Homework 8

Due: Wednesday, 3/25/26, 23:59:59

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.fft import fft, ifft, fftfreq
%matplotlib widget
```

Problem 1 — Sampling Rates

Suppose that we have sampled a repetitive signal at the sampling rate $\nu_0 = 1/\tau$ for a total of N samples. We take its fast Fourier transform (FFT), calling the output FFT1.

1. We now sample the same signal at the same rate for twice as long and compute the FFT of the $2N$ points, calling the output FFT2. How would FFT2 differ from FFT1?

2. How would their power spectra differ?

3. Suppose we now sample for the same time as in the first case, but at twice the frequency, obtaining $2N$ points as in the second example. How would resulting transform (FFT3) and its power spectrum compare to the first one?

Problem 2 — FFT Sleuthing

Data from sampling a periodic signal at 1024 samples per second are stored in a text file **Prob2.txt**, available on the course page. The values in the file are measured potentials (in volts). To load text data into a numpy array, you can use the function `np.loadtxt(filename)`.

1. Load the data and prepare a properly labeled plot of the potential as a function of time.

2. Use the FFT to deduce the frequency composition of this signal, which is the superposition of a fundamental and several harmonics with varying amplitudes. That is, determine the amplitudes and basis functions that went into generating the "measured" potentials (okay, I admit it, I didn't measure the voltages; I computed them).

Problem 3 — Deconvolution

The file **Prob3.txt** contains a mystery signal that was recorded after convolution with an instrument whose response function has the form

$$g(t) = \begin{cases} 0 & t < 0 \\ \alpha e^{-\alpha t} & t \geq 0 \end{cases} \tag{1}$$

The sample rate was 10^3 samples/s and the instrument response time constant, $\tau = \alpha^{-1}$ was 0.1 s. Use the convolution theorem to deduce the original signal. That theorem holds that

$$\mathscr{F}[f * g] = \mathscr{F}[f] \times \mathscr{F}[g] \tag{2}$$

where $\mathscr{F}[f]$ represents the Fourier transform of f. Prepare a plot of the original signal with a properly labeled time axis.

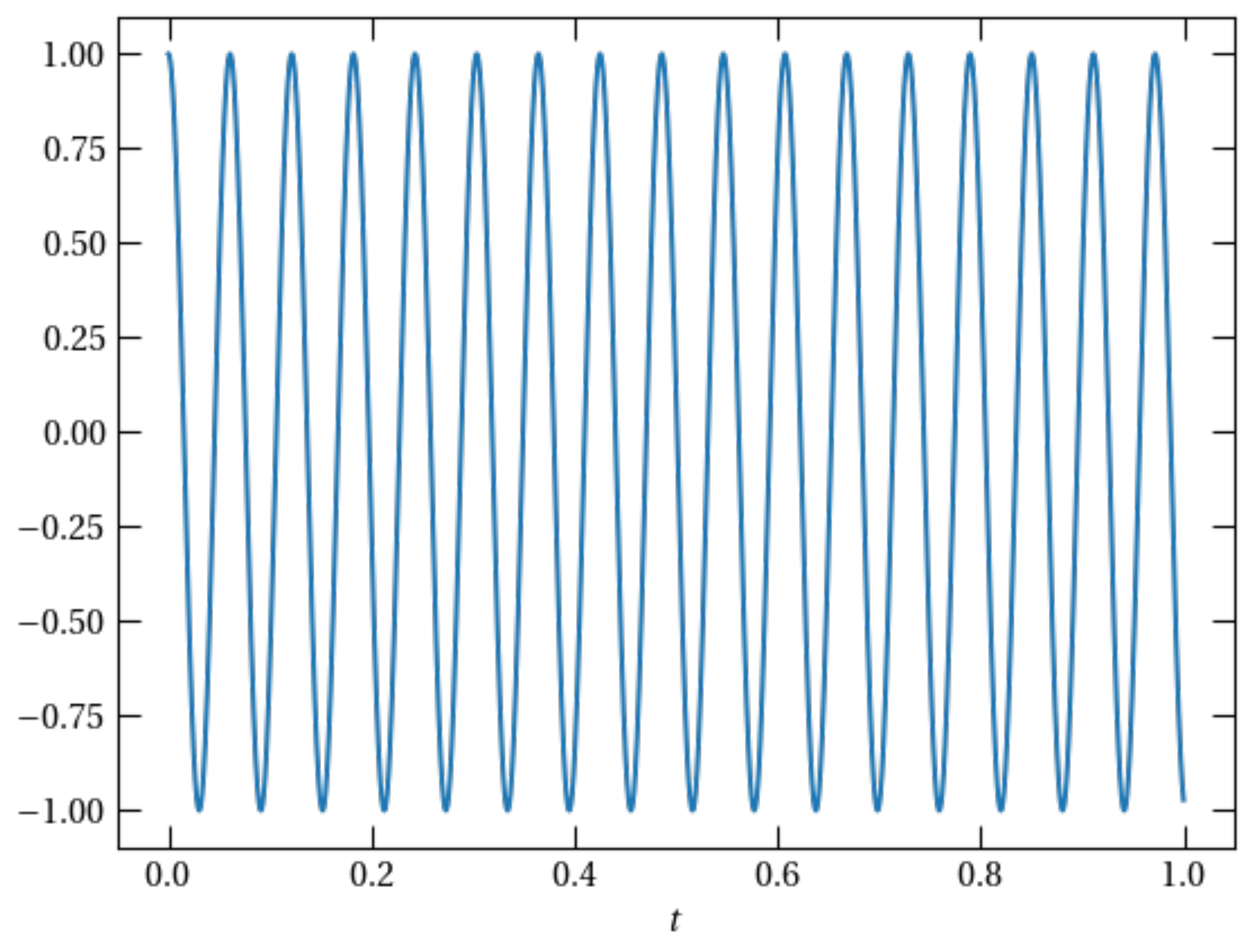
Problem 4 — Windows

Because a discrete Fourier transform is necessarily taken over a finite sampling time, it cannot capture an integral number of cycles of all frequencies below the Nyquist frequency, which means that the periodic continuation of the sampling interval implies a discontinuity. As we have seen, this discontinuity leads to significant power leaking into nearby frequencies (see Fig. 8 in the notes). A way to suppress this problem to a significant degree is to multiply the raw data by a window function that goes smoothly to zero at the edges of the sampled interval. A quick search will bring up a bunch of window functions that folks use (cosine, gaussian, Hamming, Hanning, Welch, Parzen, Tukey, etc.). A boatload of these are defined in `scipy.signal.windows`.

Use the following snippet to generate a source signal and compare the power spectrum computed from the raw signal with that obtained using two different window functions (of your own choosing). Plot the power on a logarithmic scale, use a legend to distinguish the three curves, and write a short summary of your conclusions.

In [3]:
```
t4 = np.linspace(0, 1, 2048, endpoint=False)
freq = 16.47
s4 = np.cos(2 * np.pi * freq * t4)
fig, ax = plt.subplots()
ax.plot(t4, s4)
ax.set_xlabel('$t$');
```

Figure



In [ ]: