Fourier Analysis of Time Series

by Dr. R. L. Herman, UNC Wilmington
Friday, September 20, 2002
This is a work in progress.

Introduction

Often one is interested in determining the frequency content of signals. Signals are typically represented as time dependent functions. Real signals are continuous, or analog signals. However, through sampling the signal by gathering data, the signal does not contain high frequencies and is finite in length. The data is then discrete and the corresponding frequencies are discrete and bounded. Thus, in the process of gathering data, one seriously affects the frequency content of the signal. This is true for simple a superposition of signals with fixed frequencies. The situation becomes more complicated if the data has an overall non-constant trend or even exists in the presence of noise.

From Transforms to Series

To be completed.

Fourier Series

As described in the last section (hopefully), we have seen that by restricting our data to a time interval [0, T] for period T, and extending the data to $(-\infty, \infty)$, one generates a periodic function of infinite duration at the cost of losing data outside the fundamental range. This is not unphysical, as the data is typically taken over a finite period of time. Thus, any physical results in the analysis can be obtained be restricting the outcome to the given period.

In typical problems one seeks a representation of the signal, valid for $t \in [0, T]$, as

$$f(t) = \frac{1}{2} A_0 + \sum_{p=1}^{\infty} [A_p \cos(w_p t) + B_p \sin(w_p t)],$$

where the angular frequency is given by

$$w_p = 2pf_p = \frac{2p}{T} p.$$

Note that *f(t)* is periodic with period $P=T$: $f(t+P) = f(t)$. Most signals have an infinite period, so we already have made a first approximation. Thus, this has restricted the physical data to $0 < t < T$. [That is, by limiting t, we are forced to using discrete frequencies, or the above sum as opposed to an integral in the Fourier Transform in the last section.]

One can extract the Fourier Coefficients (A_p, B_p) using the orthogonality of the trigonometric basis. Namely, such orthogonality for a set $\{f_n(x), n = 1\ldots\infty\}$ over the interval $[a,b]$ with weight $r(x)$ is given by the condition

$$\int_a^b f_n(x) f_m(x)\, r(x)\, dx = 0,\ m \neq n.$$

For the trigonometric functions, this is given by the relations

$$\int_0^T \cos(w_m t)\cos(w_n t)\ dt = \begin{cases} 0, & m \neq n \\ \dfrac{T}{2}, & m = n. \end{cases}$$

$$\int_0^T \sin(w_m t)\sin(w_n t)\ dt = \begin{cases} 0, & m \neq n \\ \dfrac{T}{2}, & m = n. \end{cases}$$

$$\int_0^T \cos(w_m t)\sin(w_n t)\, dt = 0.$$

These relations are proved in Appendix A. The coefficients are found to in Appendix B as

$$A_p = \frac{2}{T}\int_0^T y(t)\cos(w_p t)\, dt,\ p = 0,1,2,\ldots$$

$$B_p = \frac{2}{T}\int_0^T y(t)\sin(w_p t)\, dt,\ p = 1,2,\ldots$$

Discrete Series

In the previous analysis we had restricted time to the interval [0,T], leading to a Fourier series with discrete frequencies and a periodic function of time. In reality, taking data can only be done at certain frequencies, thus eliminating high frequencies. Such a restriction on the frequency will lead to a discretization of the data in time. Another way to view this is that when recording data we sample at a finite number of time steps, limiting our ability to collect data with large oscillations. Thus, we not only have discrete frequencies but we also have discrete times.

This can be captured in the following representation:

$$f(t_n) = \frac{1}{2}A_0 + \sum_{p=1}^{M}[A_p \cos(w_p t) + B_p \sin(w_p t)],$$

where the time, time step and trigonometric arguments are given by

$$t_n = n\Delta t,\ \ \Delta t = \frac{T}{N},\ \text{and}\ w_p t = \frac{2ppn}{T}.$$

We need to determine M and the unknown coefficients. As for the Fourier series, we rely on an orthogonality principle, but this time replacing the integral by a sum. Again we will determine the unknowns in terms of the given samples of the function *f(t)*. The orthogonality is provided in Appendix C. If we take N samples, we can the determine N unknown coefficients $A_0, A_1, \ldots, A_{N/2}$ and $B_1, \ldots, B_{N/2-1}$. Thus, we can fix $M = \frac{N}{2}$. Often the coefficients B_0 and $B_{N/2}$ are included for symmetry. Note that the corresponding sine function factors evaluate to zero, leaving these two coefficients arbitrary. Thus, we can take them to be zero.

The full set of coefficients are found to be (in Appendix D)

$$A_p = \frac{2}{N}\sum_{n=1}^{N} y(t_n)\cos(\tfrac{2p\,pn}{N}),\ \ p = 1,\ldots N/2 - 1$$

$$B_p = \frac{2}{N}\sum_{n=1}^{N} y(t_n)\sin(\tfrac{2ppn}{N}),\ p = 1,2,\ldots N/2 - 1$$

$$A_0 = \frac{1}{N}\sum_{n=1}^{N} y(t_n),$$

$$A_{N/2} = \frac{1}{N}\sum_{n=1}^{N} y(t_n)\cos(np),$$

$$B_0 = B_{N/2} = 0$$

Matlab Implementation

In this section we provide implementations of the discrete trigonometric transform in Matlab. The first implementation is a straightforward one which can be done in most programming languages. The second implementation makes use of matrix computations that can be performed in Matlab. Sums can be done with matrix multiplication, as describes in Appendix I. This eliminates the loops in the program.

Direct Implementation without special use of matrix algebra

```
%
% DFT in a direct implementation
%
% Enter Data in y
y=[7.6 7.4 8.2 9.2 10.2 11.5 12.4 13.4 13.7 11.8 10.1 ...
   9.0 8.9 9.5 10.6 11.4 12.9 12.7 13.9 14.2 13.5 11.4 10.9 8.1];

% Get length of data vector or number of samples
N=length(y);

% Compute Fourier Coefficients
for p=1:N/2+1
   A(p)=0;
   B(p)=0;
```

```
  for n=1:N
    A(p)=A(p)+2/N*y(n)*cos(2*pi*(p-1)*n/N)';
    B(p)=B(p)+2/N*y(n)*sin(2*pi*(p-1)*n/N)';
  end
end
A(N/2+1)=A(N/2+1)/2;

% Reconstruct Signal - pmax is number of frequencies used in increasing order
pmax=13;
for n=1:N
  ynew(n)=A(1)/2;
  for p=2:pmax
    ynew(n)=ynew(n)+A(p)*cos(2*pi*(p-1)*n/N)+B(p)*sin(2*pi*(p-1)*n/N);
  end
end

% Plot Data
plot(y,'o')

% Plot reconstruction over data
hold on
plot(ynew,'r')
hold off
```

Compact Implementation

This implementation uses matrix products and is described in Appendix H.

```
%
% DFT in a compact implementation
%
% Enter Data in y
y=[7.6 7.4 8.2 9.2 10.2 11.5 12.4 13.4 13.7 11.8 10.1 ...
  9.0 8.9 9.5 10.6 11.4 12.9 12.7 13.9 14.2 13.5 11.4 10.9 8.1];
N=length(y);

% Compute the matrices of trigonometric functions
p=1:N/2+1;
n=1:N;
C=cos(2*pi*n'*(p-1)/N);
S=sin(2*pi*n'*(p-1)/N);

% Compute Fourier Coefficients
A=2/N*y*C;
B=2/N*y*S
A(N/2+1)=A(N/2+1)/2;

% Reconstruct Signal - pmax is number of frequencies used in increasing order
pmax=13;
ynew=A(1)/2+C(:,2:pmax)*A(2:pmax)'+S(:,2:pmax)*B(2:pmax)';

% Plot Data
plot(y,'o')

% Plot reconstruction over data
```

```
hold on
plot(ynew,'r')
hold off
```

Appendices

A. Orthogonality of Trigonometric Basis

We want to prove the relations (for $w_m = \frac{2p\,m}{T},\ m = 1,2,\ldots.$)

$$\int_0^T \cos(w_m t)\cos(w_n t)\ dt = \begin{cases} 0, & m \neq n \\ \frac{T}{2}, & m = n. \end{cases}$$

$$\int_0^T \sin(w_m t)\sin(w_n t)\ dt = \begin{cases} 0, & m \neq n \\ \frac{T}{2}, & m = n. \end{cases}$$

$$\int_0^T \cos(w_m t)\sin(w_n t)\ dt = 0.$$

These are based upon the trigonometric identities in Appendix F. For $w_m \neq \pm w_n$

$$\begin{aligned} \int_0^T \cos(w_m t)\sin(w_n t)\ dt &= \frac{1}{2}\int_0^T \sin((w_n + w_m)t) + \sin((w_n - w_m)t)\ dt \\ &= \frac{1}{2}\left[\frac{-\cos((w_n + w_m)t)}{w_n + w_m} + \frac{-\cos((w_n - w_m)t)}{w_n - w_m} \right]_0^T \\ &= 0 \end{aligned}$$

since $(w_n \pm w_m)T = 2p(n \pm m)$ and $\cos(2p(n \pm m)) = 1$ for n and m integers. Similarly, the other integrals can be computed.

$$\begin{aligned} \int_0^T \cos(w_m t)\cos(w_n t)\ dt &= \frac{1}{2}\int_0^T \cos((w_n + w_m)t) + \cos((w_n - w_m)t)\ dt \\ &= \frac{1}{2}\left[\frac{\sin((w_n + w_m)t)}{w_n + w_m} + \frac{\sin((w_n - w_m)t)}{w_n - w_m} \right]_0^T \\ &= 0 \end{aligned}$$

since $\sin(2p(n \pm m)) = 0.$ Finally,

$$\int_0^T \sin(w_m t)\sin(w_n t)\ dt = \frac{1}{2}\int_0^T \cos((w_n - w_m)t) - \cos((w_n + w_m)t)\ dt$$

$$= \frac{1}{2}\left[\frac{\sin((w_n - w_m)t)}{w_n - w_m} + \frac{\sin((w_n + w_m)t)}{w_n + w_m}\right]_0^T$$

$$= 0$$

For $w_m = w_n$, we have instead

$$\int_0^T \cos(w_n t)\cos(w_n t)\ dt = \int_0^T \cos^2(w_n t)\ dt$$

$$= \frac{1}{2}\int_0^T [1 + \cos(2w_n t)\ ]dt$$

$$= \frac{1}{2}\left[t + \frac{\sin(2w_n t)}{2w_n}\right]_0^T$$

$$= \frac{T}{2}.$$

$$\int_0^T \sin(w_n t)\sin(w_n t)\ dt = \int_0^T \sin^2(w_n t)\ dt$$

$$= \frac{1}{2}\int_0^T [1 - \cos(2w_n t)\ ]dt$$

$$= \frac{1}{2}\left[t - \frac{\sin(2w_n t)}{2w_n}\right]_0^T$$

$$= \frac{T}{2}.$$

Here we have used the trigonometric identities $\cos^2 q = \frac{1}{2}(1 + \cos^2 2q)$ and $\sin^2 q = \frac{1}{2}(1 - \cos^2 2q)$, which are obtainable from the identities in Appendix F.

B. Derivation of Fourier Coefficients

We now look at the Fourier series representation

$$f(t) = \frac{1}{2}A_0 + \sum_{p=1}^{\infty}[A_p \cos(w_p t) + B_p \sin(w_p t)].$$

We can use the orthogonality relations in the last Appendix in order to arrive at expressions for the Fourier coefficients.

First, we integrate the series over one period.

$$\int_0^T f(t)\,dt = \int_0^T \left\{ \frac{1}{2}A_0 + \sum_{p=1}^{\infty} [A_p \cos(w_p t) + B_p \sin(w_p t)] \right\} dt$$

$$= \frac{1}{2}A_0 \int_0^T dt + \sum_{p=1}^{\infty} \left[A_p \int_0^T \cos(w_p t)\,dt + B_p \int_0^T \sin(w_p t)\,dt \right]$$

$$= \frac{1}{2}A_0 T + \sum_{p=1}^{\infty} \left[A_p \frac{\sin(w_p t)}{w_p}\bigg|_0^T + B_p \frac{-\cos(w_p t)}{w_p}\bigg|_0^T \right]$$

$$= \frac{1}{2}A_0 T$$

This will give the expression $A_0 = \frac{2}{T}\int_0^T f(t)\,dt$.

Now multiply the series by $\cos(w_q t)$ for some $q = 1,2,\ldots$ and integrate.

$$\int_0^T f(t)\cos(w_q t)\,dt = \int_0^T \left\{ \frac{1}{2}A_0 + \sum_{p=1}^{\infty} [A_p \cos(w_p t) + B_p \sin(w_p t)] \right\} \cos(w_q t)\,dt$$

$$= \frac{1}{2}A_0 \int_0^T \cos(w_q t)\,dt + \sum_{p=1}^{\infty} \left[A_p \int_0^T \cos(w_p t)\cos(w_q t)\,dt + B_p \int_0^T \sin(w_p t)\cos(w_q t)\,dt \right]$$

$$= \frac{1}{2}A_0 \frac{\sin(w_q t)}{w_q}\bigg|_0^T + \sum_{p=1}^{\infty} \left[A_p \frac{T}{2}\delta_{p,q} + B_p \cdot 0 \right]$$

$$= \frac{T}{2}A_q.$$

Here use was made of the known orthogonality relations from the last section. Also, the Kronecker delta was used, defined as

$$\delta_{n,m} = \begin{cases} 0, & m \neq n \\ 1, & m = n \end{cases}.$$

This renders the sum above a sum with all zero terms except for one, that for $p = q$.

For example, if $q = 3$, we have

$$\int_0^T f(t)\cos(w_3 t)\,dt = A_1 \cdot 0 + A_2 \cdot 0 + A_3 \frac{T}{2} + A_4 \cdot 0 + \cdots = \frac{T}{2}A_3.$$

Thus, one can solve for the A_q's to obtain $A_q = \frac{2}{T}\int_0^T f(t)\cos(w_q t)\,dt, \quad q = 1,2,\ldots.$

Similarly, one has

$$\int_0^T f(t)\sin(w_q t)\,dt = \int_0^T \left\{\frac{1}{2}A_0 + \sum_{p=1}^{\infty}[A_p\cos(w_p t) + B_p\sin(w_p t)]\right\}\sin(w_q t)\,dt$$

$$= \frac{1}{2}A_0\int_0^T \sin(w_q t)dt + \sum_{p=1}^{\infty}\left[A_p\int_0^T \cos(w_p t)\sin(w_q t)\,dt + B_p\int_0^T \sin(w_p t)\sin(w_q t)\,dt\right]$$

$$= \frac{1}{2}A_0\frac{-\cos(w_q t)}{w_q}\Bigg|_0^T + \sum_{p=1}^{\infty}\left[A_p\cdot 0 + B_p\frac{T}{2}d_{p,q}\right]$$

$$= \frac{T}{2}B_q.$$

So, $B_q = \frac{2}{T}\int_0^T f(t)\sin(w_q t)\,dt.$

C. Discrete Orthogonality

The derivation of the discrete Fourier coefficients can be done using the discrete orthogonality of the discrete trigonometric basis similar to the derivation of the above Fourier coefficients for the Fourier series. We first prove the following

$$\sum_{n=1}^{N}\cos\left(\frac{2\pi nk}{N}\right) = \begin{cases} 0, & k = 1,\ldots,N-1 \\ N, & k = 0, N \end{cases}$$

$$\sum_{n=1}^{N}\sin\left(\frac{2\pi nk}{N}\right) = 0, \quad k = 0,\ldots,N$$

This can be done more easily using the exponential form,

$$\sum_{n=1}^{N}\cos\left(\frac{2\pi nk}{N}\right) + i\sum_{n=1}^{N}\sin\left(\frac{2\pi nk}{N}\right) = \sum_{n=1}^{N}e^{2\pi ink/N},$$

by using Euler's formula, $e^{iq} = \cos q + i\sin q$ for each term in the sum.

The exponential sum is the sum of a geometric progression, which can be summed as done in Appendix G. Thus, we have

$$\sum_{n=1}^{N}e^{2\pi ink/N} = \sum_{n=1}^{N}\left(e^{2\pi ik/N}\right)^n$$

$$= e^{2\pi ik/N} + \left(e^{2\pi ik/N}\right)^2 + \cdots + \left(e^{2\pi ik/N}\right)^N$$

$$= \frac{e^{2\pi ik/N}\left[1 - \left(e^{2\pi ik/N}\right)^N\right]}{1 - e^{2\pi ik/N}}$$

$$= \frac{e^{2\pi ik/N}\left[1 - e^{2\pi ik}\right]}{1 - e^{2\pi ik/N}}.$$

As long as $k \neq 0, N$ the numerator is 0. In the special cases that $k = 0, N$, we have $e^{2p\,ink/N} = 1$. So,

$$\sum_{n=1}^{N} e^{2p\,ink/N} = \sum_{n=1}^{N} 1 = N.$$

Therefore,

$$\sum_{n=1}^{N} \cos\left(\frac{2p\,nk}{N}\right) + i\sum_{n=1}^{N} \sin\left(\frac{2p\,nk}{N}\right) = \begin{cases} 0, & k = 1, \ldots, N-1 \\ N, & k = 0, N \end{cases}$$ and the result follows.

We can use this to establish orthogonality relations of the following type for $p, q = 0, 1, \ldots, \frac{N}{2}$:

$$\sum_{n=1}^{N} \cos\left(\frac{2p\,pn}{N}\right)\cos\left(\frac{2p\,qn}{N}\right) = \frac{1}{2}\left[\sum_{n=1}^{N} \cos\left(\frac{2p\,(p-q)n}{N}\right) + \cos\left(\frac{2p\,(p+q)n}{N}\right)\right].$$

Splitting the above sum into two sums and then evaluating the separate sums from earlier in this section,

$$\sum_{n=1}^{N} \cos\left(\frac{2p\,(p-q)n}{N}\right) = \begin{cases} 0, & p \neq q \\ N, & p = q \end{cases},$$

$$\sum_{n=1}^{N} \cos\left(\frac{2p\,(p+q)n}{N}\right) = \begin{cases} 0, & p+q \neq N \\ N, & p+q = N \end{cases}$$

we obtain

$$\sum_{n=1}^{N} \cos\left(\frac{2p\,pn}{N}\right)\cos\left(\frac{2p\,qn}{N}\right) = \begin{cases} N/2, & p = q \neq N/2 \\ N, & p = q = N/2. \\ 0, & \text{otherwise} \end{cases}$$

Similarly, we find

$$\sum_{n=1}^{N} \sin\left(\frac{2p\,pn}{N}\right)\cos\left(\frac{2p\,qn}{N}\right) = \frac{1}{2}\left[\sum_{n=1}^{N} \sin\left(\frac{2p\,(p-q)n}{N}\right) + \sin\left(\frac{2p\,(p+q)n}{N}\right)\right] = 0.$$

and

$$\sum_{n=1}^{N} \sin\left(\frac{2p\,pn}{N}\right)\sin\left(\frac{2p\,qn}{N}\right) = \frac{1}{2}\left[\sum_{n=1}^{N} \cos\left(\frac{2p\,(p-q)n}{N}\right) - \cos\left(\frac{2p\,(p+q)n}{N}\right)\right]$$

$$= \begin{cases} N/2, & p = q \neq N/2 \\ 0, & \text{otherwise} \end{cases}.$$

D. Discrete Transform

The derivation of the coefficients for the DFT is now easily obtained using the discrete orthogonality in the last section. We start with the expansion

$$f(t_n) = \frac{1}{2}A_0 + \sum_{p=1}^{N/2}\left[A_p \cos\left(\frac{2p\,pn}{N}\right) + B_p \sin\left(\frac{2p\,pn}{N}\right)\right].$$

We first sum over n:

$$\begin{aligned}
\sum_{n=1}^{N} f(t_n) &= \sum_{n=1}^{N}\left\{\frac{1}{2}A_0 + \sum_{p=1}^{N/2}\left[A_p \cos\left(\frac{2p\,pn}{N}\right) + B_p \sin\left(\frac{2p\,pn}{N}\right)\right]\right\} \\
&= \frac{1}{2}A_0\sum_{n=1}^{N}1 + \sum_{p=1}^{N/2}\left\{A_p\sum_{n=1}^{N}\cos\left(\frac{2p\,pn}{N}\right) + B_p\sum_{n=1}^{N}\sin\left(\frac{2p\,pn}{N}\right)\right\} \\
&= \frac{1}{2}A_0 N + \sum_{p=1}^{N/2}\left\{A_p\cdot 0 + B_p\cdot 0\right\} \\
&= \frac{1}{2}A_0 N.
\end{aligned}$$

Now, we can multiply both sides of the expansion by $\cos\left(\frac{2p\,qn}{N}\right)$ and sum over n.

$$\begin{aligned}
\sum_{n=1}^{N} f(t_n)\cos\left(\frac{2p\,qn}{N}\right) &= \sum_{n=1}^{N}\left\{\frac{1}{2}A_0 + \sum_{p=1}^{N/2}\left[A_p \cos\left(\frac{2p\,pn}{N}\right) + B_p \sin\left(\frac{2p\,pn}{N}\right)\right]\right\}\cos\left(\frac{2p\,qn}{N}\right) \\
&= \frac{1}{2}A_0\sum_{n=1}^{N}\cos\left(\frac{2p\,qn}{N}\right) \\
&\quad + \sum_{p=1}^{N/2}\left\{A_p\sum_{n=1}^{N}\cos\left(\frac{2p\,pn}{N}\right)\cos\left(\frac{2p\,qn}{N}\right) + B_p\sum_{n=1}^{N}\sin\left(\frac{2p\,pn}{N}\right)\cos\left(\frac{2p\,qn}{N}\right)\right\} \\
&= \begin{cases} \displaystyle\sum_{p=1}^{N/2}\left\{A_p\frac{N}{2}d_{p,q} + B_p\cdot 0\right\} & q \neq N/2 \\ \displaystyle\sum_{p=1}^{N/2}\left\{A_p N d_{p,N/2} + B_p\cdot 0\right\} & q = N/2 \end{cases} \\
&= \begin{cases} \dfrac{1}{2}A_q N, & q \neq N/2 \\ A_{N/2}N, & q = N/2 \end{cases}.
\end{aligned}$$

So, we have found that

$$A_q = \frac{2}{N}\sum_{n=1}^{N} f(t_n)\cos\left(\frac{2p\,qn}{N}\right), \quad q \neq \frac{N}{2},$$

$$A_{N/2} = \frac{1}{N}\sum_{n=1}^{N} f(t_n)\cos\left(\frac{2p\,n(N/2)}{N}\right) = \frac{1}{N}\sum_{n=1}^{N} f(t_n)\cos(p\,n).$$

Similarly,

$$\sum_{n=1}^{N} f(t_n)\sin\left(\frac{2p\,qn}{N}\right) = \sum_{n=1}^{N}\left\{\frac{1}{2}A_0 + \sum_{p=1}^{N/2}[A_p\cos\left(\frac{2p\,pn}{N}\right) + B_p\sin\left(\frac{2p\,pn}{N}\right)]\right\}\sin\left(\frac{2p\,qn}{N}\right)$$

$$= \frac{1}{2}A_0\sum_{n=1}^{N}\sin\left(\frac{2p\,qn}{N}\right) + \sum_{p=1}^{N/2}\left\{A_p\sum_{n=1}^{N}\cos\left(\frac{2p\,pn}{N}\right)\sin\left(\frac{2p\,qn}{N}\right) + B_p\sum_{n=1}^{N}\sin\left(\frac{2p\,pn}{N}\right)\sin\left(\frac{2p\,qn}{N}\right)\right\}$$

$$= \sum_{p=1}^{N/2}\left\{A_p\cdot 0 + B_p\frac{N}{2}d_{p,q}\right\}$$

$$= \frac{1}{2}B_q N.$$

Finally, we have

$$B_q = \frac{2}{N}\sum_{n=1}^{N} f(t_n)\sin\left(\frac{2p\,qn}{N}\right), \quad q = 1,\ldots,\frac{N}{2}-1.$$

E. The Discrete Exponential Transform

The derivation of the coefficients for the DFT was obtained using the discrete orthogonality in the last section. However, this is not the form used in Matlab for spectral analysis. Matlab allows for the computation of the Fast Fourier Transform (FFT) and its description in the help section does not involve sines and cosines. Namely, Matlab defines the transform and inverse transform as

"For length N input vector x, the DFT is a length N vector X, with elements

$$X(k) = \sum_{n=1}^{N} x(n)*\exp(-j*2*pi*(k-1)*(n-1)/N),\ 1 <= k <= N.$$

The inverse DFT (computed by IFFT) is given by

$$x(n) = (1/N)\sum_{k=1}^{N} X(k)*\exp(\ j*2*pi*(k-1)*(n-1)/N),\ 1 <= n <= N.$$

The relationship between the DFT and the Fourier coefficients a and b in

$$x(n) = a0 + \sum_{k=1}^{N/2} a(k)*\cos(2*pi*k*t(n)/(N*dt)) + b(k)*\sin(2*pi*k*t(n)/(N*dt))$$

is

a0 = X(1)/N, a(k) = 2*real(X(k+1))/N, b(k) = -2*imag(X(k+1))/N,

where x is a length N discrete signal sampled at times t with spacing dt."

Or, it also provides the following:

$$X(k) = \sum_{j=1}^{N} x(j)W_N^{(k-1)(n-1)},$$

$$x(j) = \frac{1}{N}\sum_{k=1}^{N} X(k)W_N^{-(k-1)(n-1)}, \text{ where } W_N = e^{-2p\,i/N} \text{ for } 1 \le k, n \le N.$$

In this section we will derive the discrete Fourier exponential transform in preparation for a discussion of FFT in the next section. We will start with the DFT.

$$f(t_n) = \frac{1}{2}A_0 + \sum_{p=1}^{N/2}\left[A_p \cos\left(\frac{2p\,pn}{N}\right) + B_p \sin\left(\frac{2p\,pn}{N}\right)\right].$$

Again, we can employ Euler's formula to rewrite the trigonometric functions in terms of exponentials, Namely, using $e^{iq} = \cos q + i\sin q$, we have that

$$\cos q = \frac{1}{2}\left(e^{iq} + e^{-iq}\right),$$

$$\sin q = \frac{1}{2i}\left(e^{iq} - e^{-iq}\right).$$

Then we have

$$f(t_n) = \frac{1}{2}A_0 + \sum_{p=1}^{N/2}\left\{A_p \frac{1}{2}\left[e^{2p\,ipn/N} + e^{-2p\,ipn/N}\right] + B_p \frac{1}{2i}\left[e^{2p\,ipn/N} - e^{-2p\,ipn/N}\right]\right\}$$

$$= \frac{1}{2}A_0 + \sum_{p=1}^{N/2}\left\{\frac{1}{2}\left[A_p - iB_p\right]e^{2p\,ipn/N} + \frac{1}{2}\left[A_p + iB_p\right]e^{-2p\,ipn/N}\right\}.$$

We define $C_p = \frac{1}{2}\left(A_p - iB_p\right)$ and note that the above result can be written as

$$f(t_n) = C_0 + \sum_{p=1}^{N/2}\left\{C_p e^{2p\,ipn/N} + C_p^* e^{-2p\,ipn/N}\right\}.$$

Here we have introduce the complex conjugate operation $(a+ib)^* = a - ib.$

The terms in the sums look similar. We can actually combine them into one form. Note that $e^{2p\,iN} = \cos(2p\,N) + i\sin(2p\,N) = 1.$ Thus, we can write $e^{-2p\,ipn/N} = e^{-2p\,ipn/N}e^{-2p\,iN} = e^{2p\,i(N-p)n/N}$ in the second sum. Since $p = 1,\ldots,N/2$, we see that $N - p = N-1, N-2,\ldots,N/2.$ So, we can rewrite the second sum as

$$\sum_{p=1}^{N/2} C_p^* e^{-2p\,ipn/N} = \sum_{p=1}^{N/2} C_p^* e^{2p\,i(N-p)n/N} = \sum_{q=N/2}^{N-1} C_{N-q}^* e^{2p\,iqn/N}.$$

Since q is a dummy index (it can be replaced by any letter without changing the value of the sum), we can replace it with a p and combine the terms in both sums to obtain

$$f(t_n) = \sum_{p=0}^{N-1} F_p e^{2p\,ipn/N},$$

where

$$F_p = \begin{cases} \dfrac{A_0}{2}, & p = 0 \\ \dfrac{1}{2}(A_p - iB_p), & 0 < p < N/2 \\ A_{N/2}, & p = N/2 \\ \dfrac{1}{2}(A_{N-p} + iB_{N-p}), & N/2 < p < N \end{cases}$$

Notice that the real and imaginary parts of the Fourier coefficients obey certain symmetry properties over the full range of the indices since the real and imaginary parts are related between $p \in [0, N/2]$ and $p \in [N/2, N-1]$. [*A description of this will be provided later*.]

We can know determine the coefficients in terms of the sampled data.

$$C_p = \frac{1}{2}(A_p - iB_p) = \frac{1}{N}\sum_{n=1}^{N} f(t_n)\left[\cos\left(\frac{2p\,pn}{N}\right) - i\sin\left(\frac{2p\,pn}{N}\right)\right]$$

$$= \frac{1}{N}\sum_{n=1}^{N} f(t_n)e^{-2p\,ipn/N}.$$

Thus,

$$F_p = \frac{1}{N}\sum_{n=1}^{N} f(t_n)e^{-2p\,ipn/N}, \quad 0 < p < \frac{N}{2}$$

and

$$F_p = C^*_{N-p} = \frac{1}{N}\sum_{n=1}^{N} f(t_n)e^{2p\,i(N-p)n/N}, \quad \frac{N}{2} < p < N$$

$$= \frac{1}{N}\sum_{n=1}^{N} f(t_n)e^{-2p\,ipn/N}.$$

We have shown that for all F's but two, the form is

$$F_p = \frac{1}{N}\sum_{n=1}^{N} f(t_n)e^{-2p\,ipn/N}.$$

However, we can easilty show that this is also true when $p = 0$ and $p = \frac{N}{2}$.

$$F_{N/2} = A_{N/2} = \frac{1}{N}\sum_{n=1}^{N} f(t_n)\cos(np)$$

$$= \frac{1}{N}\sum_{n=1}^{N} f(t_n)\left[\cos(np) - i\sin(np)\right]$$

$$= \frac{1}{N}\sum_{n=1}^{N} f(t_n)e^{-2p\,in(N/2)/N}$$

and

$$F_0 = \frac{1}{2}A_0 = \frac{1}{N}\sum_{n=1}^{N} f(t_n)$$
$$= \frac{1}{N}\sum_{n=1}^{N} f(t_n)e^{2p\,in(0)/N}$$

Thus, all of the F_p's are of the same form. This gives us the discrete transform pair

$$f(t_n) = \sum_{p=0}^{N-1} F_p e^{2p\,ipn/N},$$

$$F_p = \frac{1}{N}\sum_{n=1}^{N} f(t_n)e^{-2p\,ipn/N}.$$

Note that this is similar to the definition of the FFT given in Matlab.

F. Fast Fourier Transform

The usual computation of the discrete Fourier transform is done using the Fast Fouier Transform (FFT). There are various implementations of it, but a standard form is the Radix-2 FFT. We describe this FFT in the current section. We begin by writing the DFT compactly using $W = e^{-2p\,i/N}$. Note that $W^{N/2} = -1, \quad W^N = 1,$ and $e^{2p\,ijk/N} = W^{jk}$. We can then write

$$F_k = \sum_{j=0}^{N-1} W^{jk} f_j.$$

The key to the FFT is that this sum can be written as two similar sums:

$$F_k = \sum_{j=0}^{N-1} W^{jk} f_j$$
$$= \sum_{j=0}^{N/2-1} W^{jk} f_j + \sum_{j=N/2}^{N-1} W^{jk} f_j$$
$$= \sum_{j=0}^{N/2-1} W^{jk} f_j + \sum_{m=0}^{N/2-1} W^{k(m+N/2)} f_{m+N/2}, \quad \text{for } m = j - \frac{N}{2}$$
$$= \sum_{j=0}^{N/2-1} \left[W^{jk} f_j + W^{k(j+N/2)} f_{j+N/2} \right]$$
$$= \sum_{j=0}^{N/2-1} W^{jk} \left[f_j + (-1)^k f_{j+N/2} \right], \text{since } W^{k(j+N/2)} = W^{kj}(W^{N/2})^k \; and \; W^{N/2} = -1.$$

Thus, the sum appears to be of the same form as before, but there are half as many terms with a different coefficient for the W^{jk}'s. In fact, we can separate the terms involving the + or – sign by looking at the even and odd values of k.

For even $k = 2m,$ we have

$$F_{2m} = \sum_{j=0}^{N/2-1} \left(W^{2m}\right)^j \left[f_j + f_{j+N/2} \right], \quad m = 0, \ldots \frac{N}{2} - 1.$$

For odd $k = 2m+1,$ we have

$$F_{2m+1} = \sum_{j=0}^{N/2-1} \left(W^{2m}\right)^j W^j \left[f_j - f_{j+N/2} \right], \quad m = 0, \ldots \frac{N}{2} - 1.$$

Each of these equations gives the Fourier coefficients in terms of a similar sum using fewer terms and with a different weight, $W^2 = \left(e^{-2\pi i/N}\right)^2 = e^{-2\pi i/(N/2)}.$ IF N is a power of 2, then this process can be repeated over and over until one ends up with a simple sum.

The process is easily seen when written out for a small number of samples. Let $N = 8.$ Then a first pass at the above gives

$$\begin{aligned}
F_0 &= f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 \\
F_1 &= f_0 + Wf_1 + W^2 f_2 + W^3 f_3 + W^4 f_4 + W^5 f_5 + W^6 f_6 + W^7 f_7 \\
F_2 &= f_0 + W^2 f_1 + W^4 f_2 + W^6 f_3 + f_4 + W^2 f_5 + W^4 f_6 + W^6 f_7 \\
F_3 &= f_0 + W^3 f_1 + W^6 f_2 + Wf_3 + W^4 f_4 + W^7 f_5 + W^2 f_6 + W^5 f_7 \\
F_4 &= f_0 + W^4 f_1 + f_2 + W^4 f_3 + f_4 + W^4 f_5 + f_6 + W^4 f_7 \\
F_5 &= f_0 + W^5 f_1 + W^2 f_2 + W^7 f_3 + W^4 f_4 + Wf_5 + W^6 f_6 + W^3 f_7 \\
F_6 &= f_0 + W^6 f_1 + W^4 f_2 + W^2 f_3 + f_4 + W^6 f_5 + W^4 f_6 + W^2 f_7 \\
F_7 &= f_0 + W^7 f_1 + W^6 f_2 + W^5 f_3 + W^4 f_4 + W^3 f_5 + W^2 f_6 + Wf_7
\end{aligned}$$

The point is that the terms in these expressions can be regrouped with $W = e^{-\pi i/8}$ and noting $W^4 = -1$:

$$\begin{aligned}
F_0 &= (f_0 + f_4) + (f_1 + f_5) + (f_2 + f_6) + (f_3 + f_7) \\
&\equiv g_0 + g_1 + g_2 + g_3 \\
F_1 &= (f_0 - f_4) + (f_1 - f_5)W + (f_2 - f_6)W^2 + (f_3 - f_7)W^3 \\
&\equiv g_4 + g_6 + g_5 + g_7 \\
F_2 &= (f_0 + f_4) + (f_1 + f_5)W^2 - (f_2 + f_6) - (f_3 + f_7)W^2 \\
&= g_0 - g_2 + (g_1 - g_3)W^2 \\
F_3 &= (f_0 - f_4) - (f_2 - f_6)W + (f_1 - f_5)WW^2 + (f_3 - f_7)WW^6 \\
&= g_4 - g_6 + g_5 W^2 + g_7 W^6 \\
F_4 &= (f_0 + f_4) + (f_1 + f_5) - (f_2 + f_6) - (f_3 + f_7) \\
&= g_0 + g_2 - g_1 - g_3
\end{aligned}$$

$$
\begin{aligned}
F_5 &= (f_0 - f_4) + (f_2 - f_6)W + (f_1 - f_5)WW^4 + (f_3 - f_7)WW^4 \\
&= g_4 + g_6 + g_5 W^4 + g_7 W^4 \\
F_6 &= (f_0 + f_4) + (f_1 + f_5)W^6 - (f_2 + f_6) - (f_3 + f_7)W^6 \\
&= g_0 - g_2 + (g_1 - g_3)W^6 \\
F_7 &= (f_0 - f_4) - (f_2 - f_6)W + (f_1 - f_5)WW^6 + (f_3 - f_7)WW^2 \\
&= g_4 - g_6 + g_5 W^6 + g_7 W^2
\end{aligned}
$$

However, each of the *g*-series can be rewritten as well, leading to

$$
\begin{aligned}
F_0 &= (g_0 + g_2) + (g_1 + g_3) \equiv h_0 + h_1 \\
F_1 &= (g_4 + g_6) + (g_5 + g_7) \equiv h_4 + h_5 \\
F_2 &= (g_0 - g_2) + (g_1 - g_3)W^2 \equiv h_2 + h_3 \\
F_3 &= (g_4 - g_6) + (g_5 - g_7)W^2 \equiv h_6 + h_7 \\
F_4 &= (g_0 + g_2) - (g_1 + g_3) = h_0 - h_1 \\
F_5 &= (g_4 + g_6) - (g_5 + g_7) = h_4 - h_5 \\
F_6 &= g_0 - g_2 - (g_1 - g_3)W^2 = h_2 - h_3 \\
F_7 &= g_4 - g_6 + g_5 W^6 + g_7 W^2 = h_6 - h_7
\end{aligned}
$$

Thus, the computation of the Fourier coefficients amounts to inputting the *f*'s and computing the *g*'s. This takes 8 additions and 4 multiplications. Then one get the *h*'s, which is another 8 additions and 4 multiplications. There are three stages, amounting to a total of 12 multiplications and 24 additions. Carrying out the process in general, one has $\log_2 N$ steps with $N/2$ multiplications and *N* additions per step. In the direct computation one has $(N-1)^2$ multiplications and $N(N-1)$ additions. Thus, for $N = 8$, that would be 49 multiplications and 56 additions.

The above process is typically shown schematically in a "butterfly diagram". Examples can be found at other sites. One might be provided here at a later date. For now, we have the basic butterfly transformation displayed as

$$
\begin{array}{ccc}
f_j & \rightarrow & f_j + f_{\frac{1}{2}N+j} \\
 & \times & \\
f_{\frac{1}{2}N+j} & \rightarrow & \left[f_j - f_{\frac{1}{2}N+j} \right] W^j
\end{array}
$$

In the actual implementation, one computes with the *h*'s in the following order:

Output and Binary Representation	Desired Order
$h_0 + h_1 = F_0$, 000	F_0, 000
$h_0 - h_1 = F_4$, 100	F_1, 001
$h_2 + h_3 = F_2$, 010	F_2, 010
$h_2 - h_3 = F_6$, 110	F_3, 011
$h_4 + h_5 = F_1$, 001	F_4, 100
$h_4 - h_5 = F_5$, 101	F_5, 101
$h_6 + h_7 = F_3$, 011	F_6, 110
$h_6 - h_7 = F_7$, 111	F_7, 111

The binary representation of the index was also listed. Notice that the output is in bit-reversed order as compared to the right side of the table which shows the coefficients in the correct order. [Just compare the columns in each set of binary representations.] So, typically there is a bit reversal routine needed to unscramble the order of the output coefficients in order to use them.

G. Trigonometric Identities

The basic trigonometric identities that one needs the product of trigonometric functions expanded as simple expressions. These are based upon the sum and difference identities:

$$\sin(A \pm B) = \sin(A)\cos(B) \pm \sin(B)\cos(A)$$
$$\cos(A \pm B) = \cos(A)\cos(B) \mp \sin(A)\sin(B).$$

Adding or subtracting one obtains

$$\sin(A+B) + \sin(A-B) = 2\sin(A)\cos(B)$$
$$\cos(A+B) + \cos(A-B) = 2\cos(A)\cos(B)$$
$$\cos(A-B) - \cos(A+B) = 2\sin(A)\sin(B).$$

H. Geometric Progression

Another frequently occurring computation is the sum of a geometric progression. This is a sum of the form $S_N = \sum_{k=0}^{N} ar^k$. This is a sum of $N+1$ terms in which consecutive terms have a constant ratio, r. The sum is easily computed. One multiplies the sum S_N by r and subtracts the resulting sum from the original sum to obtain

$S_N - rS_N = (a + ar + \cdots + ar^N) - (ar + \cdots + ar^N + ar^{N+1}) = a - ar^{N+1}$.

Factoring on both sides of this chain of equations yields the desired sum,

$$S_N = \frac{a(1-r^{N+1})}{1-r}.$$

I. Matrix Operations for Matlab

The beauty of using Matlab is that many operations can be performed using matrix operations and that one can perform complex arithmetic. This eliminates many loops and make the coding of computations quicker. However, one needs to be able to understand the formalism. In this section we elaborate on these operations so that one can see how the Matlab implementation of the direct computation of the DFT can be carried out in compact form as shown previously in the Matlab Implementation section. This is all based upon the structure of Matlab, which is essentially a MATrix LABoratory.

A key operation between matrices is matrix multiplication. An $n \times m$ matrix is simply a collection of numbers arranged in n rows and m columns. For example, the matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ is a 2×3 matrix. The entries (elements) of a general matrix A can be represented as A_{ij} which represents the ith row and jth column.

Given two matrices, A and B, we can define the multiplication of these matrices when the number of columns of A equals the number of rows of B. The product, which we represent as matrix C, is given by the ijth element of C. In particular, we let A be a $p \times m$ matrix and B an $m \times q$ matrix. The product, C, will be a $p \times q$ matrix with entries

$$\begin{aligned} C_{ij} &= \sum_{k=1}^{m} A_{ik} B_{kj}, \quad i = 1,\ldots, p, \quad j = 1,\ldots, q, \\ &= A_{i1}B_{1j} + A_{12}B_{2j} + \ldots A_{im}B_{mj}. \end{aligned}$$

If we wanted to compute the sum $\sum_{n=1}^{N} a_n b_n$, then in a typical programming language we could use a loop, such as

Sum =0
Loop n from 1 to N
Sum = Sum + a(n)*b(n)
End Loop

In Matlab we could do this with a loop as above, or we could resort to matrix multiplication. We can let **a** and **b** be $1 \times n$ and $n \times 1$ matrices, respectively. Then the product would be a 1×1 matrix; namely, the sum we are seeking. However, these matrices are not always of the suggested size.

A $1 \times n$ matrix is called a row vector and a $1 \times n$ matrix is called a column vector. Often we have that both are of the same type. One can convert a row vector into a column

vector, or vice versa, using the matrix operation called a *transpose*. More generally, the transpose of a matrix is defined as follows: A^T has the elements satisfying $\left(A^T\right)_{ij} = A_{ji}$. In Matlab, the transpose if a matrix A is A'.

Thus, if we want to perform the above sum, we have $\sum_{n=1}^{N} a_n b_n = \sum_{n=1}^{N} a_{1n} b_{n1}$. In particular, if both **a** and **b** are row vectors, the sum in Matlab is given by **ab'**, and if they are both row vectors, the sum is **a'b**. This notation is much easier to type.

In our computation of the DFT, we have many sums. For example, we want to compute the coefficients of the sine functions,

$$B_p = \frac{2}{N}\sum_{n=1}^{N} y(t_n)\sin(\tfrac{2ppn}{N}), \quad p = 0,\ldots,N/2$$

The sum can be computed as a matrix product. The function *y* only has values at times t_n. This is the sampled data. We can represent it as a vector. The sine functions take values at arguments (angles) depending upon *p* and *n*. So, we can represent the sines as an $N\times(N/2+1)$ or $(N/2+1)\times N$ matrix. The Fourier coefficient thus becomes a simple matrix multiplication, ignoring the prefactor $\frac{2}{N}$. Thus, if we put the sampled data in a $1\times N$ vector **Y** and put the sines in an $N\times(\frac{N}{2}+1)$ vector **S**, the Fourier coefficient will be the product, which has size $1\times\left(\frac{N}{2}+1\right)$. Thus, in the code we see that these coefficients are computed as **B=2/N*y*S** for the given **y** and **B** matrices. The A coefficients are computed in the same manner. Comparing the two codes in that section, we see how much easier it is to implement. However, the number of multiplications and additions has not decreased. This is why the FFT is generally better. But, seeing the direct implementation helps one to understand what is being computed before seeking a more efficient implementation, such as the FFT.