four.transform Fourier transform

We begin with the usual loading of modules.

```
import numpy as np # for numerics
import sympy as sp # for symbolics
import matplotlib.pyplot as plt # for plots!
from IPython.display import display, Markdown, Latex
```

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 δ/2.

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

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

```
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.

```
N = 201 # 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)
```

```
p = plt.figure(1)
plt.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.show() # display here
```

1. Python code in this section was generated from a Jupyter notebook named `fourier_series_to_transform.ipynb` with a `python3` kernel.

For $\delta = 2$ and $T \in [5, 15, 25]$, the left-hand column of Fig. transform.1 shows two triangle pulses for each period T.

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 ??. 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`).

```
sp.var('x,a_0,a_n,b_n',real=True)
sp.var('delta,T',positive=True)
sp.var('n',nonnegative=True)
# a0 = 2/T*sp.integrate(
#   (delta/2-sp.Abs(x)),
#   (x,-delta/2,delta/2) # otherwise zero
# ).simplify()
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()
display(sp.Eq(a_n,an),sp.Eq(b_n,bn))
```

$$a_n = \begin{cases} \frac{T\left(1-\cos\left(\frac{\pi\delta n}{T}\right)\right)}{\pi^2 n^2} & \text{for } n \neq 0 \\ \frac{\delta^2}{2T} & \text{otherwise} \end{cases}$$

$$b_n = 0$$

Furthermore, let us compute the harmonic amplitude

(f_harmonic_amplitude):

$$C_n = \sqrt{a_n^2 + b_n^2} \tag{1}$$

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

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

$$C_n = \begin{cases} \frac{T\left|\cos\left(\frac{\pi\delta n}{T}\right)-1\right|}{\pi^2 n^2} & \text{for } n \neq 0 \\ \frac{\delta^2}{2T} & \text{otherwise} \end{cases}$$

Now we lambdify the symbolic expression for a numpy function.

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

Now we can plot.

```
omega_max = 12 # rad/s max frequency in line spectrum
n_max = round(omega_max*period/(2*np.pi)) # max harmonic
n_a = np.linspace(0,n_max,n_max+1)
omega = 2*np.pi*n_a/period
p = plt.figure(2)
markerline, stemlines, baseline = plt.stem(
    omega, period/pulse_width*cn_f(n_a,period,pulse_width),
    linefmt='b-', markerfmt='bo', basefmt='r-',
    use_line_collection=True,
)
plt.xlabel('frequency $\omega$ (rad/s)')
plt.xlim([0,omega_max])
plt.ylim([0,pulse_width/2])
plt.yticks([0,pulse_width/2],['0','$\delta/2$'])
plt.show() # show here
```

The line spectra are shown in the right-hand column of Fig. transform.1. 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

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

First, we plot it.

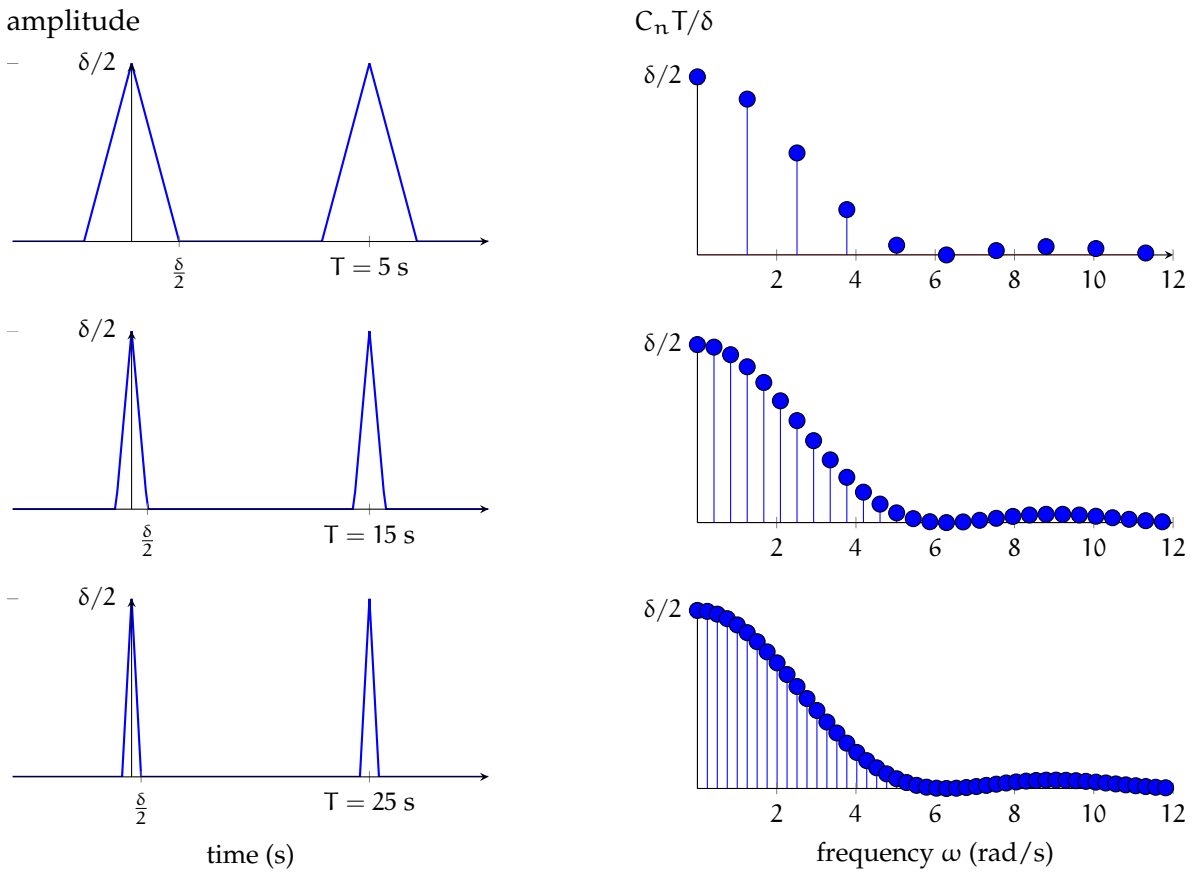

Figure transform.1: triangle pulse trains (left column) with longer periods, descending, and their corresponding line spectra (right column), scaled for convenient comparison.

```
F = lambda w: pulse_width/2* \
    np.sin(w*pulse_width/(2*2))**2/ \
    (w*pulse_width/(2*2))**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 value
axes = plt.figure(3)
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()
```

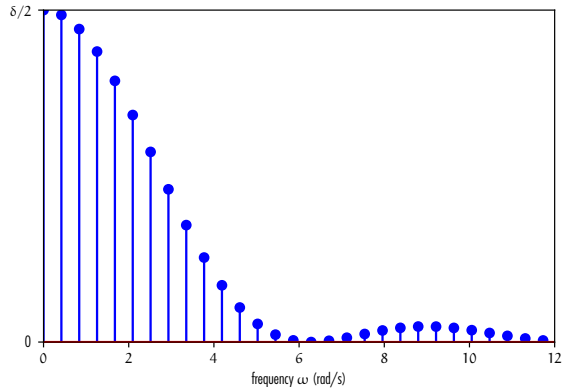

Figure transform.2: $F(\omega)$, our mysterious Fourier series amplitude analog.

Let's consider the plot in Fig. transform.2 of F. It's obviously the function emerging in

Fig. transform.1 from increasing the period of our pulse train.

Now we are ready to define the Fourier transform and its inverse.

Definition four.4: Fourier transforms: trigonometric form

Fourier transform (analysis):

$$A(\omega) = \int_{-\infty}^{\infty} y(t)\cos(\omega t)dt \tag{3}$$

$$B(\omega) = \int_{-\infty}^{\infty} y(t)\sin(\omega t)dt. \tag{4}$$

Inverse Fourier transform (synthesis):

$$y(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} A(\omega)\cos(\omega t)d\omega + \frac{1}{2\pi}\int_{-\infty}^{\infty} B(\omega)\sin(\omega t)d\omega. \tag{5}$$

Definition four.5: Fourier transforms: complex form

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

$$\mathcal{F}(y(t)) = Y(\omega) = \int_{-\infty}^{\infty} y(t)e^{-j\omega t}dt. \tag{6}$$

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

$$\mathcal{F}^{-1}(Y(\omega)) = y(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} Y(\omega)e^{j\omega t}d\omega. \tag{7}$$

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 sum.ft.

Example four.transform–1

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.

The signal is aperiodic, so the Fourier transform can be computed from Eq. 6:

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

re: a Fourier transform

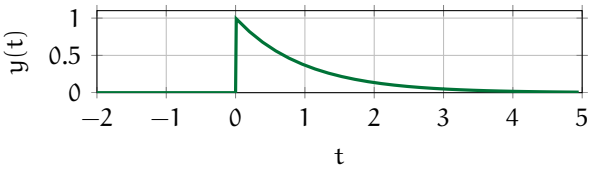

Figure transform.3: an aperiodic signal.

$$= \frac{1}{-a+j\omega}\left(\lim_{t\to\infty} e^{-at}e^{j\omega t} - 1\right)$$

(arrange)

$$= \frac{1}{-a+j\omega}\left((0)(\text{complex with mag} \leqslant 1) - 1\right)$$

(limit)

$$= \frac{-1}{-a+j\omega}$$ (consequence)

$$= \frac{1}{a-j\omega}$$

$$= \frac{a+j\omega}{a+j\omega}\cdot\frac{1}{a-j\omega}$$ (rationalize)

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

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

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

Now we can plot these functions of ω. Setting $a = 1$ (arbitrarily), we obtain the plots of Fig. transform.4.

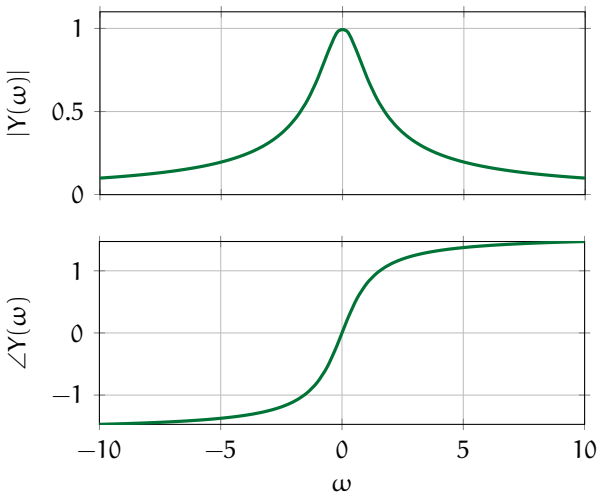

Figure transform.4: the magnitude and phase of the Fourier transform.