University of South-Eastern Norway

Interference Removal from Electro-Cardiogram (ECG) Signals

TSE2280 Signal Processing

Lab 5, spring 2025

Contents

List of Tables

List of Figures

1 Introduction

1.1 Objective

The purpose of this lab is to get training in interpreting filter responses from zeros and poles in the z-plane, and to explore basic IIR-filters.

In the first part, you will use an interactive Jupyter Notebook `pole_zero_demo` to study a few IIR filters and see how the response changes when you move poles and zeros around in the z-plane.

In the second part, you will design a sharp notch-filter to remove a single frequency interference signal from an ECG-recording. The ECG can be viewed as the electrical control signal for our heart, interpretation of these signals is an essential tool in heart diagnosis.

The lab demonstrates concepts from Chapters 9 and 10 in the textbook by McClellan et al [1], covering z-plane analysis and IIR-filters. The lab is based on lab *Lab S-9: Interference Removal from Electro-Cardiogram (ECG) Signals* [2] that comes with the textbook. Some of the text has been taken from this, while the original lab has been modified and converted from Matlab to Python.

1.2 Software Tools: Python with Spyder and JupyterLab

This lab uses the same tools as the previous labs. Python is used for for programming, with Spyder [3] as the programming environment and JupyterLab [4] for running the demo program `pole_zero_demo`. The signals are represented as NumPy [5, 6] arrays and plotted in Matplotlib [7, 8]. The signals are analysed with functions from the signal processing modules in SciPy [9, 10]. The filtering and frequency analysis tools in SciPy, `scipy.signal`, are central in this lab.

1.3 Time, Frequency and z-domains

IIR filters find a wide use in signal processing. Bandpass filtering is the most common, but nulling or notch filters are also very useful for removing interference, such as narrowband interference caused by power lines. In this lab, you will explore the connection between the time domain n, the frequency domain $\hat{\omega}$, and the z-domain. The main tool for this is to study poles and zeros of the system function $H(z) = \frac{B(z)}{A(z)}$, where $B(z)$ and $A(z)$ are polynomials in z. The frequency response of the system is given by the values of $H(z)$ on the unit circle where $|z| = 1$, written as $z = e^{j\hat{\omega}}$.

1) *Zeros,* $B(z) = 0$. When placed on the unit circle, zeros in $B(z)$ will force the frequency response $H\left(e^{j\hat{\omega}}\right)$ to be zero. This can be used to null out sinusoids at one frequency.

2) *Poles,* $A(z) = 0$. When placed inside the unit circle, the roots of $A(z)$ will create peaks in the frequency response. These can be used to form bandpass filters.

3) *IIR Notch Filters.* Notch filters require zeros on the unit circle $|z| = 1$ with poles at the same angle and just inside the unit circle. The frequency response of the notch is much sharper than a nulling filter, i.e., an FIR filter formed from zeros on the unit circle without nearby poles.

2 Theory

2.1 Poles, Zeros, and Filter Coefficients

The Jupyter Notebook `pole_zero_demo` allows simultaneous interactive exploration of the three domains, i.e., poles and zeros in the z-domain, the impulse response $h[n]$ in the n-domain, and the frequency response $H\left(e^{j\hat{\omega}}\right)$ in the $\hat{\omega}$-domain. The intention is to build an intuitive understanding of the relationship between them. The system function $H(z)$ is written as a ratio of polynomials in z^{-1}, expressed in either factored or expanded form,

$$H(z) = \frac{B(z)}{A(z)} = G\frac{\prod_{k=1}^{M}\left(1 - z_k z^{-1}\right)}{\prod_{l=1}^{N}\left(1 - p_l z^{-1}\right)} = \frac{\sum_{k=1}^{M} b_k z^{-k}}{\sum_{l=1}^{N} a_l z^{-l}}\,, \tag{1}$$

where M is the number of zeros z_k, N the number of poles p_l, and b and a the filter coefficients. G is a scaling factor giving the gain of the filter. If b and a are real, the poles and zeros ar either real or occur in complex-conjugate pairs. This follows from the mathematical property *a polynomial with real coefficients has roots that are real or occur in complex-conjugate pairs.*

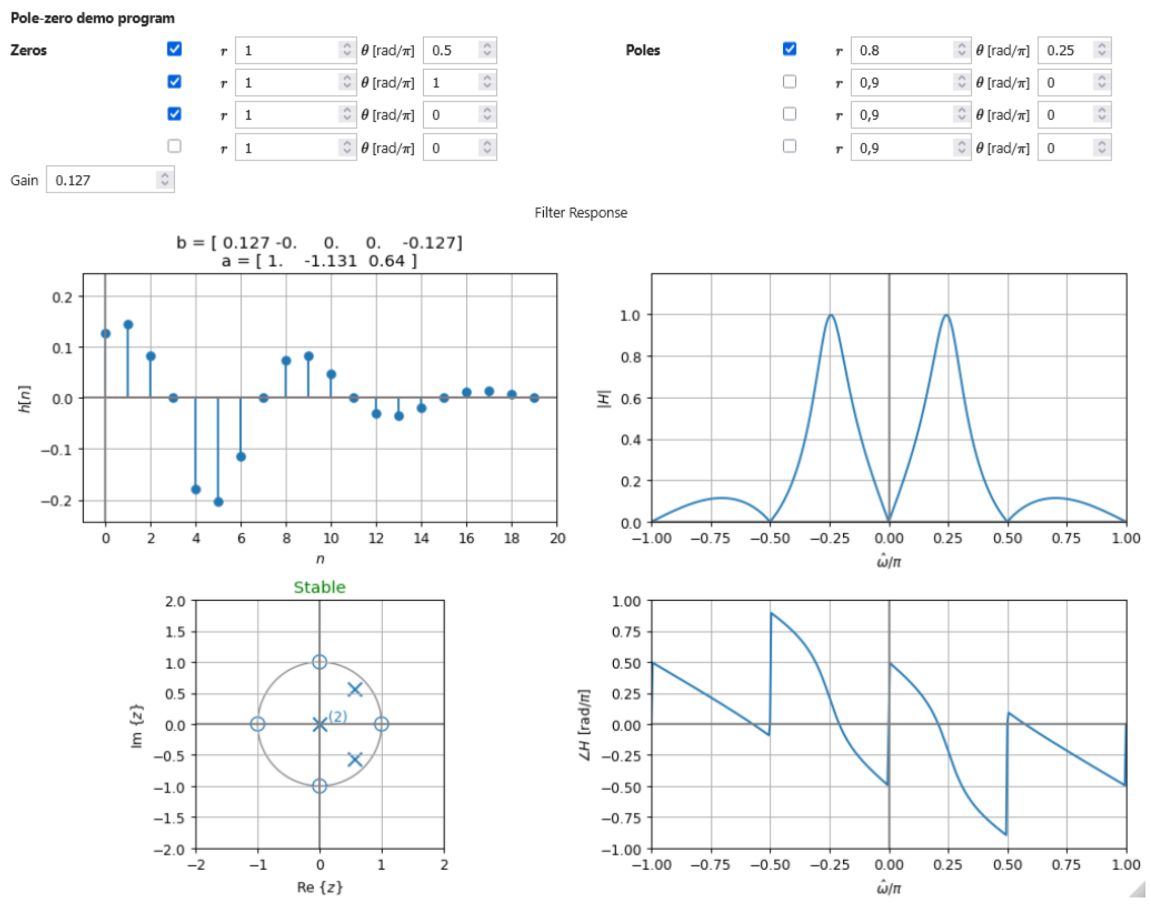
2.2 Using `pole_zero_demo`

`pole_zero_demo` is a Jupyter Notebook with interactive widget controls that update the graphs when values are changed, see Figure 1. Equation (1) defines how $H(z)$ is calculated from the zeros z_k, poles p_l, and gain G.

The control panel at the top is used to define up to 4 conjugate zero pairs and 4 conjugate pole pairs, specified by their magnitude r and phase θ. Note that the phase angle θ is specified in units of π, e.g., a value 0.5 corresponds to $\theta = 0.5\,\mathrm{rad}/\pi$. Poles and zeros are added or removed by selecting the checkbox, and moved around in the complex plane by changing their values. The program may add extra poles or zeros at origo or infinity if the specified numbers of poles and zeros are not equal. The *Gain* control scales the output without changing the shape of the curves.

Complex poles or zeros ($\theta \neq \pm\pi$) will be added as complex conjugate pairs. This is required to get real valued filter coefficients a and b. Real poles or zeroes ($\theta = \pm\pi$) will be added as single values.

The graphs in Figure 1 show the inmpulse response $h[n]$, the pole-zero diagram in the complex plane, and the frequency response $H\left(e^{j\hat{\omega}}\right)$ as magnitude $|H|$ and phase $\angle H$. If system is unstable, the frequency response in meaningless and will not be show. This occurs if not all poles are inside the unit circle. The coefficients b and a calculated for the IIR-filter are shown above the impulse response graph.



Pole-zero demo program

Zeros	✓	r 1	θ [rad/π] 0.5	Poles	✓	r 0.8	θ [rad/π] 0.25
	✓	r 1	θ [rad/π] 1		☐	r 0,9	θ [rad/π] 0
	✓	r 1	θ [rad/π] 0		☐	r 0,9	θ [rad/π] 0
	☐	r 1	θ [rad/π] 0		☐	r 0,9	θ [rad/π] 0

Gain 0.127

Filter Response

b = [0.127 -0.   0.   0.   -0.127]
a = [1.   -1.131 0.64]

Stable

Figure 1. Screenshot of the interactive Jupyter Notebook `pole_zero_demo`. The top controls define zeros and poles by their magnitude r and phase θ. Poles and zeros are added or removed by selecting the checkbox, and moved around in the complex plane by changing their values. Complex poles or zeros are added as complex conjugate pairs. The *Gain* control scales the output.
The graphs show the inpulse response $h[n]$ (upper left), the pole-zero diagram in the complex plane (lower left), and the frequency response $H\left(e^{j\hat{\omega}}\right)$ as magnitude $|H|$ (upper right) and phase $\angle H$ (lower right).
The coefficients b and a for the resulting IIR-filter are written out above the impulse response graph.
Note that all angles are given in unit $[\mathrm{radians}/\pi]$.

3 Lab Exercise 1: Zeros, Poles, and z, n, and $\hat{\omega}$ Domains

Reporting

This first part of the lab will us `pole_zero_demo` to illustrate how the poles and zeros in the z-domain determine the impulse response $h[n]$ and frequency response $H\left(e^{j\hat{\omega}}\right)$ of a filter. Write down your answers in a document and upload this as a pdf-file to Canvas.

3.1 FIR-filters. Zeros

1) Implement the following FIR system in `pole_zero_demo`

$$H(z) = 1 - z^{-1} + z^{-2}\,. \tag{2}$$

   a) Express the zeros on polar form and place them correctly in `pole_zero_demo`.
   b) Make a screenshot of the graph and enclose this in your report.
   c) Move the zero-pair around the unit circle by changing the phase θ. Observe that the location of the null in the frequency response also moves.

   Note that the the NumPy function `roots` can be used to find the roots of a polynomial. Observe the following

   - The impulse response $h[n]$ values are equal to the polynomial coefficients of $H(z)$.
   - The frequency response has nulls because the zeros of $H(z)$ lie exactly on the unit circle. Compare the frequencies of the nulls to the angles of the zeros.

2) The 6-point running-sum FIR digital filter has the system function

$$H(z) = \sum_{n=0}^{5} z^{-n} = \frac{1 - z^{-6}}{1 - z^{-1}} \tag{3}$$

   a) Find the zeros of $H(z)$ for this FIR filter. You may use `roots` function in NumPy.
   Note: The formula for $H(z)$ involves a numerator-denominator cancellation that removes the zero at $z = 1$. This is called a *pole-zero cancellation*.
   b) Use `pole_zero_demo` to place the five zeros of the FIR filter at the correct locations. Observe that the impulse response will be all ones when you have the correct zero locations. Describe the frequency response of the filter, LPF, HPF, or BPF?
   c) List all the frequencies, $\hat{\omega} \in [-\pi, \pi]$, that are nulled by the filter.
   d) Make a different FIR filter that has the system function

$$H_c(z) = 1 - z^{-6} \tag{4}$$

   which is the numerator of (3).
   Reformulate $H_c(z)$ to positive powers of z, i.e.,

$$H_c(z) = \frac{B(z)}{A(z)}\,, \tag{5}$$

   where $B(z)$ and $A(z)$ are expressed in positive powers of z.
   Find the poles and zeros of $H_c(z)$ and use `pole_zero_demo` to place them in the pole-zero plot. Hint: this can be done very simply by adding one zero to the previous configuration.
   e) Write a formula for the impulse response, $h_c[n]$ using weighted and shifted impulses.
   f) The frequency response of the FIR filter in the previous part is called a *comb filter*. Explain why is it called a comb, and determine where its nulls are located for $\hat{\omega} \in [-\pi, \pi]$.
   g) One of the zeros will be at $z = -1$. Remove this zero write down the system function $H(z)$ for the new filter. Hint: `pole_zero_demo` contains the answer.
   h) Describe the frequency response of this filter, LPF, HPF, or BPF?

3.2 IIR-filters

1) Real poles

   a) Use `pole_zero_demo` to place a single pole at $z = -\frac{1}{2}$ and a single zero at $z = 1$. Describe the important features of the impulse response $h[n]$, and also the important features of the frequency response $H\left(e^{j\hat{\omega}}\right)$.

b) Find passband of the filter, defined as the region where the magnitude response is greater than 90% of the maximum; and the stopband as the region where the magnitude response is less than 10% of the maximum. You can zoom in on `pole_zero_demo` to find these answers.

c) Move the pole from a location close to the origin out to $z = -\frac{1}{2}$, and then to out to $z = -0.99$ while staying on the real axis. Comment the changes in the impulse response $h[n]$ and the frequency response $H\left(e^{j\hat{\omega}}\right)$.

d) Move the pole from $z = -0.99$ to a radius outside the unit circle. Describe the changes in $h[n]$. Explain how the appearance of $h[n]$ validates the statement that the system is not stable. In this case, the frequency response $H\left(e^{j\hat{\omega}}\right)$ is not legitimate because the system is no longer stable.

e) In general, where should poles be placed to guarantee system stability? By stability we mean that the system's output does not blow up as $n \to \infty$.

2) First-order IIR filter

a) Implement the following first-order IIR system

$$H(z) = \frac{1 - z^{-1}}{1 + 0.9z^{-1}} \tag{6}$$

by placing its pole and zero in the z-plane.

b) Look at the frequency response and what kind of filter you have, HPF, LPF, or BPF?

c) Move the pole from $z = -0.9$ to $z = 0$ and then to $z = +0.9$. Comment how the frequency response changes. Describe the type of filter for $z = 0.9$, HPF, LPF, or BPF?

3) Second-order IIR BPF

Use the `pole_zero_demo` to implement the second-order system

$$H(z) = \frac{1 - z^{-2}}{1 + 0.8z^{-1} + 0.64z^{-2}}\ . \tag{7}$$

Factor the numerator and denominator to get the two poles and the two zeros, and place the poles and zeros at the correct locations in the z-plane. `pole_zero_demo` will add complex-conjugate pairs, so you only have to add one of the poles.

Look at the frequency response and verify that you have created a BPF.

4) Filters that are not bandpass

It is tempting to think that with two poles the frequency response always has a peak, but there are two interesting cases where this is not the case.

i. All-pass filters where $|H\left(e^{j\hat{\omega}}\right)|$= constant.

ii. IIR notch filters that null out one frequency, but have a flat nonzero magnitude response elsewhere.

a) Implement the second-order system

$$H(z) = \frac{64 + 80z^{-1} + 100z^{-2}}{1 + 0.8z^{-1} + 0.64z^{-2}}$$

by factoring to get the poles and zeros and then using `pole_zero_demo` to place them the z-plane. Adjust the gain to get the correct numerator coefficients.

b) Look at the frequency response and determine what kind of filter you have, LPF, HPF, BPF, notch, or all-pass?

c) Move the zeros to be exactly on the unit-circle at the same angle as the poles. Observe how the frequency response changes.

d) Determine $H(z)$ for this filter, which should have the form

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 + 0.8z^{-1} + 0.64z^{-2}}\ . \tag{8}$$

e) Describe the type of filter you have created, LPF, HPF, BPF, notch, or all-pass?

5) IIR Allpass Filter

An IIR allpass filter will pass all frequency components with a magnitude response $|H\left(e^{j\hat{\omega}}\right)|$ that is prefectly flat across the entire frequency band, but its phase response is nonlinear versus frequency. It has one complex-conjugate zero pair and one complex-conjugate pole pair.

$$\text{Zeros at } r^{-1}e^{\pm j\theta} \qquad\qquad \text{Poles at } re^{\pm j\theta} \tag{9}$$

where r is a number slightly less than one, so that the system is stable. Thus the system function is

$$H(z) = \frac{B(z)}{A(z)} = G\frac{\left(1 - r^{-1}e^{j\theta}z^{-1}\right)\left(1 - r^{-1}e^{-j\theta}z^{-1}\right)}{\left(1 - re^{j\theta}z^{-1}\right)\left(1 - re^{-j\theta}z^{-1}\right)}\,. \tag{10}$$

a) Create an example in `pole_zero_demo` using $r = 0.9$ and $\theta = \pi/3$

b) Adjust the gain G so that the frequency response magnitude is equal to 10.

c) Write out the expression for the numerator $B(z)$ and denominator $B(z)$ of $H(z)$ in (10).
Note that `pole_zero_demo` provides the filter coefficients b and a, so you can check your work.

4 Lab Exercise 2: IIR Notch Filter to Remove Interfrerence

A common heart test is an electrocardiogram (ECG or EKG) which records electrical activity that changes during the cardiac cycle. Electrodes placed at several locations on the body pick up these signals. Unfortunately, the electrodes also pick up signals from other electrical sources, most notably harmonics of the 50 Hz power signal (60 Hz in America). Two examples of ECG curves are shown in Figure 2. The objective of this lab is to show that you can remove a sinusoidal interference from a corrupted ECG signal, and produce a cleaned-up signal.

An IIR notch filter will null out one frequency while having a frequency response that is flat across the rest of the frequency band. It has one complex-conjugate pair of zeros and one of poles

$$\text{Zeros at } e^{\pm j\theta} \qquad\qquad \text{Poles at } re^{\pm j\theta}$$

where r is a number slightly less than one. The system function is

$$H(z) = G\frac{\left(1 - e^{j\theta}z^{-1}\right)\left(1 - e^{-j\theta}z^{-1}\right)}{\left(1 - re^{j\theta}z^{-1}\right)\left(1 - re^{-j\theta}z^{-1}\right)}\,. \tag{11}$$

This is similar to the all-pass filter in (10), the only difference is the magnitude of the zeros, which is 1, placing the zeros on the unit circle.

You can use `pole_zero_demo` to exhibit a typical frequency response for a notch filter. To see an example, select $r = 0.95$ and $\theta = \pi/4$. It is also possible to multiply out the numerator and denominator of (11) to get the filter coefficients.

4.1 IIR Filter Implementation

IIR filters can be implemented in Python using functions from SciPy under `scipy.signal`. The function `scipy.signal.lfilter(b, a, x)` filters the sequence `x` with forward coefficients `b` and backward coefficients `a`.

4.2 Notch Filter to Remove Sinusoidal Interference

The ECG signal with interference will be generated by the Python function `ecgdata.py`. The function reads data from a file `ecg_recorded.npz` and adds a random sinusoidal interference to simulate line noise. The interference signal will have frequency close to the line frequency 50 Hz or its 2nd or 3rd harmonic. The code for generating the ECG signal with random intereference is listed below.

```
import ecgdata         # Library to load and create ECG-data

# Make ECG signal with artificlial line noise
ecg, fs = ecgdata.make_ecg(seed)  # Load ECG signal, add artificial noise from seed

# Plot ECG signal
n_pts = len(ecg)                  # No. of samples in ECG-signal
t = np.arange(n_pts)/fs           # Time vector

plt.plot(t, ecg)
plt.xlabel('Time␣[s]')
plt.ylabel('ECG␣voltage␣[V]')
```

a) Run the code above to obtain the ECG signal `ecg` and its sample rate f_s.
   Select an integer number for the parameter `seed`. This is the 'seed' to the random number generator that picks the ECG signal and adds the artificial noise. A fixed value of `seed` will always give the same 'random' noise.

b) Calculate the power spectral density of the signal and use this to find the interference frequency f_{int}.

c) Use f_{int} and the sample rate f_s to find the normalised frequency $\hat{\omega}_c$ of the notch filter.

d) Design the IIR notch filter from its poles and zeros, determined from $\hat{\omega}_c$.

e) Use `pole_zero_demo` to visualise the frequency response of the notch filter for the value found for $\hat{\omega}_c$.

f) Convert the poles and zeros of the IIR notch filter to coefficients b and a. You may use the values from `pole_zero_demo`.

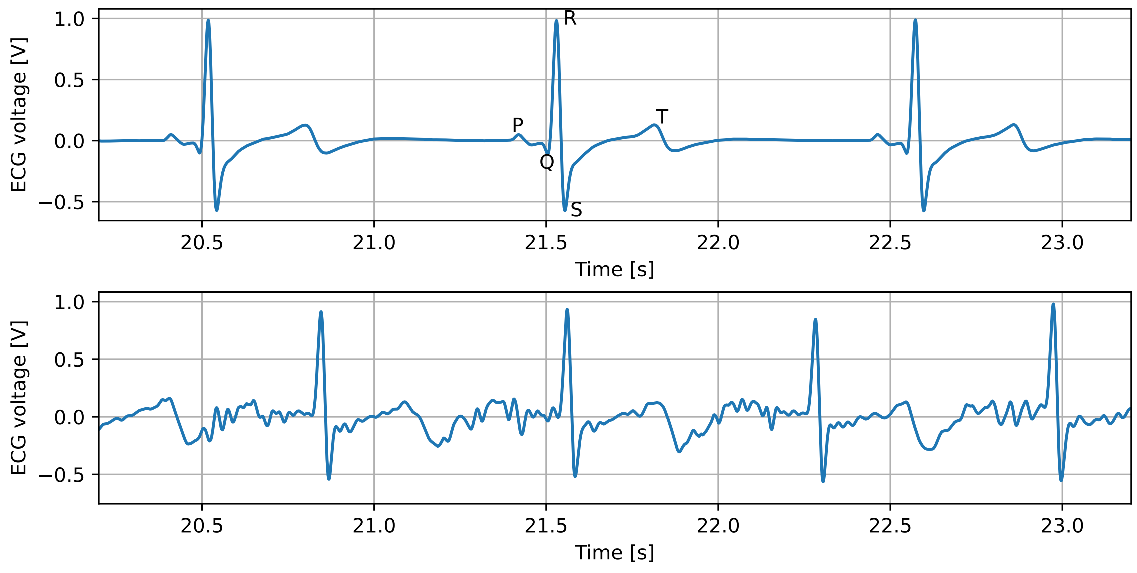


Figure 2. Examples of two ECG-curves The upper ECG follows the classic pattern. The P-wave, QRS complex, and T-wave are labeled. The P-wave results from depolarization of the sinoatrial (SA) node in the heart, leading to depolarization and contraction of the atrial cardiac muscle tissues. The QRS complex arises from depolarization of the atrioventricular (AV) node and results in depolarization and contraction of the ventricular muscle tissue. The Q and S components of the complex are not always exhibited in a normal ECG. The T-wave results from repolarization of the ventricular tissue.
The lower ECG is nonstandard. In place of the P and Q waves there is an irregular waveform. It appears that, in this case, the heart is in a constant state of atrial fibrillation. This normally is not a problem, as the atria contribute very little to the overall pumping ability of the heart, however it can present problems during strenuous exercise.
From [2], recorded and described by Georgia Tech students Josh Hammel and Chris Clarke, April 2006.

g) Apply the notch filter to the ECG signal and show that you can remove the interference. Make a two-panel subplot comparing the time-domain signals before and after filtering. Scale the x-axis to show three or four periods of the ECG..

h) Show spectrograms of the signal before and after filtering. Point out where you see the interference in the input signal's spectrogram, and where it has been removed in the output spectrogram.

References

[1] J. H. McClellan, R. Schafer, and M. Yoder, *DSP First*, 2nd ed. United Kingdom: Pearson Education Limited, 2016.

[2] ——, "Lab S-9: Interference Removal from Electro-Cardiogram (ECG) Signals," 2016. [Online]. Available: https://dspfirst.gatech.edu/database/?d=labs

[3] P. Raybaut. (2024) Spyder. [Online]. Available: https://docs.spyder-ide.org/

[4] Project Jupyter. (2024) Jupyter Lab. [Online]. Available: https://docs.jupyter.org

[5] NumPy. (2024) NumPy ver. 2.2. [Online]. Available: https://numpy.org/doc/2.2

[6] C. R. Harris et al., "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[7] Matplotlib. (2024) Matplotlib 3.10.0. [Online]. Available: https://matplotlib.org/stable

[8] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. [Online]. Available: https://doi.org/10.1109/MCSE.2007.55

[9] SciPy. (2025) SciPy ver. 1.15. [Online]. Available: https://docs.scipy.org/doc/scipy

[10] P. Virtanen et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. [Online]. Available: https://doi.org/10.1038/s41592-019-0686-2