SECTION 7: FOURIER ANALYSIS

ESC 440 – Computational Methods for Engineers

Fourier Series – Trigonometric Form

Periodic Functions

- A function is ***periodic*** if

$$f(t) = f(t + T)$$

  where T is the ***period*** of the function

- The function repeats itself every T seconds
- Here, we're assuming a function of time, but could also be a spatial function, e.g.
  - Elevation
  - Pixel intensity along rows or columns of an image

Frequency

- The ***frequency*** of a periodic function is the inverse of its period

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

- We'll refer to a function's frequency as its ***fundamental frequency***, f_0
- This is ***ordinary frequency***, and has units of ***Hertz*** (Hz) (or cycles/sec)
- Can also describe a function in terms of its ***angular frequency***, which has units of rad/sec

$$\omega_0 = 2\pi \cdot f_0 = \frac{2\pi}{T}$$

Fourier Series

- Fourier discovered that if a periodic function satisfies the ***Dirichlet conditions***:

  1) It is absolutely integrable over any period:

  $$\int_{t_0}^{t_0+T} f(t)dt < \infty$$

  2) It has a finite number of maxima and minima over any period
  3) It has a finite number of discontinuities over any period

  - In other words, ***any periodic signal of engineering interest***

- Then it can be represented as an infinite sum of harmonically-related sinusoids, the ***<u>Fourier series</u>***

Joseph Fourier
1768 – 1830

Fourier Series

- The ***Fourier series***

$$f(t) = a_0 + \sum_{k=1}^{\infty} [a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t)]$$

where ω_0 is the fundamental frequency, $\omega_0 = \frac{2\pi}{T}$

and, the Fourier coefficients are given by

$$a_0 = \frac{1}{T}\int_0^T f(t)dt$$

the average value of the function over a full period, and

$$a_k = \frac{2}{T}\int_0^T f(t)\cos(k\omega_0 t)\,dt\,, \qquad k = 1,2,3 \ldots$$

and

$$b_k = \frac{2}{T}\int_0^T f(t)\sin(k\omega_0 t)\,dt\,, \qquad k = 1,2,3 \ldots$$

Sinusoids as Basis Functions

- Harmonically-related sinusoids form a set of ***orthogonal basis functions*** for any periodic functions satisfying the Dirichlet conditions
- Not unlike the unit vectors in $\mathbf{R}^2$ space:

$$\hat{\mathbf{i}} = (1,0), \qquad \hat{\mathbf{j}} = (0,1)$$

- Any vector can be expressed as a linear combination of these basis vectors

$$\mathbf{x} = a_1\hat{\mathbf{i}} + a_2\hat{\mathbf{j}}$$

where each coefficient is given by an inner product

$$a_1 = \mathbf{x} \cdot \hat{\mathbf{i}}$$
$$a_2 = \mathbf{x} \cdot \hat{\mathbf{j}}$$

- These are the ***projections*** of $\mathbf{x}$ onto the basis vectors

Sinusoids as Basis Functions

- Similarly, any periodic function can be represented as a sum of projections onto the sinusoidal basis functions
- Similar to vector dot products, these projections are also given by inner products:

$$a_k = \frac{2}{T}\int_0^T f(t)\cos(k\omega_0 t)\,dt\,, \qquad k = 1,2,3\ldots$$

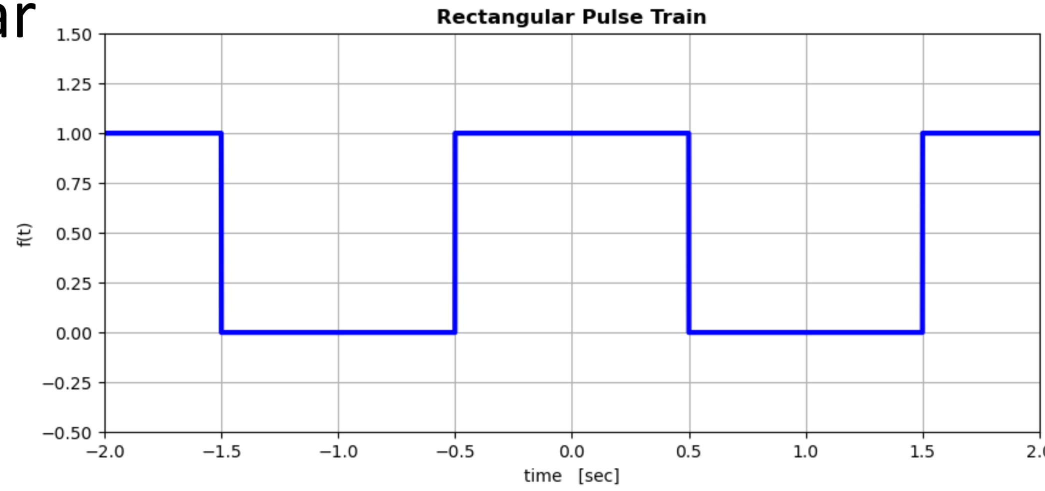
and

$$b_k = \frac{2}{T}\int_0^T f(t)\sin(k\omega_0 t)\,dt\,, \qquad k = 1,2,3\ldots$$

- These are projections of $f(t)$ onto the sinusoidal basis functions

Fourier Series – Example

- Consider a rectangular pulse train
  - $T = 2\ sec$
  - $f_0 = \frac{1}{T} = 0.5Hz$
  - $\omega_0 = \pi\ ^{rad}/_{sec}$

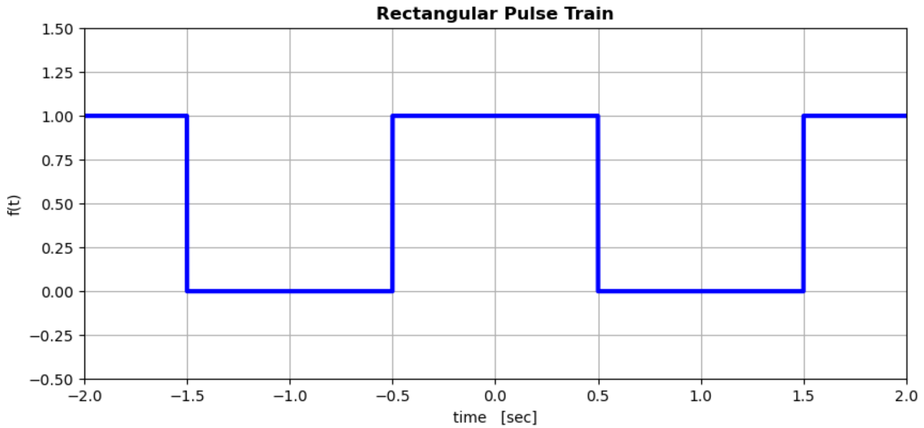


- Can determine the Fourier series by integrating over any full period, for example, $t = [0,2]$

$$f(t) = \begin{cases} 1 & 0 < t < 0.5 \\ 0 & 0.5 < t < 1.5 \\ 1 & 1.5 < t < 2.0 \end{cases}$$

Fourier Series – Example – a_0

$$f(t) = \begin{cases} 1 & 0 < t < 0.5 \\ 0 & 0.5 < t < 1.5 \\ 1 & 1.5 < t < 2.0 \end{cases}$$



- First, calculate the average value

$$a_0 = \frac{1}{T}\int_0^T f(t)dt = \frac{1}{2}\int_0^2 f(t)dt$$

$$a_0 = \frac{1}{2}\int_0^{0.5} 1dt + \frac{1}{2}\int_{0.5}^{1.5} 0dt + \frac{1}{2}\int_{1.5}^{2} 1dt$$

$$a_0 = \frac{1}{2}t\,\Big|_0^{0.5} + \frac{1}{2}t\,\Big|_{1.5}^{2} = 0.25 + 0.25$$

$a_0 = 0.5$, as would be expected

Fourier Series – Example – a_k

- Next determine the cosine coefficients, a_k

$$a_k = \frac{2}{T}\int_0^T f(t)\cos(k\omega_0 t)\,dt$$

$$a_k = \frac{2}{2}\int_0^{0.5}\cos(k\pi t)\,dt + \frac{2}{2}\int_{1.5}^{2}\cos(k\pi t)\,dt$$

$$a_k = \frac{1}{k\pi}\sin(k\pi t)\Big|_0^{0.5} + \frac{1}{k\pi}\sin(k\pi t)\Big|_{1.5}^{2}$$

$$a_k = \frac{1}{k\pi}\left[\sin\left(k\frac{\pi}{2}\right) - 0 + 0 - \sin\left(k3\frac{\pi}{2}\right)\right]$$

$$a_k = \frac{1}{k\pi}\left[\sin\left(k\frac{\pi}{2}\right) - \sin\left(k3\frac{\pi}{2}\right)\right]$$

Fourier Series – Example – a_k

- We know that

$$\sin\left(k3\frac{\pi}{2}\right) = \sin\left(k\frac{\pi}{2} + k\pi\right) = -\sin(k\frac{\pi}{2})$$

so

$$a_k = \frac{2}{k\pi}\sin\left(k\frac{\pi}{2}\right), \qquad k = 1,2,3 \ldots$$

- The first few values of a_k:

$$a_1 = \frac{2}{\pi},\ a_2 = \ 0\ ,\ a_3 = -\frac{2}{3\pi},\ a_4 = 0\ ,\ a_5 = \frac{2}{5\pi}$$

- Zero for all even values of k
  - Only odd harmonics present in the Fourier Series

Fourier Series – Example – b_k

- Next, determine the sine coefficients, b_k

$$b_k = \frac{2}{T}\int_0^T f(t)\sin(k\omega_0 t)\,dt$$

$$b_k = \frac{2}{2}\int_0^{0.5}\sin(k\pi t)\,dt + \frac{2}{2}\int_{1.5}^{2}\sin(k\pi t)\,dt$$

$$b_k = -\frac{1}{k\pi}\left[\cos(k\pi t)\Big|_0^{0.5} + \cos(k\pi t)\Big|_{1.5}^{2}\right]$$

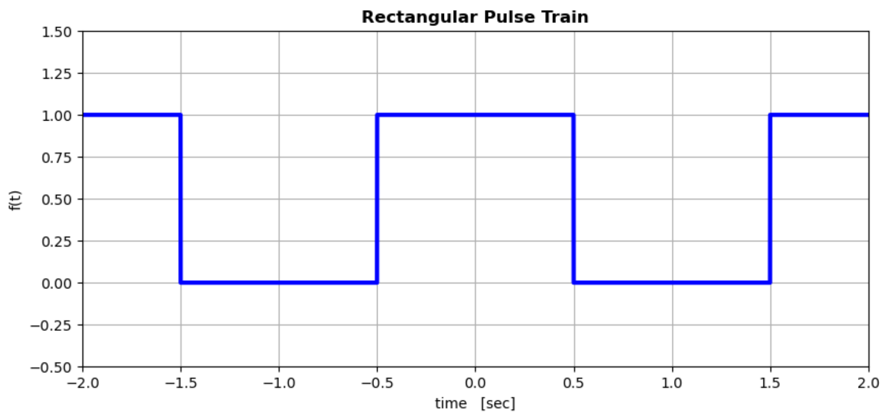
$$b_k = -\frac{1}{k\pi}\left[\cos\left(k\frac{\pi}{2}\right) - 1 + 1 - \cos\left(k\frac{\pi}{2} + k\pi\right)\right] = 0$$

$$b_k = 0\,, \qquad k = 1,2,3\,\ldots$$

- All b_k coefficients are zero
  - Only cosine terms in the Fourier series

Fourier Series – Example

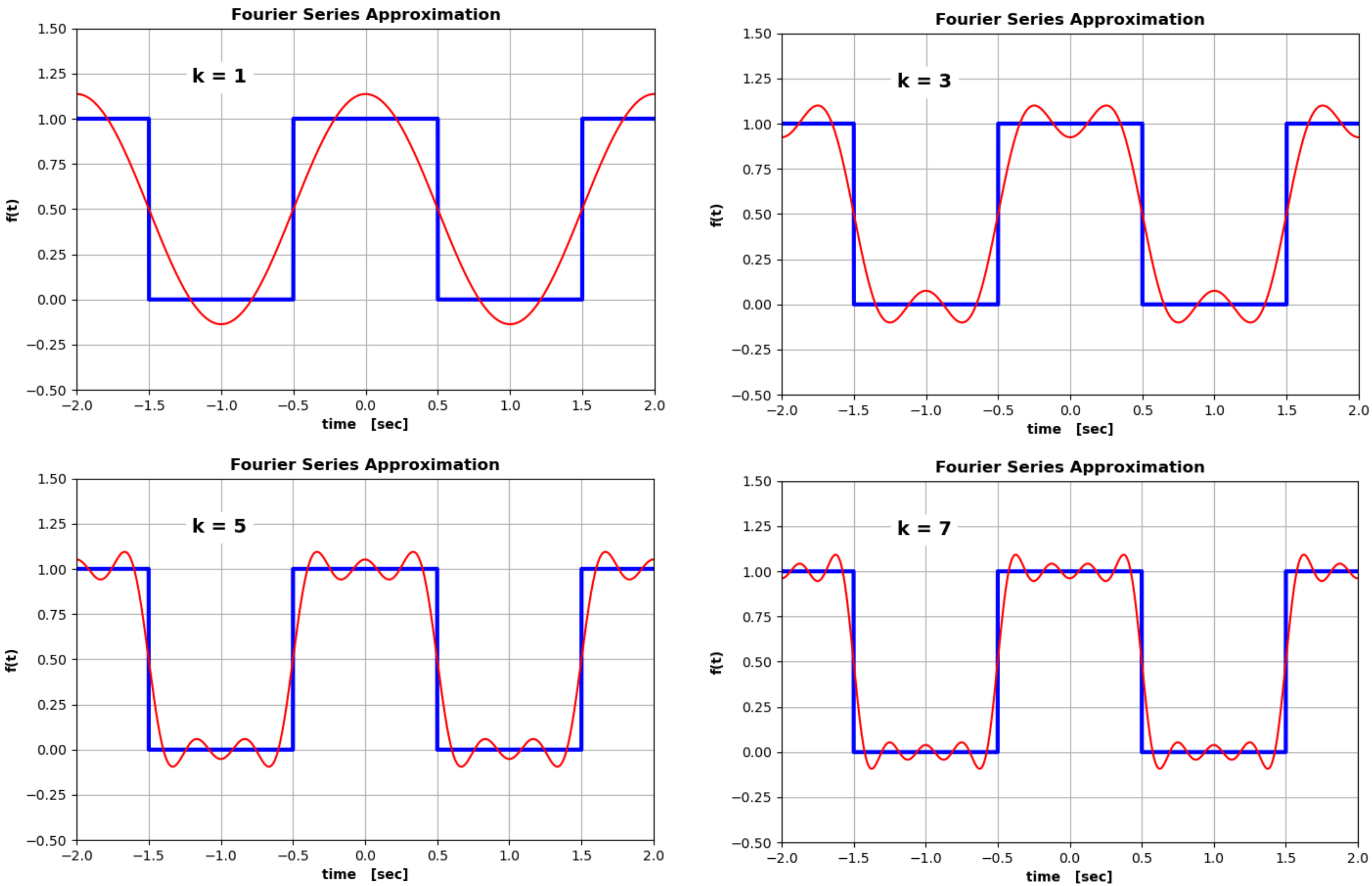
- The Fourier series for the rectangular pulse train:



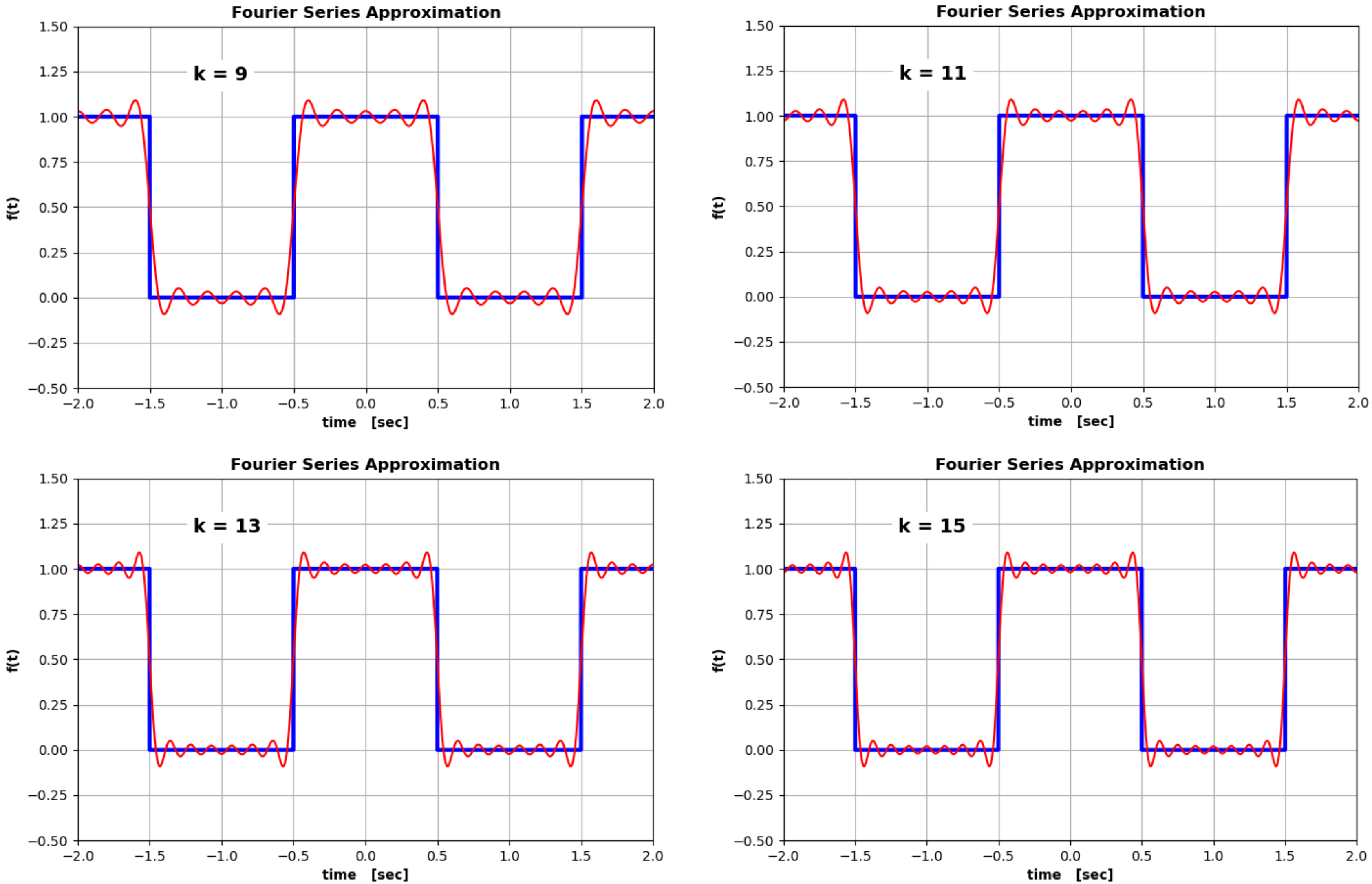
$$f(t) = 0.5 + \sum_{k=1}^{\infty} \frac{2}{k\pi} \sin\left(k\frac{\pi}{2}\right) \cos(k\pi t)$$

- Note that this is an equality as long as we include an infinite number of harmonics
- Can approximate $f(t)$ by truncating after a finite number of terms

Fourier Series – Example



Fourier Series – Example



Even and Odd Symmetry

- An ***even function*** is one for which

$$f(t) = f(-t)$$

- An ***odd function*** is one for which

$$f(t) = -f(-t)$$

- Consider two functions, $f(t)$ and $g(t)$
  - If both are even (or odd), then

$$\int_{-\alpha}^{\alpha} f(t)g(t)dt = 2\int_{0}^{\alpha} f(t)g(t)dt$$

  - If one is even, and one is odd, then

$$\int_{-\alpha}^{\alpha} f(t)g(t)dt = 0$$

Even and Odd Symmetry

- Since $\cos(k\omega_0 t)$ is even, and $\sin(k\omega_0 t)$ is odd
  - If $f(t)$ is an ***even*** function, then

$$a_k = \frac{4}{T}\int_0^{T/2} f(t)\cos(k\omega_0 t)\,dt\,, \qquad k = 1, 2, 3, \dots$$

$$b_k = 0, \qquad k = 1, 2, 3, \dots$$

  - If $f(t)$ is an ***odd*** function, then

$$a_k = 0, \qquad k = 1, 2, 3, \dots$$

$$b_k = \frac{4}{T}\int_0^{T/2} f(t)\sin(k\omega_0 t)\,dt\,, \qquad k = 1, 2, 3, \dots$$

- Recall the Fourier series for the pulse train, an even function, had only cosine terms

Fourier Series – Cosine w/ Phase Form

Cosine-with-Phase Form

- Given the trigonometric identity

$$A_1 \cos(\omega t) + B_1 \sin(\omega t) = C_1 \cos(\omega t + \theta)$$

where $\quad C_1 = \sqrt{A_1^2 + B_1^2} \quad$ and $\quad \theta = \tan^{-1}\left(-\frac{B_1}{A_1}\right)$

- We can express the Fourier series in ***cosine-with-phase form***:

$$f(t) = a_0 + \sum_{k=1}^{\infty} A_k \cos(k\omega_0 t + \theta_k)$$

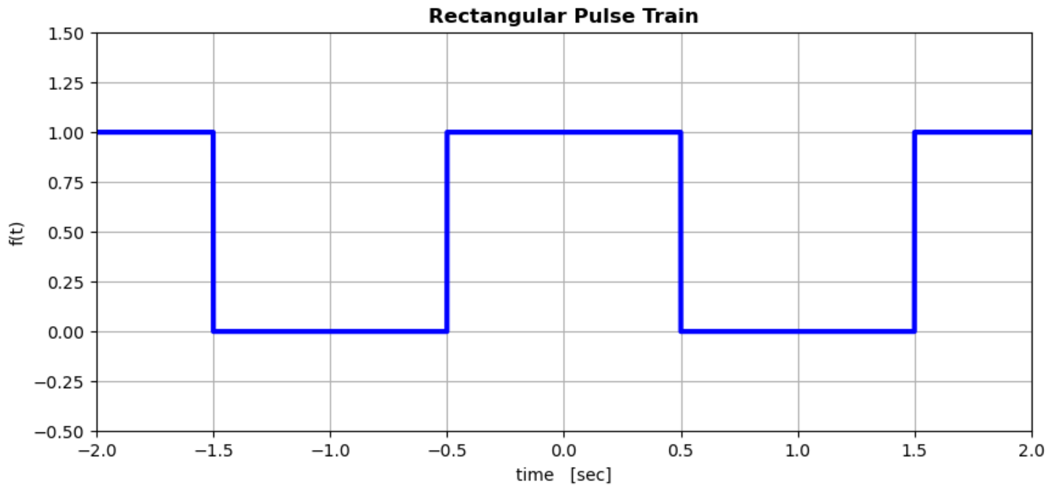
where

$$A_k = \sqrt{a_k^2 + b_k^2}$$

$$\theta_k = \begin{cases} \tan^{-1}\left(-\frac{b_k}{a_k}\right), & a \geq 0 \\ \pi + \tan^{-1}\left(-\frac{b_k}{a_k}\right), & a < 0 \end{cases}$$

Cosine-with-Phase Form – Example

- Consider, again, the rectangular pulse train
  - $a_k = \frac{2}{k\pi}\sin\left(\frac{k\pi}{2}\right)$
  - $b_k = 0$



- So,

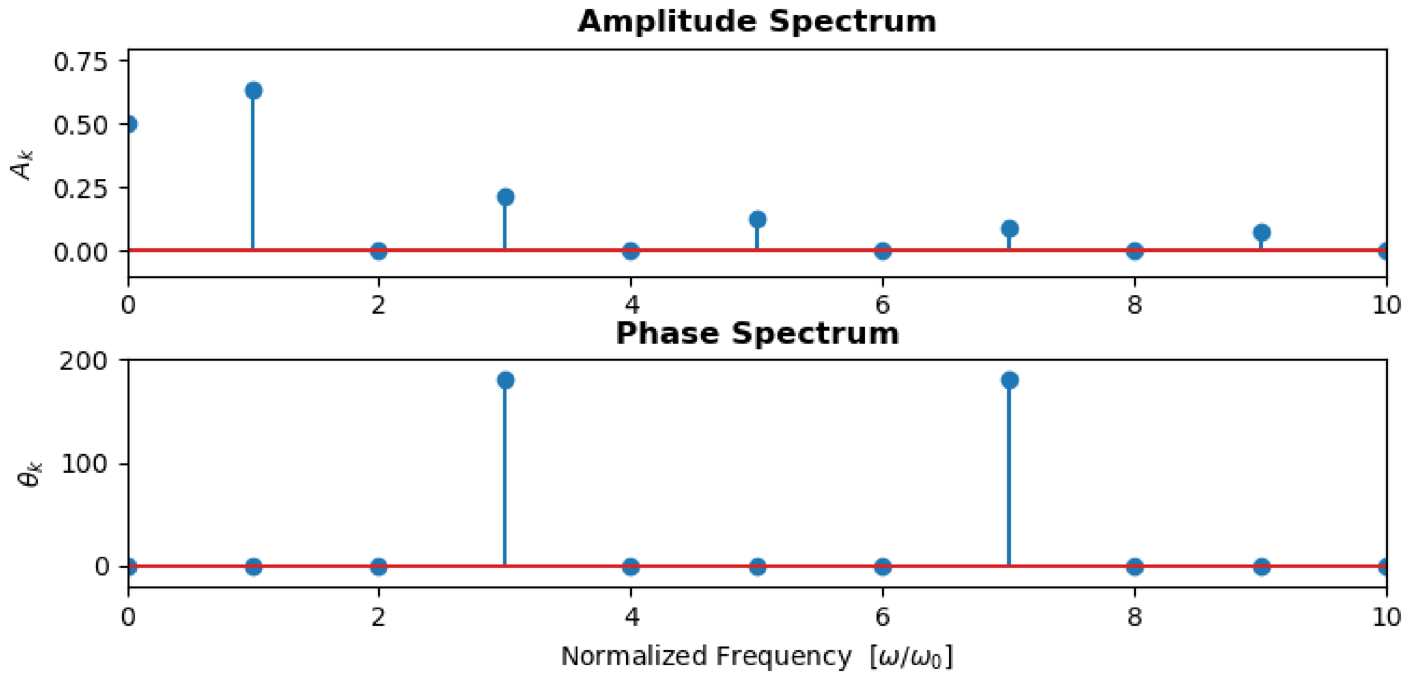
$$A_k = \sqrt{a_k^2 + b_k^2} = |a_k| = \frac{2k}{\pi}\left|\sin\left(\frac{k\pi}{2}\right)\right|$$

and

$$\theta_k = \tan^{-1}\left(-\frac{0}{\frac{2k}{\pi}\sin\left(\frac{k\pi}{2}\right)}\right) = \begin{cases} 0, & k = 1, 5, 9, \ldots \\ \pi, & k = 3, 7, 11, \ldots \end{cases}$$

Line Spectra

- The cosine-with-phase form of the Fourier series is conducive to graphical display as ***amplitude and phase line spectra***



- Average value and amplitude of odd harmonics are clearly visible

Fourier Series – Complex Exponential Form

Complex Exponential Fourier Series

- Recall ***Euler's formula***

$$e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)$$

- This allows us to express the Fourier series in a more compact, though equivalent form

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0 t}$$

where the ***complex*** coefficients are given by

$$c_k = \frac{1}{T}\int_0^T f(t)e^{-jk\omega_0 t}dt$$

- Note that the series is now computed for both ***positive and negative harmonics*** of the fundamental

Complex Exponential Fourier Series

- We can express the complex series coefficients in terms of the trigonometric series coefficients

$$c_0 = a_0$$

$$c_k = \frac{1}{2}(a_k - jb_k), \qquad k = 1, 2, 3, \ldots$$

$$c_{-k} = \frac{1}{2}(a_k + jb_k), \qquad k = 1, 2, 3, \ldots$$

- Coefficients at $\pm k$ are complex conjugates, so

$$|c_k| = |c_{-k}| \quad \text{and} \quad \angle c_k = -\angle c_{-k}$$

Complex Exponential Fourier Series

- Similarly, the coefficients of the trigonometric series in terms of the complex coefficients are

$$a_0 = c_0$$

$$a_k = c_k + c_{-k} = 2\mathcal{Re}(c_k)$$

$$b_k = j(c_k - c_{-k}) = -2\mathcal{Im}(c_k)$$

- Can also relate the complex coefficients to the cosine-with-phase series coefficients

$$|c_k| = |c_{-k}| = \frac{1}{2}A_k, \qquad k = 1, 2, 3, \ldots$$

$$\angle c_k = \begin{cases} \theta_k, & k = +1, +2, +3, \ldots \\ -\theta_k, & k = -1, -2, -3, \ldots \end{cases}$$

Even and Odd Symmetry

- For even functions, since $b_k = 0$, coefficients of the complex series are purely real:

$$c_0 = a_0$$

$$c_k = c_{-k} = \frac{1}{2} a_k, \qquad k = 1, 2, 3, \ldots$$

- For odd functions, since $a_k = 0$, coefficients of the complex series are purely imaginary (except c_0):

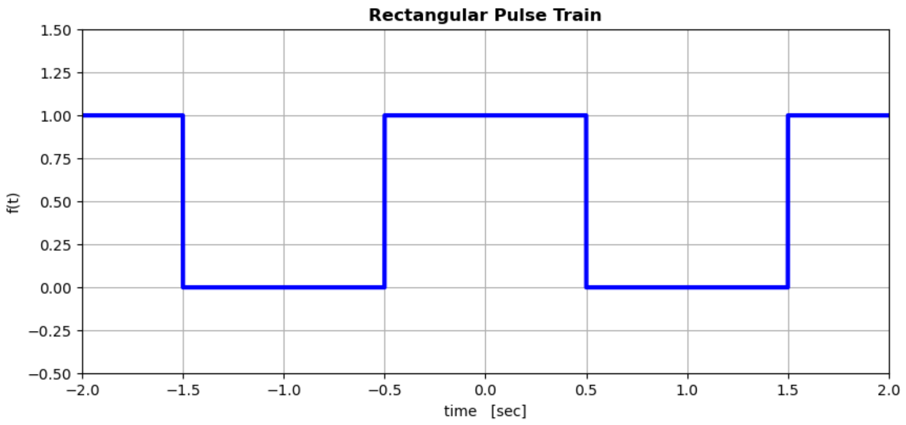
$$c_0 = a_0$$

$$c_k = -j \frac{1}{2} b_k, \qquad k = 1, 2, 3, \ldots$$

$$c_{-k} = +j \frac{1}{2} b_k, \qquad k = 1, 2, 3, \ldots$$

Complex Series – Example

$$f(t) = \begin{cases} 1 & 0 < t < 0.5 \\ 0 & 0.5 < t < 1.5 \\ 1 & 1.5 < t < 2.0 \end{cases}$$



- The complex Fourier series for the rectangular pulse train:

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0 t}$$

- The complex coefficients are given by

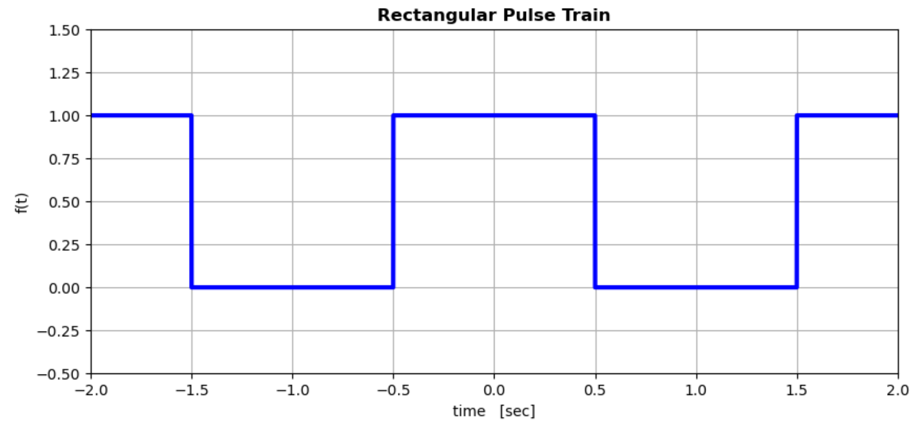
$$c_k = \frac{1}{T}\int_{-T/2}^{T/2} f(t) e^{-jk\omega_0 t} dt = \frac{1}{2}\int_{-1}^{1} f(t) e^{-jk\pi t} dt$$

$$c_k = \frac{1}{2}\int_{-0.5}^{0.5} e^{-jk\pi t} dt = -\frac{1}{2jk\pi} e^{-jk\pi t} \Big|_{-0.5}^{0.5}$$

Complex Series – Example

$$c_k = -\frac{1}{2jk\pi} e^{-jk\pi t} \Big|_{-0.5}^{0.5}$$

$$c_k = -\frac{1}{2jk\pi} \left[e^{-jk\frac{\pi}{2}} - e^{jk\frac{\pi}{2}} \right]$$



- Rearranging into the form of a sinusoid

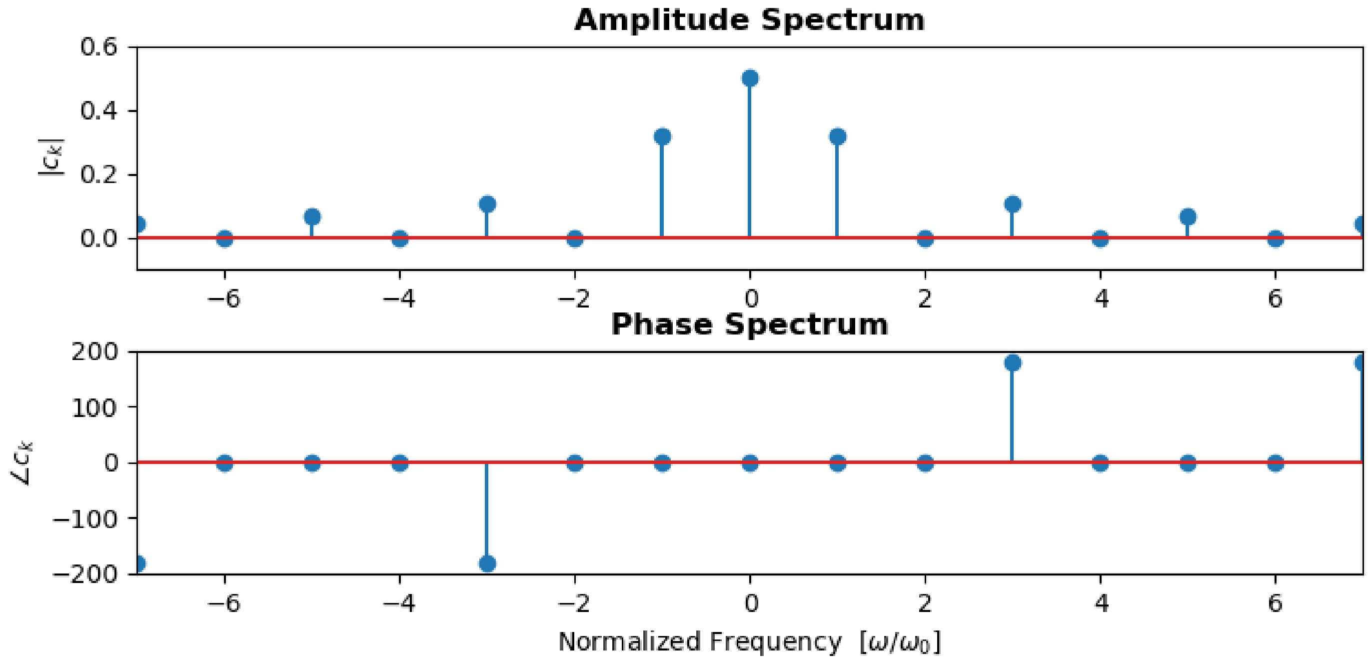
$$c_k = \frac{1}{k\pi} \left[\frac{e^{jk\frac{\pi}{2}} - e^{-jk\frac{\pi}{2}}}{2j} \right] = \frac{1}{k\pi} \sin\left(k\frac{\pi}{2}\right)$$

- Given the even symmetry of $f(t)$, all coefficients are real, and also have even symmetry

$$c_k = c_{-k} = \frac{1}{k\pi} \sin\left(k\frac{\pi}{2}\right) = \frac{1}{\pi}, 0, -\frac{1}{3\pi}, 0, \frac{1}{5\pi}, 0, \dots$$

Line Spectra

- The complex series coefficients can also be plotted as ***amplitude and phase line spectra***
  - Now, plot spectra over ***positive and negative frequencies***



- Note that the magnitude spectrum is an even function of frequency, and the phase spectrum is an odd function of frequency

Sinusoidal Curve Fitting

The Fourier series can also be understood by approaching it as a least-squares curve-fitting problem, where sinusoids are fit to a function or data set.

Sinusoidal Curve Fitting

- In a previous section of the course we saw how we can fit different functions to data using linear least-squares regression
  - Can also fit sinusoids using this technique
- The data we're fitting could be:
  - Measured data that we believe to be sinusoidal in nature
  - A periodic function, that, while not sinusoidal, we want to approximate as a sinusoid or sum of sinusoids

Sinusoidal Curve Fitting

- Our fitting function is

$$y = A_0 + C_1 \cos(\omega_0 t + \theta)$$

- The fundamental frequency is

$$\omega_0 = 2\pi f_0 = \frac{2\pi}{T}$$

where T is the period of the function or data we are fitting

- The ***three fitting parameters*** are: A_0, C_1, and θ
- In order to be able to apply linear regression, we can't have a fitting parameter in the argument of a trigonometric function
  - Apply a trig. Identity to recast the model as

$$y = A_0 + A_1 \cos(\omega_0 t) + B_1 \sin(\omega_0 t)$$

- Assuming we know ω_0, this is a ***linear least-squares model***

Sinusoidal Curve Fitting

- Assuming ω_0 is known, the linear least-squares model is

$$y = a_0 z_0 + a_1 z_1 + a_2 z_2$$

where

$$z_0 = 1, \;\; z_1 = \cos(\omega_0 t), \;\; z_2 = \sin(\omega_0 t)$$

and

$$a_0 = A_0, \;\; a_1 = A_1, \;\; \text{and} \;\; a_2 = B_1$$

- For a least-squares fit, minimize the sum of the squares of the residuals

$$S_r = \sum_{i=1}^{N} \{y_i - [A_0 + A_1 \cos(\omega_0 t) + B_1 \sin(\omega_0 t)]\}^2$$

Normal Equations

- As we saw in the curve fitting section of the course, the matrix normal equations for this least-squares fit are

$$\mathbf{Z^T Z a} = \mathbf{Z^T y}$$

where Z is the design matrix:

$$\mathbf{Z} = \begin{bmatrix} z_{01} & z_{11} & z_{21} \\ z_{02} & z_{12} & z_{22} \\ \vdots & \vdots & \vdots \\ z_{0N} & z_{1N} & z_{2N} \end{bmatrix} = \begin{bmatrix} 1 & \cos(\omega_0 t_1) & \sin(\omega_0 t_1) \\ 1 & \cos(\omega_0 t_2) & \sin(\omega_0 t_2) \\ \vdots & \vdots & \vdots \\ 1 & \cos(\omega_0 t_N) & \sin(\omega_0 t_N) \end{bmatrix}$$

$\mathbf{a}$ is the vector of fitting parameters

$$\mathbf{a} = [A_0 \; A_1 \; B_1]^T$$

and $\mathbf{y}$ is the vector of N function or data values

$$\mathbf{y} = [y_1 \; y_2 \; y_3 \; \ldots y_N]^T$$

Normal Equations – $\mathbf{Z^T Z a} = \mathbf{Z^T y}$

- The LHS of the normal equations is

$$\mathbf{Z^T Z a} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \cos(\omega_0 t_1) & \cos(\omega_0 t_2) & \cdots & \cos(\omega_0 t_N) \\ \sin(\omega_0 t_1) & \sin(\omega_0 t_2) & \cdots & \sin(\omega_0 t_N) \end{bmatrix} \begin{bmatrix} 1 & \cos(\omega_0 t_1) & \sin(\omega_0 t_1) \\ 1 & \cos(\omega_0 t_2) & \sin(\omega_0 t_2) \\ \vdots & \vdots & \vdots \\ 1 & \cos(\omega_0 t_N) & \sin(\omega_0 t_N) \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ B_1 \end{bmatrix}$$

$$\mathbf{Z^T Z a} = \begin{bmatrix} N & \Sigma\cos(\omega_0 t) & \Sigma\sin(\omega_0 t) \\ \Sigma\cos(\omega_0 t) & \Sigma\cos^2(\omega_0 t) & \Sigma\cos(\omega_0 t)\sin(\omega_0 t) \\ \Sigma\sin(\omega_0 t) & \Sigma\sin(\omega_0 t)\cos(\omega_0 t) & \Sigma\sin^2(\omega_0 t) \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ B_1 \end{bmatrix}$$

- If we assume our N data points span exactly one period, then we know the following mean values

$$\frac{\Sigma\cos(\omega_0 t)}{N} = \frac{\Sigma\sin(\omega_0 t)}{N} = \frac{\Sigma\cos(\omega_0 t)\sin(\omega_0 t)}{N} = 0$$

and

$$\frac{\Sigma\cos^2(\omega_0 t)}{N} = \frac{\Sigma\sin^2(\omega_0 t)}{N} = \frac{1}{2}$$

Normal Equations – $\mathbf{Z^T Z a} = \mathbf{Z^T y}$

- Using these known mean values, the normal equations simplify to

$$\begin{bmatrix} N & 0 & 0 \\ 0 & N/2 & 0 \\ 0 & 0 & N/2 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ B_1 \end{bmatrix} = \begin{bmatrix} \Sigma y \\ \Sigma y \cos(\omega_0 t) \\ \Sigma y \sin(\omega_0 t) \end{bmatrix}$$

- We can solve for the vector of fitting parameters, $\mathbf{a}$

$$\mathbf{a} = \left(\mathbf{Z^T Z}\right)^{-\mathbf{1}} \mathbf{Z^T y}$$

- The inverse of the diagonal matrix is a diagonal matrix, where the diagonal elements are inverted, so

$$\begin{bmatrix} A_0 \\ A_1 \\ B_1 \end{bmatrix} = \begin{bmatrix} 1/N & 0 & 0 \\ 0 & 2/N & 0 \\ 0 & 0 & 2/N \end{bmatrix} \begin{bmatrix} \Sigma y \\ \Sigma y \cos(\omega_0 t) \\ \Sigma y \sin(\omega_0 t) \end{bmatrix}$$

Sinusoidal Least-Squares Fit

- The fitting parameters are

$$A_0 = \frac{\Sigma y}{N}$$

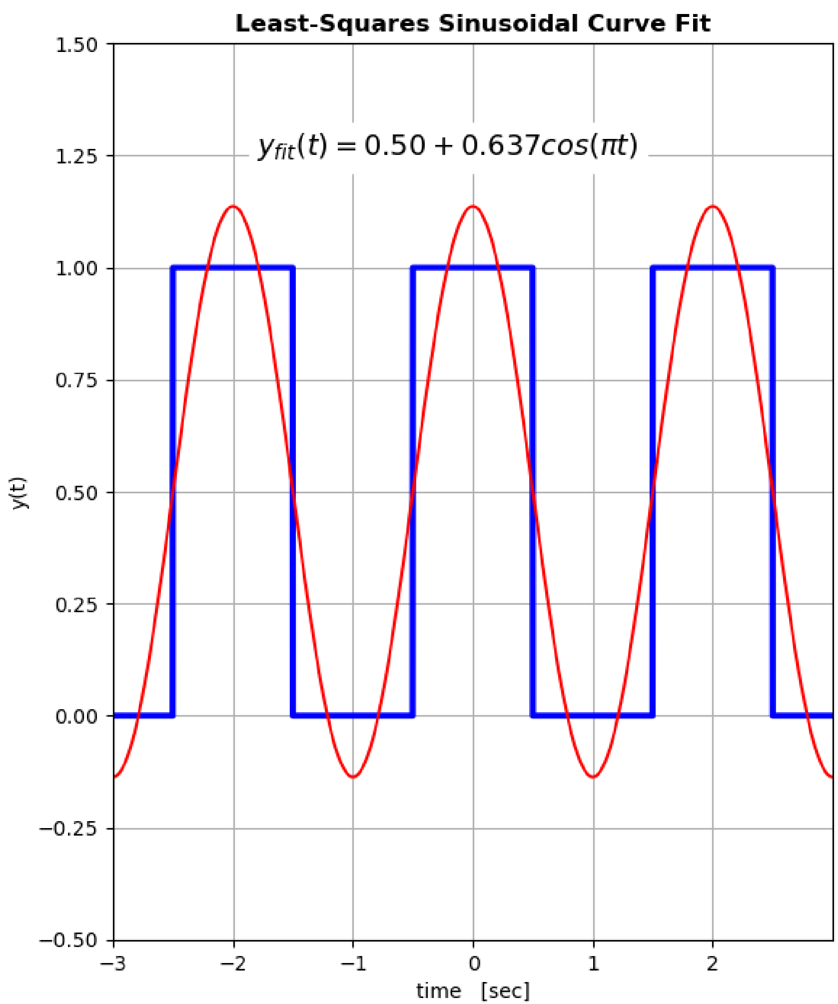
$$A_1 = \frac{2}{N}\Sigma y \cos(\omega_0 t)$$

$$B_1 = \frac{2}{N}\Sigma y \sin(\omega_0 t)$$

- Note the similarity to the Fourier series coefficients
- The least-squares, best-fit sinusoid is given by

$$y = \frac{\Sigma y}{N} + \left(\frac{2}{N}\Sigma y \cos(\omega_0 t)\right)\cos(\omega_0 t) + \left(\frac{2}{N}\Sigma y \sin(\omega_0 t)\right)\sin(\omega_0 t)$$

Sinusoidal Least-Squares Fit – Example




```python
# rect_pulse_cfit.py

import numpy as np
from matplotlib import pyplot as plt


T = 2                    # period of the pulse train
f0 = 1/T                 # fundamental frequency
w0 = 2*np.pi*f0

Ts = T/1001                          # sample period
t = np.arange(-T/2, T/2, Ts)         # time vector spans one full period

y = 0.5 + 0.5*np.sign(np.cos(np.pi*t))

N = len(y)

# create the design matrix
Z1 = np.ones((N,1))
Z2 = np.array([np.cos(w0*t)]).transpose()
Z3 = np.array([np.sin(w0*t)]).transpose()
Z = np.append(np.append(Z1, Z2, axis=1), Z3, axis=1)

# Solve normal equations for vector of fitting coefficients, a.
# Need to transpose y to a column vector.
a = np.linalg.inv(Z.transpose() @ Z) @ (Z.transpose() @ y.transpose())

A0 = a[0]
A1 = a[1]
B1 = a[2]
```

- As expected, $B_1 = 0$ due to the ***even symmetry*** of the function being fit

Least-Squares Fit of Two Harmonics

- Now, consider extending the fitting model to include the first two harmonics

$$y = A_0 + A_1 \cos(\omega_0 t) + B_1 \sin(\omega_0 t) + A_2 \cos(2\omega_0 t) + B_2 \sin(2\omega_0 t)$$

- We've added two more basis functions to the linear least-squares model
- The design matrix is now

$$\mathbf{Z} = \begin{bmatrix} 1 & \cos(\omega_0 t_1) & \sin(\omega_0 t_1) & \cos(2\omega_0 t_1) & \sin(2\omega_0 t_1) \\ 1 & \cos(\omega_0 t_2) & \sin(\omega_0 t_2) & \cos(2\omega_0 t_2) & \sin(2\omega_0 t_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \cos(\omega_0 t_N) & \sin(\omega_0 t_N) & \cos(2\omega_0 t_N) & \sin(2\omega_0 t_N) \end{bmatrix}$$

Least-Squares Fit of Two Harmonics

- If we again assume samples spanning exactly one period, the off-diagonal terms on the LHS of the normal equations go to zero, leaving

$$\mathbf{Z}^{\mathbf{T}}\mathbf{Z}\mathbf{a} = \mathbf{Z}^{\mathbf{T}}\mathbf{y}$$

$$\begin{bmatrix} N & 0 & 0 & 0 & 0 \\ 0 & N/2 & 0 & 0 & 0 \\ 0 & 0 & N/2 & 0 & 0 \\ 0 & 0 & 0 & N/2 & 0 \\ 0 & 0 & 0 & 0 & N/2 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ B_1 \\ A_2 \\ B_2 \end{bmatrix} = \begin{bmatrix} \Sigma y \\ \Sigma \mathrm{y} \cos(\omega_0 t) \\ \Sigma \mathrm{y} \sin(\omega_0 t) \\ \Sigma \mathrm{y} \cos(2\omega_0 t) \\ \Sigma \mathrm{y} \sin(2\omega_0 t) \end{bmatrix}$$

- Solve for $\mathbf{a}$ as

$$\mathbf{a} = \left(\mathbf{Z}^{\mathbf{T}}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\mathbf{T}}\mathbf{y}$$

Least-Squares Fit of Two Harmonics

- Solving for **a** gives the following fitting parameters

$$A_0 = \frac{\Sigma y}{N}$$

$$A_1 = \frac{2}{N}\Sigma y \cos(\omega_0 t)$$

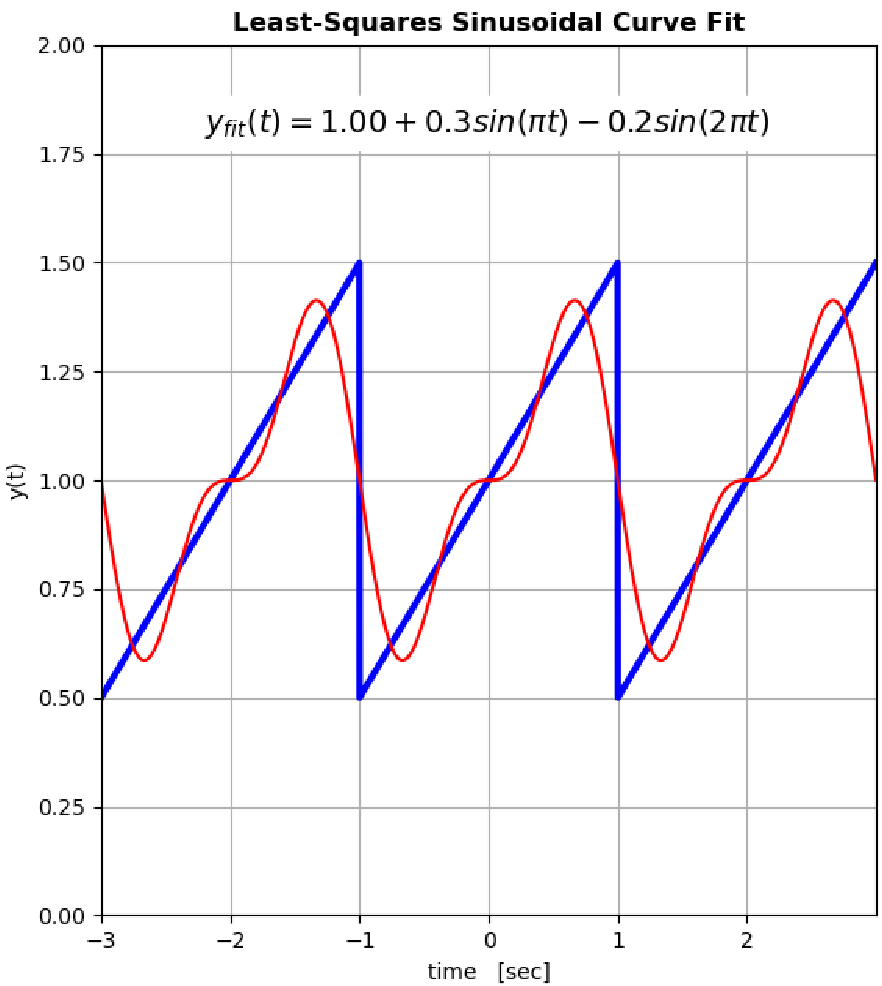
$$B_1 = \frac{2}{N}\Sigma y \sin(\omega_0 t)$$

$$A_2 = \frac{2}{N}\Sigma y \cos(2\omega_0 t)$$

$$B_2 = \frac{2}{N}\Sigma y \sin(2\omega_0 t)$$

- This model could obviously be extended to include an arbitrary number of harmonics

Least-Squares Fit – Example




```python
# sawtooth_cfit.py

import numpy as np
from matplotlib import pyplot as plt

#  fit a sum of first two harmonics to a sawtooth wave

T = 2                  # period of the pulse train
f0 = 1/T               # fundamental frequency
w0 = 2*np.pi*f0

Ts = T/1001                     # sample period
t = np.arange(-T/2, T/2, Ts)    # time vector spans one full period

y = 1 + t/2

N = len(y)

# create the design matrix
# Z = [ones(N,1),cos(w0*t'),sin(w0*t'),cos(2*w0*t'),sin(2*w0*t')];
Z1 = np.ones((N,1))
Z2 = np.array([np.cos(w0*t)]).transpose()
Z3 = np.array([np.sin(w0*t)]).transpose()
Z4 = np.array([np.cos(2*w0*t)]).transpose()
Z5 = np.array([np.sin(2*w0*t)]).transpose()
Z123 = np.append(np.append(Z1, Z2, axis=1), Z3, axis=1)
Z = np.append(np.append(Z123, Z4, axis=1), Z5, axis=1)

# Solve normal equations for vector of fitting coefficients, a.
# Need to transpose y to a column vector.
a = np.linalg.inv(Z.transpose() @ Z) @ (Z.transpose() @ y.transpose())

A0 = a[0]
A1 = a[1]
B1 = a[2]
A2 = a[3]
B2 = a[4]
```

- Sawtooth wave has ***odd symmetry***, so $A_1 = A_2 = 0$, and only sine terms are present

Fourier Transform

The Fourier transform extends the frequency-domain analysis capability provided by the Fourier series to aperiodic signals.

Fourier Transform

- The ***Fourier Series*** is a tool that provides insight into the ***frequency content*** of ***periodic signals***

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0 t}$$

where the complex coefficients are given by

$$c_k = \int_{-T/2}^{T/2} f(t) e^{-jk\omega_0 t} dt$$

- These c_k values provide a measure of the energy present in a signal at ***discrete values of frequency***
  - $k\omega_0$, integer multiples (harmonics) of the fundamental
- Frequency-domain representation is ***discrete***, because the time-domain signal is ***periodic***

Fourier Transform

- Many signals of interest are ***aperiodic***
  - They never repeat
  - Equivalent to an infinite period, $T \to \infty$
- As $T \to \infty$, the mapping from the time domain to the frequency domain is given by the ***Fourier transform***

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

  where $F(\omega)$ is a ***complex***, ***continuous*** function of frequency

- The ***continuous frequency-domain*** representation corresponds to the ***aperiodic time-domain*** signal

Inverse Fourier Transform

- We can also map frequency-domain functions back to the time domain using the ***inverse Fourier transform***

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega$$

- The forward ($-j$ or $-i$ transform) and the inverse ($+j$ or $+i$ transform) provide the mapping between ***Fourier transform pairs***

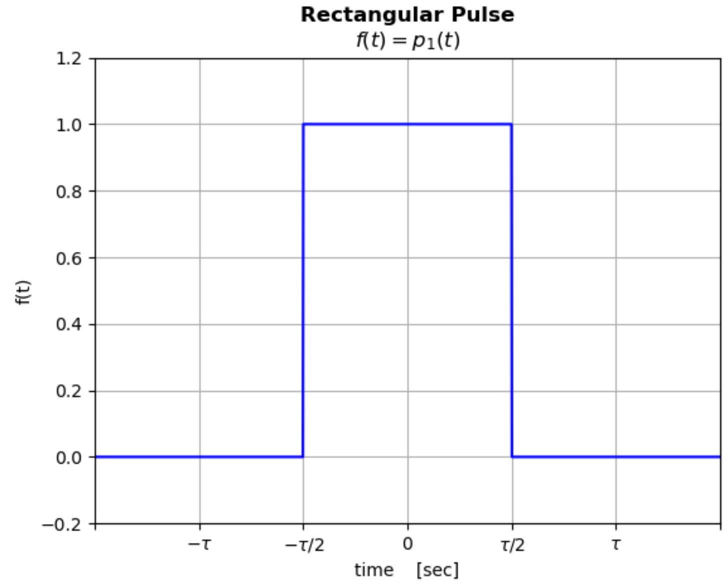
$$f(t) \leftrightarrow F(\omega)$$

Fourier Transform – Rectangular Pulse

- Consider a pulse of duration, τ

$$f(t) = p_\tau(t)$$

- Calculate the Fourier transform



$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt = \int_{-\tau/2}^{\tau/2} e^{-j\omega t} dt$$

$$F(\omega) = -\frac{1}{j\omega} e^{-j\omega t} \Big|_{-\frac{\tau}{2}}^{\frac{\tau}{2}} = -\frac{1}{j\omega}\left[e^{-j\omega\frac{\tau}{2}} - e^{j\omega\frac{\tau}{2}}\right]$$

$$F(\omega) = \frac{2}{\omega}\left[\frac{e^{j\omega\frac{\tau}{2}} - e^{-j\omega\frac{\tau}{2}}}{2j}\right] = \frac{2}{\omega}\sin\left(\frac{\tau\omega}{2}\right)$$

Fourier Transform – Rectangular Pulse

- Here, we can introduce the sinc function

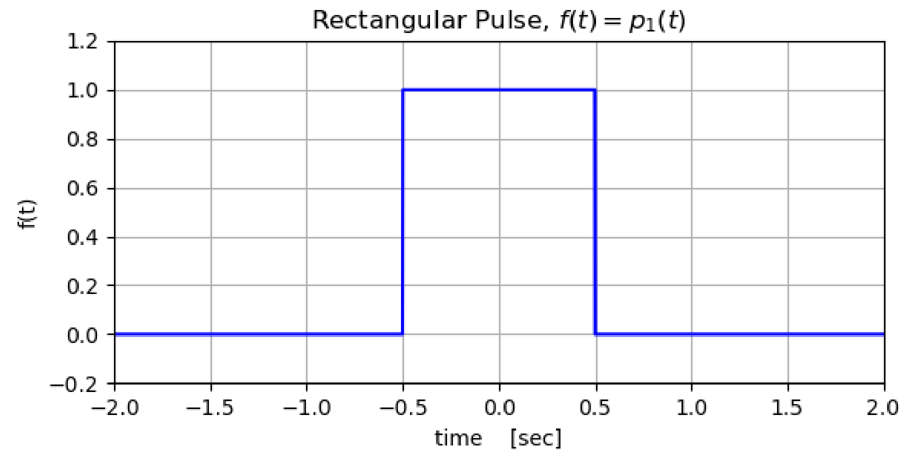
$$sinc(x) = \frac{\sin(\pi x)}{\pi x}$$

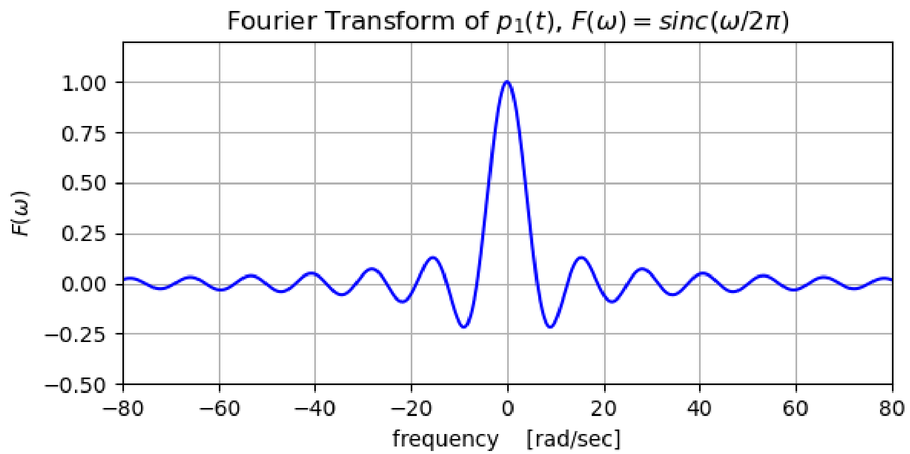
- Letting $x = \frac{\tau\omega}{2\pi}$, we have

$$F(\omega) = \frac{2}{\omega}\sin\left(\frac{\tau\omega}{2}\right)$$

$$F(\omega) = \tau\frac{\sin\left(\pi\frac{\tau\omega}{2\pi}\right)}{\pi\frac{\tau\omega}{2\pi}}$$

$$\boxed{F(\omega) = \tau\, sinc\left(\frac{\tau\omega}{2\pi}\right)}$$



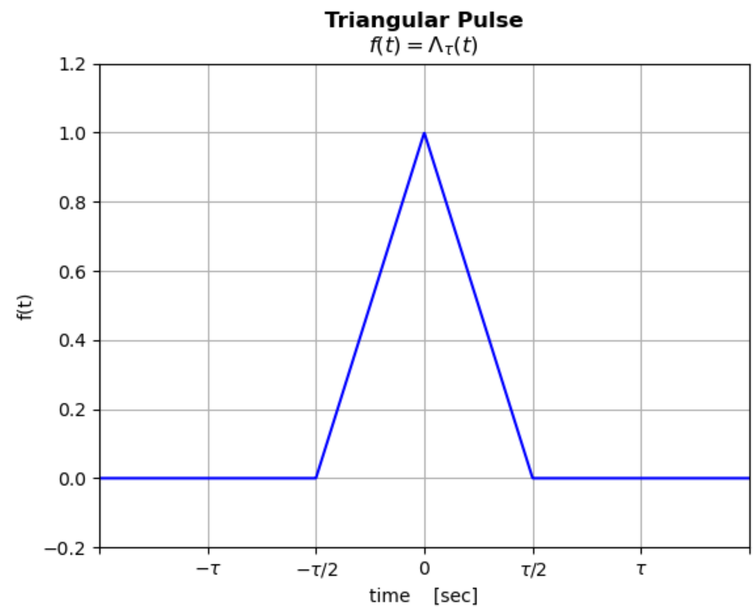


Fourier Transform – Triangular Pulse

- Next, consider a triangular pulse of duration, τ

$$f(t) = \Lambda_\tau(t)$$

$$\Lambda_\tau(t) = \begin{cases} +\dfrac{2}{\tau}t + 1, & -\dfrac{\tau}{2} \le t \le 0 \\ -\dfrac{2}{\tau}t + 1, & 0 \le t \le \dfrac{\tau}{2} \\ 0, & \text{otherwise} \end{cases}$$



- The Fourier transform is

$$F(\omega) = \int_{-\infty}^{\infty} \Lambda_\tau e^{-j\omega t} dt = \int_{-\tau/2}^{0} \left(\frac{2}{\tau}t + 1\right) e^{-j\omega t} dt + \int_{0}^{\tau/2} \left(-\frac{2}{\tau}t + 1\right) e^{-j\omega t} dt$$

- Integration by parts gives

$$F(\omega) = \frac{8}{\tau\omega^2} \sin^2\left(\frac{\tau\omega}{4}\right)$$

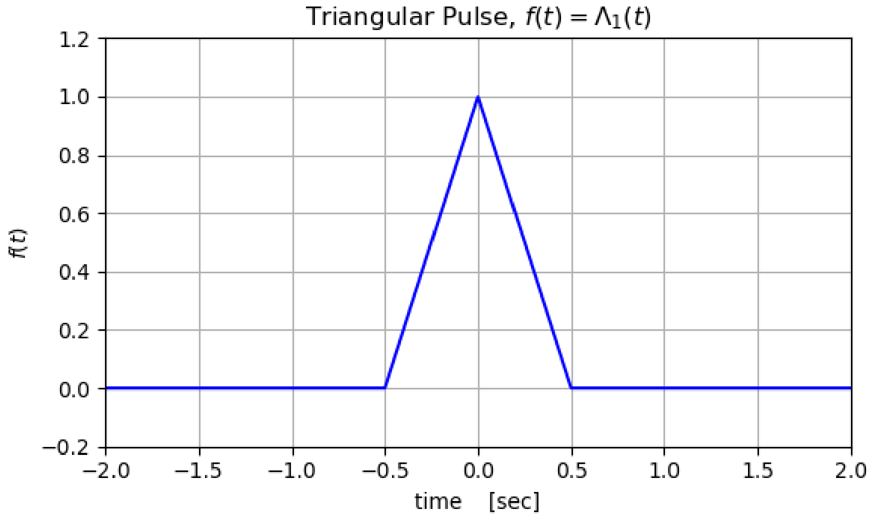
Fourier Transform – Triangular Pulse

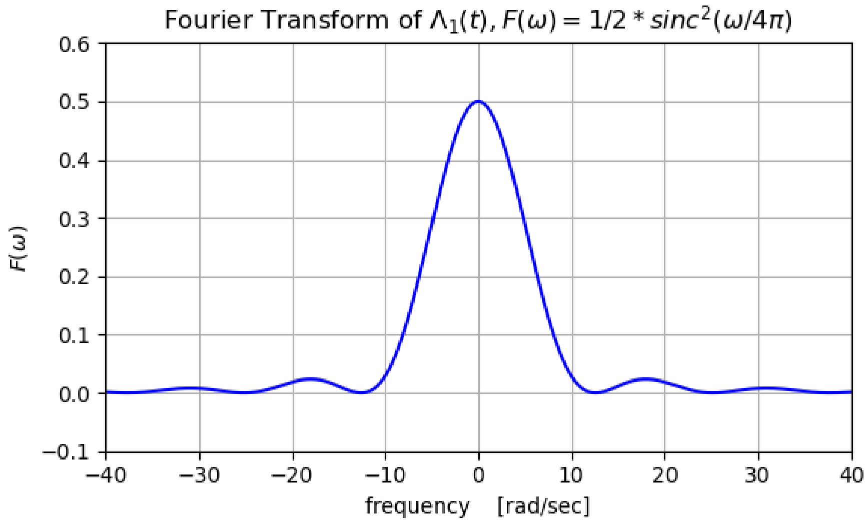
- This, too, can be recast into the form of a sinc function
- Letting $x = \frac{\tau\omega}{4\pi}$, we have

$$F(\omega) = \frac{8}{\tau\omega^2}\sin^2\left(\pi\frac{\tau\omega}{4\pi}\right)$$

$$F(\omega) = \frac{\tau}{2}\frac{\sin^2\left(\pi\frac{\tau\omega}{4\pi}\right)}{\left(\pi\frac{\tau\omega}{4\pi}\right)^2}$$

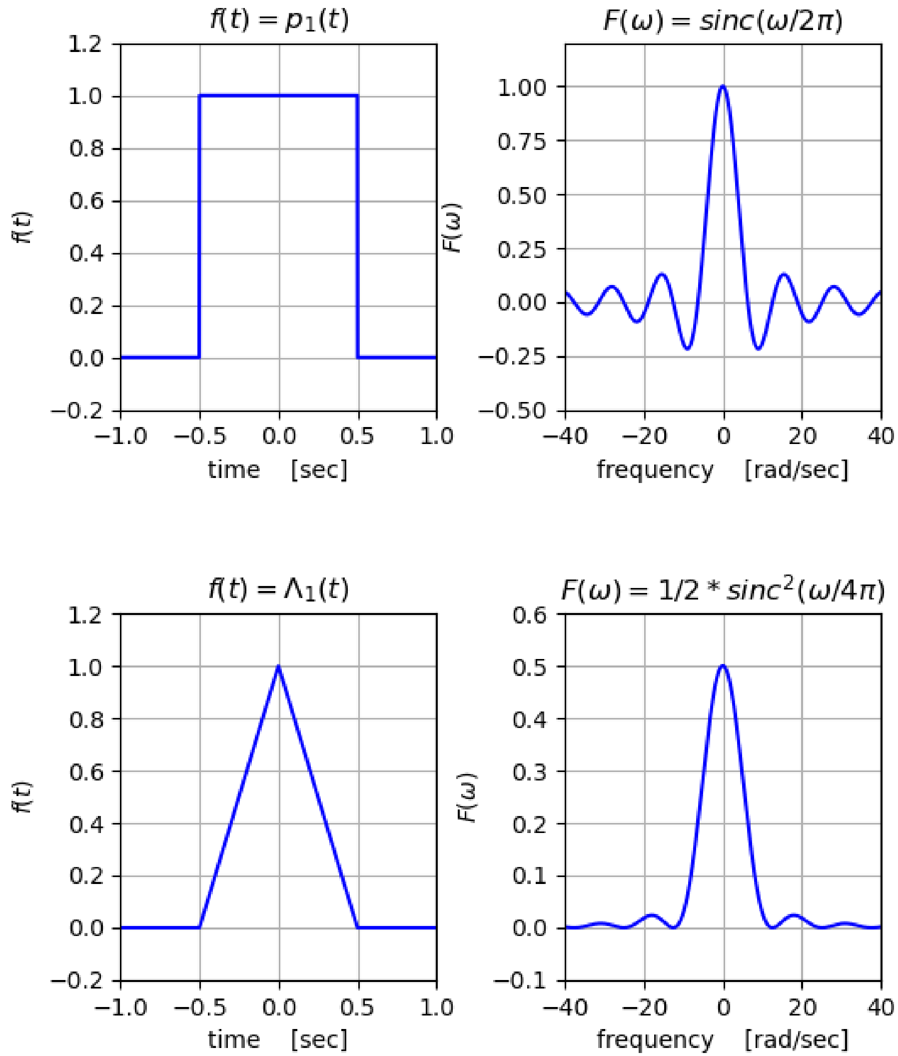
$$\boxed{F(\omega) = \frac{\tau}{2}\,sinc^2\left(\frac{\tau\omega}{4\pi}\right)}$$





Rectangular vs. Triangular Pulse

- Average value in time domain translates to $F(0)$ value in frequency domain
- More abrupt transitions in time domain correspond to more high-frequency content
- ***Multiplication in one domain corresponds to convolution in the other***
  - Convolution of two rectangular pulses is a triangular pulse
  - $sinc$ becomes $sinc^2$ in the frequency domain



Fourier Transform – Impulse Function

- The ***impulse function*** is defined as

$$\delta(t) = 0, \qquad t \neq 0$$

$$\int_{-\infty}^{\infty} \delta(t)dt = 1$$

- Its Fourier transform is

$$F(\omega) = \int_{-\infty}^{\infty} \delta(t)e^{-j\omega t}dt$$

- Since $\delta(t) = 0$ for $t \neq 0$, and since $e^{-j\omega t} = 1$ for $t = 0$

$$F(\omega) = \int_{-\infty}^{\infty} \delta(t)dt = 1$$

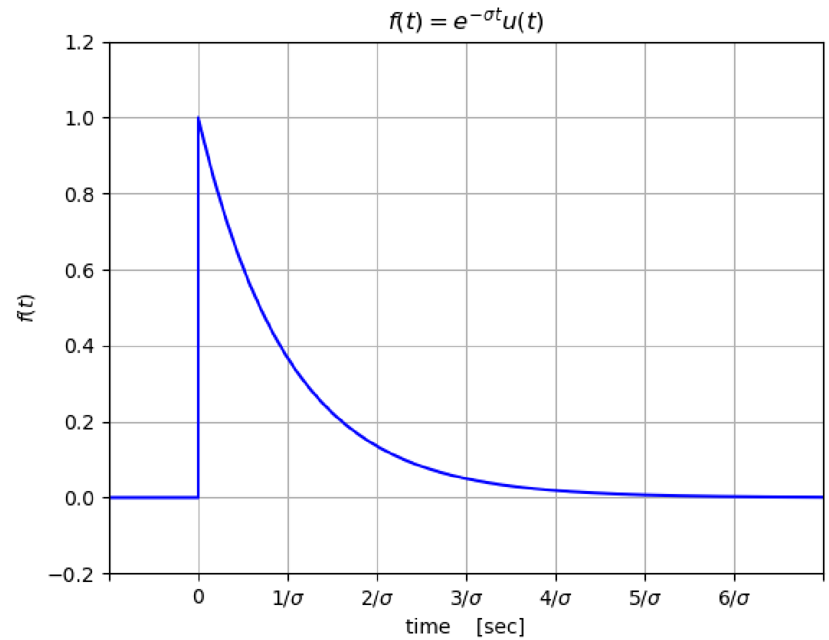
- ***The Fourier transform of the time-domain impulse function is one for all frequencies***
  - Equal energy at all frequencies

Fourier Transform – Decaying Exponential

- Consider a decaying exponential

$$f(t) = e^{-\sigma t} \cdot u(t)$$

where $u(t)$ is the unit step function



- The Fourier transform is:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

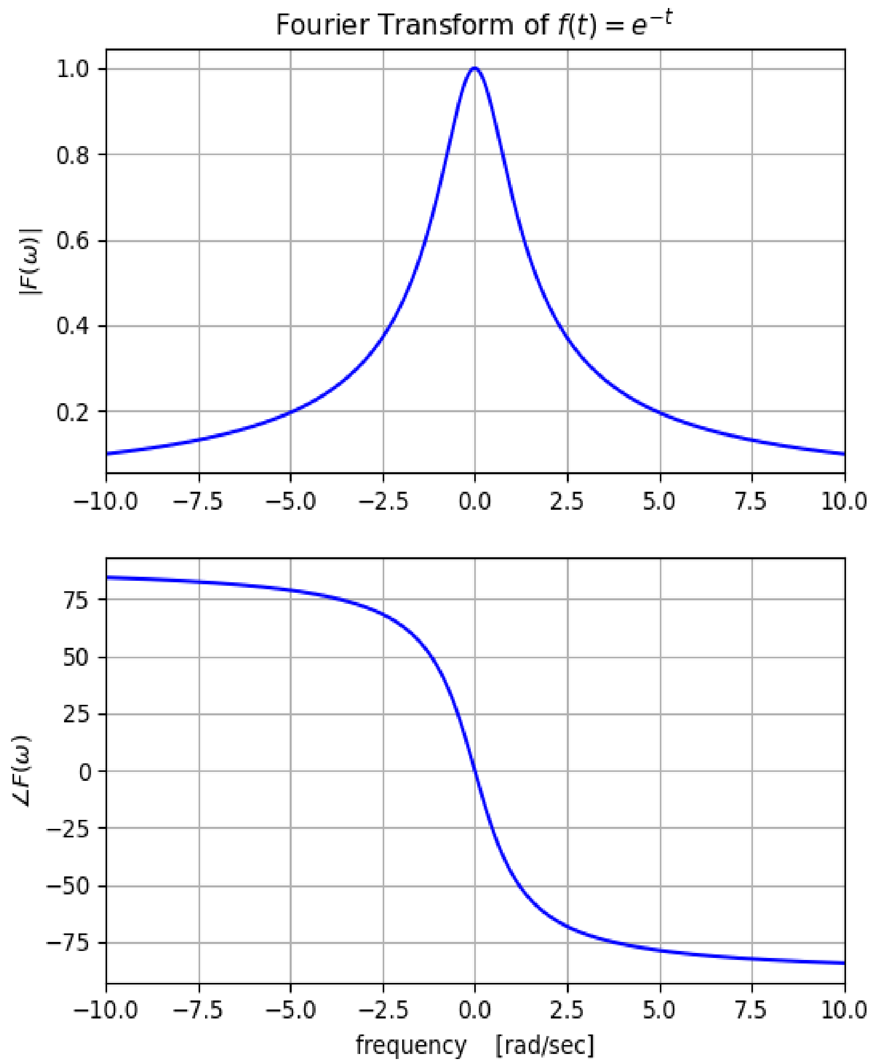
$$F(\omega) = \int_{0}^{\infty} e^{-\sigma t}e^{-j\omega t}dt$$

$$F(\omega) = \int_{0}^{\infty} e^{-(\sigma+j\omega)t}dt = -\frac{1}{\sigma + j\omega}e^{-(\sigma+j\omega)t}\Big|_0^{\infty} = -\frac{1}{\sigma + j\omega}[0-1]$$

$$F(\omega) = \frac{1}{\sigma + j\omega}$$

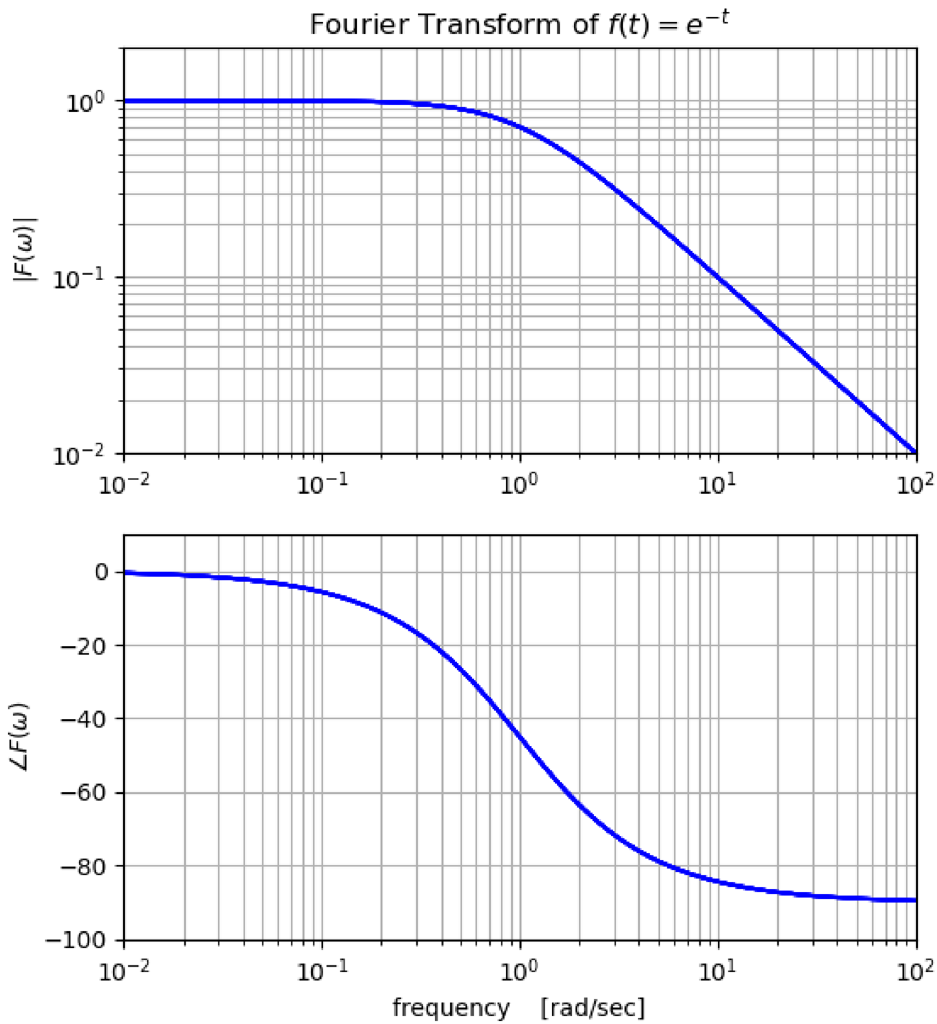
Fourier Transform – Decaying Exponential

- Fourier transform of this exponential signal is ***complex***
- Plot magnitude and phase separately
- Note the even symmetry of magnitude, and odd symmetry of the phase of $F(\omega)$



Fourier Transform – Decaying Exponential

- On logarithmic scales, this Fourier transform should look familiar
- $f(t)$ could be the impulse response of a first-order system
  - ***Convolution*** of an impulse with the system's impulse response
- $F(\omega)$ looks like the frequency response of a first-order system
  - ***Multiplication*** of the F.T. of an impulse ($F(\omega) = 1$) with the system's frequency response



Even and Odd Symmetry

- We are mostly concerned with real time-domain signals
  - Not true for all engineering disciplines, e.g. communications, signal processing, etc.
- ***For a real time-domain signal, $f(t)$,***
  - If $f(t)$ is ***even*** $F(\omega)$ will be ***real and even***
  - If $f(t)$ is ***odd***, $F(\omega)$ will be ***imaginary and odd***
  - If $f(t)$ has ***neither even nor odd*** symmetry, $F(\omega)$ will be ***complex*** with an ***even real*** part and an ***odd imaginary*** part.

Discrete Fourier Transform

For discrete-time signals, mapping from the time domain to the frequency domain is accomplished with the discrete Fourier transform (DFT).

Discrete-Time Fourier Transform (DTFT)

- The Fourier transform maps a continuous-time signal, defined for $-\infty < t < \infty$, to a continuous frequency-domain function defined for $-\infty < \omega < \infty$
- In practice we have to deal with ***discrete-time***, i.e. ***sampled***, signals
  - Only defined at discrete sampling instants

$$f(t) \rightarrow f[n]$$

- Now, mapping to the frequency domain is the ***discrete-time Fourier transform (DTFT)***

$$F(\omega) = \sum_{n=-\infty}^{\infty} f[n]e^{-j\omega n}$$

- ***DTFT maps a discrete, aperiodic, time-domain signal to a continuous, periodic function of frequency***

Aliasing

- ***Aliasing*** is a phenomena that results in a signal appearing as a lower-frequency signal as a result of ***sampling***
- In order to avoid aliasing, the sample rate must be at least the ***Nyquist rate***

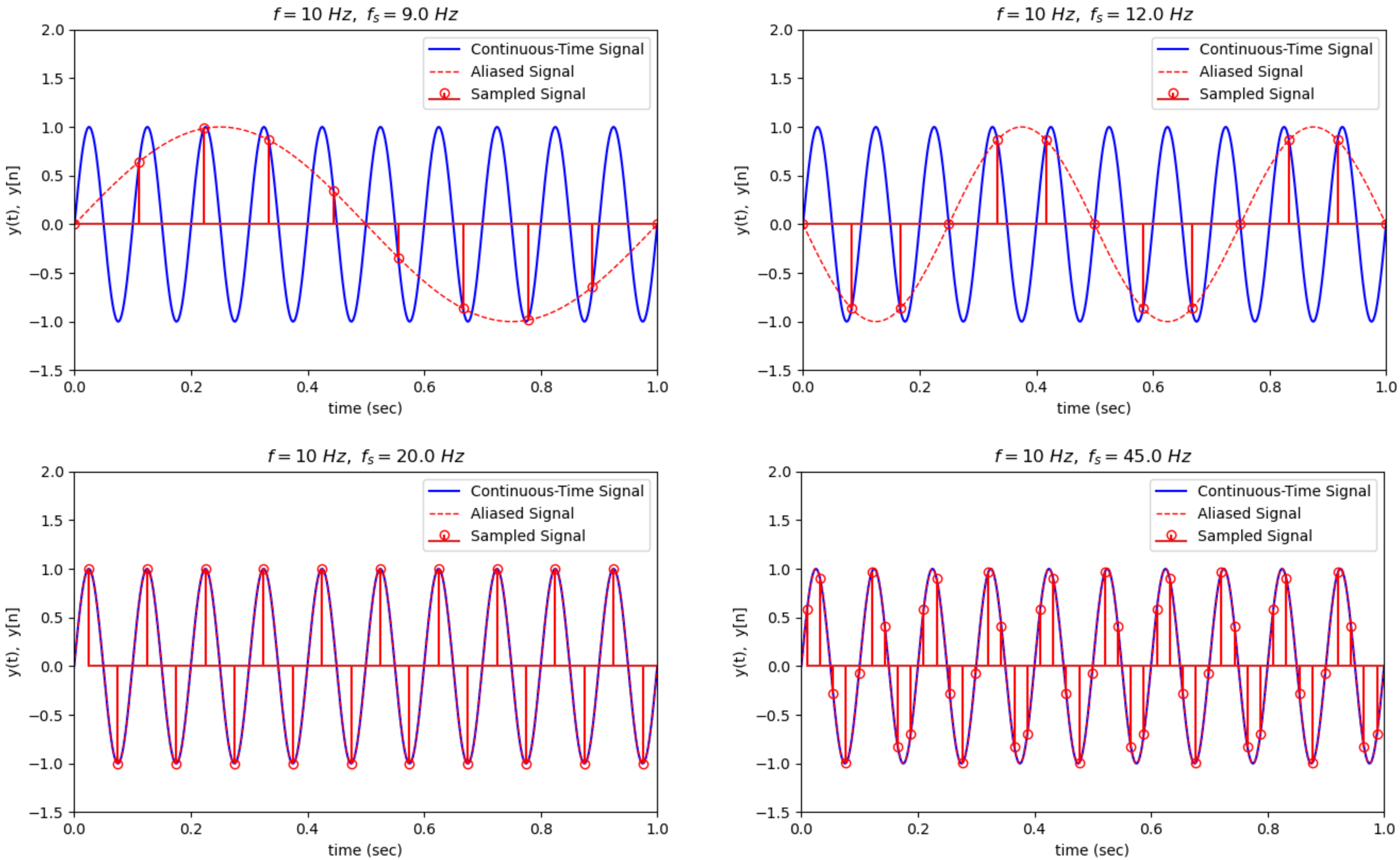
$$f_s \geq 2f_{max}$$

where f_{max} is the highest frequency component present in the signal

- For a given sample rate, the ***Nyquist frequency*** is the highest frequency signal that will not result in aliasing

$$f_{Nyquist} = \frac{f_s}{2}$$

Aliasing – Examples



Discrete-Time Fourier Transform (DTFT)

$$F(\omega) = \sum_{n=-\infty}^{\infty} f[n]e^{-j\omega n}$$

- Discrete-time $f[n]$ generated from $f(t)$ by ***sampling*** at a ***sample rate*** of f_s, with a ***sample period*** of T_s
- Sampled signals can only accurately represent frequencies up to the ***Nyquist frequency***

$$f_{max} = f_{Nyquist} = \frac{f_s}{2}$$

- Higher frequency components of $f(t)$ are ***aliased*** down to lower frequencies in the range of

$$-\frac{f_s}{2} \leq f \leq \frac{f_s}{2}$$

- The DTFT is a periodic function of frequency, with a period f_s
- Due to aliasing, sampling in the time domain corresponds to periodicity in the frequency domain

The Discrete Fourier Transform (DFT)

- The DTFT

$$F(\omega) = \sum_{n=-\infty}^{\infty} f[n]e^{-j\omega n}$$

utilizes discrete-time, sampled, data, but still requires and infinite amount of data

- In practice, our time-domain data sets are both discrete and finite
- The ***discrete Fourier transform***, ***DFT***, maps ***discrete*** and ***finite*** (periodic) ***time-domain*** signals to ***periodic*** and ***discrete frequency-domain*** signals

$$F_k = \sum_{n=0}^{N-1} f[n]e^{-jk2\pi\frac{n}{N}}$$

The Discrete Fourier Transform (DFT)

- Consider N samples of a time-domain signal, $f[n]$
  - Sampled with sampling period T_s and sampling frequency f_s
  - Total time span of the sampled data is $N \cdot T_s$
- The DFT of $f[n]$ is

$$F_k = \sum_{n=0}^{N-1} f[n] e^{-jk2\pi n/N}$$

- A discrete function of the integer value, k
- The DFT consists of N complex values: $F_0, F_1, \ldots, F_{N-1}$
- Each value of k represents a discrete value of frequency from $f = 0$ to $f = f_s$

The Inverse Discrete Fourier Transform

- A discrete, finite set of frequency-domain data can be transformed back to the time domain
- The ***inverse discrete Fourier Transform*** (IDFT)

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{jk2\pi n/N}$$

- Note the $1/N$ scaling factor
  - In practice, this is often applied when computing the DFT
  - Must exist in *either* the DFT or IDFT, not both

DFT Frequencies

$$F_k = \sum_{n=0}^{N-1} f[n] e^{-jk2\pi n/N}$$

- A dot product of $f[n]$ with a complex exponential

$$F_k = f[n] \cdot e^{-j\Omega n}$$

- The frequency of the exponential, Ω, is the ***digital frequency***:

$$\Omega = k2\pi/N$$

  which has units of $rad/sample$

- Digital frequency is related to the ordinary frequency by the sample rate, f_s

$$\Omega = \frac{2\pi f}{f_s} \quad \left[\frac{rad}{sample}\right]$$

DFT Frequencies

$$F_k = \sum_{n=0}^{N-1} f[n] e^{-jk2\pi n/N}$$

- # of samples: N, sample rate: f_s, sample period: T_s
- ***Maximum detectable frequency***

$$f_{max} = f_s/2$$

  - Nyquist frequency
  - Corresponds to $k = N/2, \Omega = \pi$
- ***Frequency increment*** (bin width, resolution)

$$\Delta f = \frac{1}{N \cdot T_s} = \frac{f_s}{N}$$

- Last $(^N/_2 - 1)$ points of F_k, $F_{N/_2+1} \ldots F_{N-1}$ correspond to ***negative frequency***

$$-\frac{f_s}{2} + \Delta f \ldots - \Delta f$$

DFT Frequencies

- For example, consider $N = 10$ samples of a signal sampled at $f_s = 100Hz, T_s = 10msec$
  - $\Delta f = \frac{1}{NT_s} = \frac{f_s}{N} = \frac{1}{10 \cdot 0.01sec} = 10Hz$
  - $f_{max} = \frac{f_s}{2} = 50Hz$
  - $\Delta\Omega = \frac{2\pi}{N} {}^{rad}/_{Sa} = 0.2\pi\ {}^{rad}/_{Sa}$

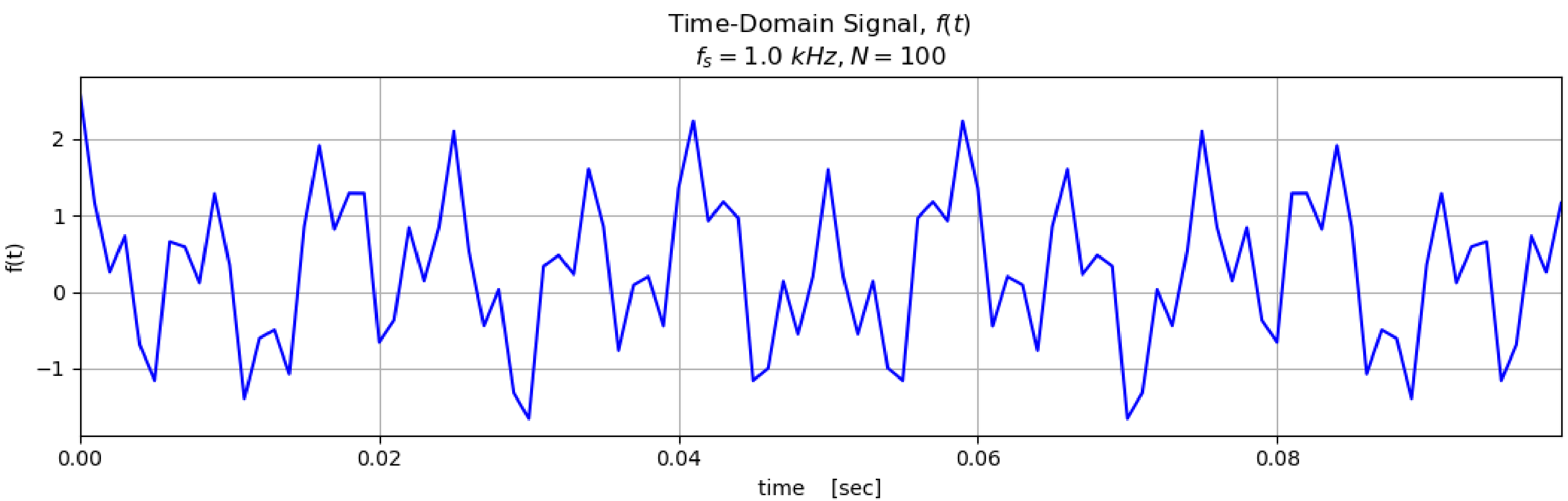
k	0	1	2	3	4	5	6	7	8	9	Units
Ω	$\mathbf{0}$	$\mathbf{0.2\pi}$	$\mathbf{0.4\pi}$	$\mathbf{0.6\pi}$	$\mathbf{0.8\pi}$	$\boldsymbol{\pi}$	$\mathbf{1.2\pi}$	$\mathbf{1.4\pi}$	$\mathbf{1.6\pi}$	$\mathbf{1.8\pi}$	rad/Sa
f/f_s	$\mathbf{0}$	$\mathbf{0.1}$	$\mathbf{0.2}$	$\mathbf{0.3}$	$\mathbf{0.4}$	$\mathbf{0.5}$	$\mathbf{0.6}$	$\mathbf{0.7}$	$\mathbf{0.8}$	$\mathbf{0.9}$	$-$
f	$\mathbf{0}$	$\mathbf{10}$	$\mathbf{20}$	$\mathbf{30}$	$\mathbf{40}$	$\mathbf{50}$	$\mathbf{-40}$	$\mathbf{-30}$	$\mathbf{-20}$	$\mathbf{-10}$	Hz

DFT - Example

- Consider the following signal

$$f(t) = 0.3 + 0.5\cos(2\pi \cdot 50 \cdot t) + \cos(2\pi \cdot 120 \cdot t) + 0.8\cos(2\pi \cdot 320 \cdot t)$$

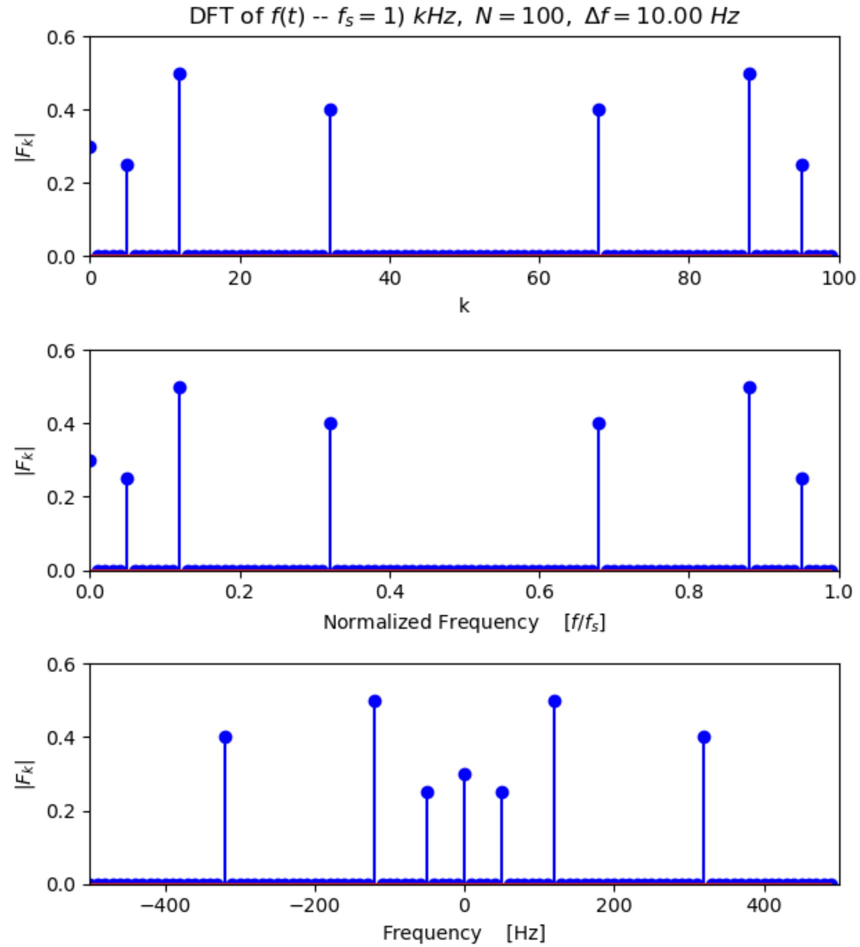
  - Sample rate: $f_s = 1kHz$
  - Record length: $N = 100$
  - Bin width: $\Delta f = 10Hz$



DFT - Example

$$f(t) = 0.3 + 0.5\cos(2\pi \cdot 50 \cdot t) + \cos(2\pi \cdot 120 \cdot t) + 0.8\cos(2\pi \cdot 320 \cdot t)$$

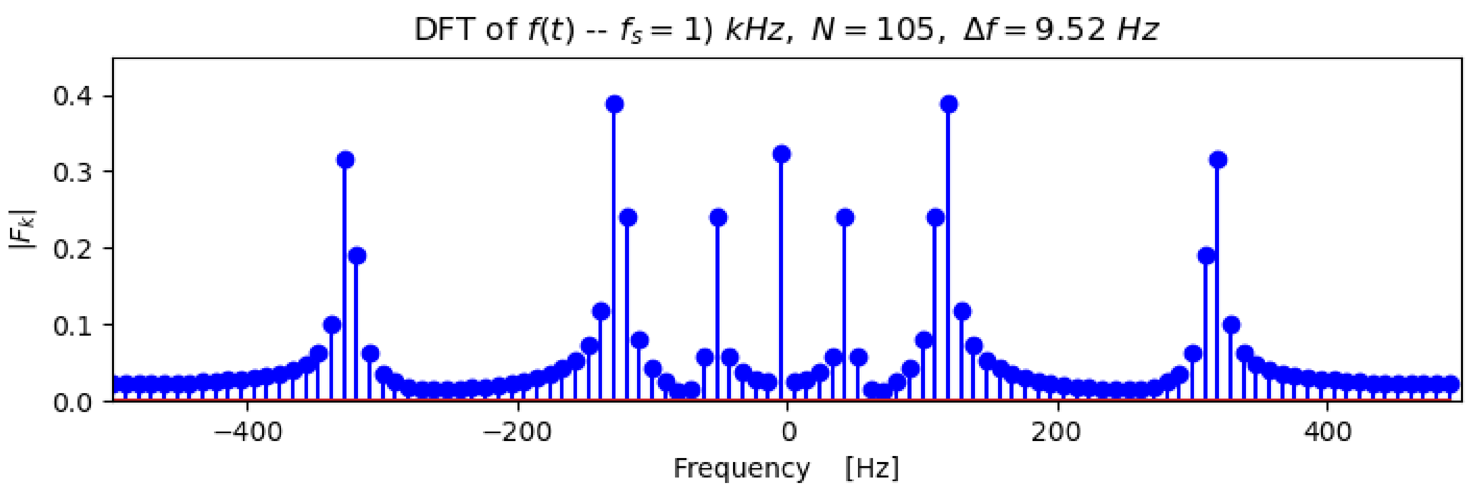
- Plotting magnitude of (real) F_k
- Components at 0, 50, 120, and $310 Hz$ are clearly visible
- Plot spectrum as a function of
  - Index value, k
  - Normalized frequency
  - Ordinary frequency
- F_k values divided by N so that F_0 is the average value of $f(t)$
  - Amplitude of other components given by the sum of F_k and F_{-k} magnitudes



Spectral Leakage

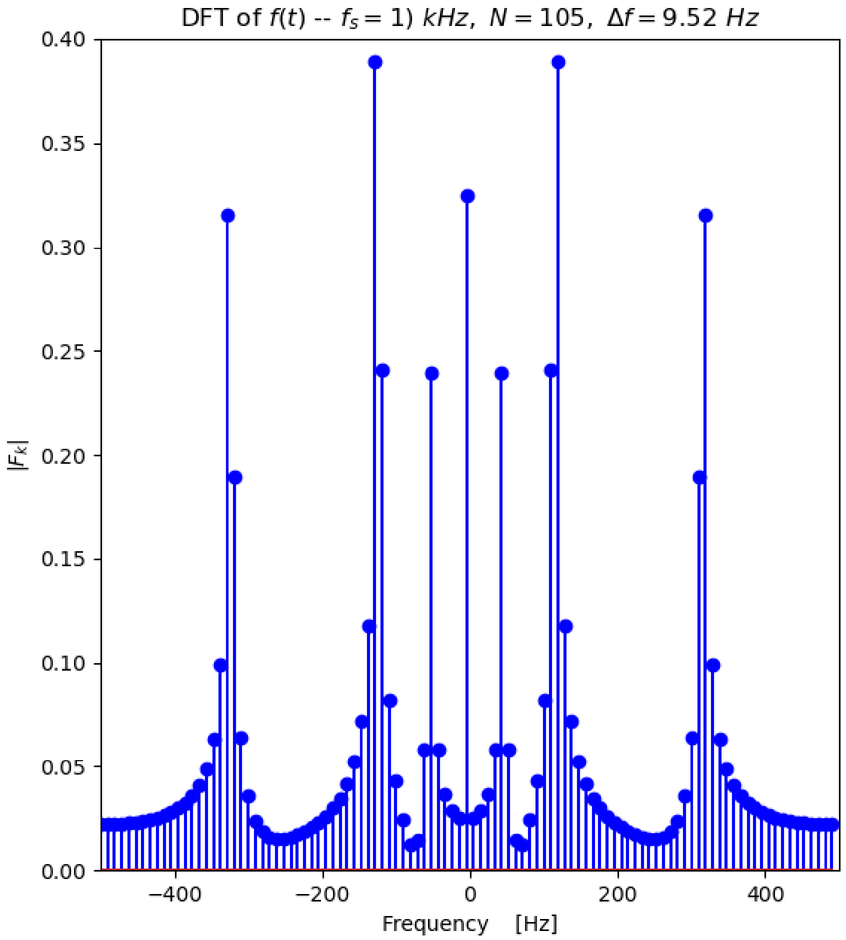
$$f(t) = 0.3 + 0.5\cos(2\pi \cdot 50 \cdot t) + \cos(2\pi \cdot 120 \cdot t) + 0.8\cos(2\pi \cdot 320 \cdot t)$$

- For $f_s = 1kHz$ and $N = 100$, $\Delta f = 10Hz$, and all signal components fall at integer multiples of Δf
  - All components lie in exactly one ***frequency bin***
- Now, increase the number of samples to $N = 105$
  - ***Bin width*** decreases to $\Delta f = 9.52Hz$
  - Each non-zero signal component now falls between frequency bins – ***Spectral Leakage***



Spectral Leakage

- Signal components now fall between ***two*** bins
- Why non-zero F_k over more than two bins?
  - ***Truncation*** (windowing)
- Finite record length is equivalent to ***multiplication*** of $f(t)$ by a ***rectangular pulse*** (window)
  - F.T. of pulse is a ***sinc***
  - Multiplication in the time domain → convolution in frequency domain
- Truncated signal is assumed periodic
  - True only if windowing function captures an integer number of periods of all signal components



Summary of Fourier Analysis Tools

	Time Domain	Frequency Domain
Fourier series	continuous periodic (or truncated)	aperiodic discrete
Fourier transform	continuous aperiodic	aperiodic continuous
DTFT	discrete aperiodic	periodic continuous
DFT	discrete periodic (or truncated)	periodic discrete

- In general:

Time Domain		Frequency Domain
Periodic	⟷	Discrete
Aperiodic	⟷	Continuous
Continuous	⟷	Aperiodic
Discrete	⟷	Periodic

DFT Algorithm

Implementing the DFT in Python

$$F_k = \sum_{n=0}^{N-1} f[n] e^{-jk2\pi n/N}$$

- A dot product of complex N-vectors for each of the N values of k

$$F_k = f[n] \cdot e^{-jk2\pi n/N}$$

  - Simple to code
  - N multiplications for each k value – N^2 operations
  - Inefficient, particularly for large N

```python
def dft(f):
    '''
    Computes the discrete Fourier transform
    of an arry, f.

    Parameters
    ----------
    f : N-vector for which to compute DFT

    Returns
    -------
    Fk : DFT of f - 1xN vector

    '''

    N = len(f)

    # initialize Fk as array pf complex zeros
    Fk = np.zeros(N, dtype=complex)

    # compute DFT
    n = np.arange(N)

    for k in range(N):
        Fk[k] = f @ np.exp(-1j*k*2*np.pi*n/N)

    return Fk
```

Fast Fourier Transform – FFT

- The fast Fourier transform (FFT) is a very efficient algorithm for computing the DFT
  - The Cooley-Tukey algorithm
- Requires on the order of $N\log_2(N)$ operations
  - Significantly fewer than N^2
- For example, for $N = 1024$:
  - DFT: $N^2 = 1{,}048{,}576$ operations
  - FFT: $N\log_2(N) = 10240$ operations – $(102 \times$ faster)
- Requires N be a power of two
  - If not, data record is padded with zeros

FFT in Python

It is very simple to implement a straight DFT algorithm in Python, but the FFT algorithm is, by far, more efficient .

FFT in Pyhton

- Both the NumPy and SciPy Python packages include many FFT-related functions
- Three most important to us:
  - `fft()`
  - `ifft()`
  - `fftshift()`
- All located in `numpy.fft` or `scipy.fft` modules
- Import to use, e.g.:

```
from scipy.fft import fft, ifft, fftshift
```

Fast Fourier Transform in Python – `fft()`


```
Xk = fft(x,n)
```

- x: vector of points for DFT computation
- n: *optional* length of the DFT to compute
- Xk: complex vector of DFT values – `len(x)` or an n-vector

- If n is specified, x will either be truncated or zero-padded so that its length is n
- If x is a matrix, the fft for each column of x is returned
- `fft()` uses the Cooley-Tukey algorithm
- Fastest for `len(x)` or n that are ***powers of two***

Inverse FFT in Python – `ifft()`


```
x = ifft(Xk,n)
```

- Xk: vector of points for inverse DFT computation
- n: *optional* length of the inverse DFT to compute
- `x`: complex vector of time-domain values – `len(Xk)` or an n-vector

- If n is specified, Xk will either be truncated or zero-padded so that its length is n
- `ifft()` uses the Cooley-Tukey algorithm
- Fastest for `len(Xk)` or n that are ***powers of two***

Shifting Negative Frequency Values – `fftshift()`


```
Xshift = fftshift(Xk)
```

- `Xk`: vector of FFT values with zero frequency point at `Xk[0]`
- `Xshift`: FFT vector with the zero-frequency point moved to the middle of the vector

- If N = `len(Xk)` is even, first and second halves of `Xk` are swapped
  - `Xshift = [Xk[N/2+1:N],Xk[1:N/2]]`
  - Frequency points are: $f = \left[-\frac{f_s}{2} \dots \left(\frac{f_s}{2} - \Delta f\right)\right]$
- If `N = length(Xk)` is odd, zero frequency point moved to the `Xshift[(N-1)/2]` position
  - `Xshift = [Xk[(N+3)/2):N],Xk[1:(N-1)/2]]`
  - Frequency points are: $f = \left[-f_s\frac{N-1}{2N} \dots f_s\frac{N-1}{2N}\right]$