of the Lomb Periodogram

We here show how equations (13.8.4) and (13.8.5) can be calculated — approximately, but to any desired precision — with an operation count only of order \( N_P \log N_P \). The method uses the FFT, but it is in no sense an FFT periodogram of the data. It is an actual evaluation of equations (13.8.4) and (13.8.5), the Lomb normalized periodogram, with exactly that method’s strengths and weaknesses. This fast algorithm, due to Press and Rybicki[8], makes feasible the application of the Lomb method to data sets at least as large as \( 10^6 \) points; it is already faster than straightforward evaluation of equations (13.8.4) and (13.8.5) for data sets as small as 60 or 100 points.

Notice that the trigonometric sums that occur in equations (13.8.5) and (13.8.4) can be reduced to four simpler sums. If we define

\[
S_h \equiv \sum_{j=1}^{N} (h_j - \bar{h}) \sin(\omega t_j), \quad C_h \equiv \sum_{j=1}^{N} (h_j - \bar{h}) \cos(\omega t_j) \tag{13.8.10}
\]

and

\[
S_2 \equiv \sum_{j=1}^{N} \sin(2\omega t_j), \quad C_2 \equiv \sum_{j=1}^{N} \cos(2\omega t_j) \tag{13.8.11}
\]

then

\[
\begin{align*}
\sum_{j=1}^{N} (h_j - \bar{h}) \cos(\omega t_j - \tau) & = C_h \cos \omega \tau + S_h \sin \omega \tau \\
\sum_{j=1}^{N} (h_j - \bar{h}) \sin(\omega t_j - \tau) & = S_h \cos \omega \tau - C_h \sin \omega \tau \\
\sum_{j=1}^{N} \cos^2(\omega t_j - \tau) & = \frac{N}{2} + \frac{1}{2} C_2 \cos(2\omega \tau) + \frac{1}{2} S_2 \sin(2\omega \tau) \\
\sum_{j=1}^{N} \sin^2(\omega t_j - \tau) & = \frac{N}{2} - \frac{1}{2} C_2 \cos(2\omega \tau) - \frac{1}{2} S_2 \sin(2\omega \tau)
\end{align*} \tag{13.8.12}
\]

Now notice that if the \( t_j \)s were evenly spaced, then the four quantities \( S_h, C_h, S_2, \) and \( C_2 \) could be evaluated by two complex FFTs, and the results could then be substituted back through equation (13.8.12) to evaluate equations (13.8.5) and (13.8.4). The problem is therefore only to evaluate equations (13.8.10) and (13.8.11) for unevenly spaced data.

Interpolation, or rather reverse interpolation — we will here call it extirpolation — provides the key. Interpolation, as classically understood, uses several function values on a regular mesh to construct an accurate approximation at an arbitrary point. Extirpolation, just the opposite, replaces a function value at an arbitrary point by several function values on a regular mesh, doing this in such a way that sums over the mesh are an accurate approximation to sums over the original arbitrary point.

It is not hard to see that the weight functions for extirpolation are identical to those for interpolation. Suppose that the function \( h(t) \) to be extirpolated is known only at the discrete (unevenly spaced) points \( h(t_i) \equiv h_i \), and that the function \( g(t) \) (which will be, e.g., \( \cos \omega t \)) can be evaluated anywhere. Let \( t_k \) be a sequence of evenly spaced points on a regular mesh. Then Lagrange interpolation (§3.1) gives an approximation of the form

\[
g(t) \approx \sum_k w_k(t) g(t_k) \tag{13.8.13}
\]

where \( w_k(t) \) are interpolation weights. Now let us evaluate a sum of interest by the following scheme:

\[
\sum_{j=1}^{N} h_j g(t_j) \approx \sum_{j=1}^{N} h_j \left[ \sum_k w_k(t_j) g(t_k) \right] = \sum_k \left[ \sum_{j=1}^{N} h_j w_k(t_j) \right] g(t_k) \equiv \sum_k \tilde{h}_k g(t_k) \tag{13.8.14}
\]
Here $\hat{h}_k \equiv \sum_j h_j w_j(t_j)$. Notice that equation (13.8.14) replaces the original sum by one on the regular mesh. Notice also that the accuracy of equation (13.8.13) depends only on the fineness of the mesh with respect to the function $g$ and has nothing to do with the spacing of the points $t_j$ or the function $h$; therefore the accuracy of equation (13.8.14) also has this property.

The general outline of the fast evaluation method is therefore this: (i) Choose a mesh size large enough to accommodate some desired oversampling factor, and large enough to have several extrapolation points per half-wavelength of the highest frequency of interest. (ii) Extrapolate the values $h_j$ onto the mesh and take the FFT; this gives $S_h$ and $C_h$ in equation (13.8.10). (iii) Extrapolate the constant values $1$ onto another mesh, and take its FFT; this, with some manipulation, gives $S_2$ and $C_2$ in equation (13.8.11). (iv) Evaluate equations (13.8.12), (13.8.5), and (13.8.4), in that order.

There are several other tricks involved in implementing this algorithm efficiently. You can figure most out from the code, but we will mention the following points: (A) A nice way to get transform values at frequencies $2\omega$ instead of $\omega$ is to stretch the time-domain data by a factor 2, and then wrap it to double-cover the original length. (This trick goes back to Tukey.) In the program, this appears as a modulo function. (B) Trigonometric identities are used to get from the left-hand side of equation (13.8.5) to the various needed trigonometric functions of $\omega t$. (C) The subroutine spread does extrapolation onto the $M$ most nearly centered mesh points around an arbitrary point; its turgid code evaluates coefficients of the Lagrange interpolating polynomials, in an efficient manner.

SUBROUTINE fasper(x,y,n,ofac,hifac,wk1,wk2,nwk,nout,jmax,prob)
INTEGER jmax,n,nout,nwk,MACC
REAL hifac,ofac,prob,wk1(nwk),wk2(nwk),x(n),y(n)
PARAMETER (MACC=4)
C USES avevar,realft,spread
C
GIVEN $n$ data points with abscissas $x$ (which need not be equally spaced) and ordinates $y$, and given a desired oversampling factor ofac (a typical value being 4 or larger), this routine fills array wk1 with a sequence of $nout$ increasing frequencies (not angular frequencies) up to hifac times the “average” Nyquist frequency, and fills array wk2 with the values of the Lomb normalized periodogram at those frequencies. The arrays $x$ and $y$ are not altered. $nwk$, the dimension of wk1 and wk2, must be large enough for intermediate work space, or an error (pause) results. The routine also returns jmax such that wk2(jmax) is the maximum element in wk2, and prob, an estimate of the significance of that maximum against the hypothesis of random noise. A small value of prob indicates that a significant periodic signal is present.

INTEGER j,k,ndim,nfreq,nfreqt
REAL ave,ck,ckk,cterm,cwt,den,df,effm,expy,fac,fndim,hc2wt,
* hs2wt,hypo,pmax,sterm,swt,var,xdif,xmax,xmin
EXTERNAL spread

nout=0.5*ofac*hifac*n
nfreqt=ofac*hifac*n*MACC
C Size the FFT as next power of 2 above nfreqt.
nfreq=64
I
if (nfreq.lt.nfreqt) then
nfreq=nfreq*2
endif
endif
ndim=2*nfreq
if(ndim.gt.nwk) pause 'workspaces too small in fasper'
call avevar(y,n,ave,var) C Compute the mean, variance, and range of the data.
xmin=x(1)
xmax=xmin
do : j=2,n
if (x(j).lt.xmin) xmin=x(j)
if (x(j).gt.xmax) xmax=x(j)
enddo
xdif=xmax-xmin
do : j=1,ndim
wk1(j)=0.
wk2(j)=0.
enddo

Extirpolate the data into the workspaces.

```fortran
fac = ndim/(xdiff*ofac)
fndim = ndim

do j = 1, n
    ck = 1. + mod((x(j) - xmin)*fac, fndim)
    ckk = 1. + mod(2.*(ck-1.), fndim)
    call spread(y(j) - ave, wk1, ndim, ck, MACC)
    call spread(wk2, ndim, ckk, MACC)
endo

call realft(wk1, ndim, 1)  Take the Fast Fourier Transforms.
call realft(wk2, ndim, 1)
df = 1. / (xdiff*ofac)
k = 3

pmmax = -1.
do j = 1, nout
    Compute the Lomb value for each frequency.
    hypo = sqrt(wk2(k)**2 + wk2(k+1)**2)
    hc2wt = 0.5*wk2(k)/hypo
    hs2wt = 0.5*wk2(k+1)/hypo
    cwt = sqrt(0.5 + hc2wt)
    swt = sign(sqrt(0.5 - hc2wt), hs2wt)
    den = 0.5*n + hc2wt*wk2(k) + hs2wt*wk2(k+1)
    cterm = (cwt*wk1(k) + swt*wk1(k+1))**2/den
    sterm = (cwt*wk1(k+1) - swt*wk1(k))**2/(n-den)
    wk1(j) = (cterm + sterm)/(2.*var)
    wk2(j) = j*df
    if (wk2(j) > pmmax) then
        pmmax = wk2(j)
        jmax = j
    endif
endo
```

Estimate significance of largest peak value.

```fortran
expy = exp(-pmmax)
effm = 2.*nout/ofac
prob = effm * expy
if(prob < 0.01) then
    prob = 1. - (1. - expy)**effm
endif
return
END
```

SUBROUTINE spread(y, yy, n, x, m)
INTEGER m, n
REAL x, y, yy(n)
Given an array yy of length n, extirpolate (spread) a value y into m actual array elements
that best approximate the “fictional” (i.e., possibly noninteger) array element number x.
The weights used are coefficients of the Lagrange interpolating polynomial.

INTEGER ihi, ilo, ix, j, nden, nfac(10)
REAL fac
SAVE nfac
DATA nfac / 1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880/
if(m < 10) pause ‘factorial table too small in spread’
ix = x
if(x < float(ix)) then
    iy(ix) = iy(ix) + y
else
    ilo = max(int(x - 0.5*m + 1.0), 1), n-m+1
    ihi = ilo + m - 1
    nden = nfac(m)
    fac = x - ilo
    do j = ilo+1, ihi
        fac = fac * (x-j)
    enddo
    iy(ihi) = iy(ihi) + y*fac/(nden*(x-ihi))
    do j = ihi+1, ilo - 1
        nden = nden / (j+1-il0) * (j-ihi)
```
13.9 Computing Fourier Integrals Using the FFT

Not uncommonly, one wants to calculate accurate numerical values for integrals of the form

\[ I = \int_a^b e^{i \omega t} h(t) dt, \]  

(13.9.1)

or the equivalent real and imaginary parts

\[ I_c = \int_a^b \cos(\omega t) h(t) dt \quad I_s = \int_a^b \sin(\omega t) h(t) dt, \]  

(13.9.2)

and one wants to evaluate this integral for many different values of \( \omega \). In cases of interest, \( h(t) \) is often a smooth function, but it is not necessarily periodic in \([a, b]\), nor does it necessarily go to zero at \( a \) or \( b \). While it seems intuitively obvious that the force majeure of the FFT ought to be applicable to this problem, doing so turns out to be a surprisingly subtle matter, as we will now see.

Let us first approach the problem naively, to see where the difficulty lies. Divide the interval \([a, b]\) into \( M \) subintervals, where \( M \) is a large integer, and define

\[ \Delta \equiv \frac{b - a}{M}, \quad t_j \equiv a + j \Delta, \quad h_j \equiv h(t_j), \quad j = 0, \ldots, M \]  

(13.9.3)

Notice that \( h_0 = h(a) \) and \( h_M = h(b) \), and that there are \( M + 1 \) values \( h_j \). We can approximate the integral \( I \) by a sum,

\[ I \approx \Delta \sum_{j=0}^{M-1} h_j \exp(i \omega t_j) \]  

(13.9.4)

which is at any rate first-order accurate. (If we centered the \( h_j \)'s and the \( t_j \)'s in the intervals, we could be accurate to second order.) Now for certain values of \( \omega \) and \( M \), the sum in equation (13.9.4) can be made into a discrete Fourier transform, or DFT, and evaluated by the fast Fourier transform (FFT) algorithm. In particular, we can choose \( M \) to be an integer power of 2, and define a set of special \( \omega \)'s by

\[ \omega_m \Delta \equiv \frac{2\pi m}{M} \]  

(13.9.5)