DOING PHYSICS WITH PYTHON

FOURIER METHODS

FOURIER SERIES

Ian Cooper
matlabvisualphysics@gmail.com

DOWNLOAD DIRECTORIES FOR PYTHON CODE

Google drive

GitHub

simFM001.py **simFM001A.py** **simFM001B.py**

simFM001C.py **simFM001D.py**

INTRODUCTION

Any function which repeats itself regularly over a given interval of space or time is called a **periodic function**. This may be expressed by writing it as $f(x) = f(x + \alpha)$ where α is the interval or period and x and α are measured in radians. The simplest examples of a periodic functions are sines and cosines of fixed frequency or wavelength, where α is the period T, or the wavelength λ.

Any periodic function may be represented using the method of Fourier Series. Only cases which involve periodic functions in time will be considered. The ideas presented can be extended to all periodic functions. The period function $f(t)$ is characterized by its amplitude A, its period T_1 [s], frequency f_1 [Hz] and angular frequency ω_1 [rad.s^{-1}].

Fourier Series

$$(1) \quad f(t) = \tfrac{1}{2} a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\left(n \omega_1 t\right) + b_n \sin\left(n \omega_1 t\right) \right)$$

The function is a superposition of sine and cosine functions where $a_0/2$ is a constant (DC off-set), and sine and cosine terms of different amplitudes, having frequencies which increase in discrete steps. So, any periodic function can be constructed from a superposition of sinusoidal functions which have a discrete set of frequencies, $f_1, 2f_1, 3f_1, 4f_1, 5f_1, 6f_1, \cdots$

The frequencies in this discrete set are called **harmonics**. The **first harmonic** is f_1 and is called the **fundamental**. The higher harmonics frequencies are the second, third, fourth, ….

Such a series must of course, satisfy certain conditions, chiefly those of convergence. These convergence criteria are met for a function with discontinuities which are not too severe and with first and second differential coefficients which are well behaved. At such discontinuities the series represents the mean of the values of the function just to the left and just to the right of the discontinuity.

The cosine part of a Fourier series represents the even part of the function and the sine terms represent the odd part of the function. Taking the argument one stage further, a function $f(t)$ which is an even function is represented by a Fourier series having only cosine terms; if $f(t)$ is odd it will have only sine terms in its Fourier representation. Whether a function is completely even or completely odd can often be determined by the position of the Y-axis (reflections about t = 0).

The Fourier series will accurately represent the function only within the chosen interval. If applied outside that interval it will not follow the function but will periodically repeat the value of the function within the chosen interval. If we represent this

interval by a Fourier cosine series the repetition will be that of an even function, if the representation is a Fourier sine series an odd function repetition will follow.

Simulation 1 Square wave (odd function) simFM001.py

The Fourier series for a square wave function is modelled in Python. The square wave that is considered has a frequency f_0 = 0.5 Hz, and amplitude h = 1.2 m and a DC off-set h_0 = 0 m or 0.75 m. The Code below shows a clever way to produce the square wave from a sine or a cosine function.

```
# Frequency of square wave pulse [Hz]
f0 = 0.5
# Number of grid points
N = 999
# amplitude of square wave pulse [a.u.]
h = 1.2
# Time span for square wave  / time mesh  [s]
t1 = -4; t2 = 12
t = linspace(t1,t2,N)
w0 = 2*pi*f0
# Sine function / square wave pulse
h0 = 0.0      # DC off-set
fSine = sin(w0*t) + h0
```

fSquare = fSine

fSquare[fSine > h0] = h+h0

fSquare[fSine < h0] = -h+h0

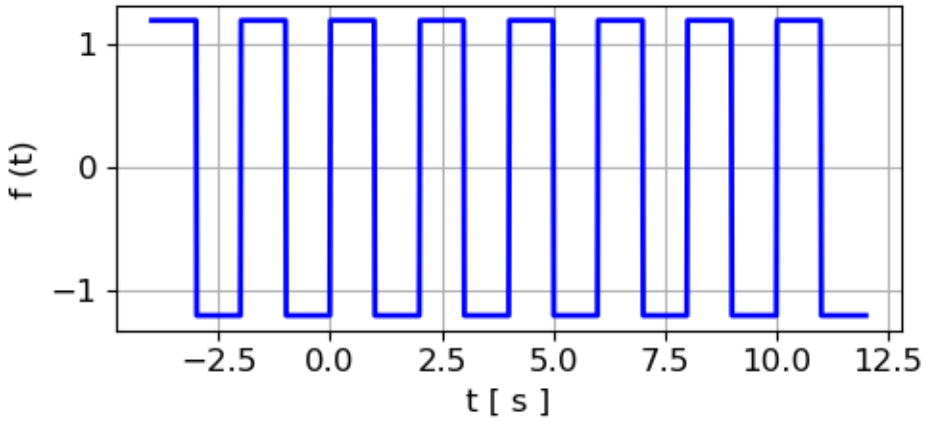


Fig. 1.1. Square wave generated from a sine function with frequency f_0 = 0.50 Hz. The amplitude of the square wave is h = 1.20 m and the DC off-set is h_0 = 0 m. From the reflection of the square wave about t = 0, the square wave is an odd function.

Since the square wave shown in figure 1.1 is an odd function, the Fourier series is expressed only in terms of the sine function. For this square wave the Fourier series is

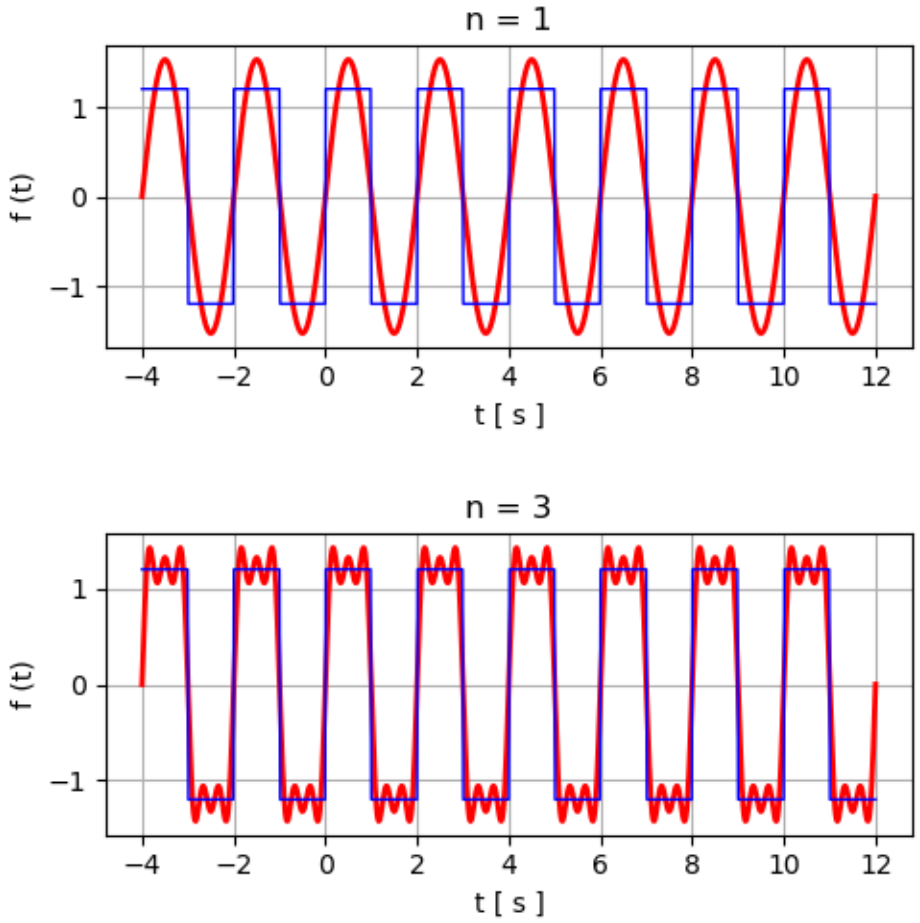
(1.1) $$f(t) = h_0 + \sum_{n=0}^{\infty}\left(\frac{1}{2n+1}\right)\left(\sin\left((2n+1)\omega_1 t\right)\right)$$

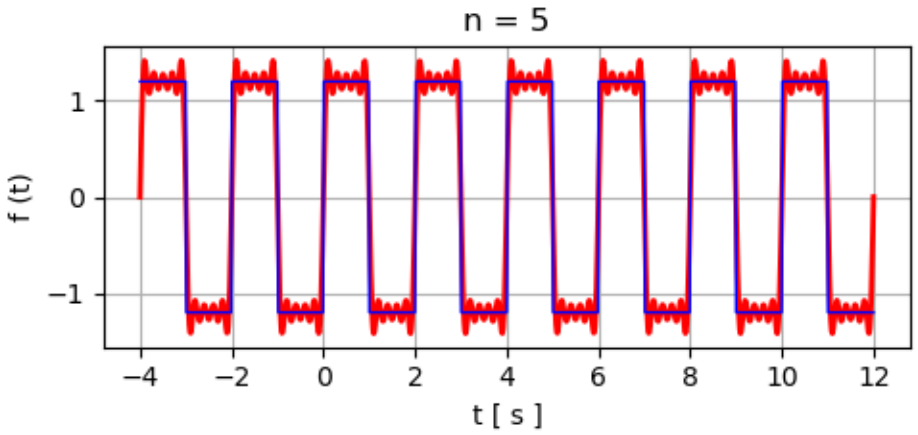
The first few terms of the series are:

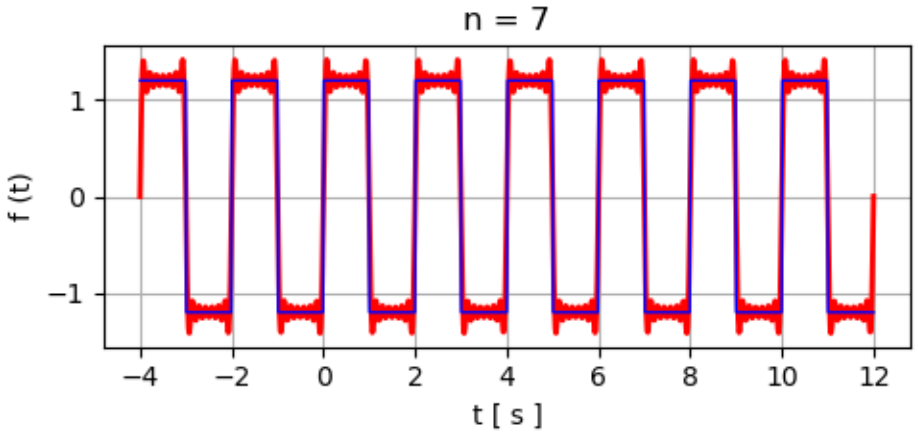
(1.2)

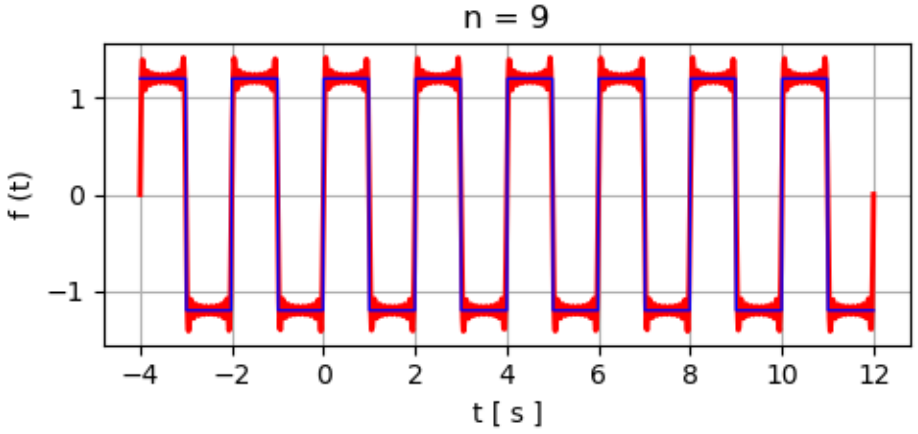
$$f(t) = h_0 + \left(\frac{4h}{\pi}\right)\left(\frac{\sin(\omega_1 t)}{1} + \frac{\sin(3\omega_1 t)}{3} + \frac{\sin(5\omega_1 t)}{5} + \frac{\sin(7\omega_1 t)}{7} + \cdots\right)$$

The more terms included in the Fourier series, the more accurate is the Fourier representation of the function. Figure 1.2 shows plots with increasing terms in equation 1.2.




n = 5
f (t)
1
0
−1
−4
−2
0
2
4
6
8
10
12
t [s]


n = 7
f (t)
1
0
−1
−4
−2
0
2
4
6
8
10
12
t [s]


n = 9
f (t)
1
0
−1
−4
−2
0
2
4
6
8
10
12
t [s]

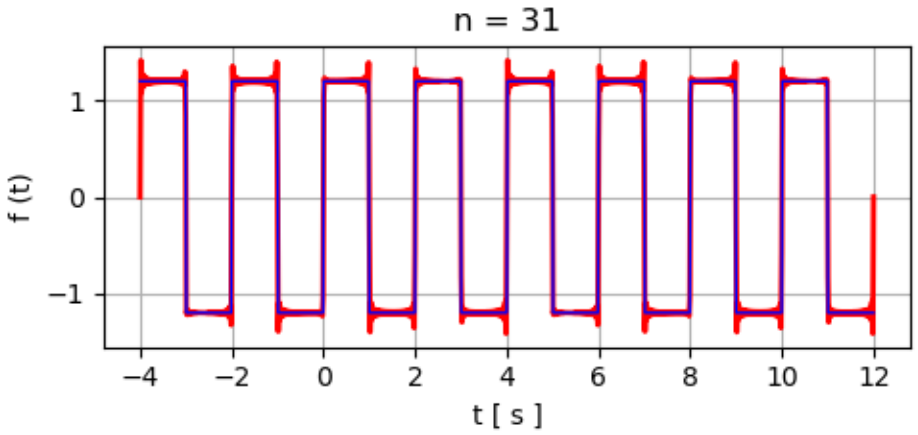


Fig. 1.2. Superposition of sinusoidal functions for the Fourier series of the square wave shown in figure 1.1.

The fundamental (first harmonic) has the frequency of the square wave and the higher frequencies build up the squareness of the wave. The highest frequencies are responsible for the sharpness of the vertical sides of the waves as shown in figure 1.2.

Fourier Transform

The **Fourier Transform** is a mathematical operation that converts a function or signal from the **time** (or spatial) **domain** into the **frequency domain**. Any sufficiently well-behaved signal can be expressed as a combination of sine and cosine waves of different frequencies.

The Fourier Transform answers:

- Which frequencies are present?
- How strong is each frequency?
- What is the phase of each frequency component?

Fourier analysis is widely used in audio processing, image compression, communications, radar and sonar, medical imaging (MRI, CT), signal processing.

The Python code **simFM001.py** computes the Fourier transform of the square wave by the direct integration of the Fourier integral.

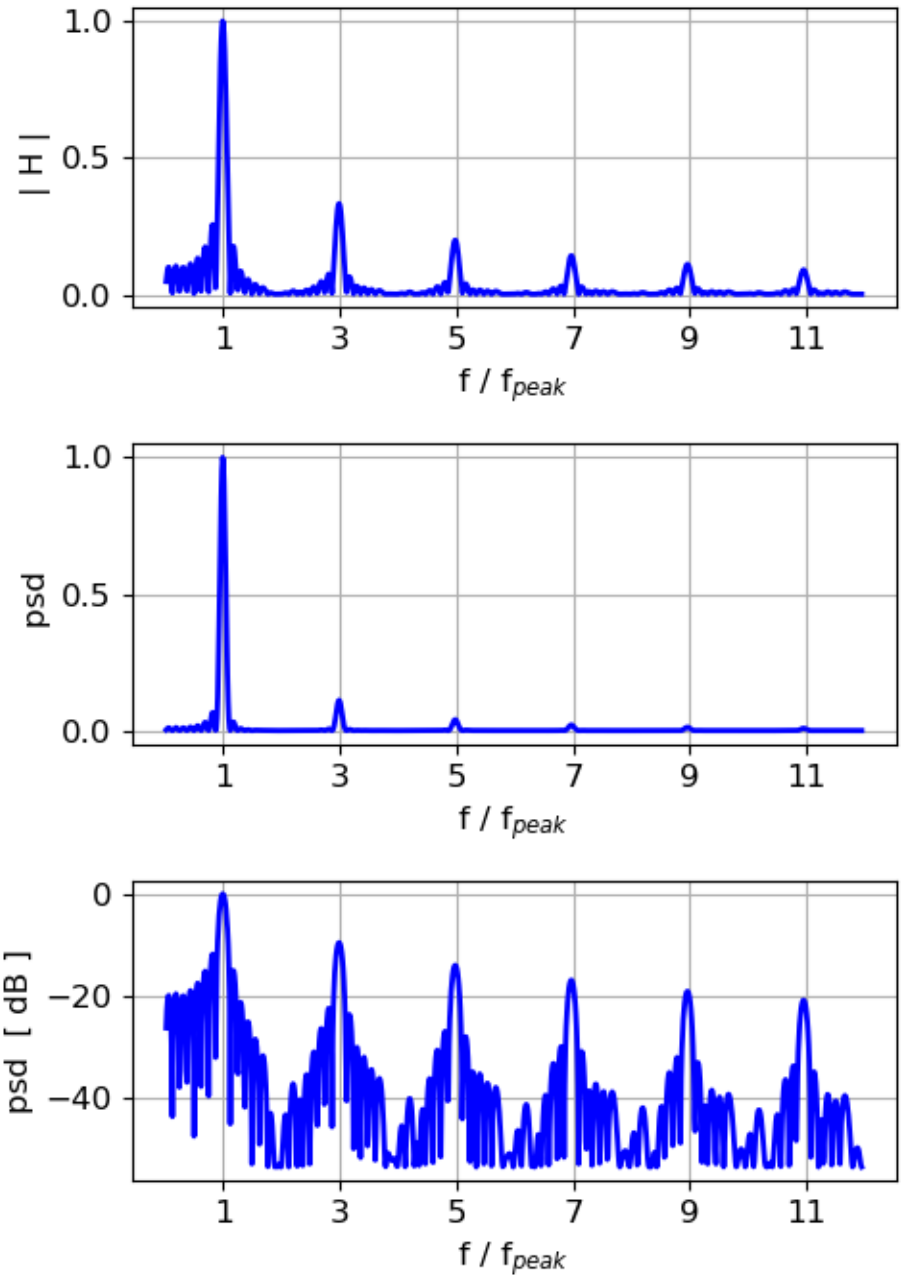


Fig. 1.3. Fourier transform of the square wave of figure 1.1. The frequency spectrum shows only the **odd** harmonics are excited $\left(1f_1,\ 3f_1,\ 5f_1, 7f_1, \cdots\right)$ and the amplitudes of the harmonics agree with equation 1.2 (1, 1/3, 1/5, 1/7 …).

We can add an off-set to the square wave. The figures below show the simulation results for the off-set h_0 = 0.75 m.

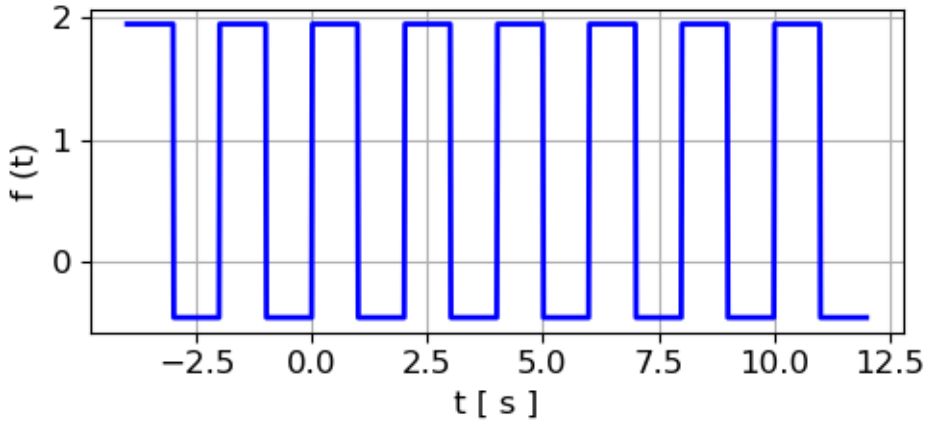


Fig 1.4. Square wave (odd) with off-set h_0 = 0.75 m off-set.

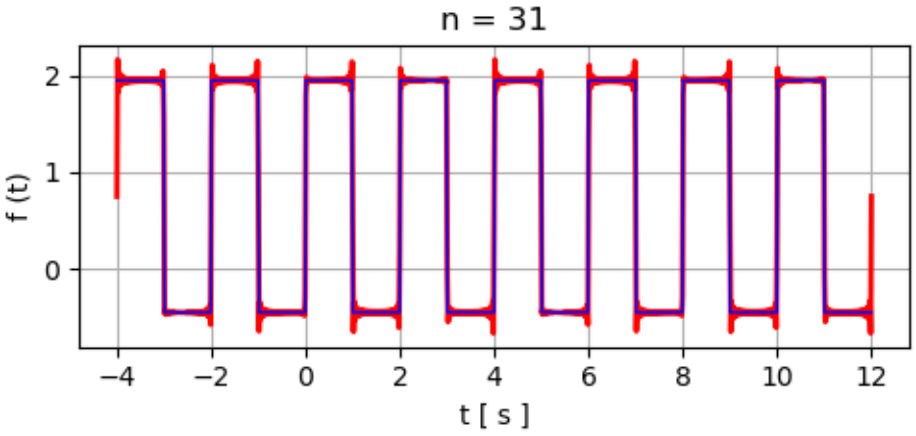


Fig. 1.5. Fourier series for the square wave with 31 terms.

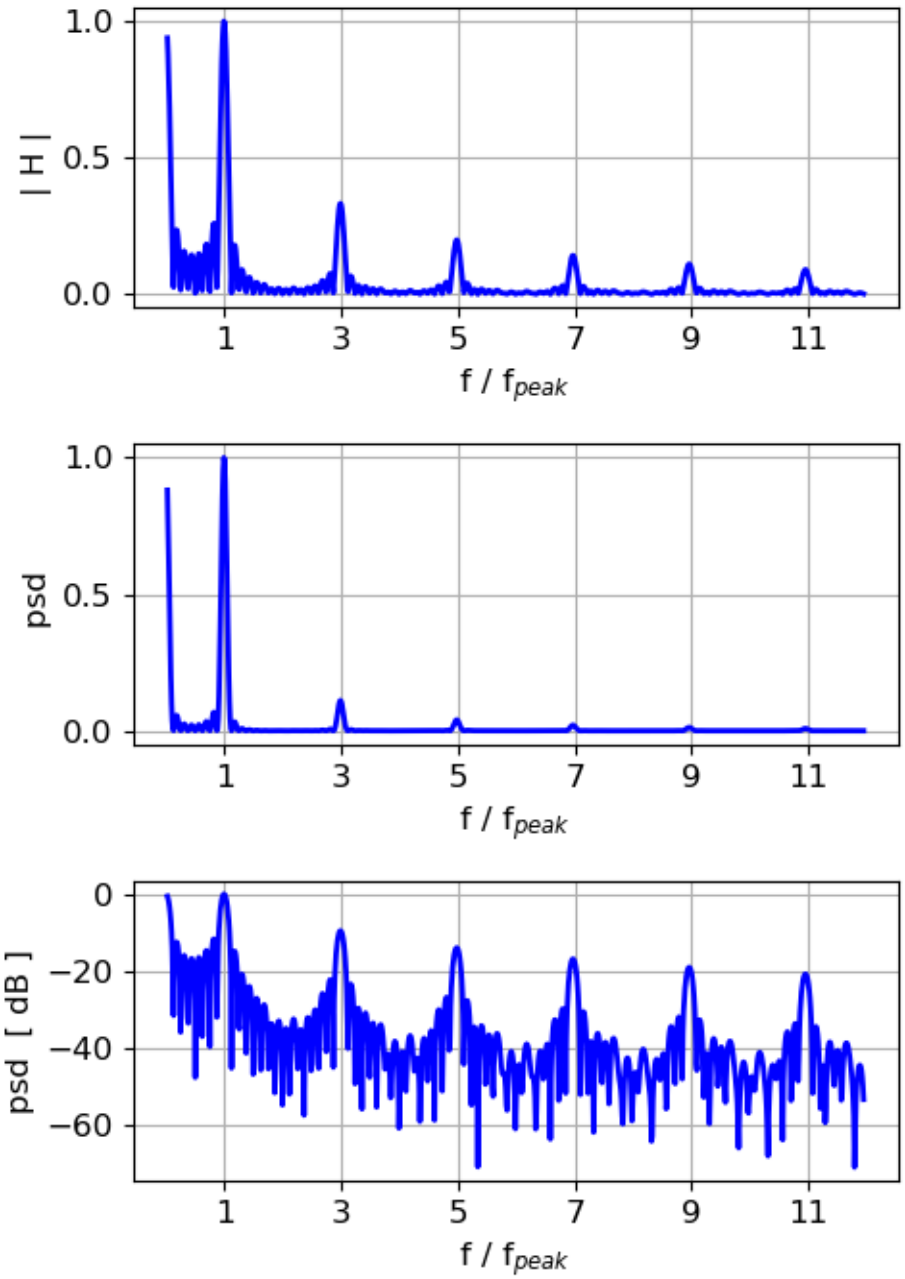


Fig. 1.6. Fourier transform of the square wave shown in figure 1.14. The frequency spectrum shows the DC component due to the off-set.

Simulation 2 Square wave (even function) simFM001A.py

Figure 2.1 shows a square wave as an even. The Fourier series is expressed only in terms of the cosine function. For this square wave the Fourier series is

$$(2.1) \quad f(t) = h_0 + \sum_{n=0}^{\infty} \left(\frac{(-1)^n}{2n+1} \right) \left(\cos\left((2n+1)\omega_1 t \right) \right)$$

The first few terms of the series are:

(2.2)

$$f(t) = h_0 + \left(\frac{4h}{\pi} \right) \left(\frac{\cos(1\omega_1 t)}{1} - \frac{\cos(3\omega_1 t)}{3} + \frac{\cos(5\omega_1 t)}{5} - \frac{\cos(7\omega_1 t)}{7} + \cdots \right)$$

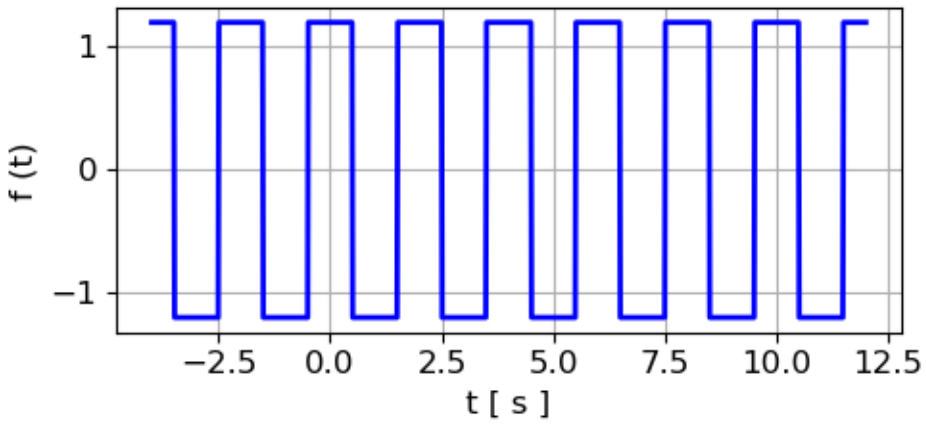


Fig. 2.1. Even function square wave.

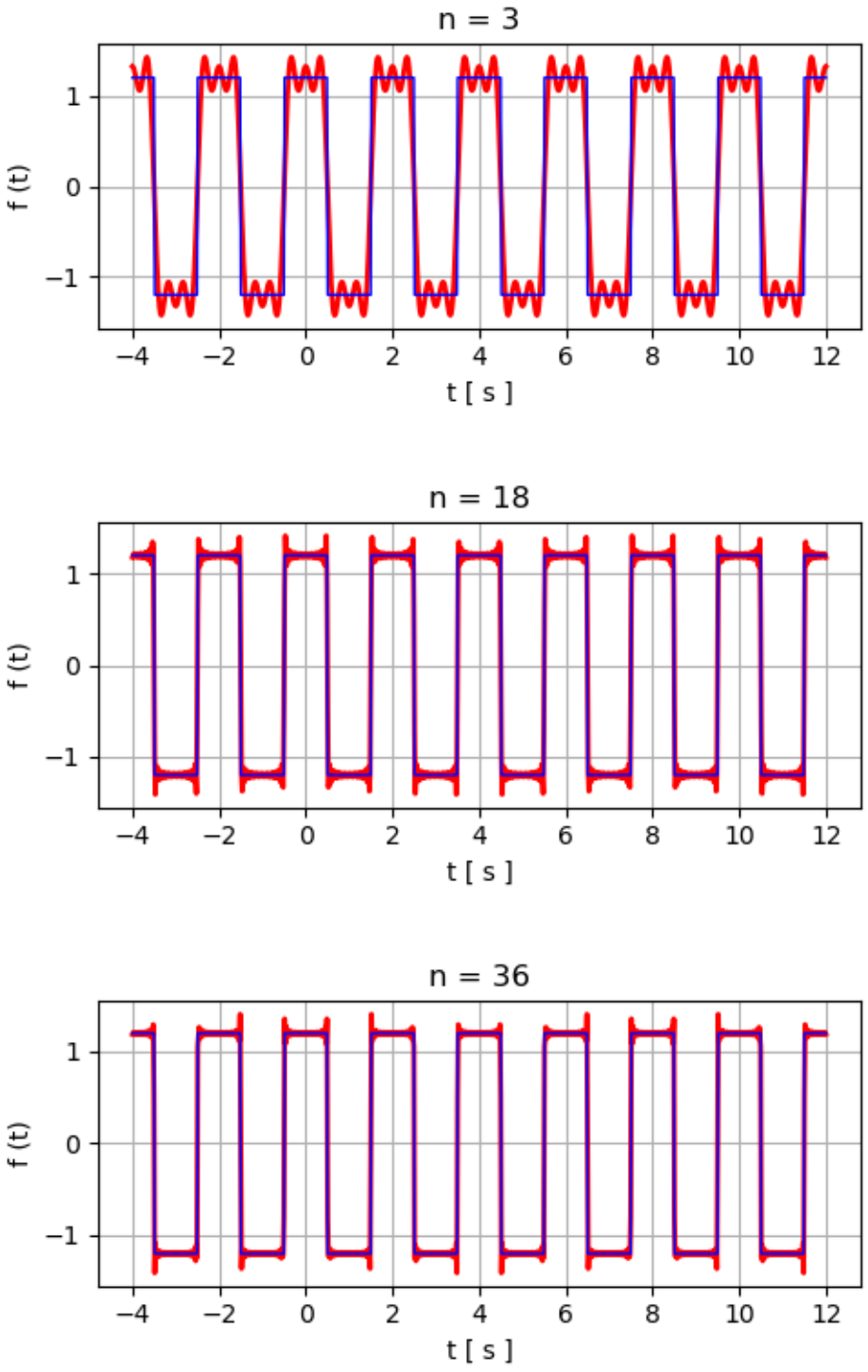


Fig. 2.2. Fourier series reconstruction of the even square wave shown in figure 2.1.

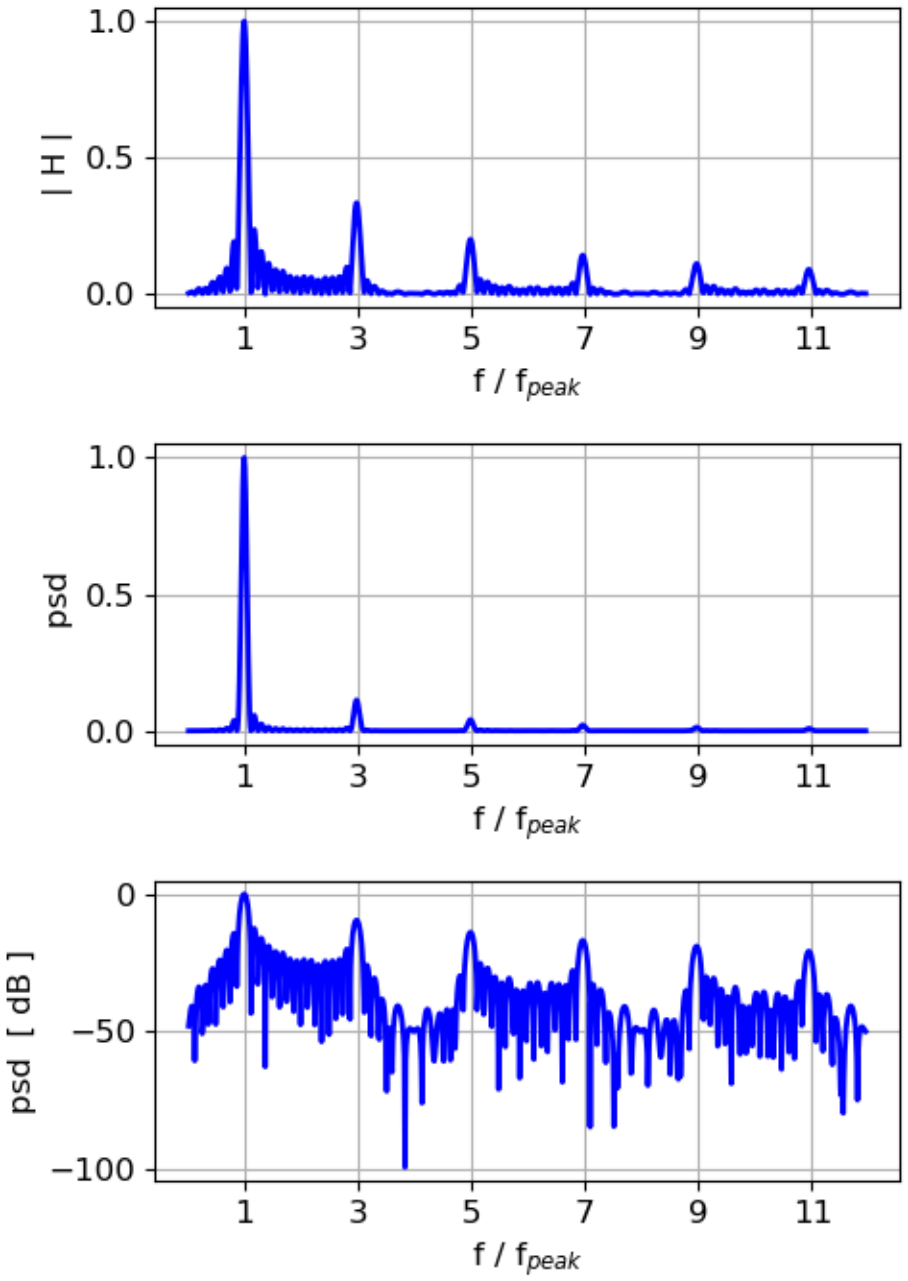


Fig. 2.3. Fourier transform of the even square wave shown in figure 2.1. Only the odd harmonics are included in the Fourier series.

Simulation 3 Sawtooth wave (odd function) simFM001B.py

You can generate and compare a **sawtooth wave** using its Fourier series and the SciPy signal library in Python.

Fourier series: Sawtooth wave

A sawtooth wave can be represented by the Fourier series:

$$f(t) = \frac{2}{\pi} \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \sin(n\omega_1 t)$$

where:

- t= time
- f_1= fundamental frequency
- $\omega_1 = 2\pi f_1$
- n= harmonic number

Python's scipy.signal library provides a built-in **sawtooth()** function.

```
# INPUTS
# Number of time steps
Nt = 999
# Number of fourier components to sum
N = 15
# time span
t1 = 0; t2 = 2
```

```
# fundamental period
T1 = 0.5
# amplitude of wave
h = 1.5

# SETUP PARAMETERS
# Time mesh
t = linspace(t1,t2,Nt)
# Frequency
f1 = 1/T1; w1 = 2*pi*f1
# sawtooth phase angle S
S = w1*t+pi
# width: sawtooth (1) or triangluar (0.5)
width = 1
# Sawtooth function  fn
fn = h*signal.sawtooth(S,width)
```

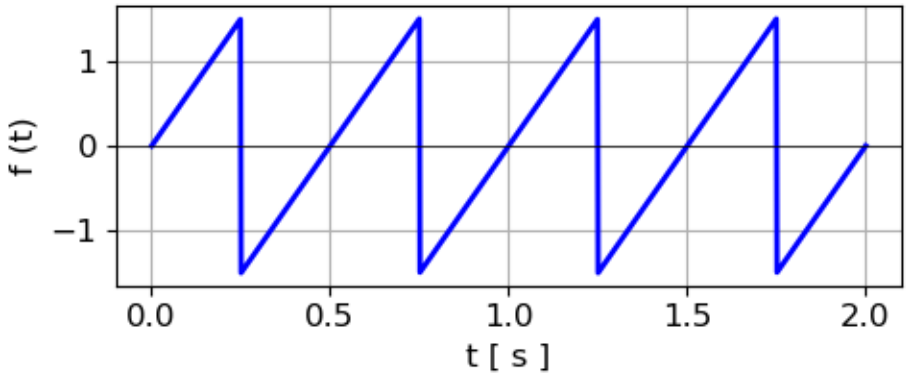


Fig. 3.1. Sawtooth wave form with amplitude h = 1.5, fundamental period T_1 = 0.50 s, and fundamental frequency f_1 = 2.0 Hz. The variable S for the singnal.sawtooth includes a phase shift of π rad so that at t = 0, $f(t)$ = 0. The variable $width$ = 1 gives a sawtooth function.

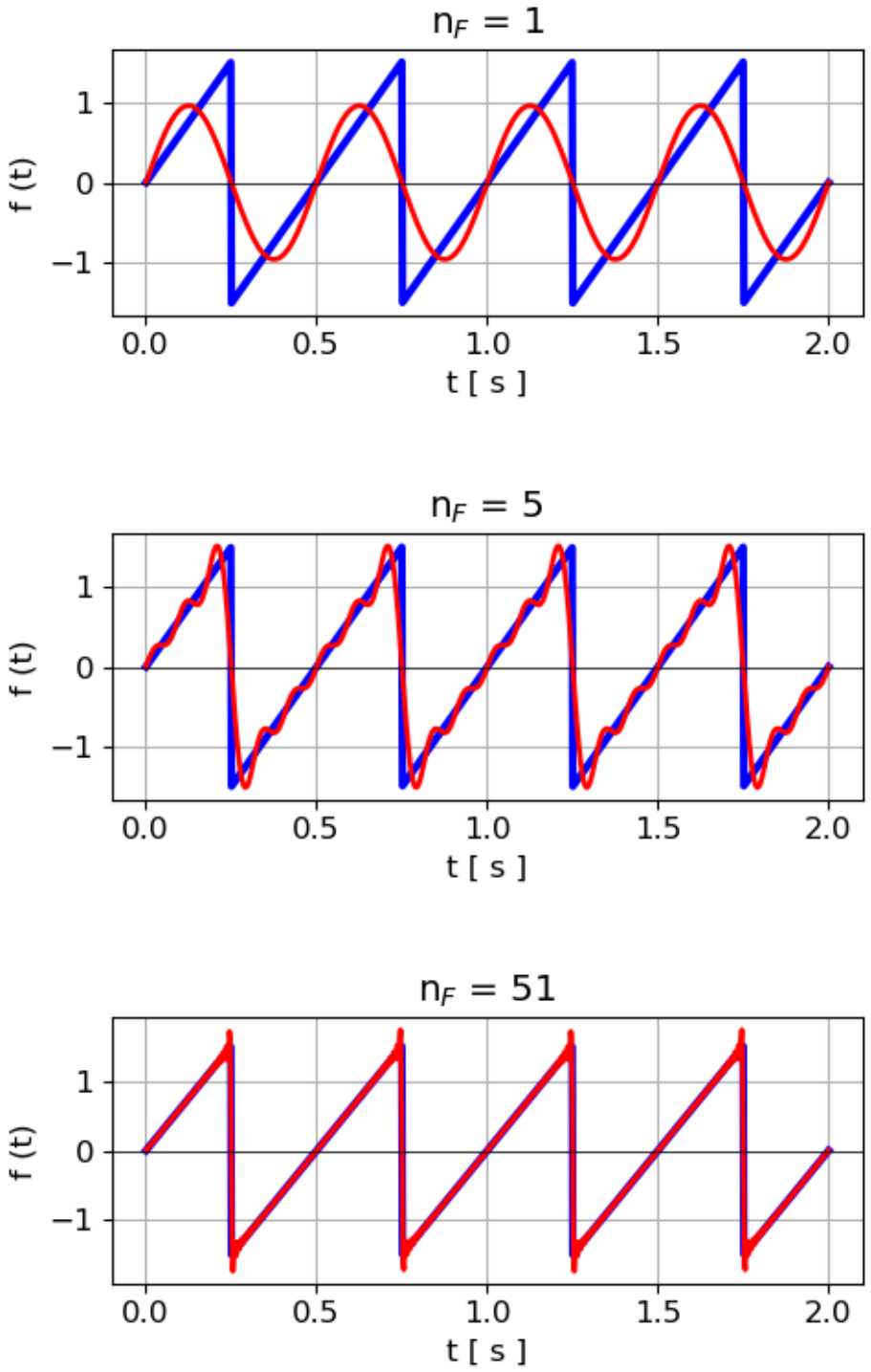


Fig. 3.2. Fourier series with increasing number of harmonics. The fundamental period is T_1 = 0.50 s.

The Fourier series has small oscillations near the discontinuities. This is called the **Gibbs phenomenon**. Increasing the number of harmonics makes the approximation better, but the overshoot does not completely disappear.

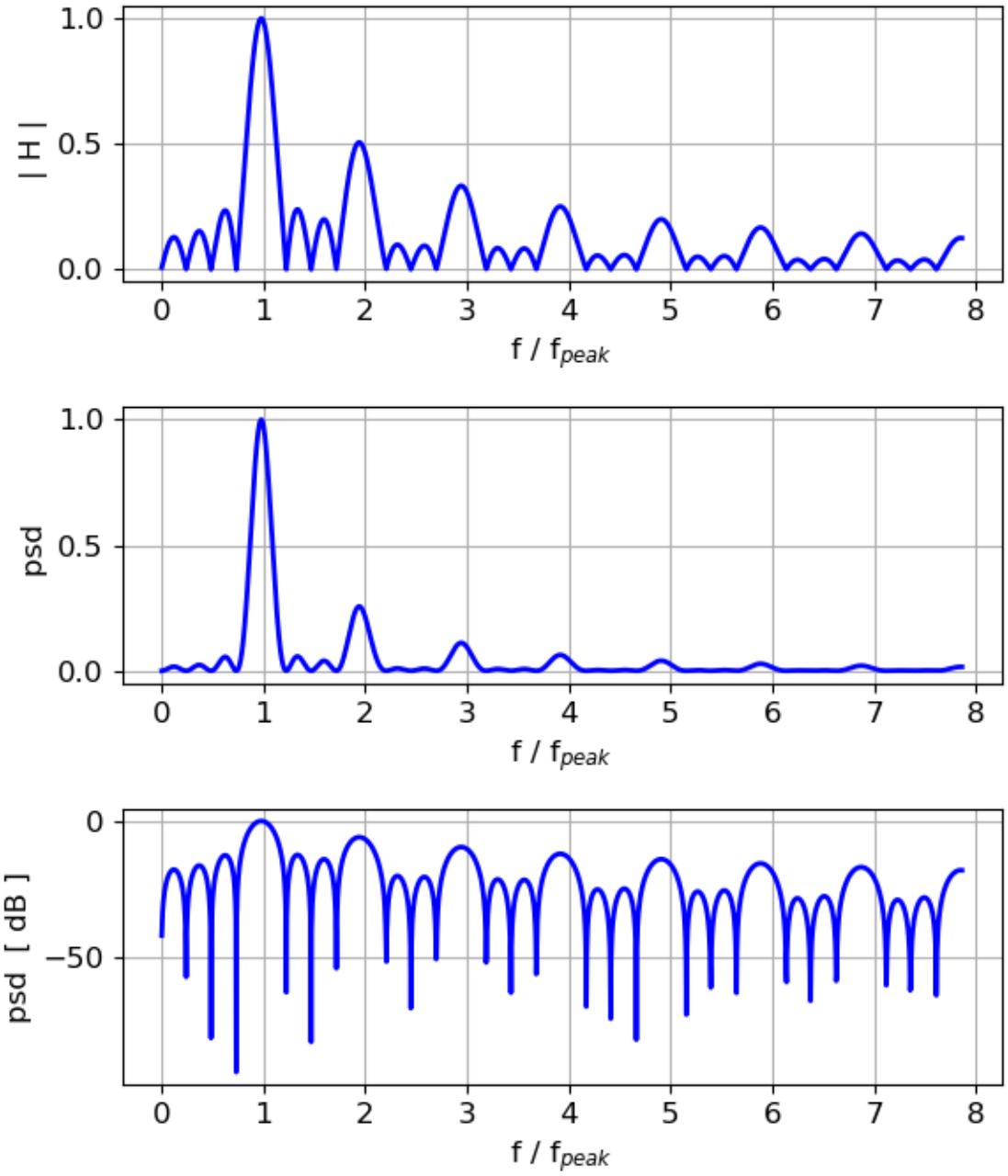


Fig. 3.3. Fourier transform of the sawtooth wave.

The Fourier series for the sawtooth wave is a superposition of all the harmonics, 1, 2, 3. …

The Fourier series coefficient for the third harmonic is

$$n = 3 \qquad c_3 = \left(\frac{2h}{3\pi} \right) = \left(\frac{(2)(1.5)}{3\pi} \right) = 0.318$$

The third harmonic peak from the Fourier transform is

$$n = 2.95 \qquad c_3 = 0.333$$

Simulation 4 Triangular wave (odd function) simFM001C.py

You can generate and compare a **triangular wave** using its Fourier series and the SciPy signal library in Python.

A triangle wave with fundamental period T_1 that has odd symmetry contains only odd sine harmonics, and its amplitudes drop off as the inverse square of the harmonic number n

$$A_n \propto 1/n^2 \quad n = 1,3,5, \cdots$$

For a symmetric odd triangle wave with angular frequency ω_1 the Fourier series is

$$f(t) = \frac{8h}{\pi^2} \sum_{n=1}^{\infty} \frac{(-1)^{(n-1)}}{(2n-1)^2} \sin\left((2n-1)\omega_1 t\right)$$

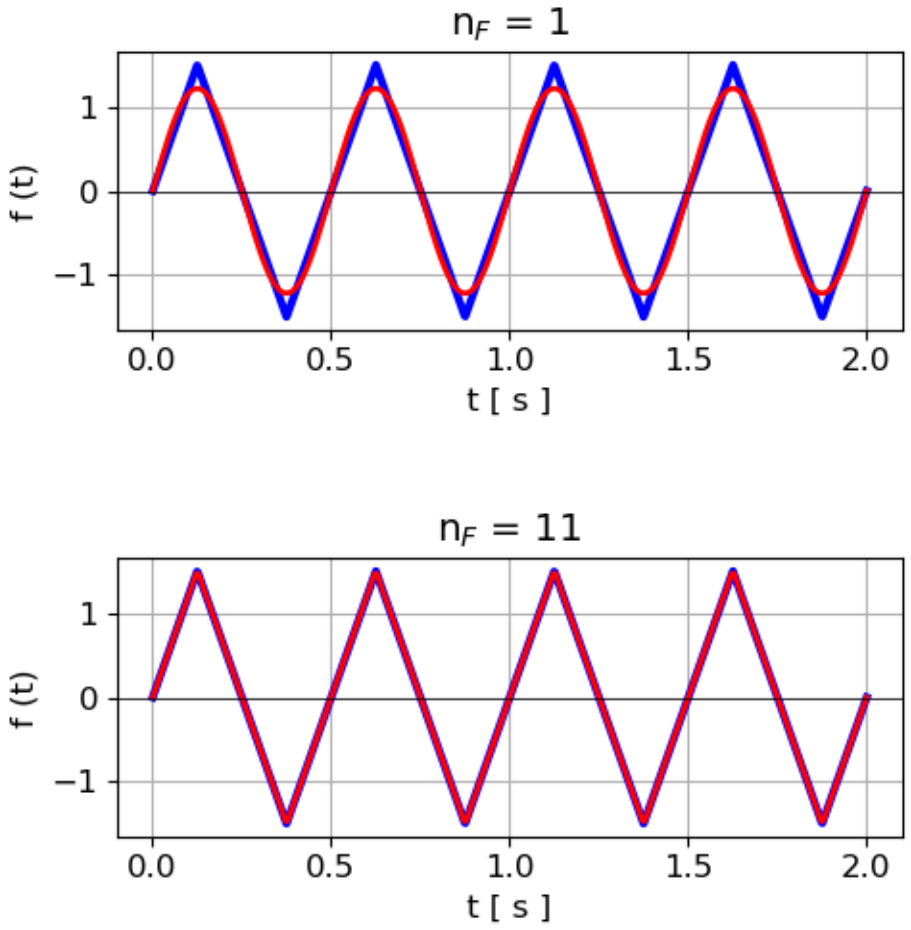


Fig. 4.1. The Fourier series converges very rapidly to the triangular waveform with increasing values of the harmonic number n.

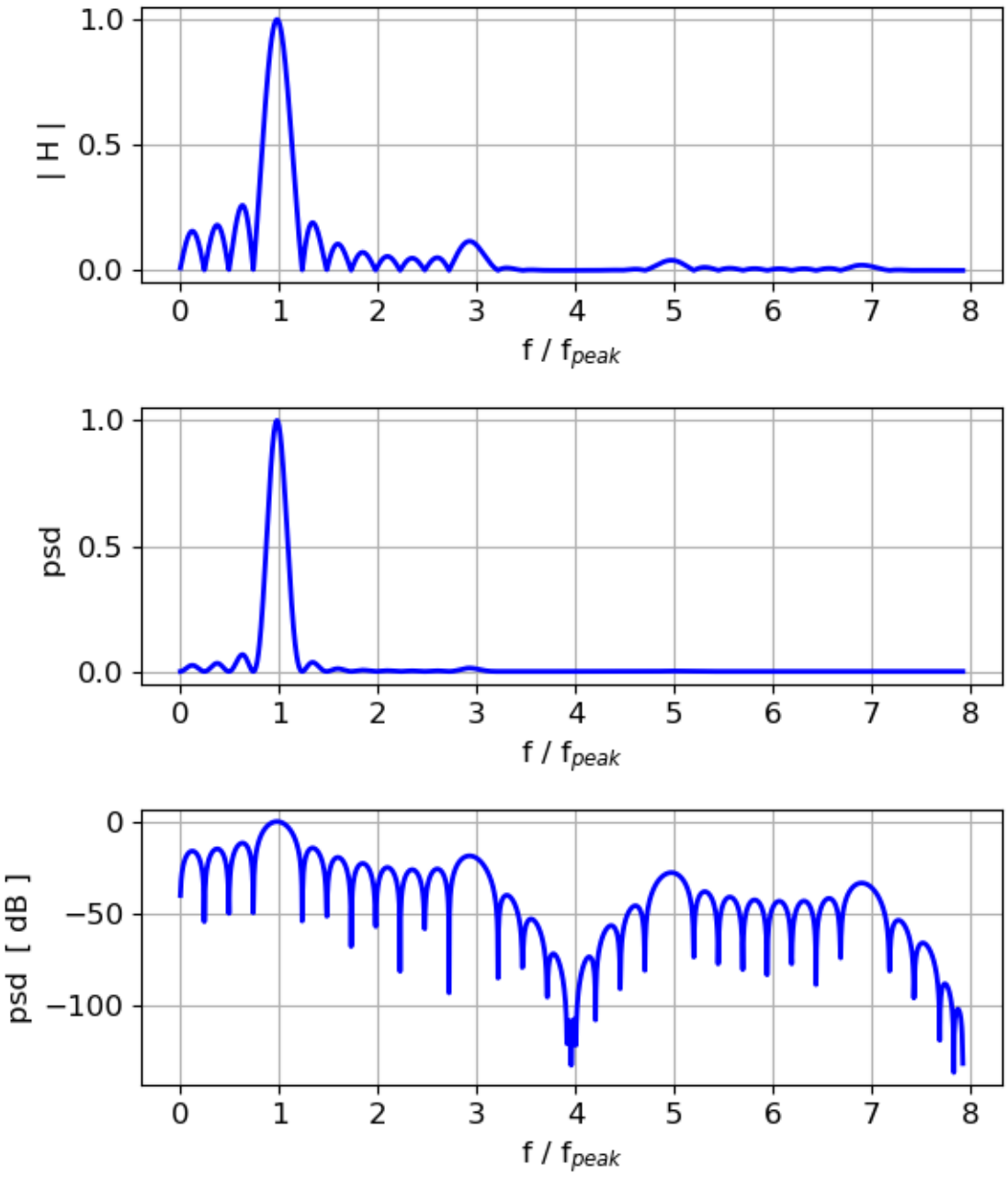


Fig. 4.2. Fourier transform of triangular waveform. The triangular waveform is a superposition of only the odd harmonics (sine functions).

Simulation 5

Numerical estimation of the Fourier series coefficients

simFM001D.py

A completely numerical approach to computing the Fourier series of a function is often more straight forward than an analytical approach.

Fourier Series n = 1, 2, 3, …

$$(1) \quad f(t) = \tfrac{1}{2}a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\left(n\,\omega_1 t\right) + b_n \sin\left(n\,\omega_1 t\right)\right)$$

Fourier series term for the harmonic number n

$$f_n(t) = a_n \cos\left(n\,\omega_1 t\right) + b_n \sin\left(n\,\omega_1 t\right)$$

To find the values of the coefficients a_n and b_n let us multiply both sides of equation 1 by $\cos\left(n\,\omega_1 t\right)$ and integrate with respect to t over the period T_1, this gives

$$a_n = \frac{2}{T_1}\int_0^{T_1} f(t)\cos\left(n\,\omega_1 t\right)dt \quad b_n = \frac{2}{T_1}\int_0^{T_1} f(t)\sin\left(n\,\omega_1 t\right)dt$$

The integrals can be evaluated numerically in Python using the function **simpson()**.

Two cases are considered: the triangular wave and the sawtooth wave with fundamental period T_1 = 1.0 s and amplitude h = 1.5.

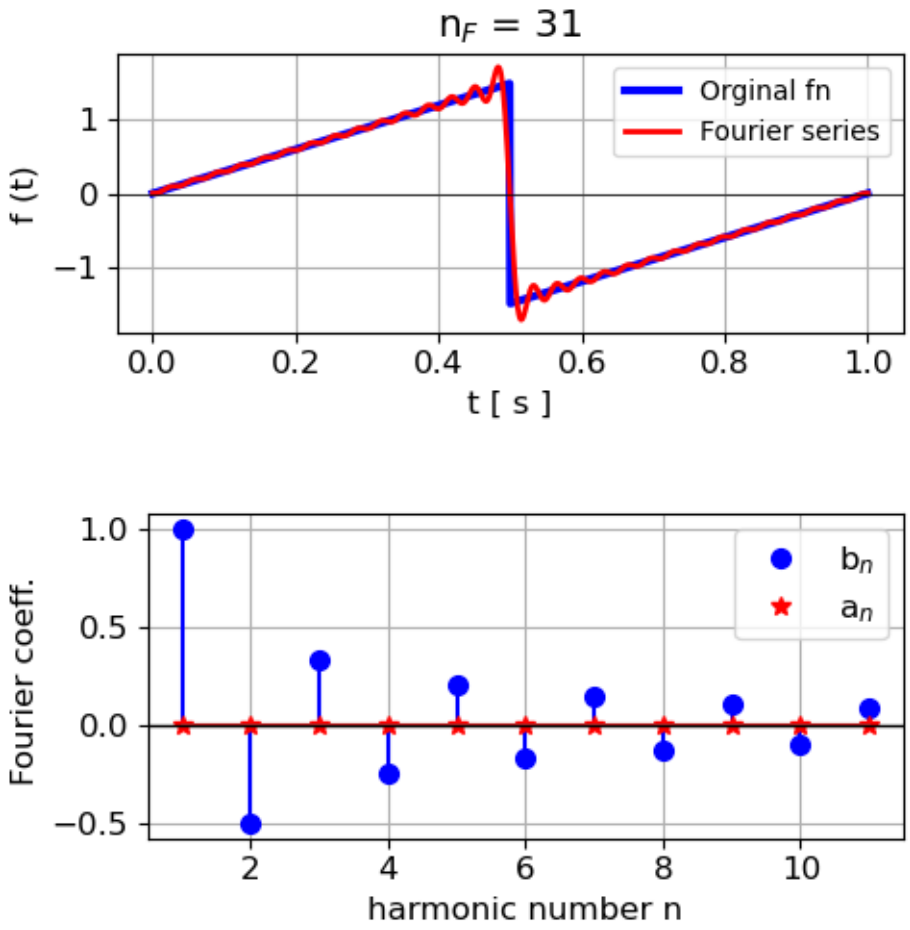


Fig. 5.1. Fourier series for the sawtooth function. The sawtooth function is an odd function. Therefore, the cosine coefficients a_n are all zero. The sawtooth function is a superposition of all the harmonics n = 1, 2, 3, … .

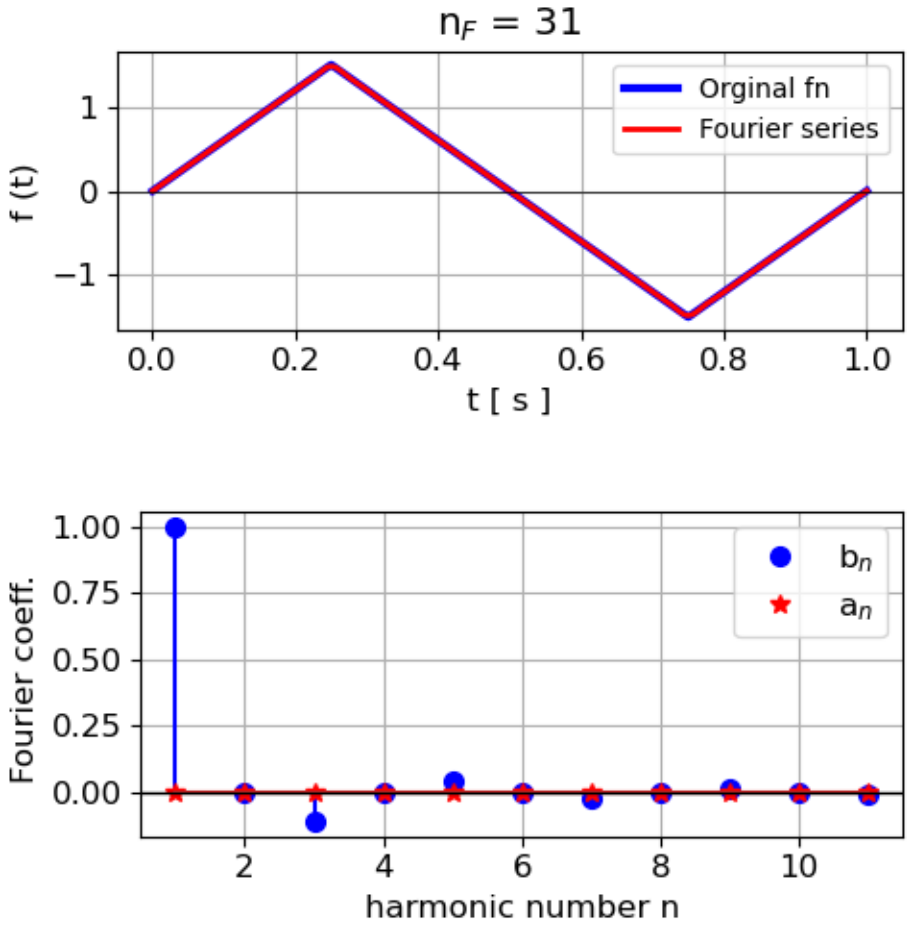


Fig. 5.2. Fourier series for the triangular function. The triangular function is an odd function. Therefore, the cosine coefficients a_n are all zero. The sawtooth function is a superposition of all the odd harmonics n = 1, 3, 5, … . Notice the rapid decline in the magnitudes of the b_n coefficients with increasing n values.