09.5 four.exe Exercises for Chapter 09 four

Exercise 09.1 stanislaw

Explain, in your own words (supplementary drawings are ok), what the *frequency domain* is, how we derive models in it, and why it is useful.

Exercise 09.2 pug

Consider the function

$$f(t) = 8\cos(t) + 6\sin(2t) + \sqrt{5}\cos(4t) + 2\sin(4t) + \cos(6t - \pi/2).$$

(a) Find the (harmonic) magnitude and (harmonic) phase of its Fourier series components. (b) Sketch its magnitude and phase spectra. *Hint: no Fourier integrals are necessary to solve this problem.*

Exercise 09.3 ponyo

Consider the function with $a > 0$

$$f(t) = e^{-a|t|}.$$

From the transform definition, derive the Fourier transform $F(\omega)$ of $f(t)$. Simplify the result such that it is clear the expression is real (no imaginary component).

Exercise 09.4 seesaw

Consider the periodic function $f : \mathbb{R} \to \mathbb{R}$ with period T defined for one period as

$$f(t) = at \quad \text{for } t \in (-T/2, T/2] \tag{1}$$

where $a, T \in \mathbb{R}$. Perform a fourier series analysis on f. Letting $a = 5$ and $T = 1$, plot f along with the partial sum of the fourier series synthesis, the first 50 nonzero components, over $t \in [-T, T]$.

Exercise 09.5 totoro

_____/ 25 p.

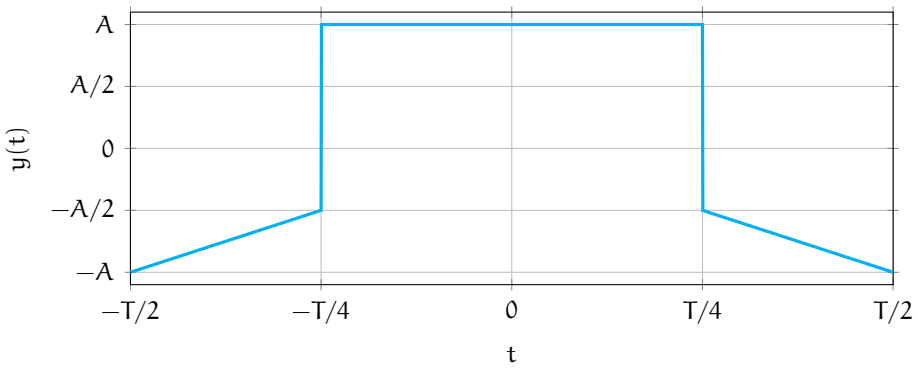

Figure exe.1: one period T of the function $y(t)$. Every line that appears straight is so.

Consider a periodic function $y(t)$ with some period $T \in \mathbb{R}$ and some parameter $A \in \mathbb{R}$ for which one period is shown in Fig. exe.1.

1. Perform a *trigonometric* Fourier series analysis of $y(t)$ and write the Fourier series $Y(\omega)$.
2. Plot the harmonic amplitude spectrum of $Y(\omega)$ for $A = T = 1$. Consider using computing software.
3. Plot the phase spectrum of $Y(\omega)$ for $A = T = 1$. Consider using computing software.

Exercise 09.6 mall

_____/ 20 p.

Consider the function $f : \mathbb{R} \to \mathbb{R}$ defined as

$$f(t) = \begin{cases} a - a|t|/T & \text{for } t \in [-T, T] \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $a, T \in \mathbb{R}$. Perform a fourier transform analysis on f, resulting in $F(\omega)$. Plot F for various a and T.

Exercise 09.7 miyazaki

Consider the function $f : \mathbb{R} \to \mathbb{R}$ defined as

$$f(t) = ae^{-b|t-T|} \tag{3}$$

where $a, b, T \in \mathbb{R}$. Perform a fourier transform analysis on f, resulting in $F(\omega)$. Plot F for various a, b, and T.

Exercise 09.8 haku

Consider the function $f : \mathbb{R} \to \mathbb{R}$ defined as

$$f(t) = a \cos \omega_0 t + b \sin \omega_0 t \tag{4}$$

where $a, b, \omega_0 \in \mathbb{R}$ constants. Perform a fourier transform analysis on f, resulting in $F(\omega)$.[2]

Exercise 09.9 secrets

This exercise encodes a "secret word" into a sampled waveform for decoding via a *discrete fourier transform* (DFT). The nominal goal of the exercise is to decode the secret word. Along the way, plotting and interpreting the DFT will be important.
First, load relevant packages.

```
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import display, Markdown, Latex
```

We define two functions: `letter_to_number` to convert a letter into an `integer` index of the alphabet (a becomes 1, b becomes 2, etc.) and `string_to_number_list` to convert a string to a `list` of `ints`, as shown in the example at the end.

[2]It may be alarming to see a Fourier transform of a periodic function! Strictly speaking, it does not exist; however, if we extend the transform to include the *distribution* (not actually a function) Dirac $\delta(\omega)$, the modified-transform does exist and is given in Table ft.1.

[2]Python code in this section was generated from a Jupyter notebook named `random_signal_fft.ipynb` with a `python3` kernel.

```
def letter_to_number(letter):
    return ord(letter) - 96

def string_to_number_list(string):
    out = [] # list
    for i in range(0,len(string)):
        out.append(letter_to_number(string[i]))
    return out # list

print(f"aces = { string_to_number_list('aces') }")
```

```
aces = [1, 3, 5, 19]
```

Now, we encode a code string `code` into a signal by beginning with "white noise," which is *broadband* (appears throughout the spectrum) and adding to it `sin` functions with amplitudes corresponding to the letter assignments of the code and harmonic corresponding to the position of the letter in the string. For instance, the string `'bad'` would be represented by noise plus the signal

$$2\sin 2\pi t + 1\sin 4\pi t + 4\sin 6\pi t. \tag{5}$$

Let's set this up for secret word `'chupcabra'`.

```
N = 2000
Tm = 30
T = float(Tm)/float(N)
fs = 1/T
x = np.linspace(0, Tm, N)
noise = 4*np.random.normal(0, 1, N)
code = 'chupcabra' # the secret word
code_number_array = np.array(string_to_number_list(code))
y = np.array(noise)
for i in range(0,len(code)):
    y = y + code_number_array[i]*np.sin(2.*np.pi*(i+1.)*x)
```

For proper decoding, later, it is important to know the fundamental frequency of the generated data.

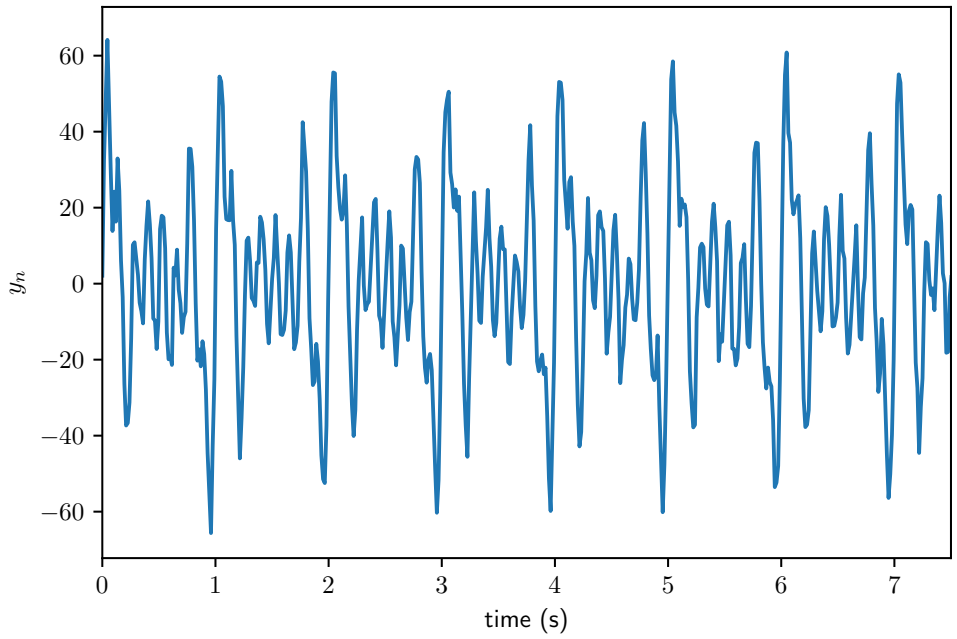

Figure exe.2: the chupacabra signal.

```
print(f"fundamental frequency = {fs} Hz")
```

```
fundamental frequency = 66.66666666666667 Hz
```

Now, we plot.

```
fig, ax = plt.subplots()
plt.plot(x,y)
plt.xlim([0,Tm/4])
plt.xlabel('time (s)')
plt.ylabel('$y_n$')
plt.show()
```

Finally, we can save our data to a `numpy` file `secrets.npy` to distribute our message.

```
np.save('secrets',y)
```

Now, I have done this (for a different secret word!) and saved the data; download it here:

ricopic.one/mathematical_foundations/source/secrets.npy.

In order to load the `.npy` file into *Python*, we can use the following command.

```
secret_array = np.load('secrets.npy')
```

Your job is to (a) perform a DFT, (b) plot the spectrum, and (c) decode the message! Here are a few hints.

1. Use `from scipy import fft` to do the DFT.
2. Use a `hanning` window to minimize the end-effects. See `numpy.hanning` for instance. The `fft` call might then look like

   ```
   2*fft(np.hanning(N)*secret_array)/N
   ```

   where `N = len(secret_array)`.
3. Use only the *positive* spectrum; you can lop off the negative side and double the positive side.

Exercise 09.10 society

Derive a fourier transform property for expressions including function $f : \mathbb{R} \to \mathbb{R}$ for

$$f(t)\cos(\omega_0 t + \psi)$$

where $\omega_0, \psi \in \mathbb{R}$.

Exercise 09.11 flapper

Consider the function $f : \mathbb{R} \to \mathbb{R}$ defined as

$$f(t) = a u_s(t) e^{-bt} \cos(\omega_0 t + \psi) \tag{6}$$

where $a, b, \omega_0, \psi \in \mathbb{R}$ and $u_s(t)$ is the unit step function. Perform a fourier transform analysis on f, resulting in $F(\omega)$. Plot F for various a, b, ω_0, ψ and T.

Exercise 09.12 eastegg

Consider the function $f : \mathbb{R} \to \mathbb{R}$ defined as

$$f(t) = g(t) \cos(\omega_0 t) \tag{7}$$

where $\omega_0 \in \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ will be defined in each part below. Perform a fourier transform analysis on f for each g below for $\omega_1 \in \mathbb{R}$ a constant and consider how things change if $\omega_1 \to \omega_0$.

a. $g(t) = \cos(\omega_1 t)$
b. $g(t) = \sin(\omega_1 t)$

Exercise 09.13 savage

An instrument called a "lock-in amplifier" can measure a sinusoidal signal $A\cos(\omega_0 t + \psi) = a\cos(\omega_0 t) + b\sin(\omega_0 t)$ at a known frequency ω_0 with exceptional accuracy even in the presence of significant noise $N(t)$. The workings of these devices can be described in two operations: first, the following operations on the signal with its noise,
$f_1(t) = a\cos(\omega_0 t) + b\sin(\omega_0 t) + N(t)$,

$$f_2(t) = f_1(t)\cos(\omega_1 t) \quad \text{and} \quad f_3(t) = f_1(t)\sin(\omega_1 t). \tag{8}$$

where $\omega_0, \omega_1, a, b \in \mathbb{R}$. Note the relation of this operation to the Fourier transform analysis of Exercise 09.12 four.. The key is to know with some accuracty ω_0 such that the instrument can set $\omega_1 \approx \omega_0$. The second operation on the signal is an aggressive low-pass filter. The filtered f_2 and f_3

are called the *in-phase* and *quadrature* components of the signal and are typically given a complex representation

$$(\text{in-phase}) + \mathrm{j}\,(\text{quadrature}).$$

Explain with fourier transform analyses on $\mathtt{f}_2$ and $\mathtt{f}_3$

a. what $\mathtt{F}_2 = \mathcal{F}(\mathtt{f}_2)$ looks like,
b. what $\mathtt{F}_3 = \mathcal{F}(\mathtt{f}_3)$ looks like,
c. why we want $\omega_1 \approx \omega_0$,
d. why a low-pass filter is desirable, and
e. what the time-domain signal will look like.

Exercise 09.14 strawman

Consider again the lock-in amplifier explored in Exercise 09.13 four.. Investigate the lock-in amplifier numerically with the following steps.

a. Generate a noisy sinusoidal signal at some frequency ω_0. Include enough broadband white noise that the signal is invisible in a time-domain plot.
b. Generate $\mathtt{f}_2$ and $\mathtt{f}_3$, as described in Exercise 09.13 four..
c. Apply a time-domain discrete low-pass filter to each $\mathtt{f}_2 \mapsto \phi_2$ and $\mathtt{f}_3 \mapsto \phi_3$, such as scipy's `scipy.signal.sosfiltfilt`, to complete the lock-in amplifier operation. Plot the results in time and as a complex (polar) plot.
d. Perform a discrete fourier transform on each $\mathtt{f}_2 \mapsto \mathtt{F}_2$ and $\mathtt{f}_3 \mapsto \mathtt{F}_3$. Plot the spectra.
e. Construct a frequency domain low-pass filter $\mathtt{F}$ and apply it (multiply!) to each $\mathtt{F}_2 \mapsto \mathtt{F}'_2$ and $\mathtt{F}_3 \mapsto \mathtt{F}'_3$. Plot the filtered spectra.
f. Perform an inverse discrete fourier transform to each $\mathtt{F}'_2 \mapsto \mathtt{f}'_2$ and $\mathtt{F}'_3 \mapsto \mathtt{f}'_3$. Plot the results in time and as a complex (polar) plot.
g. Compare the two methods used, i.e. time-domain filtering versus frequency-domain filtering.

10 freq