6
From Analog to Discrete Signals

You don't have to be a mathematician to have a feel for numbers. - John Forbes Nash (1928 -)

As you may recall, a goal of this course has been to introduce some of the tools of applied mathematics with an underlying theme of finding the connection between analog and discrete signals. We began our studies with Fourier series, which provided representations of periodic functions. We then moved on to the study of Fourier transforms, which represent functions defined over all space. Such functions can be used to describe analog signals. However, we cannot record and store analog signals. There is an uncountable amount of information in analog signals. We record a signal for a finite amount of time and even then we can only store samples of the signal over that time interval. The resulting signal is discrete. These discrete signals are represented using the Discrete Fourier Transform (DFT). In this chapter we will discuss the general steps of relating analog, periodic and discrete signals. Then we will go further into the properties of the Discrete Fourier Transform (DFT) and in the next chapter we will apply what we have learned to the study of real signals.

6.1 *Analog to Periodic Signals*

We begin by considering a typical analog signal and its Fourier transform as shown in Figure 6.1. Analog signals can be described as piecewise continuous functions defined over infinite time intervals. The resulting Fourier transforms are also piecewise continuous and defined over infinite intervals of frequencies. We represent an analog signal, $f(t)$, and its transform, $\hat{f}(\omega)$, using the inverse Fourier transform and the Fourier transform, respectively:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(\omega)e^{-i\omega t}\,d\omega, \tag{6.1}$$

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t)e^{i\omega t}\,dt. \tag{6.2}$$

Note that the figures in this section are drawn as if the transform is real-valued. (See Figure 6.1.) However, in general they are not and we will investigate how this can be handled in the next chapter.

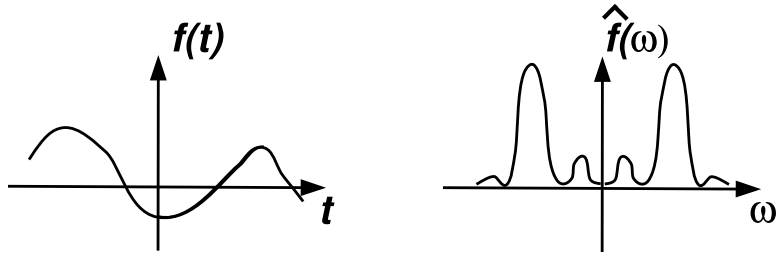

Figure 6.1: Plot of an analog signal $f(t)$ and its Fourier transform $\hat{f}(\omega)$.

Real signals cannot be studied on an infinite interval. Such signals are captured as data over a finite time interval. Let's assume that the recording starts at $t = 0$. Then, the record interval will be written as $[0, T]$, where T is called the *record length*.

The natural representation of a function, $f(t)$, for $t \in [0, T]$, is obtained by extending the signal to a periodic signal knowing that the physical signal is only defined on $[0, T]$. Recall that periodic functions can be modeled by a Fourier exponential series. We will denote the periodic extension of $f(t)$ by $f_p(t)$. The Fourier series representation of f_p and its Fourier coefficients are given by

Recall from Chapter 6 that we defined Fourier exponential series for intervals of length $2L$. It is easy to map that series expansion to one of length T, resulting in the representation used here.

$$\begin{aligned} f_p(t) &= \sum_{n=-\infty}^{\infty} c_n e^{-i\omega_n t}, \\ c_n &= \frac{1}{T}\int_0^T f_p(t) e^{i\omega_n t}\, dt. \end{aligned} \tag{6.3}$$

Here we have defined the discrete angular frequency $\omega_n = \frac{2\pi n}{T}$. The associated frequency is then $\nu_n = \frac{n}{T}$.

Given that $f_p(t)$ is a periodic function, we would like to relate its Fourier series representation to the Fourier transform, $\hat{f}_p(\omega)$, of the corresponding signal $f_p(t)$. This is done by simply computing its Fourier transform:

Here we begin with a signal $f(t)$ defined on $[0, T]$ and obtain the Fourier series representation (6.3) which gives the periodic extension of this function, $f_p(t)$. The frequencies are discrete as shown in Figure 6.2. We will determine the Fourier transform as

$$\hat{f}_p(\omega) = \frac{2\pi}{T}\hat{f}(\omega)\mathrm{comb}_{\frac{2\pi}{T}}(\omega)$$

and conclude that f_p is the convolution of the signal f with a comb function.

$$\begin{aligned} \hat{f}_p(\omega) &= \int_{-\infty}^{\infty} f_p(t) e^{i\omega t}\, dt \\ &= \int_{-\infty}^{\infty} \left(\sum_{n=-\infty}^{\infty} c_n e^{-i\omega_n t} \right) e^{i\omega t}\, dt \\ &= \sum_{n=-\infty}^{\infty} c_n \int_{-\infty}^{\infty} e^{i(\omega-\omega_n)t}\, dt. \end{aligned} \tag{6.4}$$

Recalling from Equation (5.47) that

$$\int_{-\infty}^{\infty} e^{i\omega x}\, dx = 2\pi\delta(\omega),$$

we obtain the Fourier transform of $f_p(t)$:

$$\hat{f}_p(\omega) = \sum_{n=-\infty}^{\infty} 2\pi c_n \delta(\omega - \omega_n). \tag{6.5}$$

Thus, the Fourier transform of a periodic function is a series of spikes at discrete frequencies $\omega_n = \frac{2\pi n}{T}$ of strength $2\pi c_n$. This is represented in Figure 6.2. Note that the spikes are of finite height representing the factor $2\pi c_n$.

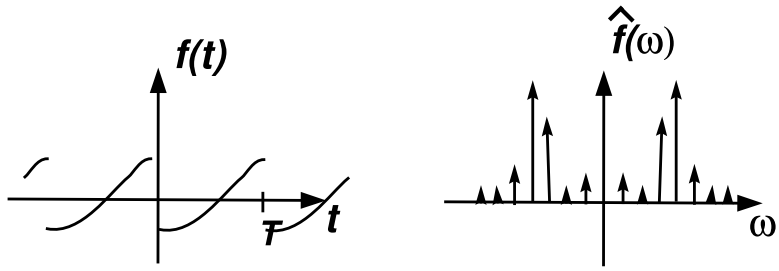

Figure 6.2: A periodic signal contains discrete frequencies $\omega_n = \frac{2\pi n}{T}$.

A simple example displaying this behavior is a signal with a single frequency, $f(t) = \cos\omega_0 t$. Restricting this function to a finite interval $[0, T]$, one obtains a version of the finite wave train, which was first introduced in Example 5.10 of the last chapter.

Example 6.1. *Find the real part of the Fourier transform of the finite wave train* $f(t) = \cos\omega_0 t$, $t \in [0, T]$.

Computing the real part of the Fourier transform, we find

$$\begin{aligned}
Re\hat{f}(\omega) &= Re\int_{-\infty}^{\infty} f(t)e^{i\omega t}\,dt \\
&= Re\int_0^T \cos\omega_0 t e^{i\omega t}\,dt \\
&= \int_0^T \cos\omega_0 t\cos\omega t\,dt \\
&= \frac{1}{2}\int_0^T \left[\cos((\omega+\omega_0)t) + \cos((\omega-\omega_0)t)\right]\,dt \\
&= \frac{1}{2}\left[\frac{\sin((\omega+\omega_0)T)}{\omega+\omega_0} + \frac{\sin((\omega-\omega_0)T)}{\omega-\omega_0}\right] \\
&= \frac{T}{2}\left[\text{sinc}\,((\omega+\omega_0)T) + \text{sinc}\,((\omega-\omega_0)T)\right]. \qquad (6.6)
\end{aligned}$$

Thus, the real part of the Fourier transform of this finite wave train consists of the sum of two sinc functions.

Two examples are provided in Figures 6.3 and 6.4. In both cases we consider $f(t) = \cos 2\pi t$, *but with* $t \in [0, 5]$ *or* $t \in [0, 2]$. *The corresponding Fourier transforms are also provided. We first see the sum of the two sinc functions in each case. Furthermore, the main peaks, centered at* $\omega = \pm\omega_0 = \pm 2\pi$, *are better defined for* $T = 5$ *than for* $T = 2$. *This indicates that a larger record length will provide a better frequency resolution.*

Next, we determine the relationship between the Fourier transform of $f_p(t)$ and the Fourier coefficients. Namely, evaluating $\hat{f}_p(\omega)$ at $\omega = \omega_k$, we have

$$\hat{f}_p(\omega_k) = \sum_{n=-\infty}^{\infty} 2\pi c_n \delta(\omega_k - \omega_n) = 2\pi c_k.$$

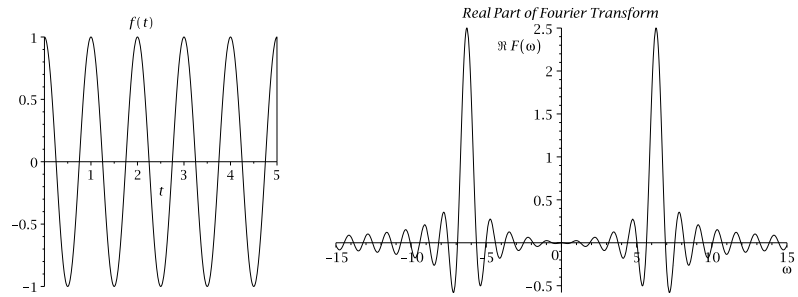

Figure 6.3: Plot of $f(t) = \cos 2\pi t$, $t \in [0,5]$ and the real and imaginary part of the Fourier transform.

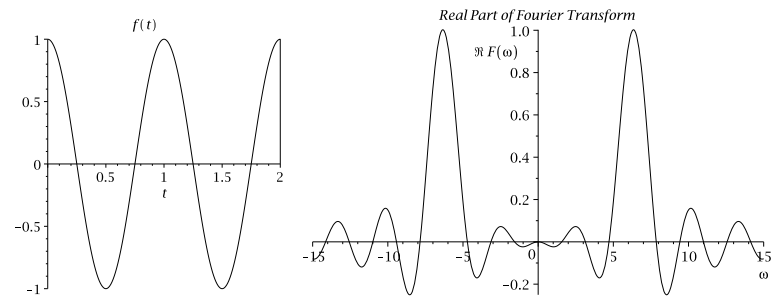

Figure 6.4: Plot of $f(t) = \cos 2\pi t$, $t \in [0,2]$ and the real and imaginary part of the Fourier transform.

Therefore, we can write the Fourier transform as

$$\hat{f}_p(\omega) = \sum_{n=-\infty}^{\infty} \hat{f}_p(\omega_n)\delta(\omega - \omega_n).$$

Further manipulation yields

$$\begin{aligned} \hat{f}_p(\omega) &= \sum_{n=-\infty}^{\infty} \hat{f}_p(\omega_n)\delta(\omega - \omega_n) \\ &= \sum_{n=-\infty}^{\infty} \hat{f}_p(\omega)\delta(\omega - \omega_n) \\ &= \hat{f}_p(\omega) \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T}). \end{aligned} \tag{6.7}$$

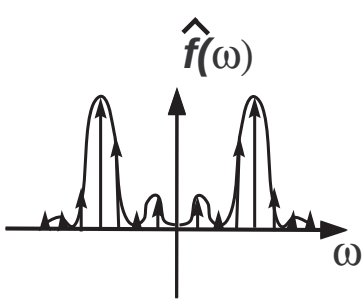

Figure 6.5: The discrete spectrum obtained from the Fourier transform of the periodic extension of $f(t)$, $t \in [0,T]$ is superimposed on the continuous spectrum of the analog signal.

This shows that the Fourier transform of a periodic signal is the product of the continuous function and a sum of δ function spikes of strength $\hat{f}_p(\omega_n)$ at frequencies $\omega_n = \frac{2\pi n}{T}$ as was shown in Figure 6.2. In Figure 6.5 the discrete spectrum is superimposed on the continuous spectrum emphasizing this connection.

The sum in the last line of Equation (6.7) is a special function, the Dirac comb function,

$$\text{comb}_{\frac{2\pi}{T}}(\omega) = \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T}),$$

which we discuss further in the next section.

6.2 *The Dirac Comb Function*

A FUNCTION THAT OFTEN OCCURS in signal analysis is the *Dirac comb function* defined by

$$\text{comb}_a(t) = \sum_{n=-\infty}^{\infty} \delta(t - na). \tag{6.8}$$

This function is simply a set of translated delta function spikes as shown in Figure 6.6. It is also called an *impulse train* or a *sampling function.* It is a periodic distribution and care needs to be taken in studying the properties of this function. We will show how the comb function can be used to relate analog and finite length signals.

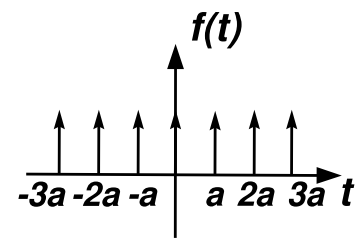

Figure 6.6: The Dirac comb function, $\text{comb}_a(t) = \sum_{n=-\infty}^{\infty} \delta(t - na)$.

In some fields, the comb function is written using the Cyrillic uppercase *Sha* function,

$$\text{Ш}(x) = \sum_{n=-\infty}^{\infty} \delta(x - n).$$

This is just a comb function with unit spacing, $a = 1$.

Employing the properties of the Dirac delta function, we can derive several properties of the Shah function. First, we have dilation and translation properties:

$$\begin{aligned} \text{Ш}(ax) &= \sum_{n=-\infty}^{\infty} \delta(ax - n) \\ &= \frac{1}{|a|} \sum_{n=-\infty}^{\infty} \delta(x - \frac{n}{a}) \\ &= \frac{1}{|a|} \text{comb}_{\frac{1}{a}}(x). \end{aligned} \tag{6.9}$$

$$\text{Ш}(x + k) = \text{Ш}(x), \quad k \text{ an integer.} \tag{6.10}$$

Also, we have the *Sampling Property*,

$$\text{Ш}(x) f(x) = \sum_{n=-\infty}^{\infty} f(n) \delta(x - n) \tag{6.11}$$

and the *Replicating Property*

$$(\text{Ш} * f)(x) = \sum_{n=-\infty}^{\infty} f(x - n). \tag{6.12}$$

These properties are easily confirmed. The Sampling Property is shown by

$$\begin{aligned} \text{Ш}(x) f(x) &= \sum_{n=-\infty}^{\infty} \delta(x - n) f(x) \\ &= \sum_{n=-\infty}^{\infty} f(n) \delta(x - n), \end{aligned} \tag{6.13}$$

while the Replicating Property is verified by a simple computation:

$$(\text{Ш} * f)(x) = \int_{-\infty}^{\infty} \text{Ш}(\xi) f(x - \xi)\, d\xi$$

$$
\begin{aligned}
&= \int_{-\infty}^{\infty} \left(\sum_{n=-\infty}^{\infty} \delta(\xi - n) \right) f(x-\xi)\, d\xi \\
&= \sum_{n=-\infty}^{\infty} \left(\int_{-\infty}^{\infty} \delta(\xi - n) f(x-\xi)\, d\xi \right) \\
&= \sum_{n=-\infty}^{\infty} f(x-n). \qquad (6.14)
\end{aligned}
$$

Thus, the convolution of a function with the Shah function results in a sum of translations of the function.

The comb function inherits these properties, since it can be written using the Shah function,

$$
\begin{aligned}
\mathrm{comb}_T(t) &= \sum_{n=-\infty}^{\infty} \delta(t - nT) \\
&= \sum_{n=-\infty}^{\infty} \delta(T(\frac{t}{T} - n)) \\
&= \sum_{n=-\infty}^{\infty} \frac{1}{|T|} \delta(\frac{t}{T} - n) \\
&= \frac{1}{|T|} \text{Ш}\left(\frac{t}{T}\right). \qquad (6.15)
\end{aligned}
$$

In the following we will only use the comb function.

We further note that $\mathrm{comb}_T(t)$ has period T :

$$
\begin{aligned}
\mathrm{comb}_T(t+T) &= \sum_{n=-\infty}^{\infty} \delta(t + T - nT) \\
&= \sum_{n=-\infty}^{\infty} \delta(t - nT) = \mathrm{comb}_T(t). \qquad (6.16)
\end{aligned}
$$

Thus, the comb function has a Fourier series representation on $[0, T]$.

Useful results from this discussion are the exponential Fourier series representation,

$$\mathrm{comb}_T(t) = \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{-2\pi i n t/T},$$

and the sum of exponentials

$$\sum_{n=-\infty}^{\infty} e^{-inat} = \frac{2\pi}{a} \mathrm{comb}_{\frac{2\pi}{a}}(t).$$

Example 6.2. *Find the Fourier exponential series representation of* $\mathrm{comb}_T(t)$.

Since $\mathrm{comb}_T(t)$ *has period* T*, the Fourier series representation takes the form*

$$\mathrm{comb}_T(t) = \sum_{n=-\infty}^{\infty} c_n e^{-i\omega_n t},$$

where $\omega_n = \frac{2\pi n}{T}$.

We can easily compute the Fourier coefficients:

$$
\begin{aligned}
c_n &= \frac{1}{T} \int_0^T \mathrm{comb}_T(t) e^{i\omega_n t}\, dt \\
&= \frac{1}{T} \int_0^T \sum_{k=-\infty}^{\infty} \delta(t - kT) e^{i\omega_n t}\, dt \\
&= \frac{1}{T} e^0 = \frac{1}{T}. \qquad (6.17)
\end{aligned}
$$

Note, the sum collapses because the only δ *function contribution comes from the* $k = 0$ *term (since* $kT \in [0, T]$ *only for* $k = 0$*). As a result, we have obtained the Fourier series representation of the comb function,*

$$\mathrm{comb}_T(t) = \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{-2\pi i n t/T}.$$

From this result we note that

$$\begin{aligned}\sum_{n=-\infty}^{\infty} e^{inat} &= \sum_{n=-\infty}^{\infty} e^{2\pi int/(2\pi/a)} \\ &= \frac{2\pi}{a}\mathrm{comb}_{\frac{2\pi}{a}}(t). \end{aligned} \tag{6.18}$$

Example 6.3. *Find the Fourier transform of* $\mathrm{comb}_a(t)$.

We compute the Fourier transform of the comb function directly:

$$\begin{aligned}F[\mathrm{comb}_a(t)] &= \int_{-\infty}^{\infty} \mathrm{comb}_a(t)e^{i\omega t}\,dt \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t-na)e^{i\omega t}\,dt \\ &= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(t-na)e^{i\omega t}\,dt \\ &= \sum_{n=-\infty}^{\infty} e^{i\omega na} \\ &= \frac{2\pi}{a}\mathrm{comb}_{\frac{2\pi}{a}}(\omega). \end{aligned} \tag{6.19}$$

The Fourier transform of a comb function is a comb function.

$$F[\mathrm{comb}_a(t)] = \frac{2\pi}{a}\mathrm{comb}_{\frac{2\pi}{a}}(\omega).$$

Example 6.4. *Show that the convolution of a function with* $\mathrm{comb}_T(x)$ *is a periodic function with period* T. *Since* $\mathrm{comb}_T(t)$ *is a periodic function with period* T, *we have*

$$\begin{aligned}(f * \mathrm{comb}_T)(t+T) &= \int_{-\infty}^{\infty} f(\tau)\mathrm{comb}_T(t+T-\tau)\,d\tau \\ &= \int_{-\infty}^{\infty} f(\tau)\mathrm{comb}_T(t-\tau)\,d\tau \\ &= (f * \mathrm{comb}_T)(t). \end{aligned} \tag{6.20}$$

Next, we will show that the periodic function $f_p(t)$ is nothing but a (Fourier) convolution of the analog function and the comb function. We first show that $(f * \mathrm{comb}_a)(t) = \sum_{n=-\infty}^{\infty} f(t-na)$.

Example 6.5. *Evaluate the convolution* $(f * \mathrm{comb}_a)(t)$ *directly.*

We carry out a direct computation of the convolution integral. We do this by first considering the convolution of a function $f(t)$ *with a shifted Dirac delta function,* $\delta_a(t) = \delta(t-a)$. *This convolution is easily computed as*

$$(f * \delta_a)(t) = \int_{-\infty}^{\infty} f(t-\tau)\delta(\tau-a)\,d\tau = f(t-a).$$

Therefore, the convolution of a function with a shifted delta function is a copy of $f(t)$ *that is shifted by* a.

Now convolve $f(t)$ *with the comb function:*

$$\begin{aligned}(f * \mathrm{comb}_a)(t) &= \int_{-\infty}^{\infty} f(t-\tau)\mathrm{comb}_a(\tau)\,d\tau \\ &= \int_{-\infty}^{\infty} f(t-\tau) \sum_{n=-\infty}^{\infty} \delta(\tau-na)\,d\tau \end{aligned}$$

$$
\begin{aligned}
&= \sum_{n=-\infty}^{\infty} (f * \delta_{na})(t) \\
&= \sum_{n=-\infty}^{\infty} f(t - na). \qquad (6.21)
\end{aligned}
$$

From this result we see that the convolution of a function $f(t)$ with a comb function is then the sum of shifted copies of $f(t)$, as shown in Figure 6.7. If the function has compact support on $[-\frac{a}{2}, \frac{a}{2}]$, i.e., the function is zero for $|t| > 1/a$, then the convolution with the comb function will be periodic.

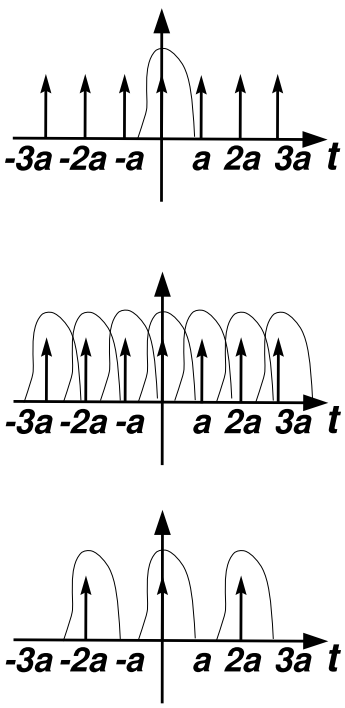

Figure 6.7: The convolution of $f(t)$ with the comb function, $\text{comb}_a(t)$. The first plot shows the function and the comb function. In the second of these plots we add the sum of several translations of $f(t)$. Incorrect sampling will lead to overlap in the translates and cause problems like aliasing. In the last of these plots, one has no overlap and the periodicity is evident.

Example 6.6. *Find the Fourier transform of the convolution* $(f * \text{comb}_a)(t)$.

This is done using the result of the last example and the first shift theorem.

$$
\begin{aligned}
F[f * comb_a] &= F\left[\sum_{n=-\infty}^{\infty} f(t - na)\right] \\
&= \sum_{n=-\infty}^{\infty} F[f(t - na)] \\
&= \sum_{n=-\infty}^{\infty} \hat{f}(\omega) e^{ina\omega} \\
&= \hat{f}(\omega) \sum_{n=-\infty}^{\infty} e^{ina\omega} \\
&= \frac{2\pi}{a} \hat{f}(\omega) comb_{\frac{2\pi}{a}}(\omega). \qquad (6.22)
\end{aligned}
$$

Another way to compute the Fourier transform of the convolution $(f * comb_a)(t)$ *is to note that the Fourier of this convolution is the product of the transforms of* f *and* $comb_a$ *by the Convolution Theorem. Therefore,*

$$
\begin{aligned}
F[f * comb_a] &= \hat{f}(\omega) F[comb_a](\omega) \\
&= \frac{2\pi}{a} \hat{f}(\omega) comb_{\frac{2\pi}{a}}(\omega). \qquad (6.23)
\end{aligned}
$$

We have obtained the same result, though in fewer steps.

For a function of period T, $f_p(t) = (f * \text{comb}_T)(t)$, we then have

$$
\hat{f}_p(\omega) = \frac{2\pi}{T} \hat{f}(\omega) \text{comb}_{\frac{2\pi}{T}}(\omega).
$$

Thus, the resulting spectrum is a series of scaled delta functions separated by $\Delta\omega = \frac{2\pi}{T}$.

The main message of this section has been that the Fourier transform of a periodic function produces a series of delta function spikes. This series of spikes is the transform of the convolution of $f(t)$ and a comb function. This is essentially a sampling of $\hat{f}(\omega)$ in frequency space. A similar result is obtained if we instead had a periodic Fourier transform due to a sampling in t-space. Combining both discrete functions in t and ω spaces, we have discrete signals, as will be described next.

6.3 Discrete Signals

WE WOULD LIKE TO SAMPLE A GIVEN SIGNAL at a discrete set of equally spaced times, $t_n \in [0, T]$. This is how one normally records signals. One samples the signal with a sampling frequency ν_s, such as ten samples per second, or 10 Hz. Therefore, the values are recorded in time steps of $T_s = 1/\nu_s$ For sampling at 10 Hz, this gives a sampling period of 0.1 s.

We can model this sampling at discrete time points by multiplying $f(t)$ by the comb function $\text{comb}_{T_s}(t)$. The Fourier transform will yield a convolution of the Fourier transform of $f(t)$ with the Fourier transform of the comb function. But this is a convolution of $\hat{f}(\omega)$ with another comb function, since the transform of a comb function is a comb function. Therefore, we will obtain a periodic representation in Fourier space.

Example 6.7. *Sample $f(t) = \cos\omega_0 t$ with a sampling frequency of $\nu_s = \frac{1}{T_s}$.*

As noted, the sampling of this function can be represented as

$$f_s(t) = \cos\omega_0 t\, \text{comb}_{T_s}(t).$$

We now evaluate the Fourier transform of $f_s(t)$:

$$\begin{aligned} F[\cos\omega_0 t\, \text{comb}_{T_s}(t)] &= \int_{-\infty}^{\infty} \cos\omega_0 t\, \text{comb}_{T_s}(t) e^{i\omega t}\, dt \\ &= \int_{-\infty}^{\infty} \cos\omega_0 t \left(\sum_{n=-\infty}^{\infty} \delta(t - nT_s) \right) e^{i\omega t}\, dt \\ &= \sum_{n=-\infty}^{\infty} \cos(\omega_0 n T_s) e^{i\omega n T_s} \\ &= \frac{1}{2} \sum_{n=-\infty}^{\infty} \left[e^{i(\omega+\omega_0)nT_s} + e^{i(\omega-\omega_0)nT_s} \right] \\ &= \frac{\pi}{T_s} \left[\text{comb}_{\frac{2\pi}{T_s}}(\omega + \omega_0) + \text{comb}_{\frac{2\pi}{T_s}}(\omega - \omega_0) \right]. \end{aligned} \tag{6.24}$$

Thus, we have shown that sampling $f(t) = \cos\omega_0 t$ with a sampling period of T_s results in a sum of translated comb functions. Each comb function consists of spikes separated by $\omega_s = \frac{2\pi}{T_s}$. Each set of spikes are translated by ω_0 to the left or the right and then added. Since each comb function is periodic with "period" ω_s in Fourier space, then the result turns out to be periodic as noted above.

In collecting data, we not only sample at a discrete set of points, but we also sample for a finite length of time. By sampling like this, we will not gather enough information to obtain the high frequencies in a signal. Thus, there will be a natural cutoff in the spectrum of the signal. This is represented in Figure 6.8. This process will lead to the Discrete Fourier Transform, the topic in the next section.

Here we will show that the sampling of a function defined on $[0, T]$ with sampling period T_s can represented by the convolution of f_p with a comb function. The resulting transform will be a sum of translations of $\hat{f}_p(\omega)$

Once again, we can use the comb function in the analysis of this process. We define a discrete signal as one which is represented by the product of a

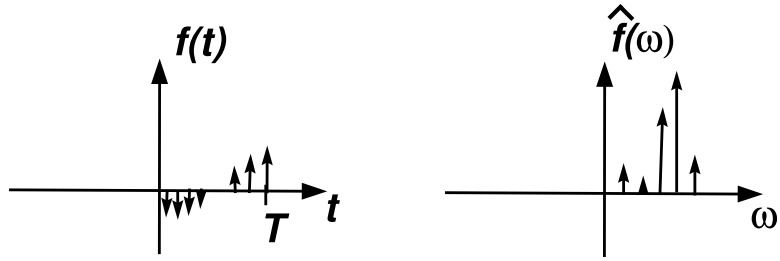

Figure 6.8: Sampling the original signal at a discrete set of times defined on a finite interval leads to a discrete set of frequencies in the transform that are restricted to a finite interval of frequencies.

periodic function sampled at discrete times. This suggests the representation

$$f_d(t) = f_p(t)\text{comb}_{T_s}(t). \tag{6.25}$$

Here T_s denotes the period of the sampling and $f_p(t)$ has the form

$$f_p(t) = \sum_{n=-\infty}^{\infty} c_n e^{-i\omega_n t}.$$

Example 6.8. *Evaluate the Fourier transform of $f_d(t)$.*

The Fourier transform of $f_d(t)$ can be computed:

$$\begin{aligned}
\hat{f}_d(\omega) &= \int_{-\infty}^{\infty} f_p(t) comb_{T_s}(t) e^{i\omega t}\, dt \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} \hat{f}_p(\mu) e^{-i\mu t}\, d\mu\right] comb_{T_s}(t) e^{i\omega t}\, dt \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}_p(\mu) \underbrace{\int_{-\infty}^{\infty} comb_{T_s}(t) e^{i(\omega-\mu)t}\, dt}_{\textit{Transform of } comb_{T_s} \textit{ at } \omega-\mu}\, d\mu \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}_p(\mu) \frac{2\pi}{T_s} comb_{\frac{2\pi}{T_s}}(\omega-\mu)\, d\mu \\
&= \frac{1}{T_s}\sum_{n=-\infty}^{\infty} \hat{f}_p(\omega - \frac{2\pi}{T_s} n).
\end{aligned} \tag{6.26}$$

Note that the Convolution Theorem for the convolution of Fourier transforms needs a factor of 2π.

We note that in the middle of this computation we find a convolution integral:

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}_p(\mu) \frac{2\pi}{T_s} \text{comb}_{\frac{2\pi}{T_s}}(\omega-\mu)\, d\mu = \left(\hat{f}_p * \text{comb}_{\frac{2\pi}{T_s}}\right)(\omega)$$

Also, it is easily seen that $\hat{f}_d(\omega)$ is a periodic function with period $\frac{2\pi}{T_s}$:

$$\hat{f}_d(\omega + \frac{2\pi}{T_s}) = \hat{f}_d(\omega).$$

We have shown that sampling a function $f_p(t)$ with sampling frequency $\nu_s = 1/T_s$, one obtains a periodic spectrum, $\hat{f}_d(\omega)$.

6.3.1 Summary

In this chapter we have taken an analog signal defined for $t \in (-\infty, \infty)$ and shown that restricting it to interval $t \in [0, T]$ leads to a periodic function

of period T,

$$f_p(t) = (f * \text{comb}_T)(t),$$

whose spectrum is discrete:

$$\hat{f}_p(\omega) = \hat{f}_p(\omega) \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T}). \tag{6.27}$$

We then sampled this function at discrete times,

$$f_d(t) = f_p(t)\text{comb}_{T_s}(t).$$

The Fourier transform of this discrete signal was found as

$$\hat{f}_d(\omega) = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} \hat{f}_p(\omega - \frac{2\pi}{T_s} n). \tag{6.28}$$

This function is periodic with period $\frac{2\pi}{T_s}$.

In Figure 6.9 we summarize the steps for going from analog signals to discrete signals. In the next chapter we will investigate the Discrete Fourier Transform.

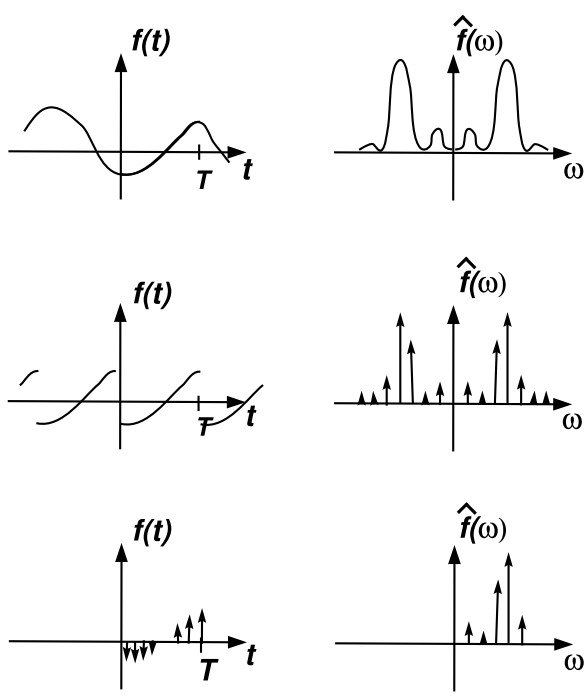

Figure 6.9: Summary of transforming analog to discrete signals. One starts with a continuous signal $f(t)$ defined on $(-\infty, \infty)$ and a continuous spectrum. By only recording the signal over a finite interval $[0, T]$, the recorded signal can be represented by its periodic extension. This in turn forces a discretization of the transform as shown in the second row of figures. Finally, by restricting the range of the sampled, as shown in the last row, the original signal appears as a discrete signal. This is also interpreted as the sampling of an analog signal leads to a restricted set of frequencies in the transform.

6.4 *The Discrete (Trigonometric) Fourier Transform*

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 duration. The data is then discrete and the corresponding frequencies are discrete and bounded. Thus, in the process of gathering data, one may seriously affect the frequency content of the signal. This is true for a simple superposition of signals with fixed frequencies. The situation becomes more complicated if the data has an overall non-constant (in time) trend or even exists in the presence of noise.

As described earlier in this chapter, we have seen that by restricting the data to a time interval $[0, T]$ for record length T, and periodically extending the data for $t \in (-\infty, \infty)$, one generates a periodic function of infinite duration at the cost of losing data outside the fundamental period. This is not unphysical, as data is typically taken over a finite time period. In addition, if one samples a finite number of values of the data on this finite interval, then the signal will contain frequencies limited to a finite range of values.

This process, as reviewed in the last section, leads us to a study of what is called the *Discrete Fourier Transform,* or *DFT.* We will investigate the discrete Fourier transform in both trigonometric and exponential form. While applications such as MATLAB rely on the exponential version, it is sometimes useful to deal with real functions using the familiar trigonometric functions.

We will change the notation to using $y(t)$ for the signal and f for the frequency.

Recall that in using Fourier series one seeks a representation of the signal, $y(t)$, valid for $t \in [0, T]$, as

$$y(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}[a_n \cos \omega_n t + b_n \sin \omega_n t], \tag{6.29}$$

where the angular frequency is given by $\omega_n = 2\pi f_n = \frac{2\pi n}{T}$. Note: In discussing signals, we will now use $y(t)$ instead of $f(t)$, allowing us to use f to denote the frequency (in Hertz) without confusion.

The frequency content of a signal for a particular frequency, f_n, is contained in both the cosine and sine terms when the corresponding Fourier coefficients, a_n, b_n, are not zero. So, one may desire to combine these terms. This is easily done by using trigonometric identities (as we had seen in Equation(2.1)). Namely, we show that

$$a_n \cos \omega_n t + b_n \sin \omega_n t = C_n \cos(\omega_n t + \psi), \tag{6.30}$$

where ψ is the *phase shift.* Recalling that

$$\cos(\omega_n t + \psi) = \cos \omega_n t \cos \psi - \sin \omega_n t \sin \psi, \tag{6.31}$$

one has

$$a_n \cos \omega_n t + b_n \sin \omega_n t = C_n \cos \omega_n t \cos \psi - C_n \sin \omega_n t \sin \psi.$$

Equating the coefficients of $\sin \omega_n t$ and $\cos \omega_n t$ in this expression, we obtain

$$a_n = C_n \cos \psi, \quad b_n = -C_n \sin \psi.$$

Therefore,

$$C_n = \sqrt{a_n + b_n} \quad \text{and} \quad \tan \psi = -\frac{b_n}{a_n}.$$

Recall that we had used orthogonality arguments in order to determine the Fourier coefficients ($a_n, n = 0, 1, 2, \ldots$ and $b_n, n = 1, 2, 3, \ldots$). In particular, using the orthogonality of the trigonometric basis, we found that

$$\begin{aligned} a_n &= \tfrac{2}{T}\int_0^T y(t)\cos\omega_n t\,dt, \quad n = 0, 1, 2, \ldots \\ b_n &= \tfrac{2}{T}\int_0^T y(t)\sin\omega_n t\,dt, \quad n = 1, 2, \ldots \end{aligned} \tag{6.32}$$

In Chapter 2 we had shown that one can write $a_n \cos\omega_n t + b_n \sin\omega_n t = C_n \sin(\omega_n t + \phi)$. It is easy to shown that *psi* and ϕ are related.

In the next section we will introduce the trigonometric version of the Discrete Fourier Transform. Its appearance is similar to that of the Fourier series representation in Equation (6.29). However, we will need to do a bit of work to obtain the discrete Fourier coefficients using discrete orthogonality.

6.4.1 *Discrete Trigonometric Series*

For the Fourier series analysis of a signal, 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 leads 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.

We first note that the data is sampled at N equally spaced times

$$t_n = n\Delta t, \quad n = 0, 1, \ldots, N-1,$$

where Δt is the time increment. For a record length of T, we have $\Delta t = T/N$. We will denote the data at these times as $y_n = y(t_n)$.

The DFT representation that we are seeking takes the form:

$$y_n = \frac{1}{2}a_0 + \sum_{p=1}^{M}\left[a_p \cos\omega_p t_n + b_p \sin\omega_p t_n\right],\ n = 0, 1, \ldots, N-1. \tag{6.33}$$

Here we note the discretizations used for future reference. Defining $\Delta t = \frac{T}{N}$ and $\Delta f = \frac{1}{T}$, we have

$$\begin{aligned} t_n &= n\Delta t = \frac{nT}{N}, \\ \omega_p &= p\Delta\omega = \frac{2\pi p}{T}, \\ f_p &= p\Delta f = \frac{p}{T}, \\ \omega_p t_n &= \frac{2\pi n p}{N}, \end{aligned}$$

for $n = 0, 1, \ldots, N-1$ and $p = 1, \ldots, M$.

The trigonometric arguments are given by

$$\omega_p t_n = \frac{2\pi p}{T} n\delta t = \frac{2\pi p n}{N}.$$

Note that $p = 1, \ldots, M$, thus allowing only for frequencies $f_p = \frac{\omega_p}{2\pi} = \frac{p}{T}$. Or, we could write

$$f_p = p\Delta f$$

for

$$\Delta f = \frac{1}{T}.$$

We need to determine M and the unknown coefficients. As for the Fourier series, we will need some orthogonality relations, but this time the orthogonality statement will consist of a sum and not an integral.

Since there are N sample values, (6.33) gives us a set of N equations for the unknown coefficients. Therefore, we should have N unknowns. For N

samples, we want to determine N unknown coefficients $a_0, a_1, \ldots, a_{N/2}$ and $b_1, \ldots, b_{N/2-1}$. Thus, we need to 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 at $p = 0, \frac{N}{2}$, leaving these two coefficients arbitrary. Thus, we can take them to be zero when they are included.

We claim that for the Discrete (Trigonometric) Fourier Transform

$$y_n = \frac{1}{2}a_0 + \sum_{p=1}^{M} [a_p \cos \omega_p t_n + b_p \sin \omega_p t_n], \; n = 0, 1, \ldots, N-1. \tag{6.34}$$

the DFT coefficients are given by

$$\begin{aligned}
a_p &= \frac{2}{N} \sum_{n=0}^{N-1} y_n \cos(\frac{2\pi pn}{N}), p = 1, \ldots N/2 - 1 \\
b_p &= \frac{2}{N} \sum_{n=0}^{N-1} y_n \sin(\frac{2\pi pn}{N}), p = 1, 2, \ldots N/2 - 1 \\
a_0 &= \frac{1}{N} \sum_{n=0}^{N-1} y_n, \\
a_{N/2} &= \frac{0}{N} \sum_{n=1}^{N-1} y_n \cos n\pi \\
b_0 &= b_{N/2} = 0
\end{aligned} \tag{6.35}$$

6.4.2 *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 in Chapter 2. We first prove the following

Theorem 6.1.

$$\begin{aligned}
\sum_{n=0}^{N-1} \cos\left(\tfrac{2\pi nk}{N}\right) &= \begin{cases} 0, & k = 1, \ldots, N-1 \\ N, & k = 0, N \end{cases} \\
\sum_{n=0}^{N-1} \sin\left(\tfrac{2\pi nk}{N}\right) &= 0, \quad k = 0, \ldots, N
\end{aligned} \tag{6.36}$$

Proof. This can be done more easily using the exponential form,

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi nk}{N}\right) + i \sum_{n=0}^{N-1} \sin\left(\frac{2\pi nk}{N}\right) = \sum_{n=0}^{N-1} e^{2\pi ink/N}, \tag{6.37}$$

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

The exponential sum is the sum of a geometric progression. Namely, we note that

$$\sum_{n=0}^{N-1} e^{2\pi ink/N} = \sum_{n=0}^{N-1} \left(e^{2\pi ik/N}\right)^n.$$

Recall from Chapter 2 that a geometric progression is a sum of the form $S_N = \sum_{k=0}^{N-1} ar^k$. This is a sum of N 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

$$S_N - rS_N = (a + ar + \cdots + ar^{N-1}) - (ar + \cdots + ar^N + ar^N) = a - ar^N. \tag{6.38}$$

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

$$S_N = \frac{a(1-r^N)}{1-r}. \tag{6.39}$$

Thus, we have

$$\begin{aligned}
\sum_{n=0}^{N-1} e^{2\pi ink/N} &= \sum_{n=0}^{N-1} \left(e^{2\pi ik/N}\right)^n \\
&= 1 + e^{2\pi ik/N} + \left(e^{2\pi ik/N}\right)^2 + \cdots + \left(e^{2\pi ik/N}\right)^{N-1} \\
&= \frac{1 - \left(e^{2\pi ik/N}\right)^N}{1 - e^{2\pi ik/N}} \\
&= \frac{1 - e^{2\pi ik}}{1 - e^{2\pi ik/N}}.
\end{aligned} \tag{6.40}$$

As long as $k \neq 0, N$ the numerator is 0 and $1 - e^{2\pi ik/N}$ is not zero.

In the special cases that $k = 0, N$, we have $e^{2\pi ink/N} = 1$. So,

$$\sum_{n=0}^{N-1} e^{2\pi ink/N} = \sum_{n=0}^{N-1} 1 = N.$$

Therefore,

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi nk}{N}\right) + i \sum_{n=0}^{N-1} \sin\left(\frac{2\pi nk}{N}\right) = \begin{cases} 0, & k = 1, \ldots, N-1 \\ N, & k = 0, N \end{cases} \tag{6.41}$$

and the Theorem is proved. □

We can use this result to establish orthogonality relations.

Example 6.9. *Evaluate the following:*

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi pn}{N}\right) \cos\left(\frac{2\pi qn}{N}\right),$$

$$\sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right) \cos\left(\frac{2\pi qn}{N}\right),$$

and

$$\sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right) \sin\left(\frac{2\pi qn}{N}\right).$$

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi pn}{N}\right) \cos\left(\frac{2\pi qn}{N}\right) = \frac{1}{2} \sum_{n=0}^{N-1} \left[\cos\left(\frac{2\pi(p-q)n}{N}\right) + \cos\left(\frac{2\pi(p+q)n}{N}\right)\right]. \tag{6.42}$$

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

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

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

we obtain

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi pn}{N}\right)\cos\left(\frac{2\pi qn}{N}\right) = \begin{cases} N/2, & p = q \neq N/2 \\ N, & p = q = N/2 \\ 0, & \textit{otherwise} \end{cases}. \tag{6.43}$$

Similarly, we find

$$\begin{aligned} & \sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right)\cos\left(\frac{2\pi qn}{N}\right) \\ = & \frac{1}{2}\sum_{n=0}^{N-1}\left[\sin\left(\frac{2\pi(p-q)n}{N}\right) + \sin\left(\frac{2\pi(p+q)n}{N}\right)\right] \\ = & 0. \end{aligned} \tag{6.44}$$

and

$$\begin{aligned} & \sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right)\sin\left(\frac{2\pi qn}{N}\right) \\ = & \frac{1}{2}\sum_{n=0}^{N-1}\left[\cos\left(\frac{2\pi(p-q)n}{N}\right) - \cos\left(\frac{2\pi(p+q)n}{N}\right)\right] \\ = & \begin{cases} N/2, & p = q \neq N/2 \\ 0, & \textit{otherwise} \end{cases}. \end{aligned} \tag{6.45}$$

We have proven the following orthogonality relations

Theorem 6.2.

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi pn}{N}\right)\cos\left(\frac{2\pi qn}{N}\right) = \begin{cases} N/2, & p = q \neq N/2 \\ N, & p = q = N/2 \\ 0, & \textit{otherwise} \end{cases}. \tag{6.46}$$

$$\sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right)\cos\left(\frac{2\pi qn}{N}\right) = 0. \tag{6.47}$$

$$\sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right)\sin\left(\frac{2\pi qn}{N}\right) \begin{cases} N/2, & p = q \neq N/2 \\ 0, & \textit{otherwise} \end{cases}. \tag{6.48}$$

6.4.3 *The Discrete Fourier Coefficients*

THE DERIVATION OF THE COEFFICIENTS FOR THE DFT is now easily done using the discrete orthogonality from the last section. We start with the

expansion

$$y_n = \frac{1}{2}a_0 + \sum_{p=1}^{N/2}\left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right)\right],\ n = 0, \ldots, N-1. \tag{6.49}$$

We first sum over n:

$$\begin{aligned}
\sum_{n=0}^{N-1} y_n &= \sum_{n=0}^{N-1}\left(\frac{1}{2}a_0 + \sum_{p=1}^{N/2}\left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right)\right]\right) \\
&= \frac{1}{2}a_0 \sum_{n=0}^{N-1} 1 + \sum_{p=1}^{N/2}\left[a_p \sum_{n=0}^{N-1}\cos\left(\frac{2\pi pn}{N}\right) + b_p \sum_{n=0}^{N-1}\sin\left(\frac{2\pi pn}{N}\right)\right] \\
&= \frac{1}{2}a_0 N + \sum_{p=1}^{N/2}\left[a_p \cdot 0 + b_p \cdot 0\right] \\
&= \frac{1}{2}a_0 N. \qquad (6.50)
\end{aligned}$$

Therefore, we have $a_0 = \frac{2}{N}\sum_{n=0}^{N-1} y_n$

Now, we can multiply both sides of the expansion (6.77) by $\cos\left(\frac{2\pi qn}{N}\right)$ and sum over n. This gives

$$\begin{aligned}
&\sum_{n=0}^{N-1} y_n \cos\left(\frac{2\pi qn}{N}\right) \\
= &\sum_{n=0}^{N-1}\left(\frac{1}{2}a_0 + \sum_{p=1}^{N/2}\left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right)\right]\right)\cos\left(\frac{2\pi qn}{N}\right) \\
= &\frac{1}{2}a_0 \sum_{n=0}^{N-1}\cos\left(\frac{2\pi qn}{N}\right) \\
&+ \sum_{p=1}^{N/2} a_p \sum_{n=0}^{N-1}\cos\left(\frac{2\pi pn}{N}\right)\cos\left(\frac{2\pi qn}{N}\right) \\
&+ \sum_{p=1}^{N/2} b_p \sum_{n=0}^{N-1}\sin\left(\frac{2\pi pn}{N}\right)\cos\left(\frac{2\pi qn}{N}\right) \\
= &\begin{cases} \sum_{p=1}^{N/2}\left[a_p \frac{N}{2}\delta_{p,q} + b_p \cdot 0\right], & q \neq N/2, \\ \sum_{p=1}^{N/2}\left[a_p N \delta_{p,N/2} + b_p \cdot 0\right], & q = N/2, \end{cases} \\
= &\begin{cases} \frac{1}{2}a_q N, & q \neq N/2 \\ a_{N/2} N, & q = N/2 \end{cases}. \qquad (6.51)
\end{aligned}$$

So, we have found that

$$a_q = \frac{2}{N}\sum_{n=0}^{N-1} y_n \cos\left(\frac{2\pi qn}{N}\right), \quad q \neq \frac{N}{2}, \tag{6.52}$$

$$\begin{aligned}
a_{N/2} &= \frac{1}{N}\sum_{n=0}^{N-1} y_n \cos\left(\frac{2\pi n(N/2)}{N}\right) \\
&= \frac{1}{N}\sum_{n=0}^{N-1} y_n \cos(\pi n). \qquad (6.53)
\end{aligned}$$

Similarly, $\sum_{n=0}^{N-1} y_n \sin\left(\frac{2\pi qn}{N}\right) =$

$$
\begin{aligned}
&= \sum_{n=0}^{N-1}\left(\frac{1}{2}a_0 + \sum_{p=1}^{N/2}\left[a_p\cos\left(\frac{2\pi pn}{N}\right) + b_p\sin\left(\frac{2\pi pn}{N}\right)\right]\right)\sin\left(\frac{2\pi qn}{N}\right)\\
&= \frac{1}{2}a_0\sum_{n=0}^{N-1}\sin\left(\frac{2\pi qn}{N}\right)\\
&\quad + \sum_{p=1}^{N/2} a_p \sum_{n=0}^{N-1}\cos\left(\frac{2\pi pn}{N}\right)\sin\left(\frac{2\pi qn}{N}\right)\\
&\quad + \sum_{p=1}^{N/2} b_p \sum_{n=0}^{N-1}\sin\left(\frac{2\pi pn}{N}\right)\sin\left(\frac{2\pi qn}{N}\right)\\
&= \sum_{p=1}^{N/2}\left[a_p\cdot 0 + b_p\frac{N}{2}\delta_{p,q}\right]\\
&= \frac{1}{2}b_q N. \qquad (6.54)
\end{aligned}
$$

Finally, we have

$$b_q = \frac{2}{N}\sum_{n=0}^{N-1} y_n \sin\left(\frac{2\pi qn}{N}\right), \quad q = 1,\ldots,\frac{N}{2}-1. \qquad (6.55)$$

6.5 *The Discrete Exponential Transform*

The derivation of the coefficients for the trigonometric DFT was obtained in the last section using the discrete orthogonality. However, applications like MATLAB do not typically use the trigonometric version of DFT for spectral analysis. MATLAB instead uses a discrete Fourier exponential transform in the form of the Fast Fourier Transform (FFT)[1] . Its description in the help section does not involve sines and cosines. Namely, MATLAB defines the transform and inverse transform (by typing **help fft**) as

[1] The Fast Fourier Transform, or FFT, refers to an efficient algorithm for computing the discrete Fourier transform. It was originally developed by James Cooley and John Tukey in 1965 for computers based upon an algorithm invented by Gauß.

```
For length N input vector x, the DFT is a length N vector X,
with elements
        N
 X(k) = sum  x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <= k <= N.
       n=1
The inverse DFT (computed by IFFT) is given by
             N
x(n) = (1/N) sum  X(k)*exp( j*2*pi*(k-1)*(n-1)/N), 1 <= n <= N.
            k=1
```

It also provides in the new help system,

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

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

where $\omega_N = e^{-2\pi i/N}$ are the Nth roots of unity. You will note a few differences between these representations and the discrete Fourier transform in the last section. First of all, there are no trigonometric functions. Next, the sums do not have a zero index, a feature of indexing in MATLAB. Also, in the older definition, MATLAB uses a "j" and not an "i" for the imaginary unit.

In this section we will derive the discrete Fourier exponential transform in the form

$$F_k = \sum_{j=0}^{N-1} W^{jk} f_j. \tag{6.56}$$

where $W_N = e^{-2\pi i/N}$ and $k = 0, \ldots, N-1$. We will find the relationship between what MATLAB is computing and the discrete Fourier trigonometic series. This will also be useful as a preparation for a discussion of the FFT in the next chapter.

We will start with the DFT (Discrete Fourier Transform):

$$y_n = \frac{1}{2}a_0 + \sum_{p=1}^{N/2}\left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right)\right] \tag{6.57}$$

for $n = 0, 1, \ldots, N-1$.

We use Euler's formula to rewrite the trigonometric functions in terms of exponentials. The DFT formula can be written as

$$\begin{aligned} y_n &= \frac{1}{2}a_0 + \sum_{p=1}^{N/2}\left[a_p \frac{e^{2\pi ipn/N} + e^{-2\pi ipn/N}}{2} + b_p \frac{e^{2\pi ipn/N} - e^{-2\pi ipn/N}}{2i}\right] \\ &= \frac{1}{2}a_0 + \sum_{p=1}^{N/2}\left[\frac{1}{2}\left(a_p - ib_p\right)e^{2\pi ipn/N} + \frac{1}{2}\left(a_p + ib_p\right)e^{-2\pi ipn/N}\right]. \end{aligned} \tag{6.58}$$

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

$$y_n = C_0 + \sum_{p=1}^{N/2}\left[C_p e^{2\pi ipn/N} + \bar{C}_p e^{-2\pi ipn/N}\right], \quad n = 0, 1, \ldots, N-1. \tag{6.59}$$

The terms in the sums look similar. We can actually combine them into one form. Note that $e^{2\pi iN} = \cos(2\pi N) + i\sin(2\pi N) = 1$. Thus, we can write

$$e^{-2\pi ipn/N} = e^{-2\pi ipn/N}e^{-2\pi iN} = e^{2\pi i(N-p)n/N}$$

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

$$\sum_{p=1}^{N/2} \bar{C}_p e^{-2\pi ipn/N} = \sum_{p=1}^{N/2} \bar{C}_p e^{2\pi i(N-p)n/N} = \sum_{q=N/2}^{N-1} \bar{C}_{N-q} e^{2\pi iqn/N}.$$

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

$$y_n = \sum_{p=0}^{N-1} Y_p e^{2\pi ipn/N}, \quad n = 0, 1, \ldots, N-1, \tag{6.60}$$

where

$$Y_p = \begin{cases} \frac{a_0}{2}, & p = 0, \\ \frac{1}{2}(a_p - ib_p), & 0 < p < N/2, \\ a_{N/2}, & p = N/2, \\ \frac{1}{2}(a_{N-p} + ib_{N-p}), & N/2 < p < N. \end{cases} \tag{6.61}$$

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 $p \in (N/2, N)$. Namely, since

$$Y_{N-j} = \frac{1}{2}(a_j + ib_j) = \overline{Y}_j, \quad j = 1, \ldots, N-1,$$

$Re(Y_{N-j}) = ReY_j$ and $Im(Y_{N-j}) = -ImY_j$ for $j = 1, \ldots, N-1$.

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

$$\begin{aligned} C_p &= \frac{1}{2}(a_p - ib_p) \\ &= \frac{1}{N}\sum_{n=1}^{N} y_n \left[\cos\left(\frac{2\pi pn}{N}\right) - i\sin\left(\frac{2\pi pn}{N}\right)\right] \\ &= \frac{1}{N}\sum_{n=1}^{N} y_n e^{-2\pi ipn/N}. \end{aligned} \tag{6.62}$$

Thus,

$$Y_p = \frac{1}{N}\sum_{n=1}^{N} y_n e^{-2\pi ipn/N}, \quad 0 < p < \frac{N}{2} \tag{6.63}$$

and

$$\begin{aligned} Y_p &= \bar{C}_{N-p} \\ &= \frac{1}{N}\sum_{n=1}^{N} y_n e^{2\pi i(N-p)n/N}, \quad \frac{N}{2} < p < N \\ &= \frac{1}{N}\sum_{n=1}^{N} y_n e^{-2\pi ipn/N}. \end{aligned} \tag{6.64}$$

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

$$Y_p = \frac{1}{N}\sum_{n=1}^{N} y_n e^{-2\pi ipn/N}. \tag{6.65}$$

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

$$Y_{N/2} = a_{N/2}$$

$$
\begin{aligned}
&= \frac{1}{N}\sum_{n=1}^{N} y_n \cos n\pi \\
&= \frac{1}{N}\sum_{n=1}^{N} y_n \left[\cos n\pi - i \sin n\pi\right] \\
&= \frac{1}{N}\sum_{n=1}^{N} y_n e^{-2\pi i n(N/2)/N} \qquad (6.66)
\end{aligned}
$$

and

$$
\begin{aligned}
Y_0 &= \frac{1}{2}a_0 \\
&= \frac{1}{N}\sum_{n=1}^{N} y_n \\
&= \frac{1}{N}\sum_{n=1}^{N} y_n e^{2\pi i n(0)/N}. \qquad (6.67)
\end{aligned}
$$

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

$$
y_n = \sum_{p=0}^{N-1} Y_p e^{2\pi i p n/N}, \quad n = 1, \ldots, N, \qquad (6.68)
$$

$$
Y_p = \frac{1}{N}\sum_{n=1}^{N} y_n e^{-2\pi i p n/N}, \quad p = 0, 1, \ldots, N-1. \qquad (6.69)
$$

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

6.6 FFT: The Fast Fourier Transform

The usual computation of the discrete Fourier transform (DFT) is done using the Fast Fourier 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 $W = e^{-2\pi i/N}$. Note that $W^{N/2} = -1, \quad W^N = 1$, and $e^{2\pi i j k/N} = W^{jk}$. We can then write

$$
F_k = \sum_{j=0}^{N-1} W^{jk} f_j. \qquad (6.70)
$$

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

$$
\begin{aligned}
F_k &= \sum_{j=0}^{N-1} W^{jk} f_j \\
&= \sum_{j=0}^{N/2-1} W^{jk} f_j + \sum_{j=N/2}^{N-1} W^{jk} f_j \\
&= \sum_{j=0}^{N/2-1} W^{jk} f_j + \sum_{m=0}^{N/2-1} W^{k(m+N/2)} f_{m+N/2}, \quad \text{for } m = j - \frac{N}{2}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{j=0}^{N/2-1} \left[W^{jk} f_j + W^{k(j+N/2)} f_{j+N/2} \right] \\
&= \sum_{j=0}^{N/2-1} W^{jk} \left[f_j + (-1)^k f_{j+N/2} \right] \qquad (6.71)
\end{aligned}
$$

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 the initial sum, 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

$$
F_{2m} = \sum_{j=0}^{N/2-1} \left(W^{2m} \right)^j \left[f_j + f_{j+N/2} \right], \quad m = 0, \ldots \frac{N}{2} - 1. \qquad (6.72)
$$

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

$$
F_{2m+1} = \sum_{j=0}^{N/2-1} \left(W^{2m} \right)^j W^j \left[f_j - f_{j+N/2} \right], \quad m = 0, \ldots \frac{N}{2} - 1. \qquad (6.73)
$$

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

$$
\begin{aligned}
F_0 &= f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7. \\
F_1 &= f_0 + W f_1 + W^2 f_2 + W^3 f_3 + W^4 f_4 + W^5 f_5 + W^6 f_6 + W^7 f_7. \\
F_2 &= f_0 + W^2 f_1 + W^4 f_2 + W^6 f_3 + f_4 + W^2 f_5 + W^4 f_6 + W^6 f_7. \\
F_3 &= f_0 + W^3 f_1 + W^6 f_2 + W f_3 + W^4 f_4 + W^7 f_5 + W^2 f_6 + W^5 f_7. \\
F_4 &= f_0 + W^4 f_1 + f_2 + W^4 f_3 + f_4 + W^4 f_5 + f_6 + W^4 f_7. \\
F_5 &= f_0 + W^5 f_1 + W^2 f_2 + W^7 f_3 + W^4 f_4 + W f_5 + W^6 f_6 + W^3 f_7. \\
F_6 &= f_0 + W^6 f_1 + W^4 f_2 + W^2 f_3 + f_4 + W^6 f_5 + W^4 f_6 + W^2 f_7. \\
F_7 &= f_0 + W^7 f_1 + W^6 f_2 + W^5 f_3 + W^4 f_4 + W^3 f_5 + W^2 f_6 + W f_7.
\end{aligned} \qquad (6.74)
$$

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

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

$$
\begin{aligned}
&= g_0 - g_2 + (g_1 - g_3)W^2. \\
F_3 &= (f_0 - f_4) - (f_2 - f_6)W + (f_1 - f_5)WW^2 + (f_3 - f_7)WW^6 \\
&= g_4 - g_6 + g_5 W^2 + g_7 W^6. \\
F_4 &= (f_0 + f_4) + (f_1 + f_5) - (f_2 + f_6) - (f_3 + f_7) \\
&= g_0 + g_2 - g_1 - g_3. \\
F_5 &= (f_0 - f_4) + (f_2 - f_6)W + (f_1 - f_5)WW^4 + (f_3 - f_7)WW^4 \\
&= g_4 + g_6 + g_5 W^4 + g_7 W^4. \\
F_6 &= (f_0 + f_4) + (f_1 + f_5)\, W^6 - (f_2 + f_6) - (f_3 + f_7)\, W^6 \\
&= g_0 - g_2 + (g_1 - g_3)W^6. \\
F_7 &= (f_0 - f_4) - (f_2 - f_6)W + (f_1 - f_5)WW^6 + (f_3 - f_7)WW^2 \\
&= g_4 - g_6 + g_5 W^6 + g_7 W^2. \qquad (6.75)
\end{aligned}
$$

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

$$
\begin{aligned}
F_0 &= (g_0 + g_2) + (g_1 + g_3) \equiv h_0 + h_1. \\
F_1 &= (g_4 + g_6) + (g_5 + g_7) \equiv h_4 + h_5. \\
F_2 &= (g_0 - g_2) + (g_1 - g_3)W^2 \equiv h_2 + h_3. \\
F_3 &= (g_4 - g_6) + (g_5 - g_7)W^2 \equiv h_6 + h_7. \\
F_4 &= (g_0 + g_2) - (g_1 + g_3) = h_0 - h_1. \\
F_5 &= (g_4 + g_6) - (g_5 + g_7) = h_4 - h_5. \\
F_6 &= g_0 - g_2 - (g_1 - g_3)W^2 = h_2 - h_3. \\
F_7 &= g_4 - g_6 + g_5 W^6 + g_7 W^2 = h_6 - h_7. \qquad (6.76)
\end{aligned}
$$

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." The basic butterfly transformation is displayed in Figure 6.10. An 8 point FFT is shown in Figure 6.11.

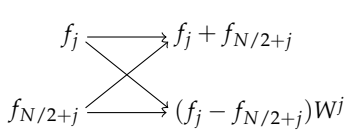

Figure 6.10: This is the basic FFT butterfly.

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

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

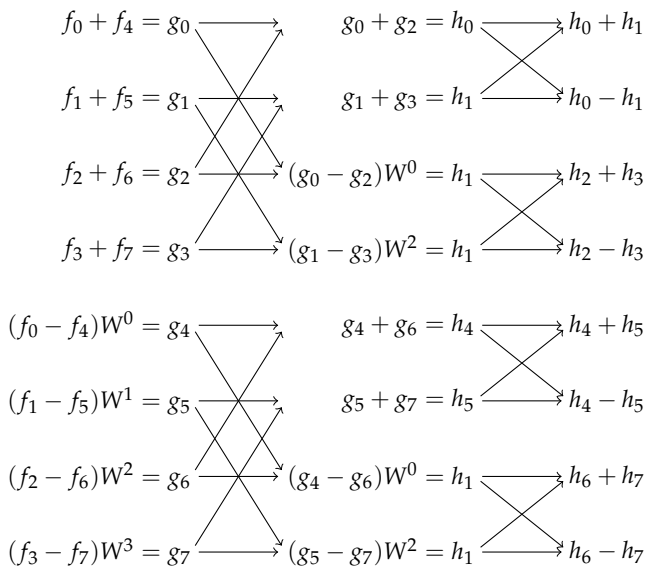

Figure 6.11: This is an 8 point FFT butterfly diagram.

Table 6.1: Output, desired order and binary representation for the Fourier Coefficients.

Output		**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

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.

6.7 Applications

IN THE LAST SECTION WE SAW that given a set of data, y_n, $n = 0, 1, \ldots, N-1$, that one can construct the corresponding discrete, finite Fourier series. The series is given by

$$y_n = \frac{1}{2}a_0 + \sum_{p=1}^{N/2}\left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right)\right],\ n = 0, \ldots, N-1. \tag{6.77}$$

and the Fourier coefficients were found as

$$\begin{aligned}
a_p &= \tfrac{2}{N}\sum_{n=0}^{N-1} y_n \cos(\tfrac{2\pi pn}{N}),\ p = 1, \ldots N/2-1\\
b_p &= \tfrac{2}{N}\sum_{n=0}^{N-1} y_n \sin(\tfrac{2\pi pn}{N}),\ p = 1, 2, \ldots N/2-1\\
a_0 &= \tfrac{1}{N}\sum_{n=0}^{N-1} y_n,\\
a_{N/2} &= \tfrac{1}{N}\sum_{n=0}^{N-1} y_n \cos n\pi,\\
b_0 &= b_{N/2} = 0
\end{aligned} \tag{6.78}$$

In this section we show how this is implemented using MATLAB.

Example 6.10. *Analysis of monthly mean surface temperatures.*

Consider the data[2] *of monthly mean surface temperatures at Amphitrite Point, Canada shown in Table 6.2. The temperature was recorded in ^{0}C and averaged for each month over a two year period. We would like to look for the frequency content of this time series.*

[2] This example is from *Data Analysis Methods in Physical Oceanography*, W. J. Emery and R.E. Thomson, Elsevier, 1997.

Month	1	2	3	4	5	6	7	8	9	10	11	12
1982	7.6	7.4	8.2	9.2	10.2	11.5	12.4	13.4	13.7	11.8	10.1	9.0
1983	8.9	9.5	10.6	11.4	12.9	12.7	13.9	14.2	13.5	11.4	10.9	8.1

Table 6.2: Monthly mean surface temperatures (^{0}C) at Amphitrite Point, Canada for 1982-1983.

In Figure 6.12 we plot the above data as circles. We then use the data to compute the Fourier coefficients. These coefficients are used in the discrete Fourier series an plotted on top of the data in red. We see that the reconstruction fits the data.

The implementations of DFT are done using MATLAB. We provide the code at the end of this section.

Example 6.11. *Determine the frequency content of $y(t) = \sin(10\pi t)$.*

Generally, we are interested in determining the frequency content of a signal. For example, we consider a pure note,

$$y(t) = \sin(10\pi t).$$

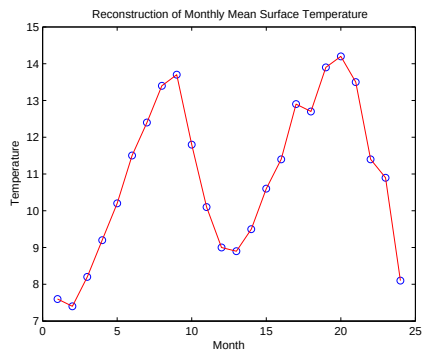

Figure 6.12: Plot and reconstruction of the monthly mean surface temperature data.

Sampling this signal with $N = 128$ points on the interval $[0, 5]$, we find the discrete Fourier coefficients as shown in Figure 6.13. Note the spike at the right place in the B plot. The others spikes are actually very small if you look at the scale on the plot of the A coefficients.

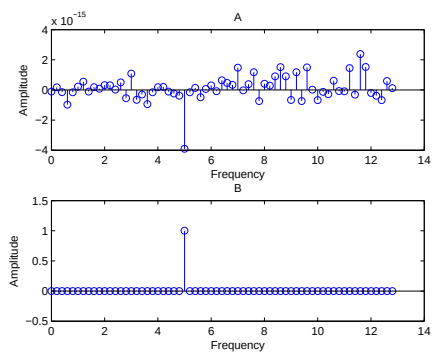

Figure 6.13: Computed discrete Fourier coefficients for $y(t) = \sin(10\pi t)$, with $N = 128$ points on the interval $[0, 5]$.

One can use these coefficients to reconstruct the signal. This is shown in Figure 6.14

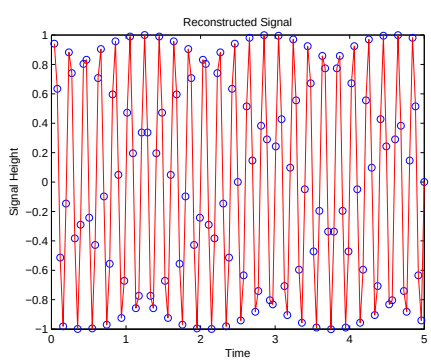

Figure 6.14: Reconstruction of $y(t) = \sin(10\pi t)$ from its Fourier coefficients.

Example 6.12. *Determine the frequency content of $y(t) = \sin(10\pi t) - \frac{1}{2}\cos(6\pi t)$.*

We can look at more interesting functions. For example, what if we add two pure notes together, such as

$$y(t) = \sin(10\pi t) - \frac{1}{2}\cos(6\pi t).$$

We see from Figure 6.15 that the implementation works. The Fourier coefficients for a slightly more complicated signal,

$$y(t) = e^{\alpha t} \sin(10\pi t)$$

for $\alpha = 0.1$*, is shown in Figure 6.16 and the corresponding reconstruction is shown in Figure 6.17. We will look into more interesting features in discrete signals later in the chapter.*

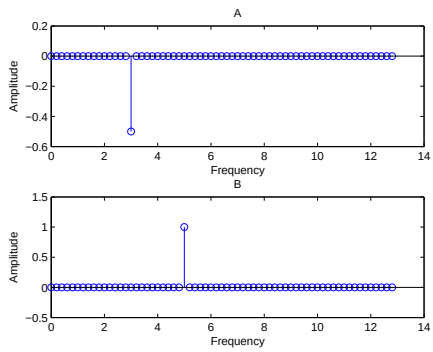

Figure 6.15: Computed discrete Fourier coefficients for $\sin(10\pi t) - \frac{1}{2}\cos(6\pi t)$ with $N = 128$ points on the interval $[0, 5]$.

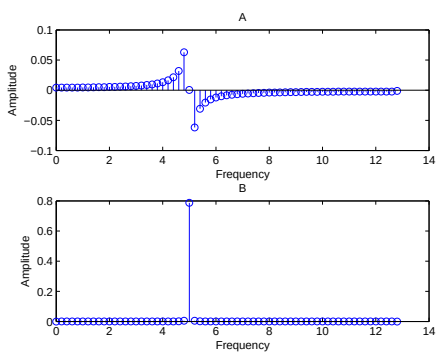

Figure 6.16: Computed discrete Fourier coefficients for $y(t) = e^{\alpha t} \sin(10\pi t)$ with $\alpha = 0.1$ and $N = 128$ points on the interval $[0, 5]$.

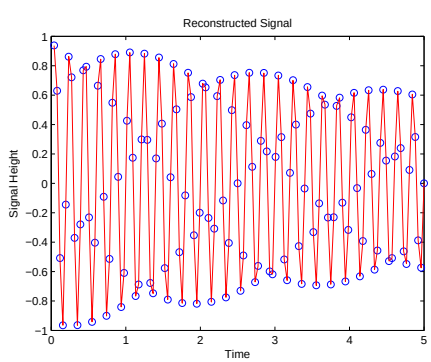

Figure 6.17: Reconstruction of $y(t) = e^{\alpha t} \sin(10\pi t)$ with $\alpha = 0.1$ from its Fourier coefficients.

6.8 MATLAB Implementation

Discrete Fourier Transforms and FFT are easily implemented in computer applications. In this section we describe the MALAB routines used in this course.

6.8.1 MATLAB for the Discrete Fourier Transform

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 or similar programs like GNU Octave. Sums can be done with matrix multiplication, as described in the next section. This eliminates the loops in the first program below and speeds up the computation for large data sets.

Direct Implementation

The following code was used to produce Figure 6.12. It shows a direct implementation using loops to compute the trigonometric DFT as developed in this chapter.

```
%
% 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
title('Reconstruction of Monthly
Mean Surface Temperature')
xlabel('Month')
ylabel('Temperature')
```

The next routine shows how we can determine the spectral content of a signal, given in this case by a function and not a measured time series. The output is the original data and reconstructed Fourier series in Figure 1, the trigonometric DFT coefficients in Figure 2, and the the power spectrum in Figure 3. This code will be referred to as **ftex.m**.

```
% ftex.m
% IMPLEMENTATION OF DFT USING TRIGONOMETRIC FORM
% N = Number of samples
% T = Record length in time
% y = Sampled signal
%
clear
N=128;
T=5;
dt=T/N;
t=(1:N)*dt;
f0=30;
y=sin(2*pi*f0*t);

% Compute arguments of trigonometric functions
for n=1:N
    for p=0:N/2
        Phi(p+1,n)=2*pi*p*n/N;
    end
end

% Compute Fourier Coefficients
for p=1:N/2+1
    A(p)=2/N*y*cos(Phi(p,:))';
    B(p)=2/N*y*sin(Phi(p,:))';
end
A(1)=2/N*sum(y);
```

```
A(N/2+1)=A(N/2+1)/2;
B(N/2+1)=0;

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

% Plot Data
figure(1)
plot(t,y,'o')

% Plot reconstruction over data
hold on
plot(t,ynew,'r')
xlabel('Time')
ylabel('Signal Height')
title('Reconstructed Signal')
hold off

% Compute Frequencies
n2=N/2;
f=(0:n2)/(n2*2*dt);

% Plot Fourier Coefficients
figure(2)
subplot(2,1,1)
stem(f,A)
xlabel('Frequency')
ylabel('Amplitude')
title('A')
subplot(2,1,2)
stem(f,B)
xlabel('Frequency')
ylabel('Amplitude')
title('B')

% Plot Fourier Spectrum
figure(3)
Power=sqrt(A.^2+B.^2);
stem(f,Power(1:n2+1))
```

```
xlabel('Frequency')
ylabel('Power')
title('Periodogram')

% Show Figure 1
figure(1)
```

Compact Implementation

The next implementation uses matrix products to eliminate the for loops in the previous code. The way this works is described in the next section.

```
%
% 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
title('Reconstruction of Monthly
Mean Surface Temperature')
xlabel('Month')
ylabel('Temperature')
```

6.8.2 *Matrix Operations for MATLAB*

THE BEAUTY OF USING PROGRAMS like MATLAB or GNU Octave 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

$$\begin{aligned} C_{ij} &= \textstyle\sum_{k=1}^{m} a_{ik} b_{kj}, \quad i = 1, \ldots, p, \quad j = 1, \ldots, q, \\ &= a_{i1} b_{1j} + a_{i2} b_{2j} + \ldots a_{im} b_{mj}. \end{aligned} \tag{6.79}$$

If we wanted to compute the sum $\sum\limits_{n=1}^{N} a_n b_n$, 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 $\sum\limits_{n=1}^{N} a_n b_n = \sum\limits_{n=1}^{N} a_{1n} b_{n1}$. In particular, if both **a** and **b** are row vectors, the sum in MATLAB is given

by $\mathbf{ab}'$, and if they are both row vectors, the sum is $\mathbf{a}'\mathbf{b}$. This notation is much easier to type.

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

$$b_p = \frac{2}{N}\sum_{n=1}^{N} y_n \sin(\tfrac{2\pi pn}{N}), \quad p = 0, \ldots, N/2 \tag{6.80}$$

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. Finding the Fourier coefficients becomes a simple matrix multiplication, ignoring the prefactor $\frac{2}{N}$. Thus, if we put the sampled data in a $1 \times N$ vector $\mathbf{Y}$ and put the sines in an $N \times (\frac{N}{2}+1)$ vector $\mathbf{S}$, 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.

6.8.3 *MATLAB Implementation of FFT*

In this section we provide implementations of the Fast Fourier Transform in MATLAB. The MATLAB code provided in Section 6.8.1 can be simplified by using the built-in **fft** function.

Much of the MATLAB code described here can be run directly in GNU Octave or easily mapped to other programming environments.

```
% fanal.m
%
% Analysis Using FFT
%
clear
n=128;
T=5;
dt=T/n;
f0=6.2;
f1=10;
y=sin(2*pi*f0*(1:n)*dt)+2*sin(2*pi*f1*(1:n)*dt);
% y=exp(-0*(1:n)*dt).*sin(2*pi*f0*(1:n)*dt);
Y=fft(y,n);
n2=n/2;
Power=Y.*conj(Y)/n^2;
f=(0:n2)/(n2*2*dt);
```

```
stem(f,2*Power(1:n2+1))
xlabel('Frequency')
ylabel('Power')
title('Periodogram')
figure(2)
subplot(2,1,1)
stem(f,real(Y(1:n2+1)))
xlabel('Frequency')
ylabel('Amplitude')
title('A')
subplot(2,1,2)
stem(f,imag(Y(1:n2+1)))
xlabel('Frequency')
ylabel('Amplitude')
title('B')
figure(1)
```

One can put this into a function which performs the FFT analysis. Below is the function **fanalf**, which can make the main program more compact.

```
function z=fanalf(y,T)
%
% FFT Analysis Function
%
% Enter Data in y and record length T
% Example:
%    n=128;
%    T=5;
%    dt=T/n;
%    f0=6.2;
%    f1=10;
%    fanal(sin(2*pi*f0*(1:n)*dt)+2*sin(2*pi*f1*(1:n)*dt));
% or
%    fanal(sin(2*pi*6.2*(1:128)/128*5),5);
n=length(y);
dt=T/n;
Y=fft(y,n);
n2=floor(n/2);
Power=Y.*conj(Y)/n^2;
f=(0:n2)/(n2*2*dt);
z=Power;
stem(f,2*Power(1:n2+1))
xlabel('Frequency')
ylabel('Power')
title('Periodogram')
```

Examples of the use for doing a spectral analysis of functions, data sets, and sound files are provided below.

This code snippet can be used with **fanalf** to analyze a given function.

```
% fanal2.m

clear
n=128;
T=5;
dt=T/n;
f0=6.2;
f1=10;
t=(1:n)*dt;
y=sin(2*pi*f0*t)+2*sin(2*pi*f1*t);
fanalf(y,T);
```

This code snippet can be used with **fanalf** to analyze a data set inserted in vector **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);
y=y-mean(y);
T=24;
dt=T/n;
fanalf(y,T);
```

One can also enter data stored in an ASCII file. This code show how this is done.

```
[year,y]=textread('sunspot.txt','%d %f');
n=length(y);
t=year-year(1);
y=y-mean(y);
T=t(n);
dt=T/n;
fanalf(y',T);
```

This code snippet can be used with **fanalf** to analyze a given sound stored in a wav file.

```
[y,NS,NBITS]=wavread('sound1.wav');
n=length(y);
T=n/NS
dt=T/n;
figure(1)
plot(dt*(1:n),y)
figure(2)
fanalf(y,T);
```

Problems

1. Find the imaginary part of the Fourier transform of the finite wave train $f(t) = \cos\omega_0 t$, $t \in [0, T]$ in Example . For $\omega_0 = \pi, 5, frm{-}e\pi, 10$, plot the real, imaginary, and modulus of the Fourier transform.

2. Recall that $\text{III}(ax) = \frac{1}{|a|}\text{comb}_{\frac{1}{a}}(x)$. Find a similar relation for $\text{comb}_T(at)$ in terms of the Shah function.

3. Write the following sums in the form $a\text{comb}_b(t)$ for some constants a and b.

a. $\text{comb}_T(\alpha t)$.

b. $\text{comb}_T(t+k)$, k and integer.

4. Evaluate $\mathcal{F}^{-1}[\text{comb}_\Omega(\omega)]$.

5. Evaluate the following sums:

a. $\sum_{n=1}^{8} e^{\pi i(n/4)-2}$

b. $\sum_{n=-7}^{8} \sin\left(\frac{\pi pn}{8}\right)\sin\left(\frac{\pi qn}{8}\right)$

6. Prove

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

7. Compute a table for the trigonometric discrete Fourier transform for the following and sketch A_k, B_k and $A_k^2 + B_k^2$.

a. $y_n = n^2$, $n \in [0, 15]$.

b. $y_n = \cos n$, $n \in [0, 15]$.

8. Here you will prove a shift theorem for the discrete exponential Fourier transform: The transform of y_k is given as $Y_j = \frac{1}{N}\sum_{k=1}^{N} y_k e^{-2\pi ijk/N}$, $j = 0, 1, \ldots N-1$. Show that for fixed $n \in [0, N-1]$ the discrete transform of y_{k-n} is $Y_j e^{-2\pi ijn/N}$, $j \in [0, N-1]$.

9. Consider the finite wave train

$$f(t) = \begin{cases} 2\sin 4t, & 0 \leq t \leq \pi \\ 0, & \text{otherwise.} \end{cases}$$

a. Plot this function.

b. Find the Fourier transform of $f(t)$.

c. Find the Fourier coefficients in the series expansion

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos 2nt + b_n \sin 2nt.$$

10. A plot of $\hat{f}(\omega)$ from the last problem is shown in Figure 7.72.

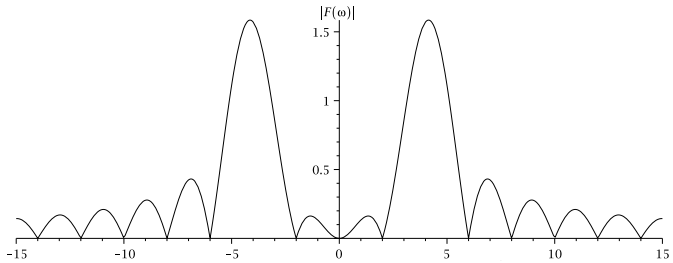

Figure 6.18: Fourier transform of the wave train in Problems 9-10.

a. What do the main peaks tell you about $f(t)$?

b. Use this plot to explain how the Fourier coefficients are related to the Fourier transform.

c. How would this plot appear if $f(t)$ were nonzero on a larger time interval?

11. Consider the sampled function in Figure 6.20. For each case compute the discrete Fourier transform.

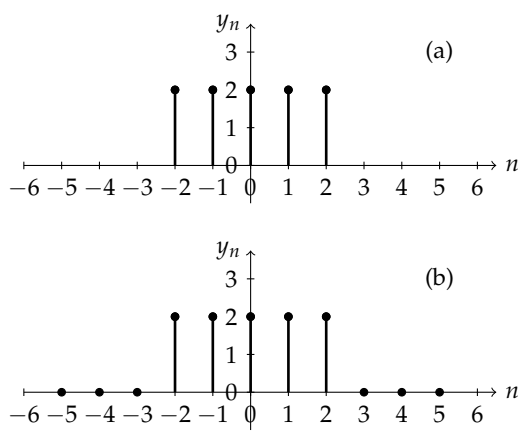

Figure 6.19: Figure for Problem 12.

12. Consider the sampled function in Figure 6.20. For each case compute the discrete Fourier transform.

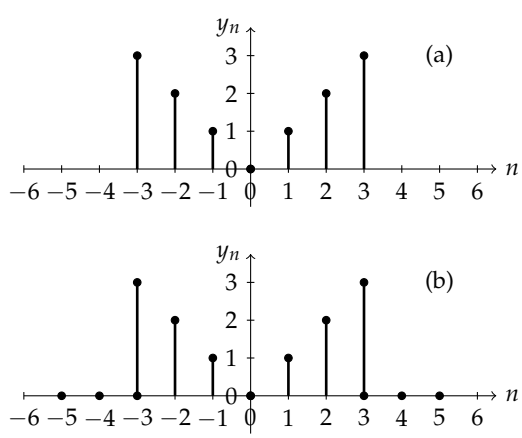

Figure 6.20: Figure for Problem 12.

13. The goal of this problem is to use MATLAB to investigate the use of the Discrete Fourier Transform (DFT) for the spectral analysis of time series. Running the m-files is done by typing the filename without the extension (.m) after the prompt (») in the Command Window, which is the middle panel of the MATLAB program. Note, most of the code provided can be run in GNU Octave.

1. Analysis of simple functions.
   a. File **ftex.m** - See the MATLAB section for the code.
      This file is a MATLAB program implementing the discrete Fourier transform using trigonometric functions like that derived in the text. The input is a function, sometimes with different frequencies. The output is a plot of the data points and the function fit, the Fourier coefficients and the periodogram giving the power spectrum.
      i. Save the file **ftex.m** in your working directory under MATLAB.
      ii. View the file by entering edit **ftex**. Note how the first function is defined by the variable **y**.
      iii. Run the file by typing **ftex** in MATLAB's Command window.
      iv. Change the parameters in **ftex.m**, remembering the original ones. In particular, change the number of points, **N**, (keeping them even), the frequency, **f0**, and the record length, **L**. Note the effects. If you get an error, enter clear and try again. Always save the m-file after making any changes before running ftex.
      v. Reset the parameters to the original values. What happens for frequencies of **f0** $= 5$, 5.5 and 5.6? Is this what you expected?
      vi. Repeat the last set of frequencies for double the record length, **T**. Is there a change?
      vii. Reset the parameters. Put in frequencies of **f0** $= 20, 30, 40$. What frequencies are present in the periodogram? Is this what you expected?
   b. Look at sums of several trigonometric functions.
      i. Reset the parameters.
      ii. Change the function in **y** to `y=sin(2*pi*f0*t)+sin(2*pi*f1*t);` and add a line defining **f1**. Start with **f1** $= 3$; and look at several other values for the two frequencies. Try different amplitudes; for example, `3*sin(2*pi*f0*t)` has an amplitude of 3. Record your observations.
      iii. Change one of the sines to a cosine. What is the effect? What happens when the sine and cosine terms have the same frequency?

c. Investigate non-sinusoidal functions.

   i. Investigate the following functions:

      1. `y=t;`
      2. `y=t.^2;`
      3. `y=sin(2*pi*f0*(t-T/5))./(t-T/5);` What is this function?
      4. `y(1,1:M)=ones(1,M); y(1,M+1:N)=zeros(1,N-M);` Start with $M = N/2$; What is this function? How are the last two problems related? Do they relate to anything from earlier class lectures? What effect results from changing **M**?
      5. Try multiplying the function in 4 by a simple sinusoid; for example, add the line `y=sin(2*pi*f0*t).*y`, for $M = N/2$. How does this affect what you had gotten for the sinusoid without multiplication?

2. Use the FFT function.

   In MATLAB there is a built in set of functions, **fft** and **ifft** for the computation of the Discrete Exponential Transform and its inverse using the Fast Fourier Transform (FFT). The files needed to do this are **fanal.m** or **fanal2.m** and **fanalf.m**. [See Section 6.8.3 for the code.] Put these codes into the MATLAB editor and see what they look like. Note that **fanal.m** was split into the two files **fanal2.m** and **fanalf.m**. This will allow us to be confident when we later create new data files and then using fanalf.m. Test this set of functions for simple sine functions to see that you get results similar to Part 1. First run **fanal** and then **fanal2**. Is there any difference between these last two approaches?

3. Analysis of data sets.

   One often does not have a function to analyze. Some measurements are made over time of a certain quantity. This is called a time series. It could be a set of data describing things like the stock market fluctuations, sunspots activity, ocean wave heights, etc. Large sets of data can be read into y and small sets can be input as vectors. We will look at how this can be done. After the data is entered, one can analyze the time series to look for any periodic behavior, or its frequency content. In the two cases below, make sure you look at the original time series using **plot(y,âĂŹ+âĂŹ)**.

   a. Ocean Waves

      In this example the data consists of monthly mean sea surface temperatures (^{0}C) at one point over a 2 year period. The temperatures are placed in **y** as a row vector. Note how the data is continued to a second line through the use of ellipsis. Also, one typically subtracts the average from the data. What affect should this have on the spectrum? Copy the following code into a new m-file called **fdata.m** and run **fdata**. Determine the dominant periods in the monthly mean sea surface temperature.

```
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);
y=y-mean(y);
T=24;
dt=T/n;
fanalf(y,T);
```

b. Sunspots

Sunspot data was exported to a text file. Download the file sunspot.txt. You can open the data in Notepad and see the two column format. The times are given as years (like 1850). This example shows how a time series can be read and analyzed. From your spectrum, determine the major period of sunspot activity. Note that we first subtracted the average so as not to have a spike at zero frequency. Copy and paste into the editor and save as **fanaltxt.m**. Note: Copy and Paste of single quotes often does not work correctly. Retype the single quotes after copying.

```
[year,y]=textread('sunspot.txt','%d %f');
n=length(y);
t=year-year(1);
y=y-mean(y);
T=t(n);
dt=T/n;
fanalf(y',T);
```

4. Analysis of sounds. Sounds can be input into MATLAB. You can create your own sounds in MATLAB or sound editing programs like Audacity or Goldwave to create audio files. These files can be input into MATLAB for analysis.

Save the following sample code as **fanalwav.m** and save the first sound file. This code shows how one can read in a WAV file. There will be two plots, the first showing the wave profile and the second giving the spectrum. Try some of the other wav files and report your findings. Note: Copy and Paste of single quotes often does not work correctly. Retype the single quotes after copying.

```
[y,NS,NBITS]=wavread('sound1.wav');
n=length(y);
T=n/NS
dt=T/n;
figure(1)
plot(dt*(1:n),y)
figure(2)
fanalf(y,T);
```

Several wav files are be provided for you to analyze. If you are able to hear the sounds, you can run them from MATLAB by typing `sound(y,NS)`. In fact, you can even create your own sounds based upon simple functions and save them as wav files. For example, try the following code: Note: Copy and Paste of single quotes often does not work correctly. Retype the single quotes after copying.

```
smp=11025;
t=(1:2000)/smp;
y=0.75*sin(2*pi*440*t);
sound(y,smp,8);
wavwrite(y,smp,8, 'myfile.wav');
```

Try some other functions, using several frequencies. If you get a clipping error, then reduce the amplitudes you are using.