1. Low-pass Filter Transfer Function

- The low-pass filter transfer function in Laplace domain is

$$H(s) = \frac{\omega_0}{s + \omega_0}$$

- Arbitrary selected cut-off frequency is

$$\omega_0 = 2 \cdot \pi \cdot 5 = 31.41 rad$$

- The transfer function for the low-pass filter is computed using `signal.TransferFunction` class from signal module of Python's scipy library
- The Bode plot shows the frequency response of transfer function $H(s)$
  - Low frequencies are not attenuated (this is the *pass band*)
  - High frequencies are attentutated (this is the *stop band*)

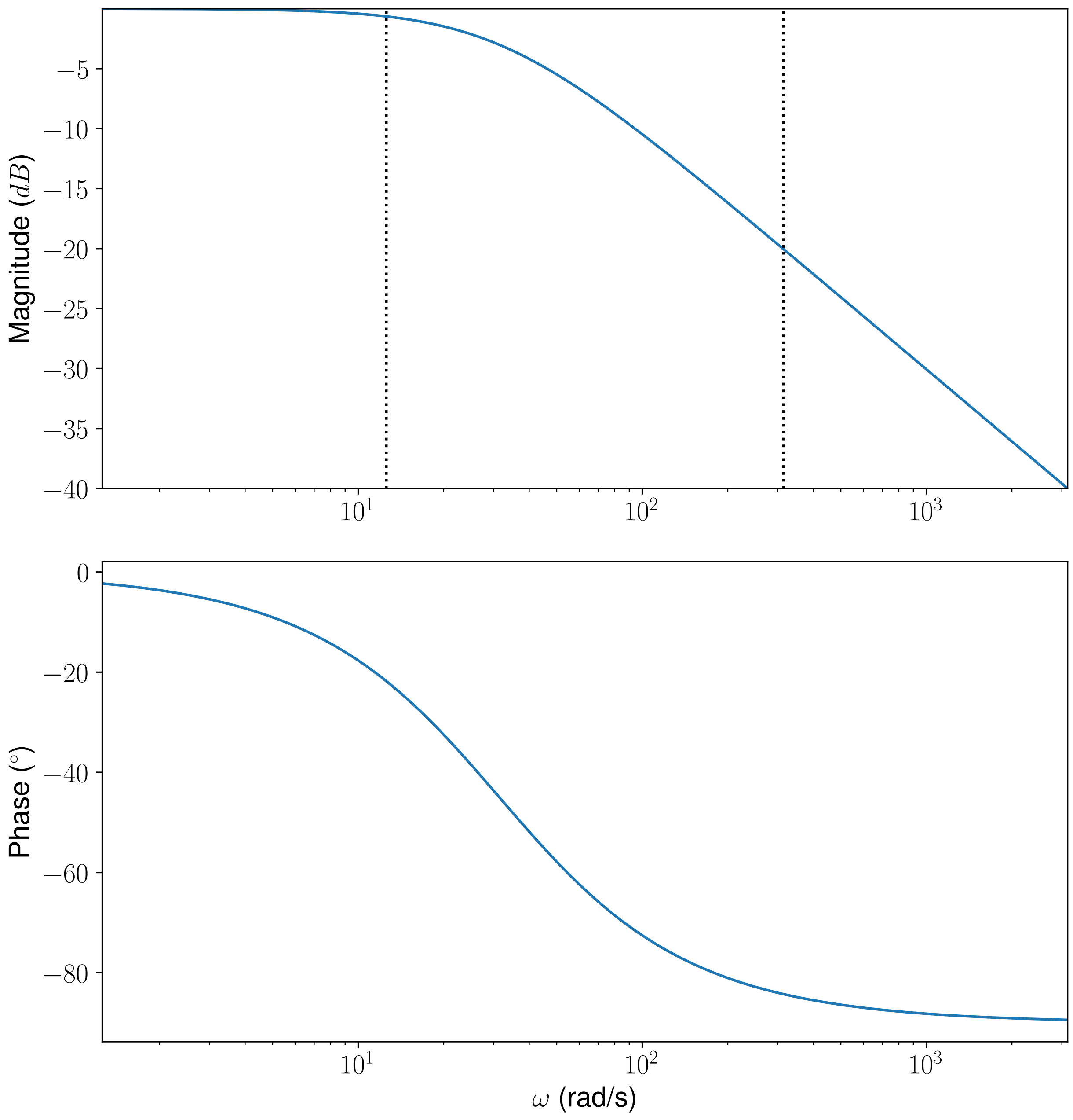

Figure 1: Bode Plot

2. Transformation from Continuous Laplace Domain to Discrete Time Domain

To implement the Low-pass Filter as a part of software for embeded system, the transformation from Laplace continues domain to time descrite domain shall be conducted. Tustin's discretization method (also known as bilinear approximation) is chosen amoung known continuous-discrete conversion methods. This method yields the best match between the continuous-time and discretized systems.

- Arbitrary chosen time step is $\Delta t = 0.001 sec$.
- Computing the discrete transfer function using Tustin's method

$$s = \frac{2}{\Delta t}(\frac{1-z^{-1}}{1+z^{-1}})$$

Substituting s in transfer function with the above

$$H(z) = \frac{\omega_0}{\frac{2}{\Delta t}\frac{1-z^{-1}}{1+z^{-1}} + \omega_0} = \frac{\Delta t\omega_0(z+1)}{(\Delta t\omega_0 + 2)z + \Delta t\omega_0 - 2}$$

- The `to_discrete` method of signal class that is a part of scipy Python's library used to compute the bilinear transform (Tustin's method). The below coeficients of diferential equation are calculated:
  TransferFunctionDiscrete
  (
  array([0.01546504, 0.01546504]), - numrator
  array([1. , -0.96906992]), - denumerator
  dt: 0.001 - discretization time
)

The discrete form of the diferential equation is:

$$y[n] = a_1 y[n-1] + a_2 y[n-2] + ... + b_0 x[n] + b_1 x[n-1] + ...$$

Where $y[n]$ - is filter output at sample time n, and $x[n]$ - is raw signal i.e. input of the filter at sample time n.
Important: The coeficients a are the values in denumerator array that is returned with TransferFunctionDiscrete function i.e. array ([1. , -0.96906992]). However these coeficents have to be applied with opposite sign to the differntial equation,

$$a_0 y[n] + a_1 y[n-1] + a_2 y[n-2] + ... + b_0 x[n] + b_1 x[n-1] + ... = 0$$

$$-1y[n] + 0.96906992y[n-1] + 0y[n-2] + ... + b_0 x[n] + b_1 x[n-1] + ... = 0$$

The coeficients in the numerator are applied as they are:

$$a_0 y[n] + a_1 y[n-1] + a_2 y[n-2] + ... + b_0 x[n] + b_1 x[n-1] + ... = 0$$

$$a_0 y[n] + a_1 y[n-1] + a_2 y[n-2] + ... + 0.01546504x[n] + 0.01546504x[n-1] + ... = 0$$

The final representation of the diferential equaiton is

$$y[n] = 0.96906992y[n-1] + 0.01546504x[n] + 0.01546504x[n-1]$$

The coeficients for different cut-off frequency and time step can easily be recalculated using LowPassFilter.ipynb Python program changing the initial values of $w0$ and dt variables.

Test Signal Generation

To check the response of discrete time Low-pass Filter implementation, the function that consists of sine function with arbitrary chosen amplitude $m_0 = 1$ and frequency $f_0 = 2Hz$ to represent signal to be filteres and since function with arbitrary chosen frequency $f_1 = 50Hz$ and amplitude $m_1 = 0.2$ to represent the noise

$$y(t) = m_0 \sin(2\pi f_0 t) + m_1 \sin(2\pi f_1 t)$$

The test signal is represented with the below figure as signal in time domain and its magnitude in frequency domain using Discrete Fourier Transform (DFT)

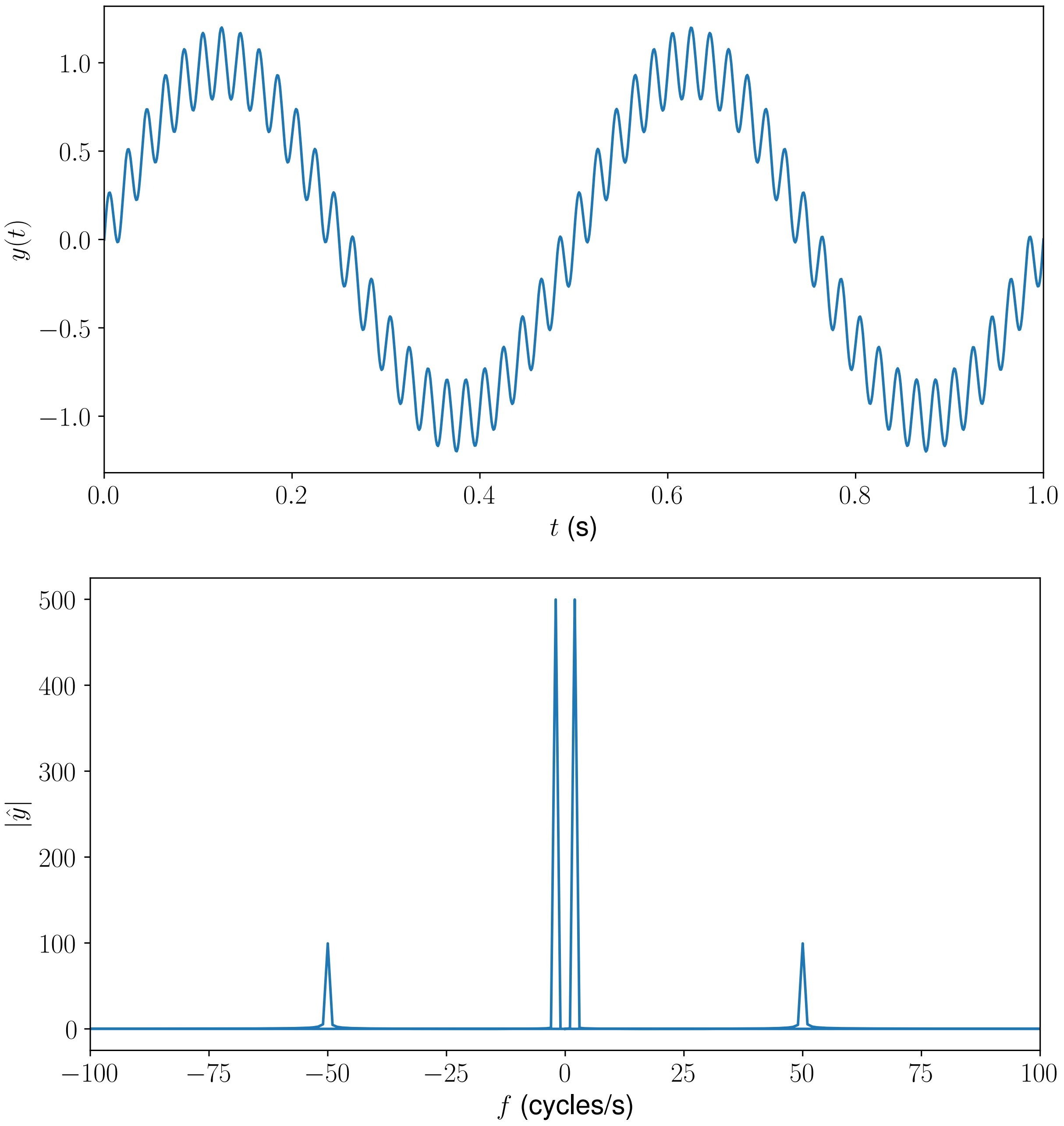

Figure 2: Test Signal

Testing the Low-pass Filter

The result of filtering the test signal is shown in the below plot:

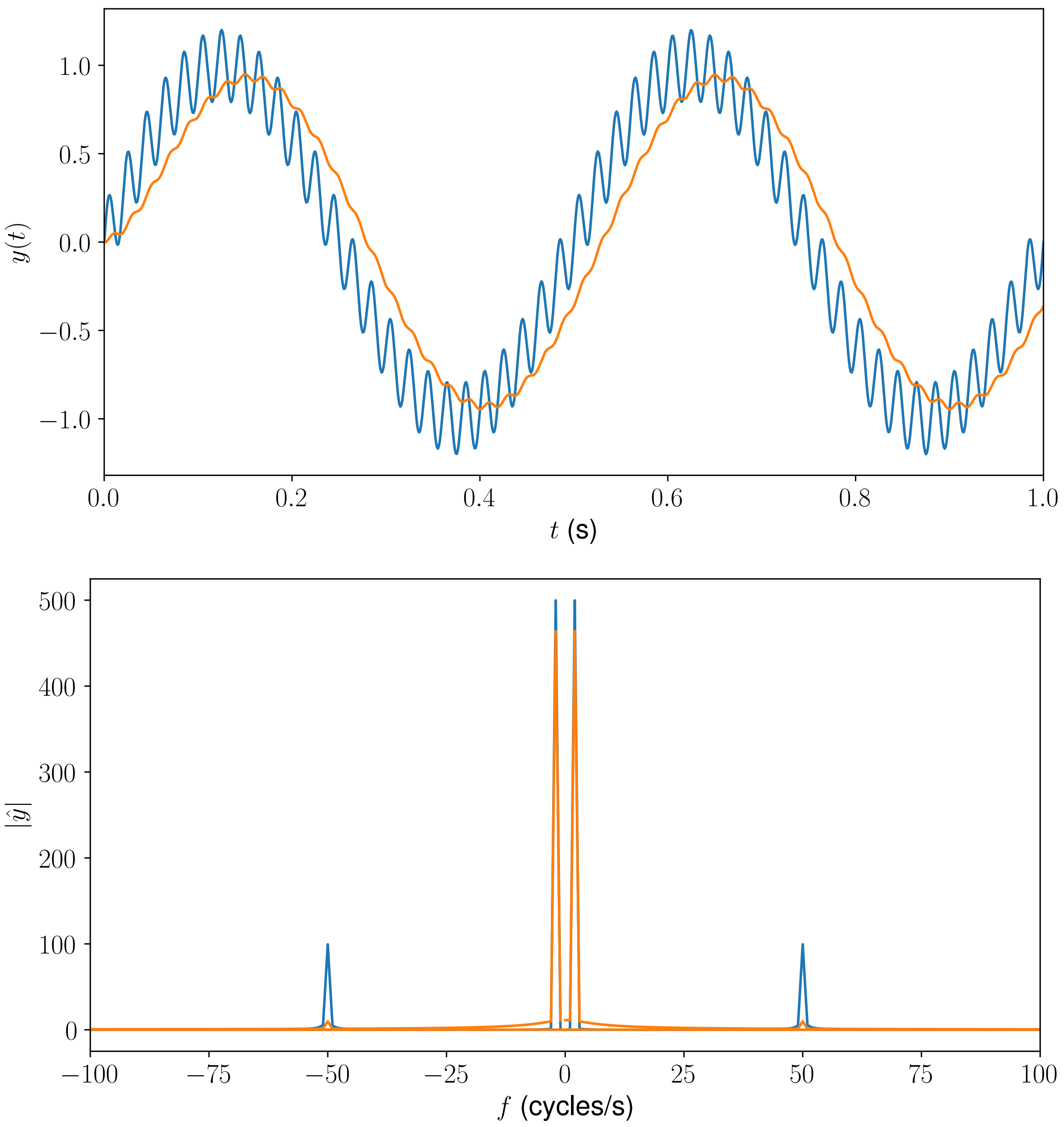

Figure 3: Low-pass Filter Test

The magnitude of filtered signal drops on 5.2% (1.05485 times) The phase of filtered signal shifts on $0.0247 sec$ (approx. 25ms), delaying the data.