1 Theory

1.1 Fourier Transform

The definition of the Fourier transform pair used here is

$$H(\omega) = \mathcal{F}[h(t)] = \int_{-\infty}^{\infty} h(t)e^{-i\omega t}dt \Leftrightarrow h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(\omega)e^{+i\omega t}d\omega, \tag{1}$$

or upon defining $\omega = 2\pi f$,

$$H(f) = \int_{-\infty}^{\infty} h(t)e^{-i2\pi ft}dt \Leftrightarrow h(t) = \int_{-\infty}^{\infty} H(f)e^{+i2\pi ft}df. \tag{2}$$

Some properties of the Fourier transform are

Table 1: Fourier transform properties and pairs

$h(t)$	$\Leftrightarrow$	$H(\omega)$
$H(t)$	$\Leftrightarrow$	$2\pi h(-\omega)$
$h(t-a)$	$\Leftrightarrow$	$H(\omega)e^{-i\omega a}$
$h(t) * g(t) = \int_{-\infty}^{\infty} h(\tau)g(t-\tau)d\tau$	$\Leftrightarrow$	$H(\omega)G(\omega)$
$h(t)g(t)$	$\Leftrightarrow$	$\frac{1}{2\pi}H(\omega) * G(\omega)$
$\delta(t)$	$\Leftrightarrow$	1
1	$\Leftrightarrow$	$2\pi\delta(\omega)$
$\mathrm{sgn}(t) = \begin{cases} -1 & t<0 \\ 0 & t=0 \\ 1 & t>0 \end{cases}$	$\Leftrightarrow$	$\frac{2}{i\omega}$
$i\frac{1}{\pi t}$	$\Leftrightarrow$	$\mathrm{sgn}(\omega)$
$U(t) = \begin{cases} 0 & t<0 \\ \frac{1}{2} & t=0 \\ 1 & t>0 \end{cases}$	$\Leftrightarrow$	$\pi\delta(\omega) + \frac{1}{i\omega}$
$\cos\omega_0 t$	$\Leftrightarrow$	$\pi\delta(\omega-\omega_0) + \pi\delta(\omega+\omega_0)$
$i\sin\omega_0 t$	$\Leftrightarrow$	$\pi\delta(\omega-\omega_0) - \pi\delta(\omega+\omega_0)$

The * operator represents convolution.

Now define the complex time series and its Fourier transform as

$$g(t) = h(t) + iq(t) \Leftrightarrow G(\omega) = R(\omega) + iI(\omega) \tag{3}$$

where $h(t)$, $q(t)$, $R(\omega)$ and $I(\omega)$ are real functions. Some special cases are as follow:

- If $g(t)$ is real, e.g., $g(t) = h(t)$, then $R(-\omega) = R(\omega)$ and $I(-\omega) = -I(\omega)$ or equivalently $h(t) \Leftrightarrow R_e(\omega) + iI_o(\omega)$, where the subscripts indicate even and odd.
- If $g(t)$ is imaginary, e.g., $g(t) = iq(t)$, then $R(-\omega) = -R(\omega)$ and $I(-\omega) = I(\omega)$, or $G(\omega) = R_o(\omega) + iI_e(\omega)$.

1.2 Hilbert Transform

The Hilbert transform of a function $x(t)$ is defined as

$$\mathcal{H}[x(t)] = \frac{1}{\pi} p.v \int_{-\infty}^{\infty} \frac{x(\tau)}{t-\tau} d\tau, \tag{4}$$

where the $p.v$ indicates a principal-value integral. In interesting this can be easily evaluated using Fourier transforms. Given $x(t) \Leftrightarrow X(\omega)$, then $\mathcal{H}[x(t)] \Leftrightarrow \text{sgn}(\omega)X(\omega)$. Given this definition of the Hilbert transform and the Fourier transform properties in Table 1, one can show that $\mathcal{H}[\cos\omega_0 t] = \sin\omega_0 t$ for $\omega_0 > 0$. Thus the Hilbert transform of a sinusoid returns another sinusoid that is 90° out of phase.

If we want the $q(t)$ above to be 90 degrees out of phase with the $h(t)$, perform the following steps:

1. Form $\mathcal{F}[h(t)] = H(\omega) = R_e(\omega) + iI_o(\omega)$.
2. Multiply by twice the unit step function, e.g., form $2U(\omega)H(\omega)$
3. Evaluate the inverse Fourier transform to give , $h(t) + iq(t) = \mathcal{F}^{-1}[2U(\omega)H(\omega)]$.
4. The desired $q(t)$ is the imaginary part of the result. The $h(t)$ returned is the input $h(t)$.[1]

The value of this approach is that only one complex array is used. More importantly the envelope of $h(t)$ is $e(t) = \sqrt{h^2(t) + q^2(t)}$.

1.3 Discrete Fourier Transform

A numerical implementation of the Fourier transform pair is the Discrete Fourier Transform (DFT). The fast algorithm is known as the Fast Fourier Transform (FFT). The DFT with dimensions is defined as

$$\begin{aligned} H(n\Delta f) &= \sum_{k=0}^{N-1} h(k\Delta t) e^{-i2\pi nk\Delta f\Delta t}\Delta t \Leftrightarrow h(k\Delta t) = \sum_{n=0}^{N-1} H(n\Delta f) e^{+i2\pi nk\Delta f\Delta t}\Delta f \\ H(n\Delta f) &= \sum_{k=0}^{N-1} h(k\Delta t) e^{-i2\pi nk/N}\Delta t \Leftrightarrow h(k\Delta t) = \sum_{n=0}^{N-1} H(n\Delta f) e^{+i2\pi nk/N}\Delta f \end{aligned} \tag{5}$$

where N is the number of evenly spaced samples, $t = k\Delta t$, $f = n\Delta f$ and $\Delta f = \frac{1}{N\Delta t}$. Note that the textbook definition of the DFT transforms a one-dimensional array into another one-dimensional array of numbers without any sense of approximating an integration. The definition given in 5 modifies the usual definition to add the dimensions, and thus approximates the Fourier integral. An important aspect of the DFT is that both $h(k)$ and $H(n)$ are periodic, e.g., $h(k) = h(k + mN)$ and $H(n) = H(n + mN)$ for $m = \ldots, -2, -1, 0, 1, 2, \ldots$. Thus any use of the DFT to approximate the Fourier transform and its inverse must be careful about periodicity, or in other words, wrap-around or aliasing.

Normally one starts with an $h(k) = h(k\Delta t)$ and computes an $H(n) = H(n\Delta f)$. Thus the negative frequency part of the spectrum is given by $H(-n) = H(N - n)$. In implementation, if one starts with the $h(t - k\Delta t)$ and computes the $H(n\Delta f)$ everything is straightforward. When implementing theory, it is efficient to just compute the response for positive frequencies and then fill in the negative frequencies to ensure that the inverse DFT gives a real time series. Pseudocode in FORTRAN [2] to yield the $H(n)$ and C corresponding to a real time series is

[1] Note: there may be a slight difference in values and some distance in length when evaluated numerically.

[2] FORTRAN is used here because complex numbers are easily used. However, default array processing will be from 1 to N instead of 0 to N-1

FORTRAN

```
COMPLEX FUNC, DATA(NPTS)
N21 = N/2 + 1
do i = 1, N21
   freq = (i-1)*df
   DATA(i) = FUNC(freq)
   if(i.eq.N21)then
       DATA(i) = CMPLX(real(DATA(i),0.0)
   endif
   DATA(N*2-i) = CONJG(DATA(i))
enddo
```

C

```
/* n is a power of 2 */
n2 = n/2 ; /* n2 is Nyquist */
for(i=0;i <= n2 ; i+=2){
   freq = i*df ;
   evalfunc(freq,&real,&imag);
   data(i)   = real ;
   data(i+1) = imag ;
   /* apply symmetry to get
              negative frequency */
   if(i> 0 & i < n2){
     data(n-i   ) = real;
     data(n-i+1 ) = imag;
   }
}
data(n2+1) = 0.0 ;
cfour(data,n,+1,&dt,&dt);
```

The index $i = N21$ in FORTRAN corresponds to the Nyquist frequency, $f_N = \frac{1}{2\Delta t}$ at which the imaginary part of the DFT must be zero.

The code to implement the Hilbert transform is

FORTRAN

```
      COMPLEX FUNC, DATA(NPTS)
      REAL h(NPTS), q(NPTS)
c  fill complex array with real function
c  assume N is a power of 2
      do i=1,N
          DATA(i) = cmplx(h(i),0.0)
      enddo
c  forward DFT
      call zfour(DATA,N,-1,dt,df)
      N21 = N/2 + 1
      do i = 1, N
         if(i.eq.1)then
               DATA(i) = 1.0*DATA(i)
         if(i.gt.1.and.i.le.N21)then
               DATA(i) = 2.0*DATA(i)
         else
c   zero at negative frequencies
               DATA(i) = cmplx(0.0,0.0)
         endif
      enddo
c  inverse DFT
      call zfour(DATA,N,+1,dt,df)
c  unpack to recover the h(t) and form q(t)
      do i=1,n
         h(i) =  real(DATA(i))
         q(i) = aimag(DATA(i))
      enddo
```

C

```
/* n is a power of 2 */
n2 = n/2 ; /* n2 is Nyquist */
for(i=0, j=0; i < n ; i++){
    if(i<n){
        data[j++] = h[i] ;
        data[j++] = 0.0
    } else {
        data[j++] = 0.0
        data[j++] = 0.0
    }
}
* get the DFT */
df = 0.0;
cfour(data, n2, -1, &dt, &df);
/* to get the Hilbert transform zero the
 * negative frequencies and inverse transform */
for(j=0; j < 2*n ; j++){
    if(j == 0)   /* freq = 0 */
       data[j] *= 1.0;
    else if(j == 1) /* freq=0 */
       data[j] *= 0.0;
    else if(j == n2 /* Nyquist frequency */
       data[j] *= 1.0;
    else if(j == n2 +1)
       data[j] *= 0.0;
    else if(j < n && j > 0)
       data[j] *= 2.0;
    else /* negative freq */
       data[j] = 0.0;
}
/* get the inverse FFT */
cfour(data, n2, +1, &dt, &df);
/* fill the arrays q[i] is the Hilbert */
for(i=0, j=0; i < n; i++){
     h[i] = data[j++];
     q[i] = data[j++];
}
```

The index $i = N21$ in the FORTRAN code corresponds to the Nyquist frequency, $f_N = \frac{1}{2\Delta t}$ at which the imaginary part of the DFT must be zero.

1.4 Surface wave

In the frequency domain the surface wave signal at a distance r from the source in a cylindrical coordinate system can be represented as

$$H(\omega) = \frac{1}{\sqrt{r}} A(\omega) e^{-ik(\omega)r} \tag{6}$$

The surface-wave wavenumber $k(\omega) = \omega/c$, where $c()$ is the phase velocity.

Implementing the inverse transform of (6) using (1) will give a time series that starts at $t = 0$. This is often inconvenient in making synthetics, since there may be many seconds before the first arrival appears. To avoid this, one may modify (6) so that the first sample of the time series is at $t = t_0$, by using the transform pair

$$h(t - t_0) \Leftrightarrow e^{i\omega t_0} H(\omega) = \frac{1}{\sqrt{r}} A(\omega) e^{i(\omega t_0 - k(\omega r))} \tag{7}$$

Now assume that the observed seismogram at distance r starts at t_0 seconds after the origin time and that N evenly spaced samples are used. Call this time series $h(k)$ for $k = 0, \ldots, N-1$. The $h(0)$ is the observation at absolute/travel-time t_0. Applying the DFT gives the $H(n)$ for $N = 0, \ldots, N-1$. To correct the transform so that the inverse Fourier transform gives a time series such that the first sample corresponds to a reference time of $t = 0$, one must form

$$e^{-\omega t_0} H(\omega), \tag{8}$$

where the $\omega = 2\pi f = 2\pi n \Delta f$. Also note that the complex spectrum at negative frequencies will be the complex conjugate of the spectrum at positive frequencies.

If the surface wave signal consists of just one mode, then one can correct for dispersion and geometrical spreading by taking the inverse Fourier transform of

$$\sqrt{r} e^{i(-\omega t_0 + kr)} H(\omega), \tag{9}$$

where the $H(\omega)$ is the Fourier transform of the time series starting at t_0.