6.2 Fourier Transform

0J

We begin with the usual loading of modules.

```python
import numpy as np # for numerics
import sympy as sp # for symbolics
import matplotlib.pyplot as plt # for plots!
```

Let's consider a periodic function f with period T (`T`). Each period, the function has a triangular pulse of width δ (`pulse_width`) and height $\delta/2$.

```python
period = 15 # period
pulse_width = 2 # pulse width
```

First, we plot the function f in the time domain. Let's begin by defining f.

```python
def pulse_train(t,T,pulse_width):
  f = lambda x:pulse_width/2-abs(x) # pulse
  tm = np.mod(t,T)
  if tm <= pulse_width/2:
    return f(tm)
  elif tm >= T-pulse_width/2:
    return f(-(tm-T))
  else:
    return 0
```

Now, we develop a numerical array in time to plot f.

```python
N = 151 # number of points to plot
tpp = np.linspace(-period/2,5*period/2,N) # time values
fpp = np.array(np.zeros(tpp.shape))
for i,t_now in enumerate(tpp):
  fpp[i] = pulse_train(t_now,period,pulse_width)
```

Now we plot.

```python
fig, ax = plt.subplots()
ax.plot(tpp,fpp,'b-',linewidth=2) # plot
plt.xlabel('time (s)')
plt.xlim([-period/2,3*period/2])
plt.xticks(
  [0,period],
  [0,'$T='+str(period)+'$ s']
)
plt.yticks([0,pulse_width/2],['0','$\delta/2$'])
plt.draw()
```

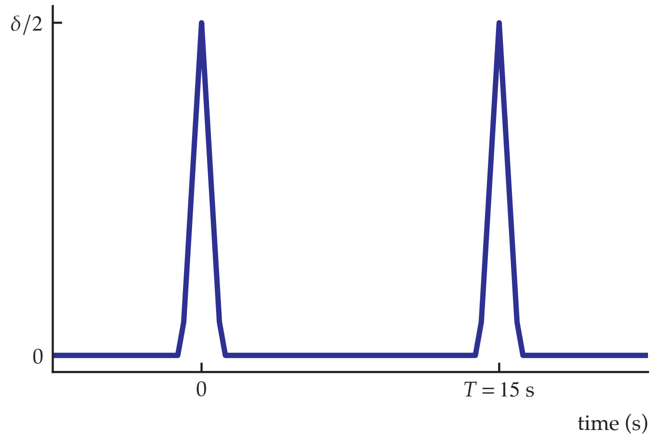

Figure 6.2. Triangle pulse train with period T and pulse width δ.

Consider the following argument. Just as a Fourier series is a frequency domain representation of a periodic signal, a Fourier transform is a frequency domain representation of an *aperiodic* signal (we will rigorously define it in a moment). The Fourier series components will have an analog, then, in the Fourier transform. Recall that they can be computed by integrating over a period of the signal. If we increase that period infinitely, the function is effectively aperiodic. The result (within a scaling factor) will be the Fourier transform analog of the Fourier series components.

Let us approach this understanding by actually computing the Fourier series components for increasing period T using definition 6.1. We'll use sympy to compute the Fourier series cosine and sine components a_n and b_n for component n (n) and period T (T).

```
x, a_0, a_n, b_n = sp.symbols('x, a_0, a_n, b_n', real=True)
delta, T = sp.symbols('delta, T', positive=True)
n = sp.symbols('n', nonnegative=True)
an = sp.integrate(
  2/T*(delta/2-sp.Abs(x))*sp.cos(2*sp.pi*n/T*x),
  (x,-delta/2, delta/2) # otherwise zero
).simplify()
bn = 2/T*sp.integrate(
  (delta/2-sp.Abs(x))*sp.sin(2*sp.pi*n/T*x),
  (x, -delta/2, delta/2) # otherwise zero
).simplify()
print(sp.Eq(a_n,an), sp.Eq(b_n,bn))
```

```
Eq(a_n, Piecewise((T*(1 - cos(pi*delta*n/T))/(pi**2*n**2), n > 0),
 ↪  (delta**2/(2*T), True))) Eq(b_n, 0)
```

Furthermore, let us compute the *harmonic amplitude* (`f_harmonic_amplitude`):

$$C_n = \sqrt{a_n^2 + b_n^2} \tag{6.2}$$

which we have also scaled by a factor T/δ in order to plot it with a convenient scale.

```
C_n = sp.symbols('C_n', positive=True)
cn = sp.sqrt(an**2+bn**2)
print(sp.Eq(C_n, cn))
```

```
Eq(C_n, Piecewise((T*Abs(cos(pi*delta*n/T) - 1)/(pi**2*n**2), n > 0),
 ↪  (delta**2/(2*T), True)))
```

Now we lambdify the symbolic expression for a numpy function.

```
cn_f = sp.lambdify((n, T, delta), cn)
```

Now we can plot. Write a function to plot pulses in the time domain with the corresponding frequency spectrum.

```
def plot_pulses_and_spectrum(T, pulse_width, omega_max):
  n_max = round(omega_max*T/(2*np.pi)) # max harmonic
  n_a = np.linspace(0,n_max,n_max+1)
  omega = 2*np.pi*n_a/T
  fig, ax = plt.subplots(1, 2)
  plt.sca(ax[0])
  for i in range(0, 3):
    tpp = np.linspace(-T/2, 5*T/2,N)
    fpp = np.array(np.zeros(tpp.shape))
    for i,t_now in enumerate(tpp):
      fpp[i] = pulse_train(t_now, T, pulse_width)
    plt.plot(tpp, fpp, 'b-', linewidth=2)
  plt.xlim([-T/2, 3*T/2])
  plt.xticks([0, T], [0, '$T='+str(T)+'$ s'])
  plt.yticks([0, pulse_width/2], ['0', '$\delta/2$'])
  plt.xlabel('time (s)')
  plt.sca(ax[1])
  plt.stem(
    omega, cn_f(n_a, T, pulse_width)*T/pulse_width, 'bo-'
  )
  plt.xlim([0, omega_max])
  plt.ylim([0, 1.1])
  plt.xlabel('Frequency $\omega$ (rad/s)')
  plt.ylabel('$C_n T/\delta$')
  return fig
```

Now we plot the pulses and their spectra for $T \in [5, 15, 25]$ rad/s and $\delta = 2$.

```
omega_max = 12  # Maximum frequency to plot
fig = plot_pulses_and_spectrum(5, pulse_width, omega_max)
plt.draw()
```

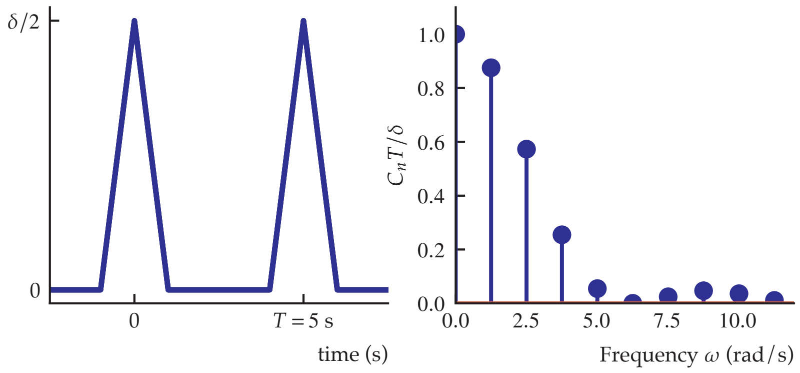

Figure 6.3. Triangle pulse train with period T and pulse width δ and its Fourier series components for $T = 5$ s.

```
fig = plot_pulses_and_spectrum(15, pulse_width, omega_max)
plt.draw()
```

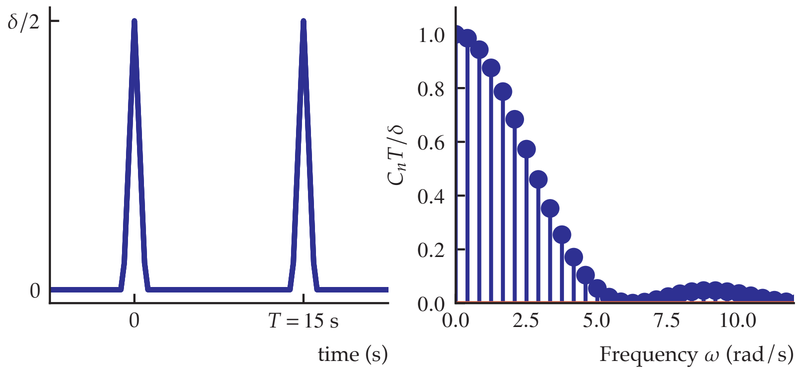

Figure 6.4. Triangle pulse train with period T and pulse width δ and its Fourier series components for $T = 15$ s.

```
fig = plot_pulses_and_spectrum(25, pulse_width, omega_max)
plt.draw()
```

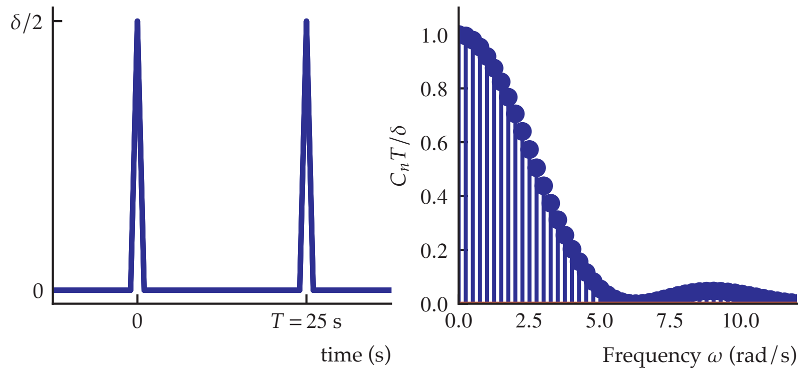

Figure 6.5. Triangle pulse train with period T and pulse width δ and its Fourier series components for $T = 25$ s.

The line spectra are shown in the right-hand columns of the plots above. Note that with our chosen scaling, as T increases, the line spectra reveal a distinct waveform.

Let F be the continuous function of angular frequency ω

$$F(\omega) = \frac{\delta}{2} \cdot \frac{\sin^2(\omega\delta/4)}{(\omega\delta/4)^2}. \tag{6.3}$$

First, we plot it.

```python
def F(w):
  return pulse_width/2*np.sin(w*pulse_width/4)**2 / \
    (w*pulse_width/4)**2
N = 201 # number of points to plot
wpp = np.linspace(0.0001, omega_max,N)
Fpp = []
for i in range(0,N):
    Fpp.append(F(wpp[i])) # build array of function values
fig, ax = plt.subplots()
plt.plot(wpp, Fpp, 'b-', linewidth=2) # plot
plt.xlim([0, omega_max])
plt.yticks([0, pulse_width/2],['0','$\delta/2$'])
plt.xlabel('Frequency $\omega$ (rad/s)')
plt.ylabel('$F(\omega)$')
plt.show()
```

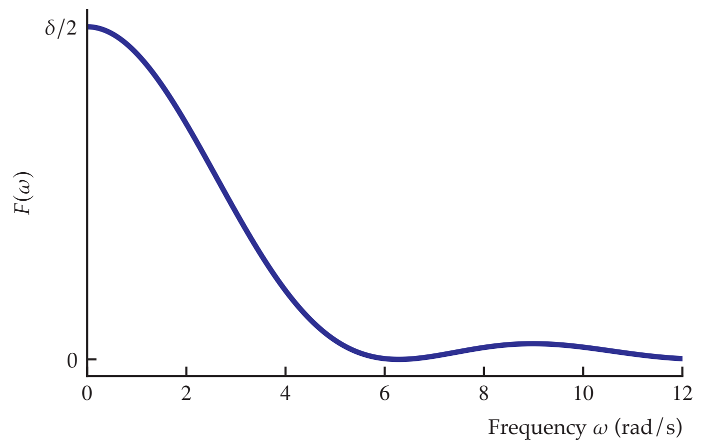

Figure 6.6. Continuous function $F(\omega)$.

The plot of F is clearly emerging from the preceding line spectra as the period T increases.

Now we are ready to define the Fourier transform and its inverse. We will define the Fourier transform in two ways: as a trigonometric transform and as a complex transform. We begin with the trigonometric transform and its inverse.

Definition 6.4: Fourier Transform (Trigonometric)

Fourier transform (analysis):

$$A(\omega) = \int_{-\infty}^{\infty} y(t)\cos(\omega t)dt \tag{6.4}$$

$$B(\omega) = \int_{-\infty}^{\infty} y(t)\sin(\omega t)dt. \tag{6.5}$$

Inverse Fourier transform (synthesis):

$$y(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} A(\omega)\cos(\omega t)d\omega + \frac{1}{2\pi}\int_{-\infty}^{\infty} B(\omega)\sin(\omega t)d\omega. \tag{6.6}$$

The Fourier transform is a generalization of the Fourier series to aperiodic functions (i.e., functions with infinite period). The complex form of the Fourier transform is more convenient for analysis and computation, as we will see.

Definition 6.5: Fourier Transform (Complex)

Fourier transform $\mathcal{F}$ (analysis):

$$\mathcal{F}(y(t)) = Y(\omega) = \int_{-\infty}^{\infty} y(t) e^{-j\omega t} dt. \tag{6.7}$$

Inverse Fourier transform $\mathcal{F}^{-1}$ (synthesis):

$$\mathcal{F}^{-1}(Y(\omega)) = y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} Y(\omega) e^{j\omega t} d\omega. \tag{6.8}$$

So now we have defined the Fourier transform. There are many applications, including solving differential equations and *frequency domain* representations—called *spectra*—of *time domain* functions.

There is a striking similarity between the Fourier transform and the Laplace transform, with which you are already acquainted. In fact, the Fourier transform is a special case of a Laplace transform with Laplace transform variable $s = j\omega$ instead of having some real component. Both transforms convert differential equations to algebraic equations, which can be solved and inversely transformed to find time-domain solutions. The Laplace transform is especially important to use when an input function to a differential equation is not absolutely integrable and the Fourier transform is undefined (for example, our definition will yield a transform for neither the unit step nor the unit ramp functions). However, the Laplace transform is also preferred for *initial value problems* due to its convenient way of handling them. The two transforms are equally useful for solving steady state problems. Although the Laplace transform has many advantages, for spectral considerations, the Fourier transform is the only game in town.

A table of Fourier transforms and their properties can be found in appendix B.2.

Example 6.2

Consider the aperiodic signal $y(t) = u_s(t)e^{-at}$ with u_s the unit step function and $a > 0$. The signal is plotted below. Derive the complex frequency spectrum and plot its magnitude and phase.

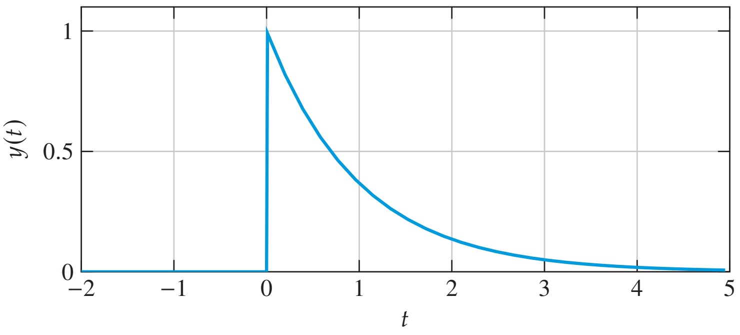

Figure 6.7. An aperiodic signal.

The signal is aperiodic, so the Fourier transform can be computed from equation (6.7):

$$
\begin{aligned}
Y(\omega) &= \int_{-\infty}^{\infty} y(t)e^{j\omega t}\,dt \\
&= \int_{-\infty}^{\infty} u_s(t)e^{-at}e^{j\omega t}\,dt && \text{(def. of } y\text{)} \\
&= \int_{0}^{\infty} e^{-at}e^{j\omega t}\,dt && (u_s \text{ effect}) \\
&= \int_{0}^{\infty} e^{(-a+j\omega)t}\,dt && \text{(multiply)} \\
&= \frac{1}{-a+j\omega}e^{(-a+j\omega)t}\Big|_{0}^{\infty}\,dt && \text{(antiderivative)} \\
&= \frac{1}{-a+j\omega}\left(\lim_{t\to\infty} e^{(-a+j\omega)t} - e^{0}\right) && \text{(evaluate)} \\
&= \frac{1}{-a+j\omega}\left(\lim_{t\to\infty} e^{-at}e^{j\omega t} - 1\right) && \text{(arrange)} \\
&= \frac{1}{-a+j\omega}\left((0)(\text{complex with mag} \le 1) - 1\right) && \text{(limit)} \\
&= \frac{-1}{-a+j\omega} && \text{(consequence)} \\
&= \frac{1}{a-j\omega} \\
&= \frac{a+j\omega}{a+j\omega}\cdot\frac{1}{a-j\omega} && \text{(rationalize)}
\end{aligned}
$$

$$= \frac{a + j\omega}{a^2 + \omega^2}.$$

The magnitude and phase of this complex function are straightforward to compute:

$$\begin{aligned} |Y(\omega)| &= \sqrt{\Re(Y(\omega))^2 + \Im(Y(\omega))^2} \\ &= \frac{1}{a^2 + \omega^2}\sqrt{a^2 + \omega^2} \\ &= \frac{1}{\sqrt{a^2 + \omega^2}} \\ \angle Y(\omega) &= \arctan(\omega / a). \end{aligned}$$

Now we can plot these functions of ω. Setting $a = 1$ (arbitrarily), we obtain the plots of figure 6.8.

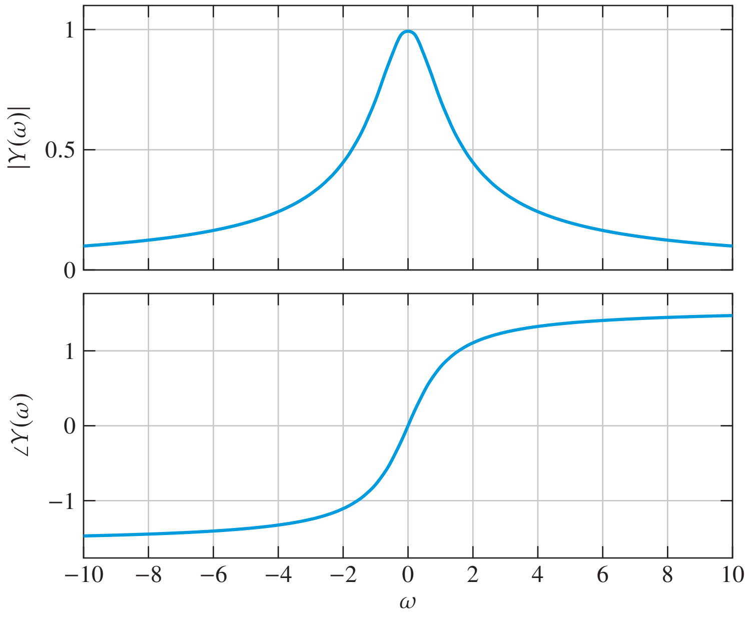

Figure 6.8. The magnitude and phase of the Fourier transform.

6.3 Generalized Fourier Series and Orthogonality

UJ

Let $f:\mathbb{R}\to\mathbb{C}$, $g:\mathbb{R}\to\mathbb{C}$, and $w:\mathbb{R}\to\mathbb{C}$ be complex functions. For square-integrable[2] f, g, and w, the **inner product** of f and g with **weight function** w over the interval $[a,b]\subseteq\mathbb{R}$ is[3]

$$\langle f, g\rangle_w = \int_a^b f(x)\overline{g}(x)w(x)\,\mathrm{d}x$$

where $\overline{g}$ denotes the complex conjugate of g. The inner product of functions can be considered analogous to the inner (or dot) product of vectors.

The fourier series components can be found by a special property of the sin and cos functions called **orthogonality**. In general, functions f and g from above are *orthogonal* over the interval $[a,b]$ iff

$$\langle f, g\rangle_w = 0$$

for weight function w. Similar to how a set of orthogonal vectors can be a basis for a vector space, a set of orthogonal functions can be a **basis** for a **function space**: a vector space of functions from one set to another (with certain caveats).

In addition to some sets of sinusoids, there are several other important sets of functions that are orthogonal. For instance, sets of **legendre polynomials** (Kreyszig 2011; § 5.2) and **bessel functions** (§ 5.4) are orthogonal.

As with sinusoids, the orthogonality of some sets of functions allows us to compute their series components. Let functions $f_0, f_1, \cdots$ be orthogonal with respect to weight function w on interval $[a,b]$ and let $\alpha_0, \alpha_1, \cdots$ be real constants. A **generalized fourier series** is (§ 11.6)

$$f(x) = \sum_{m=0}^{\infty} \alpha_m f_m(x)$$

and represents a function f as a convergent series. It can be shown that the **Fourier components** α_m can be computed from

$$\alpha_m = \frac{\langle f, f_m\rangle_w}{\langle f_m, f_m\rangle_w}.$$

In keeping with our previous terminology for fourier series, section 6.3 and section 6.3 are called general fourier **synthesis** and **analysis**, respectively.

For the aforementioned legendre and bessel functions, the generalized fourier series are called **fourier-legendre** and **fourier-bessel series** (§ 11.6). These and

2. A function f is square-integrable if $\int_{-\infty}^{\infty} |f(x)|^2\,\mathrm{d}x < \infty$.

3. This definition of the inner product can be extended to functions on $\mathbb{R}^2$ and $\mathbb{R}^3$ domains using double- and triple-integration. See (Schey 2005; p. 261).