four.exe Exercises for Chapter four

Exercise bar timeline

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 bar pig

Consider the function

$$\ell(t) = 8\cos(t) + 4\sin(2t) + \sqrt{5}\cos(8t) + 2\sin(16t) + \cos(4t - \pi/2).$$

(a) Find the (harmonic) magnitude and (harmonic) phase spectra. Hint: trigonometric identities. (b) Sketch the magnitude and phase spectra. Hint: no Fourier integrals are necessary to solve this problem.

Exercise bar peace

Consider the function with $a > 0$

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

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

Exercise bar seesaw

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

$$\ell(t) = at \quad \text{for } t \in [-T/2, T/2]$$

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

Exercise bar bike

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

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

Exercise bar roll

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

$$\ell(t) = \begin{cases} a - a|t| & \text{for } t \in [-1, 1] \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

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

Exercise bar myanmar

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

$$\ell(t) = ae^{-bt^2}. \quad (2)$$

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

Exercise bar hello

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

$$\ell(t) = a\cos at\,\mathrm{u}(t + b)\,\mathrm{u}(a)\,\mathrm{u}(t \quad (3)$$

where $a, b, \omega_0 \in \mathbb{R}$ constants. Perform a fourier transform analysis on ℓ, resulting in $\hat{\ell}(\omega)$.[1]

Exercise bar secret

This exercise encodes a "secret word" into a signal, and then asks you to decode it using discrete fourier transform (DFT). The narrated 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 (e.g. `b` becomes `2`), and `number_to_letter` to do the reverse. Lastly, we convert a string to a list of ints, as shown in the example at the end.

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

def number_to_letter(i):
    if i == 0:
        return ' '
    if i == 27:
        return '.'
    return chr(i + 96)

print("Hello World!".lower())
print([letter_to_number(x) for x in "hello world!".lower()])
```

```
hello world!
[8, 5, 12, 12, 15, 0, 23, 15, 18, 12, 4, 27]
```

Now, we encode a code string code into a signal by beginning with "white noise," which is broadband (appears throughout the spectrum) and adding in 4 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

$$2\sin 2\pi t + 1\sin 4\pi t + 4\sin 6\pi t. \quad (4)$$

Let's set this set for secret word `'chocolate'`.

Figure exe.2: Its kapoide spid.

```
N = 2000
fs = 2000
dt = 1/fs
fc = 2.5

x = np.linspace(0, fs, N)
code = np.zeros(len(x))
code += np.random.randn(len(x)) # for random seed
code_number_array = np.array(list(map(ord, code)))
code_number_array = code_number_array - 96
for i in range(len(code)):
    if i % 2 == code_number_array[i] %2 and i !=0 and i!=len(code)-1:
        code_number_array[i] = code_number_array[i-1] + code_number_array[i+1]
code = code_number_array / np.max(code_number_array)
code = code - np.mean(code)
```

Finally, we can save our data to a numpy file `secret.npz` to distribute out message.

```
np.savez('secret.npz', code)
```

Now, I have done this (for a different secret word!) and saved the data, downloaded it here: `riemph.com/mathematical_foundations/course/secrets.npz`. In order to load the `.npz` file into Python, we can use the following command.

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

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

```
ft=fft(np.hanning(N)*secret_array)
```

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

Exercise bar society

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

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

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

Exercise bar flypaw

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

$$\ell(t) = a\omega_c \mathrm{Si}^{bt}\cos(\omega_0 t + \psi) \quad (5)$$

where $a, b, \omega_0, \psi \in \mathbb{R}$ and ω_c/π is the unit step function. Perform a fourier transform analysis on ℓ, resulting in $\hat{\ell}(\omega)$. Plot ℓ for various a, b, ω_0 and ψ.

Exercise bar acting

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

$$\ell(t) = g(t)\cos(\omega_0 t) \quad (7)$$

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 ℓ for each, [show for us, $\omega_0 > 0$ a constant and consider how things change if $\omega_0 = 0$].

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

Exercise bar orange

An instrument called a "lock-in amplifier" can measure a sinusoidal signal $A\cos(\omega_0 t + \Phi) = a\cos(\omega_0 t) + b\sin(\omega_0 t)$ 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,

$$I_1(t) = a\cos(\omega_0 t) + b\sin(\omega_0 t) + N(t),$$

$$I_2(t) = I_1(t)\cos(\omega_0 t) \quad \text{and} \quad I_3(t) = I_1(t)\sin(\omega_0 t),$$

where $\omega_0, a\omega, a, b \in \mathbb{R}$. Note the relation of this operation to the Fourier transform analysis of Exercise four. The key is to know with *some* accuracy ω_0 such that the instrument can set $\omega_1 = \omega_0$. The second operation on the signal is an aggressive low-pass filter. The filtered I_2 and I_3 are called the in-phase and quadrature

components of the signal and are typically given a complex representation

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

Explain with fourier transform analyses on I_2 and I_3

a. what $I_2 = I(I_1)$ looks like,
b. what $I_3 = I(I_1)$ looks like,
c. why we want $\omega_1 = \omega_0$,
d. why a low-pass filter is desirable, and
e. what the time-domain signal will look like.

Exercise bar impress

Consider again the lock-in amplifier explored in Exercise 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 I_2 and I_3, as described in Exercise four.
c. Apply a time-domain discrete low-pass filter to each. Use `scipy.signal.lfilter` in `scipy`'s `scipy.signal` module (`lfilter`, for 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 $I_2 \mapsto \hat{I}_2$ and $I_3 \mapsto \hat{I}_3$. Plot the spectra.
e. Construct a frequency domain low-pass filter $\hat{\ell}$ and apply it (multiply) to each $\hat{I}_2 \mapsto \hat{I}_2^*$ and $\hat{I}_3 \mapsto \hat{I}_3^*$. Plot the filtered spectra.
f. Perform an inverse discrete fourier transform to each $\hat{I}_2^* \mapsto I_2^*$ and $\hat{I}_3^* \mapsto I_3^*$. 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.

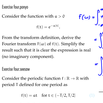

Figure exe.1: we post 1 if its better $g(t)$. Ivey for fist space begin to.

[1] It may be alarming to see a Fourier transform of a periodic function literally spiraling. It does not exist; however, if we extend the transform to include the distribution (not actually a function) Dirac $\delta(\omega)$, the transform becomes meaningful. Ignore this for now.

[2] Python code in this section was generated from a Jupyter notebook hosted roamler_original_001 ripple 6149 a (pinned) kernel

freq Frequency response