Lab 2: Signals and Systems

Signals and Systems (for AI) - WBAI016-05, Season 2024-25

Course coordinator: J.D. Cardenas Cartagena [j.d.cardenas.cartagena@rug.nl]

Topics

- FIR Filters
- Frequency response of a FIR filters

---

1 Instructions

1.1 Deliverables, Submission and Deadline

- Submit the code to Themis at the following link: themis.housing.rug.nl/course/2024-2025/ai-SS/lab2
- Submit the documentation as a PDF on the Brightspace page for Lab 2.
- You can find the report template in the following link: www.overleaf.com/...
- The deadline is on **Friday, December 20, 2024, at 17:30.**

1.2 About the Documentation

The code documentation is a PDF file in which you, as the programmer, provide all the necessary high-level information to help users or other programmers understand the information workflow within your code. For this assignment, please consider the following requirements when writing the code documentation:

- Explain the process of developing the main idea behind the code.
- List any external libraries and dependencies.
- Describe the purpose of each function or class, detailing their input-output variables and parameters, including data types and definitions.
- Include any relevant equations from lectures or the textbook, and explain how they are implemented in the code.
- Describe how each function is used within the main routine.
- Outline any complex or non-trivial code sections, such as those involving global variables.
- Include references to other relevant documentation or external resources when necessary.
- Ensure diagrams, figures, and tables have descriptive labels, units, and captions, and are appropriately sized.
- Place captions above tables and below figures. Reference equations with `\eqref{}`. See Appendix A in the template for examples.
- Use pseudocode and diagrams to illustrate the information workflow in your code.

- Maintain rigorous mathematical notation and style.
- The report should be a maximum of six pages, including references and appendices but excluding the cover page, with a preferred length of fewer than six pages.

Consider the following examples of documentation about the implementation of the stochastic gradient descent optimizer as a class:

- **Pytorch:** pytorch.org/...
  Source code: pytorch.org/...
- **Keras:** keras.io/...
  Source code: github.com/...

> **Remark:** The examples above explain how to use the class SGD. Such a class includes inputs, outputs, parameters, and functions. However, their code is more complex than expected for this assignment. The requested documentation for this lab assignment is a high-level description of your code using the criteria above. If you have questions about this deliverable, please contact the Teaching Team.

1.3 About the Code

- The code should be written in Python.
- Use basic clean code practices when writing the code: follow OOP principles when needed, use meaningful names for variables and functions, use comments to explain the code to someone else, keep lines short, and functions should do one thing.
- Ensure that the results are replicable.
- Type hinting[1] and following PEP8 style[2] are recommended to improve code readability

1.4 Grading Criteria

- The grade in the lab is a weighted sum of two deliverables: code (70%) and documentation (30%).
- Coherence with the theories and concepts discussed during the course and related literature.
- Simplicity and clarity in the solutions. Easy-to-read-and-understand answers are preferred.
- Code is manually checked for readability, hard coding, and plagiarism.
- The teaching team expects students to discuss the assessment with peers and utilize various tools to solve it; however, we request originality on the submitted work.
- Review lecture 0 and the course information page at Brightspace for further details about the assessment policy.
- This assessment is strongly recommended for pairs.

[1] For further information, refer to mypy.readthedocs.io/...
[2] For further information, refer to peps.python.org/...

1.5 Late Submission Policy

If you submit one of the components (code or documentation) late, a point from that component will be deducted every 24 hours, with a maximum of three days. After 72 hours, that component is considered not delivered and will be graded with a 0.

1.6 Support and Feedback

Each week, the TAs lead the computer labs where you can ask questions and individual support on your work. In case you require additional support in the assessment, first contact the Teaching Team:

- Laura Quiros Conesa: l.m.quiros.conesa@student.rug.nl
- Aleksandar Todorov: a.todorov.4@student.rug.nl
- Iulia Capralova: i.capralova@student.rug.nl
- Łukasz Sawala: l.h.sawala@student.rug.nl
- And always add Juan Diego [j.d.cardenas.cartagena@rug.nl] in CC.

Feel free to point out topics in the assignment to reinforce during the lectures and tutorials. The teaching team appreciates such feedback, and it does not influence the grades.

2 Problems

2.1 FIR Filter

The input for this problem is a kernel $h[n]$ of an FIR filter and a discrete input signal $x[n]$. The output should be the signal $y[n]$ obtained by feeding the FIR filter the input $x[n]$. The program should only show the samples of $y[n]$ where $x[n]$ and $h[n]$ have overlap in the sliding window process.

Example 1:

```
input:
2: [1, -1]
6: [0, 1, 1, 1, 1, 0]
output:
7: [0, 1, 0, 0, 0, -1, 0]
```

Example 2:

```
input:
6: [0, 1, 1, 1, 1, 0]
2: [1, -1]
output:
7: [0, 1, 0, 0, 0, -1, 0]
```

Example 3:

```
input:
3: [0, 1, 0]
3: [1, 2, 3]
output:
5: [0, 1, 2, 3, 0]
```

2.2 Cascading FIRs

This problem involves placing k FIR filters in a cascade. The output of filter i is the input to filter $i + 1$. The first line of the input contains k. Next follow the k filter kernels $h_1, h_2, \ldots, h_k$. The last input line is the input signal $x[n]$. The output should be the signal $y[n]$.

Example 1:

```
input:
2
2: [1, -1]
2: [1, -1]
6: [0, 1, 1, 1, 1, 0]
output:
8: [0, 1, -1, 0, 0, -1, 1, 0]
```

Example 2:

```
input:
5
2: [1,-1]
2: [1,-1]
2: [1,-1]
2: [1,-1]
2: [1,-1]
10: [0,0,0,1,1,1,1,0,0,0]
output:
15: [0,0,0,1,-4,6,-4,0,4,-6,4,-1,0,0,0]
```

2.3 Inverse of an FIR Filter

Write the documentation for this problem.

The input consists of a discrete input signal $x[n]$ followed by the output $y[n]$ obtained by feeding $x[n]$ to an unknown system. The program should determine if the system might be an FIR system. If it is not, the output should be **NO FIR**. If it is, the program should output the impulse response $h[n]$.

Example 1:

```
input:
2: [1, -1]
5: [4, -2, 2, -2, -2]
output:
4: [4, 2, 4, 2]
```

Example 2:

```
input:
2: [1, -1]
5: [4, -2, 1, -2, -2]
output:
NO FIR
```

Example 3:

```
input:
2: [1, -1]
4: [3, -1, -1, -1]
output:
3: [3, 2, 1]
```

2.4 Frequency Response of a FIR Filter

When the input $x[n]$ of a FIR system takes the form $x[n] = A_x \cos(\hat{\omega}n+\phi_x)$, the output $y[n] = h[n]*x[n]$ can also be represented in the form $y[n] = A_y \cos(\hat{\omega}n + \phi_y)$. The input consists of two lines: The first line holds the impulse response $h[n]$ of the FIR filter. The second line holds three floating point numbers $A_x, \hat{\omega}$, and ϕ_x representing the input signal $x[n]$. The program should calculate the output signal $y[n]$. The values of A_y, $\hat{\omega}$, and ϕ_y should be rounded to two decimal places. Note that if $A_y = 0$, the output should be $y[n] = 0.00$, and that $-\pi < \phi_y \leq \pi$.

Example 1:

```
input:
3: [1, 2, 1]
3 1.0472 -1.5708
output:
y[n] = 9.00 cos(1.05n - 2.62)
```

Example 2:

```
input:
3: [1, 1, 1]
6 2.094395 0
output:
y[n] = 0.00
```

Example 3:

```
input:
3: [1, 2, 1]
3 2.7489 2.75
output:
y[n] = 0.46 cos(2.75n + 0.00)
```

2.5 Parallel FIR Filter Frequency Response

Write the documentation for this problem.

This problem is similar to the previous problem, except that we have placed two FIR filters in parallel, so that the input $x[n]$ passes through two FIR filters $h_1[n]$ and $h_2[n]$ before being combined into the output $y[n]$. The layout of this system is depicted in the figure below.

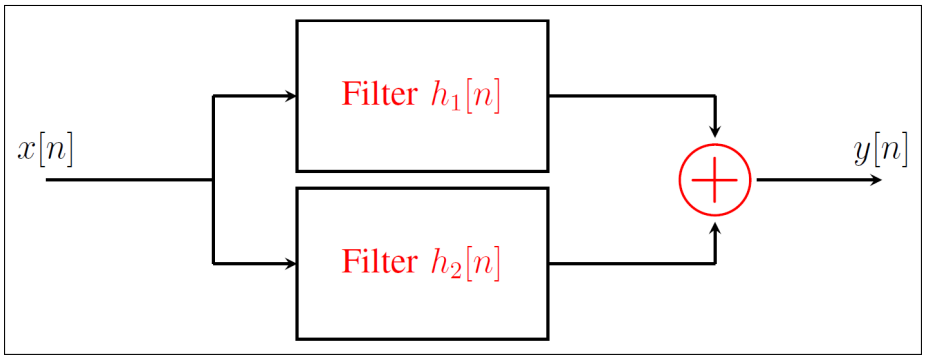

Figure 1: Parallel FIR filter

The input of this problem consists of three lines: The first two lines holds the impulse reponses $h_1[n]$ and $h_2[n]$ of the FIR filters. The final line holds three floating point numbers $A_x, \hat{\omega}$, and ϕ_x representing the input signal $x[n] = A_x \cos(\hat{\omega}n + \phi_x)$. Your program should calculate the output signal $y[n] = A_y \cos(\hat{\omega}n + \phi_y)$. The values of $A_y, \hat{\omega}$, and ϕ_y should be rounded to two digits after the decimal dot. Note that if $A_y = 0$, the output should be $y[n] = 0.00$, and that $-\pi < \phi_y \leq \pi$.

Example 1:

```
input:
3: [0, 1, 1]
2: [1, 1]
3 1.0472 -1.5708
output:
y[n] = 9.00 cos(1.05n - 2.62)
```

Example 2:

```
input:
2: [4, -2]
3: [-3, 3, 1]
6 2.094395 0
output:
y[n] = 0.00
```

Example 3:

```
input:
4: [0, 2, 0, -2]
4: [1, 0, 1, 2]
3 2.7489 2.75
output:
y[n] = 0.46 cos(2.75n + 0.00)
```

2.6 Sobel Edge Detection

Sobel edge detection makes use of convolution to determine where edges exist in a grayscale image. In particular, Sobel edge detection makes use of two kernels H_x and H_y, given by the matrices

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

For a given image X, the Sobel edge detection image Y is given by

$$Y[n, m] = \sqrt{(H_x * X)[n, m]^2 + (H_y * X)[n, m]^2}.$$

The goal of this exercise is to implement this edge detection algorithm. We will ignore edge effects. That is, for any given input X (say, 100×200 pixels), the output Y is 2 pixels shorter and narrower than the input (that is, 98×198 pixels).
Starter code is available on Brightspace and Themis. This starter code includes functionality to read in a grayscale image and to produce the appropriate output. Your task is to implement the function `performSobelEdgeDetection` that produces a grayscale image that is exactly 2 pixels narrower and 2 pixels shorter than the input, and that represents the result of performing Sobel edge detection on the input image.
This exercise has only two inputs, both of which can be downloaded from Brightspace and Themis.

Example 1:
`input:`

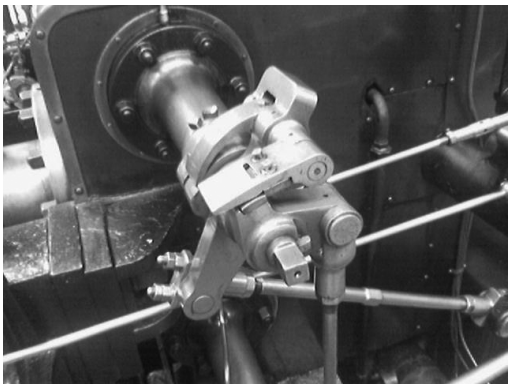

Figure 2: Machine Image

`output:`
`15384661`

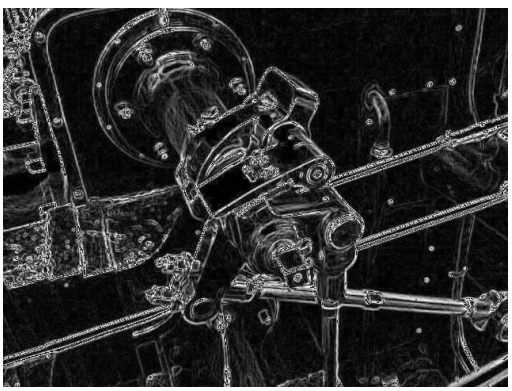

Figure 3: Machine Image after edge detection

Example 2:
`input:`

Figure 4: Flower Image

`output:`
`15494790`

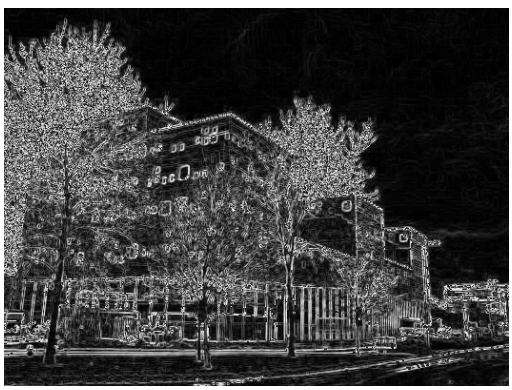

Figure 5: Flower Image after edge detection