Lab 3: 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

- Fourier Transform
- Z-Transform

---

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/lab3
- Submit the documentation as a PDF on the Brightspace page for Lab 3.
- You can find the report template in the following link: www.overleaf.com/...
- The deadline is on **Friday, January 17, 2025, 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 eight pages, including references and appendices but excluding the cover page, with a preferred length of fewer than eight 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.
- You cannot use libraries that implement the algorithms directly, e.g., `numpy.fft`. You must implement the algorithms from scratch.
- 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 Template matching using Pearson correlator

While convolution can be used to perform template matching, this approach does not take into account the magnitude of the input. To counter this problem, a Pearson correlator can be used, which calculates the Pearson correlation between two equal length inputs, in our case the filter $h[n]$ and the signal input $x[n]$.

$$y[n] = \frac{\sum_{k=0}^{L-1}(x[n+k]-\bar{x})(h[k]-\bar{h})}{\sqrt{\sum_{k=0}^{L-1}(x[n+k]-\bar{x})^2}\cdot\sqrt{\sum_{k=0}^{L-1}(h[k]-\bar{h})^2}},$$

$$= \frac{L(\sum_{k=0}^{L-1}x[n+k]h[k]) - (\sum_{k=0}^{L-1}x[n+k])(\sum_{k=0}^{L-1}h[k])}{\sqrt{L(\sum_{k=0}^{L-1}(x[n+k])^2) - (\sum_{k=0}^{L-1}x[n+k])^2}\cdot\sqrt{L(\sum_{k}(h[k])^2) - (\sum_{k}h[k])^2}},$$

where L is the length of the filter, and $\bar{h} = \frac{1}{L}\sum_{k=0}^{L-1}h[k]$ is the mean of the filter coefficients.
This non-linear filter scales output to a value between -1 and 1. The value 1 indicates a perfect positive linear relation between the inputs (perfect match), whereas -1 indicate a perfect negative linear relation between the inputs (inverse match). The value 0 indicates no linear relation between inputs (no match).
The input for this problem are a template $h[n]$, and a discrete input signal $x[n]$. The output should be sequence of Pearson correlations $y[n]$ between $x[n]$ and $h[n]$, rounded to 5 digits after the decimal dot.

- **Example 1:**
  input:
  3: [2,3,5]
  10: [1,2,3,5,4,420,1,12,13,15]
  output:
  8: [0.98198,1.00000,0.32733,0.94423,-0.19509,-0.74065,0.80296,1.00000]

- **Example 2:**
  input:
  3: [17,2,19]
  10: [1,6,12,11,19,4,21,12,3,9]
  output:
  8: [0.15959,-0.54127,0.67900,-0.92966,1.00000,-0.82675,-0.10762,0.90419]

- **Example 3:**
  input:
  2: [1,42]
  10: [1,42,1,42,1,42,10,52,10,52]
  output:
  9: [1.00000,-1.00000,1.00000,-1.00000,1.00000,-1.00000,1.00000,-1.00000,1.00000]

2.2 Discrete Fourier transform using Vandermonde matrix

In this exercise you are asked to implement the discrete Fourier transform by utilizing a Vandermonde matrix. An $N \times N$ Vandermonde matrix is defined by:

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}, \quad (1)$$

where $\omega = e^{-j2\pi/N}$ is the N-th root of unity. Consider an input $x[n]$ of length L. Apply the DFT by matrix-vector multiplication with an L $\times$ L Vandermonde matrix and input $x[n]$.
The input for this assignment consists of a single signal $x[n]$. On the output, print the complex number(s) $X[k]$ corresponding to the DFT of $x[n]$ in the form $a+jb$, where a and b are rounded to 2 digits after the decimal dot.

Example 1:

```
input:
4: [1,2,3,4]
output:
10.00+j0.00
-2.00+j2.00
-2.00+j0.00
-2.00-j2.00
```

Example 2:

```
input:
5: [1,2,3,3,4]
output:
13.00+j0.00
-2.00+j1.90
-2.00+j1.18
-2.00-j1.18
-2.00-j1.90
```

Example 3:

```
input:
3: [3,1,4]
output:
8.00+j0.00
0.50+j2.60
0.50-j2.60
```

2.3 Inverse DFT using Vandermonde matrix

Write the documentation for this problem.

Implement the inverse discrete Fourier transform by adapting the code of the previous exercise. Note that the structure of the input and output is identical.

Example 1:

```
input:
4: [1,2,3,4]
output:
2.50+j0.00
-0.50-j0.50
-0.50+j0.00
-0.50+j0.50
```

Example 2:

```
input:
5: [1,2,3,3,4]
output:
2.60+j0.00
-0.40-j0.38
-0.40-j0.24
-0.40+j0.24
-0.40+j0.38
```

Example 3:

```
input:
3: [3,1,4]
output:
2.67+j0.00
0.17-j0.87
0.17+j0.87
```

2.4 Fast Fourier transform

The goal of this problem is to implement the recursive version of the Fast Fourier Transform (FFT), as presented in class. The input for this problem is a sequence $x[n]$. The output should be the discrete Fourier transform $X[k]$ of the input sequence, as obtained by the FFT where every component is represented as a complex number $a + jb$, with a, b rounded to 2 digits after the decimal dot. Note that if the length of the input is not a power of 2, you will need to pad the input sequence.

Example 1:

```
input:
4: [1,2,3,4]
output:
10.00+j0.00
-2.00+j2.00
-2.00+j0.00
-2.00-j2.00
```

Example 2:

```
input:
5: [1,2,3,3,4]
output:
13.00+j0.00
-3.71-j6.54
2.00+j1.00
-2.29-j0.54
3.00+j0.00
-2.29+j0.54
2.00-j1.00
-3.71+j6.54
```

Example 3:

```
input:
3: [3,1,4]
output:
8.00+j0.00
-1.00-j1.00
6.00+j0.00
-1.00+j1.00
```

2.5 z-domain to time domain

Write the documentation for this problem.

A FIR filter is described, up to a constant, by the roots or zeros of the system function X(z). The input of this problem consists of two lines. The first line contains the number M of complex-valued roots of the system function X(z). The second line contains M numbers that represent the roots $\{z_k\}$ as angles ϕ_k, so that $z_k = e^{j\phi_k}$. The output of the program should be the corresponding time-domain impulse response function h[n], assuming that the constant factor is 1. Each value of h[n] should be real-valued (i.e. not complex-valued) and rounded to 2 digits after the decimal point.

Example 1:

```
input:
1
0
output:
2: [1.00,-1.00]
```

Example 2:

```
input:
2
1.047198 -1.047198
output:
3: [1.00,-1.00,1.00]
```

Example 3:

```
input:
4
1.256 2.513 -1.256 -2.513
output:
5: [1.00,1.00,1.00,1.00,1.00]
```

2.6 Pattern matching using a 2D Pearson correlator

In this problem you will perform Pearson correlation on digital gray scale images. The input for this problem are a PGM (simple image format) image, and a small template image (also a PGM image). Most of the code has been written for you, and can be found in the file pearson2D.c on Brightspace. The input image that we will use is called smb_w11.pgm. It is a screenshot taken from Super Mario Bros, world 1-1. The template is a question mark box that appear throughout the levels, and is called box.pgm. These two file names are hard-coded in the code of this exercise. Of course, you normally would not do that, but in this case it is necessary to pass the test in Themis. So, do not change these file names! You can display these images using the Linux command display. The objective of this exercise is to count (and of course, not by hand!) how many boxes are present in the screenshot.

The number of occurrences must be printed on the output. Note that this problem has only one test case in Themis. Of course, hardcoding the number without doing calculation results in zero score for this problem. The file pearson2D.py is a completely functional program, except for the routine fft2D, which should compute the 2-dimensional FFT of an image. In the file you will see that the code of this routine looks like:

```
def fft2D(direction, re, im):
    '''YOU HAVE TO IMPLEMENT THE BODY OF THIS FUNCTION YOURSELF'''
    '''YOU CAN USE THE fft1D() FUNCTION ABOVE'''

    # 1) Get the shape of the real part of the image

    # 2) Perform 1D FFT on each row of the real and imaginary parts of the image

    # 3) Initialize arrays to hold the real and imaginary parts of the column transform

    # 4) Perform 1D FFT on each column of the real and imaginary parts of the image
```

A nice property of the 2D FFT is that it can be implemented using only 1D FFTs, by performing two passes. In the first pass, we perform a 1D FFT on each row (real to complex) of the image. In the second pass, we perform a 1D FFT on each column resulting from the first pass (complex to complex). The same holds for the inverse 2D FFT. The one-dimensional FFTs have been implemented for you, and are already present in the file pearson2D.py. Note that this implementation of the 1D FFT is not recursive, but it functions the same way as the recursive version.
Once you implemented the function fft2D correctly, you can run the program. The output is the number of boxes in the screenshot. Moreover, as a side-product, the program produces the file matches.pgm. Use the program display to see the result.

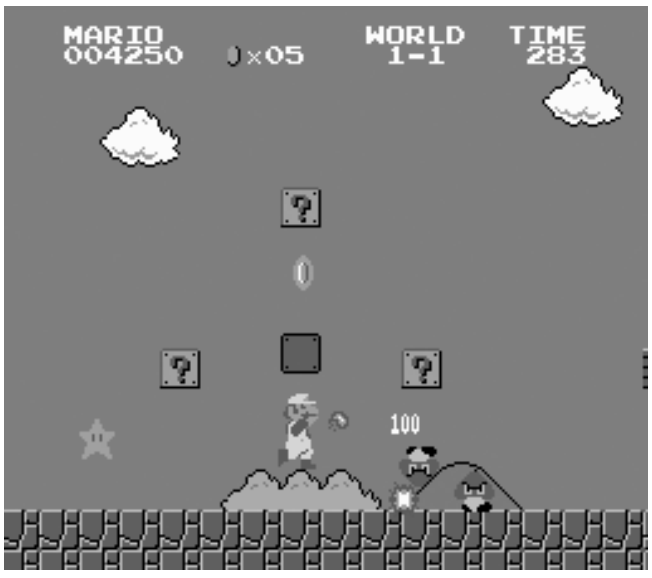

Figure 1: Input image for the pattern matching problem