Lab 1: 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

- Sinusoids
- Spectrum representation
- Sampling and aliasing

---

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/lab1
- Submit the documentation as a PDF on the Brightspace page for Lab 1.
- You can find the report template in the following link: www.overleaf.com/...
- The deadline is on **Friday, December 6, 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 Cartesian coordinates to Polar coordinates

Write a program that reads a Cartesian coordinate (x, y) from the input and converts it into the corresponding Polar coordinate (r, θ). The input consists of two integers: x and y. The output of your program should be two floating-point numbers (use doubles in your programs): r (length, so a non-negative number) and the principal value of θ (in radians). The output values should be rounded to two digits after the decimal point.

Example 1:

```
input:
2 0
output:
2.00 0.00
```

Example 2:

```
input:
0 2
output:
2.00 1.57
```

Example 3:

```
input:
-2 0
output:
2.00 3.14
```

2.2 Polar coordinates to Cartesian coordinates

Write a program that reads a Polar coordinate (r, θ) from the input and converts it into the corresponding Cartesian coordinate (x, y). The input consists of two floating-point numbers: r (a non-negative number) and θ (in radians). The output should be two floating-point numbers (rounded to two digits after the decimal point).

Example 1:

```
input:
1.00 1.57
output:
0.00 1.00
```

Example 2:

```
input:
1.00 0.00
output:
1.00 0.00
```

Example 3:

```
input:
1.00 -1.57
output:
0.00 -1.00
```

2.3 Sum of sinusoids of the same frequency

In this problem, you are constructing the sum of a number of sinusoids with the same frequency. The input consists of several lines. The first line contains two integers: the frequency f (in Hz) and the number n of sinusoids to sum.
The following n lines each contain two floating-point numbers, representing the amplitude A_i and phase ϕ_i of the i-th sinusoid $A_i \cos(2\pi f t + \phi_i)$. Your program should calculate the resulting signal

$$x(t) = \sum_i A_i \cos(2\pi f t + \phi_i) = A \cos(2\pi f t + \phi)$$

The values of A and ϕ should be rounded to two digits after the decimal point. Note that if $A = 0$, the output should be $x(t) = 0.00$.

Example 1:

```
input:
42 2
1 1.047198
1 0.523599
output:
x(t)=1.93cos(2*pi*42*t+0.79)
```

Example 2:

```
input:
100 3
5 3.665191
5 1.570796
5 -0.5235988
output:
x(t)=0.00
```

Example 3:

```
input:
10 2
3 1.570796
4 0
output:
x(t)=5.00cos(2*pi*10*t+0.64)
```

2.4 Aliasing

The input for this problem is a single line containing two integers representing the signal frequency f_0 (in Hz) and the sampling frequency f_s (in Hz). The output should be a single integer that represents the frequency (in Hz) of the reconstructed sinusoid after a sinusoid with frequency f_0 has been sampled with frequency f_s.

Example 1:

```
input:
100 500
output:
100
```

Example 2:

```
input:
100 100
output:
0
```

Example 3:

```
input:
100 90
output:
10
```

2.5 Products of sinusoids

Write the documentation for this problem.

A beat note can be represented as a product of two sinusoids. In this problem, we consider the product

$$\prod_i \cos(2\pi f_i t)$$

of arbitrary numbers of sinusoids $\cos(2\pi f_i t)$ and determine what non-negative frequencies are present in the spectrum.

The input consists of several lines. Each line contains a single integer representing the frequency f_i (in Hz) of one of the sinusoids in the product. The integer 0 signals the end of the input. The output should be a list of all the non-negative frequencies (in Hz) in the spectrum, ordered from smallest to largest. Note that the spectrum does not contain duplicate frequencies.

Example 1:

```
input:
2000
515
27
0
output:
1458
1512
2488
2542
```

Example 2:

```
input:
300
20
0
output:
280
320
```

Example 3:

```
input:
100
100
100
0
output:
100
300
```

2.6 Nyquist rate

A beat note can be represented as a product of two sinusoids. In this problem, we consider the product

$$\prod_i \cos(2\pi f_i t)$$

of arbitrary numbers of sinusoids $\cos(2\pi f_i t)$ and determine what the Nyquist rate would be for sampling the signal.

The input consists of several lines. Each line contains a single integer that represents the frequency f_i (in Hz) of one of the sinusoids in the product. The integer 0 signals the end of the input. The output should be a single integer representing the Nyquist rate (in Hz) needed to accurately reproduce the signal.

Example 1:

```
input:
2000
515
27
0
output:
5084
```

Example 2:

```
input:
300
20
0
output:
640
```

Example 3:

```
input:
100
100
100
0
output:
600
```

2.7 Multipath fading

Write the documentation for this problem.

When radio waves waves reflect off smooth surfaces, these reflections may interfere with the original signal. In such reflection cases, the received signal is a sum of the original signal and the reflected signal. Since these signals travel different distances, they differ in their time delay. Given a transmitted signal $s(t)$, the received signal $r(t)$ is therefore given by

$$r(t) = s(t - t_1) + s(t - t_2)$$

Consider the case where a receiver located at $(x, 0)$ receives a signal from a transmitter at $(0, 0)$, which is reflected off a reflector at (d_r, d_t), as shown below.

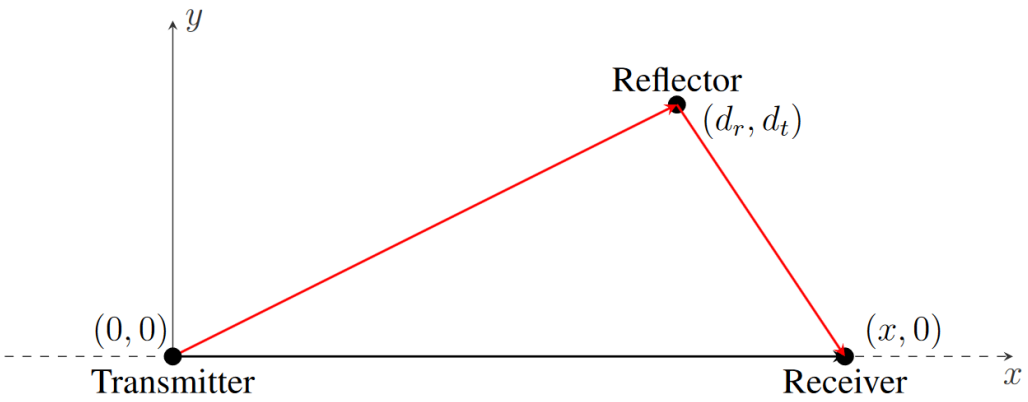

Figure 1: Multipath fading

In this case, the transmitter sends out the signal $s(t) = \cos(2\pi(150 \times 10^6)t)$, which travels at a speed of 3×10^8 m/s. For simplicity, we ignore propagation losses. That is, the amplitude is not affected by distance or reflection.

Write a program that reads a single line containing the integers d_r, d_t, and x (in m). The output should be a floating-point number (rounded to two digits after the decimal dot) that represents the amplitude of the received signal r.

Example 1:

```
input:
30 40 30
output:
2.00
```

Example 2:

```
input:
30 40 31
output:
0.04
```

Example 3:

```
input:
10 100 50
output:
1.90
```