## Lecture 02.03 Fourier transforms

The source for this lecture is in *SageMath* kernel *Jupyter* notebook. For more information, see jupyter.org and sagemath.org.

See ricopic.one/measurement/notebooks for the source code notebook. First, we import packages and all that. We use `matplotlib` for plotting and `numpy` for numerics.

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

```
save_figures = False # true to save LaTeX figures
T = 35 # period
pulse_width = 2 # pulse width
f1(x) = pulse_width/2-abs(x) # first pulse
f2(x) = pulse_width/2-abs(x-T) # second pulse
omega_max = 12 # rad/s max frequency in line spectrum
n_max = round(omega_max*T/(2*pi)) # corresponding max harmonic
```

First, we plot the function `f` in the time domain. Using the *SageMath* `piecewise` function due to its Fourier Series methods (used momentarily), we define it and use `matplotlib` to plot it.

```
f = piecewise([[(-pulse_width/2,pulse_width/2),f1]]) # for FS series
fp = piecewise( # for plotting
    [
        [[-T/2,-pulse_width/2],0],
        [(-pulse_width/2,pulse_width/2),f1],
        [[pulse_width/2,T/2],0],
        [(T/2,T-pulse_width/2),0],
        [[T-pulse_width/2,T+pulse_width/2],f2],
        [(T+pulse_width/2,T+T/2),0],
        [[T+T/2,T+T/2],0]
    ]
)
N = 201 # number of points to plot
tpp = np.linspace(-T/2,3*T/2,N) # numeric array of time values
fpp = []
for i in range(0,N):
    fpp.append(fp(tpp[i])) # build array of function values
axes = plt.figure(1)
plt.plot(tpp,fpp,'b-',linewidth=2) # plot
plt.xlabel('time (s)')
```

```
plt.xlim([-T/2,3*T/2])
plt.xticks([pulse_width/2,T],['$\\frac{\delta}{2}$','$T='+str(T)+'$ s'])
plt.yticks([0,pulse_width/2],['0','$\delta/2$'])
if save_figures:
    tikz_save( # save for LaTeX's pgfplots
        'figures/fourier_series_to_transform_pulse'+
        str(T)+'.tex',
        figureheight='.5\linewidth',
        figurewidth='1\linewidth'
    )
plt.show() # display here
```

For $\delta = 2$ and $T \in [5, 15, 25]$, the left-hand column of Figure 02.1 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. *SageMath* has nice methods for its `piecewise` class, `fourier_series_cosine_coefficient(n,T/2)` and `fourier_series_sine_coefficient(n,T/2)`, that can compute the Fourier series cosine and sine components $a_n$ and $b_n$ for component n (`n`) and period T (`T`).

```
f_cos = [];
f_sin = [];
f_harmonic_amplitude = [];
omega = [];
for i in range(0,n_max):
    f_cos.append(f.fourier_series_cosine_coefficient(i,T/2))
    f_sin.append(f.fourier_series_sine_coefficient(i,T/2))
    f_harmonic_amplitude.append(
        T/pulse_width*sqrt(f_cos[i]**2+f_sin[i]**2)
    )
    omega.append(2*pi*i/T)
```

Furthermore, we have computed the *harmonic amplitude* (`f_harmonic_amplitude`):

$$C_n = \sqrt{a_n^2 + b_n^2} \tag{02.20}$$

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

```
axes = plt.figure(2)
markerline, stemlines, baseline = plt.stem(
    omega, f_harmonic_amplitude,
    linefmt='b-', markerfmt='bo', basefmt='r-'
)
plt.xlabel('frequency $\omega$ (rad/s)')
plt.xlim([0,omega_max])
plt.yticks([0,pulse_width/2],['0','$\delta/2$'])
if save_figures:
    tikz_save( # save for LaTeX
        'figures/fourier_series_to_transform_spectrum'+
        str(T)+'.tex',
        figureheight='.5\linewidth',
        figurewidth='1\linewidth'
    )
plt.show() # show here
```

{

} The line spectra are shown in the right-hand column of Figure 02.1. 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 $\omega$

$$F(\omega) = \frac{\delta}{2} \cdot \frac{\sin^2(\omega\delta/4)}{(\omega\delta/4)^2}. \tag{02.21}$$

First, we plot it.

```
F(w) = pulse_width/2* \
    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) # numeric array of time values
```

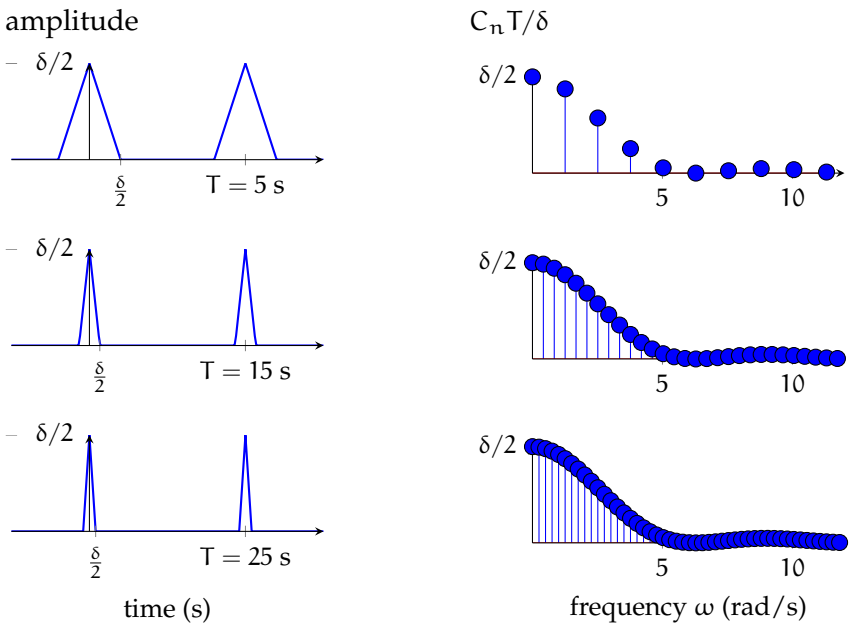


**Figure 02.1:** triangle pulse trains (left column) with longer periods, descending, and their corresponding line spectra (right column), scaled for convenient comparison.

```
Fpp = []
for i in range(0,N):
    Fpp.append(F(wpp[i])) # build array of function values
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('$F(\omega)$')
if save_figures:
    tikz_save( # save for LaTeX
        'figures/fourier_series_to_transform_transform.tex',
        figureheight='.5\linewidth',
        figurewidth='1\linewidth'
    )
plt.show()
```

Let's consider the plot in Figure 02.2 of `F`. It's obviously the function emerging in Figure 02.1 from increasing the period of our pulse train.

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

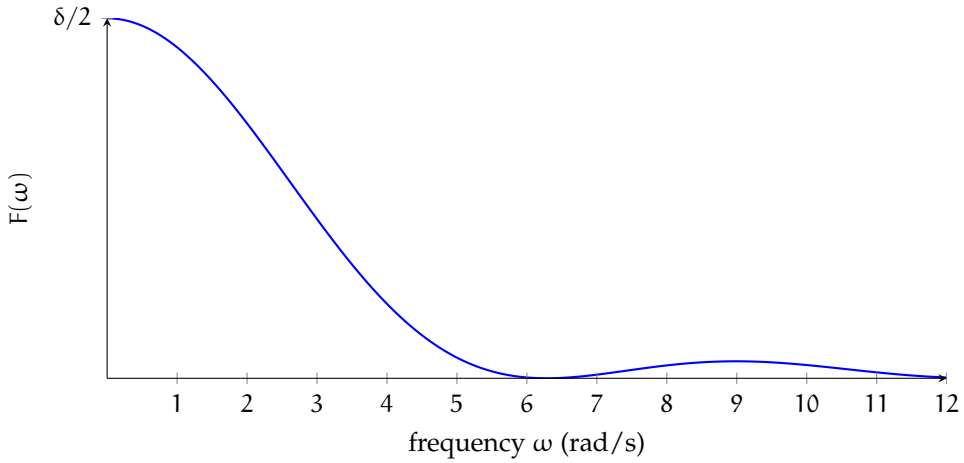


**Figure 02.2:** $F(\omega)$, our mysterious Fourier series amplitude analog.

**Definition 02.03.1: Fourier transforms: trigonometric form**

Fourier transform (analysis):

$$A(\omega) = \int_{-\infty}^{\infty} y(t)\cos(\omega t)dt \tag{02.22}$$

$$B(\omega) = \int_{-\infty}^{\infty} y(t)\sin(\omega t)dt. \tag{02.23}$$

Inverse Fourier transform (synthesis):

$$y(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} A(\omega)\cos(\omega t)d\omega + \frac{1}{2\pi}\int_{-\infty}^{\infty} B(\omega)\sin(\omega t)d\omega. \tag{02.24}$$

**Definition 02.03.2: Fourier transforms: complex form**

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

$$\mathcal{F}(y(t)) = Y(\omega) = \int_{-\infty}^{\infty} y(t) e^{j\omega t} dt. \tag{02.25}$$

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

$$\mathcal{F}^{-1}(Y(\omega)) = y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} Y(\omega) e^{-j\omega t} d\omega. \tag{02.26}$$

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 a step or ramp function). 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 on the course website in the "Resources" section.