Course introduction

General information

Lecture starts at **9:15** and ends at **10:45.** It consists of **Questions** 15 min, **Overview** 45 min and **Applications** 30 min.

Goal of the course

is to understand the logic behind methods: **problem statement** and **solution**. This course is not about programming rather about data analysis using programming language.

Interaction in the course

If you have any doubts about the method, please ask your questions at the beginning and after the **Overview**.

Assignments

There will be one assignment per lecture.
The assignments will be graded from **0** to **5**, and the lowest grade should be **2** in order to get credits.

The Monday's assignments should be uploaded before **Tuesday 16:00**, and the Friday's assignment should be uploaded before next **Monday 16:00**.

Feedback

Was it difficult (why)? Was it unclear (why)? Was it interesting (why)? Was it relevant (why)?

Course structure

Lecture 1, “Introduction to time series in neuroscience” / Origin of MEG, EEG and fMRI time series

Lecture 2, “Programming languages, software and scripts” / Writing scripts for data analysis in Python

Lecture 3, “Generative models for univariate time series, part I” / Modelling of time series (periodic functions and noise)

Lecture 4, “Generative models for univariate time series, part II” / Autoregressive models (AR and ARMA)

Lecture 5, “Analysis of univariate time series, part I” / Signal transformations (Fourier transform, etc.)

Lecture 6, “Analysis of univariate time series, part II” / Filtering of time series (FIR/IIR filters)

Lecture 7, “Analysis of univariate time series, part III” / Statistical filtering of time series

Lecture 8, “Analysis of multivariate time series, part I” / Vector autoregressive models (VAR)

Lecture 9, “Analysis of multivariate time series, part II” / Interactions between time series (correlation and causality)

Lecture 10, “Analysis of multivariate time series, part III” / Principal and independent component analysis

Lecture 11, “Clustering and classification, part I” / Clustering methods (k-means, hierarchical clustering, etc.)

Lecture 12, “Clustering and classification, part II” / Classification methods (linear discriminant analysis, SVM, etc.)

Motivation

Often do not know the exact mechanisms of the brain. **What do we do?** We make a hypothesis on how brain works and measure brain activity during specific task. Then we analyze these data to either **support** or **reject** our hypothesis.

Data analysis

The data analyses can be divided to **signal detection** and **information processing**.

signal detection

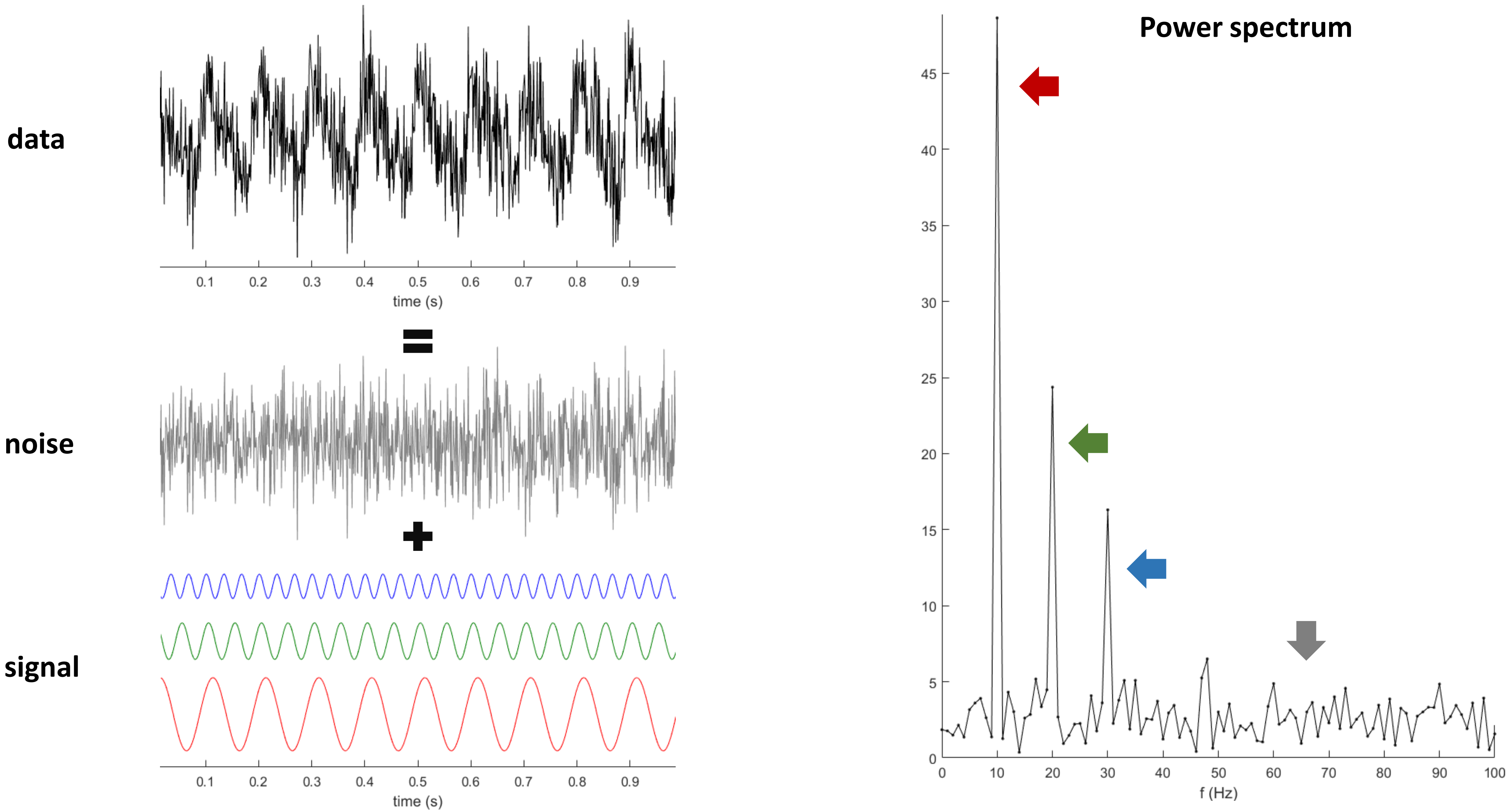
- **transformation** / e.g., Fourier transform
- **filtering** / e.g., FIR/IIR filters, wavelets
- **statistical filtering** / e.g., Wiener filter

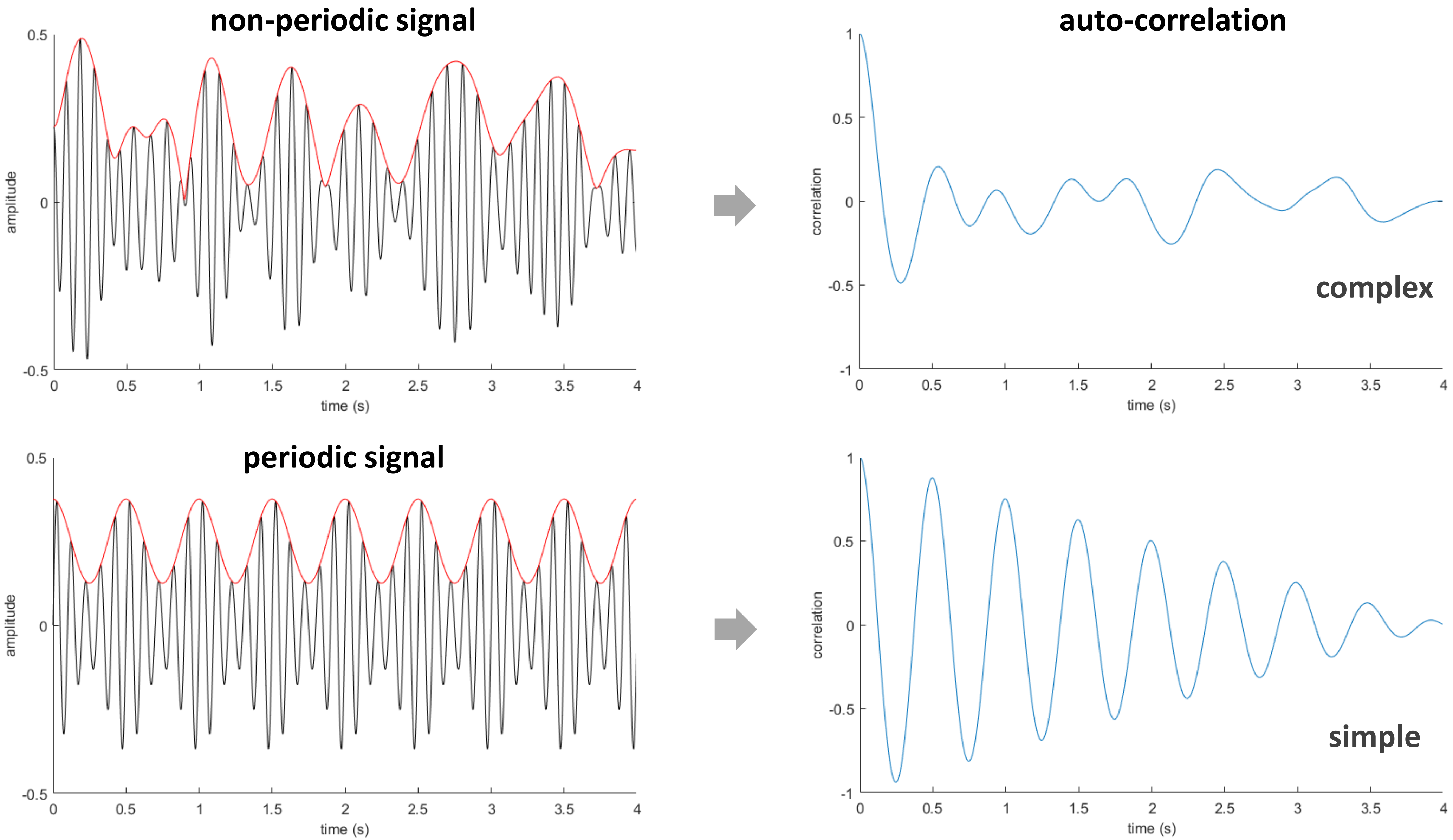
information processing

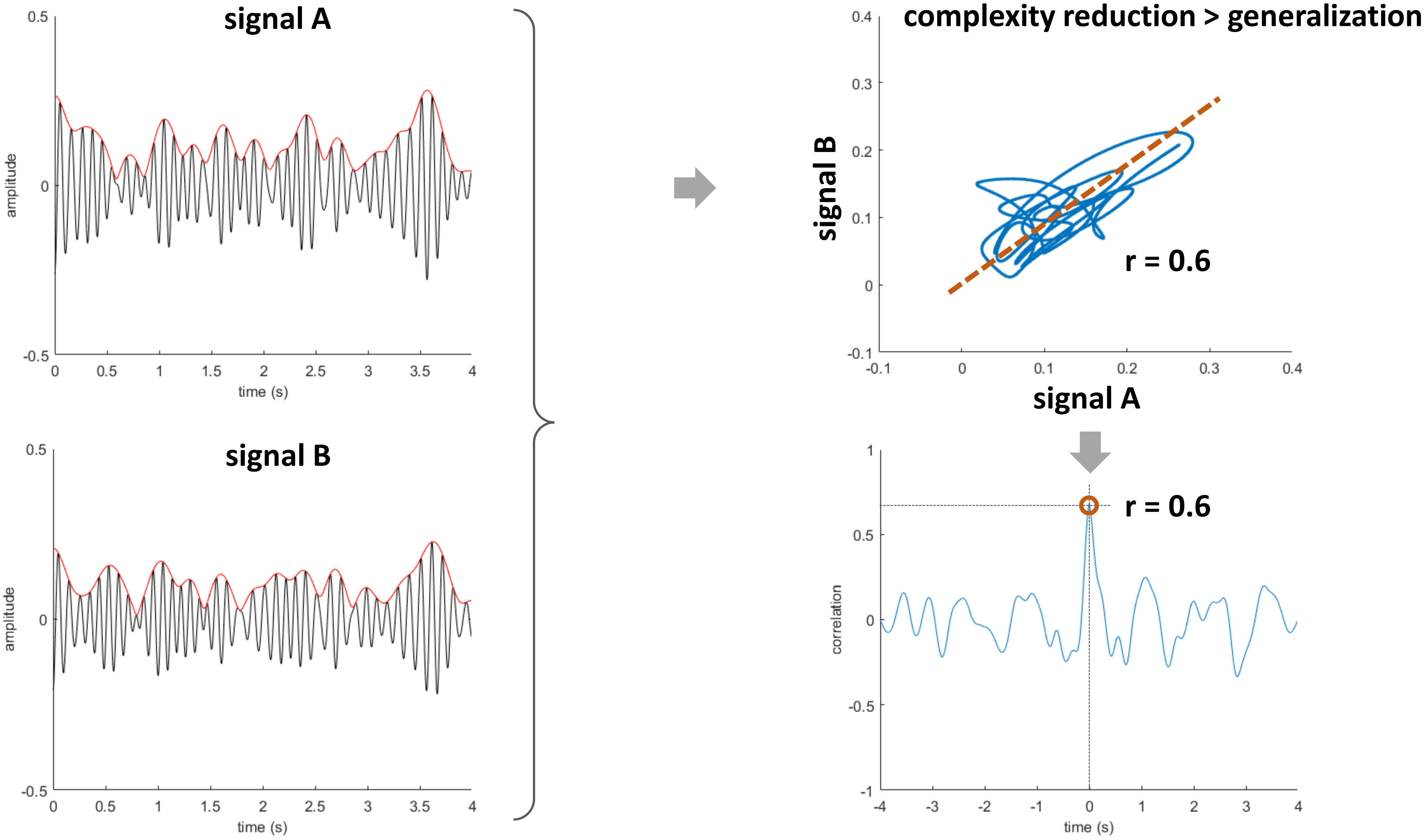
- **auto-correlation** / i.e., self-similarity
- **cross-correlation** / i.e., similarity with others
- **feature extraction** / e.g., PCA, ICA
- **clustering and classification**

Limitations

Most of the signal detection methods are borrowed from **radio engineering**. In radio engineering, there is a clear definition **what is signal** and **what is noise**. Often, it does not work in neuroscience because the brain mechanisms are extremely complicated. In this case, biologically realistic computational models could be the only solution.


data
noise
signal
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
time (s)
Power spectrum
0
5
10
15
20
25
30
35
40
45
0
10
20
30
40
50
60
70
80
90
100
f (Hz)


non-periodic signal
auto-correlation
complex
periodic signal
simple
amplitude
correlation
time (s)
0.5
0
-0.5
1
-1
0
0.5
1
1.5
2
2.5
3
3.5
4


signal A
0.5
0
-0.5
amplitude
0 0.5 1 1.5 2 2.5 3 3.5 4
time (s)
signal B
0.5
0
-0.5
amplitude
0 0.5 1 1.5 2 2.5 3 3.5 4
time (s)
complexity reduction > generalization
0.4 0.3 0.2 0.1 0 -0.1
signal B
-0.1 0 0.1 0.2 0.3 0.4
signal A
r = 0.6
1 0.5 0 -0.5 -1
correlation
-4 -3 -2 -1 0 1 2 3 4
time (s)
r = 0.6

Lecture 1. Introduction to time series in neuroscience

Outline / **overview**

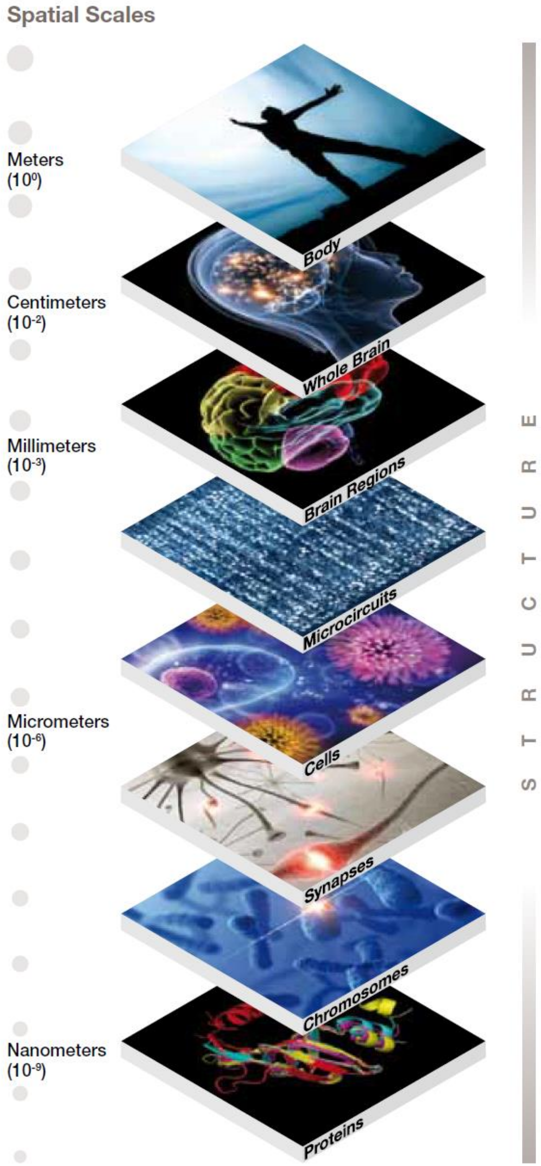
- **Section 1.** Origin of time series in neuroscience (EEG/MEG, fMRI, behavioral, physiological)
- **Section 2.** Recording of time series (ADC principles, signal sampling, …)
- **Section 3.** Programming languages (Python, MATLAB, …)

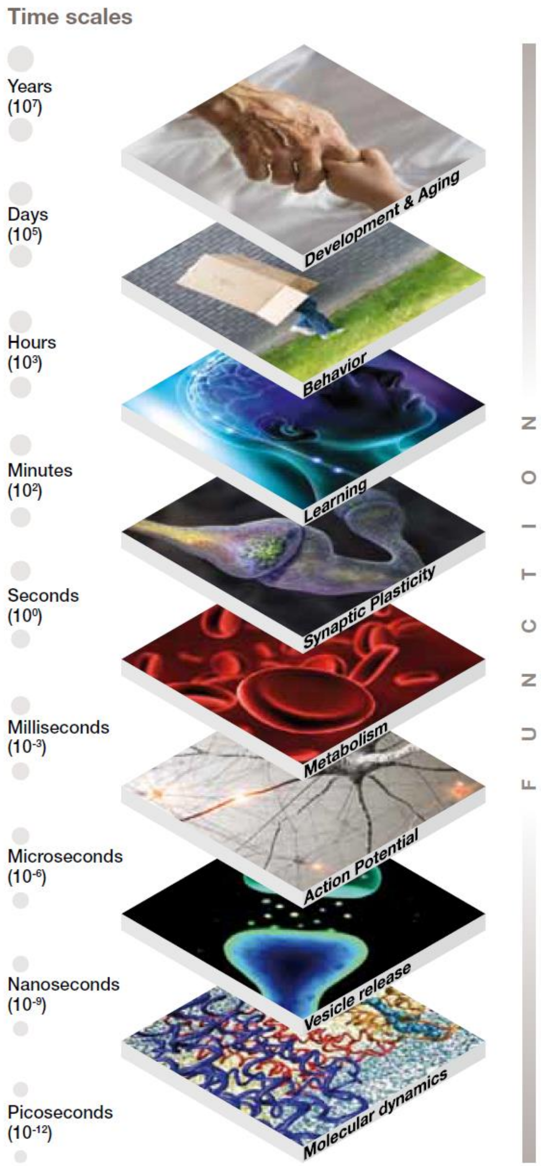
Outline / **application**

- **Section A1.** Installing Python
- **Section A2.** Basic libraries and examples

Section 1. Origin of time series in neuroscience

Time and space scales

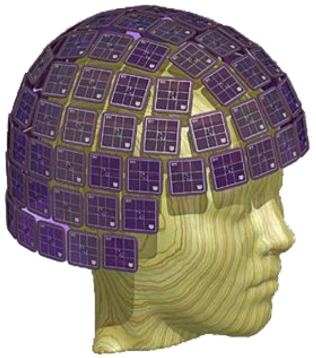


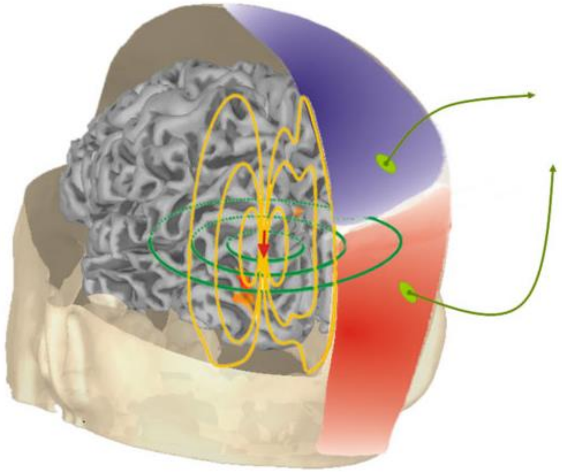


- **Electro-/Magneto-encephalography (EEG/MEG)**
- Functional magnetic resonance imaging (fMRI)
- Behavioral responses (e.g., reaction times)
- Physiological responses (e.g., heart-beats, temperature)

EEG

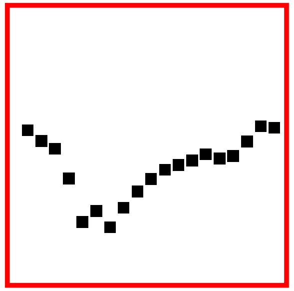
MEG

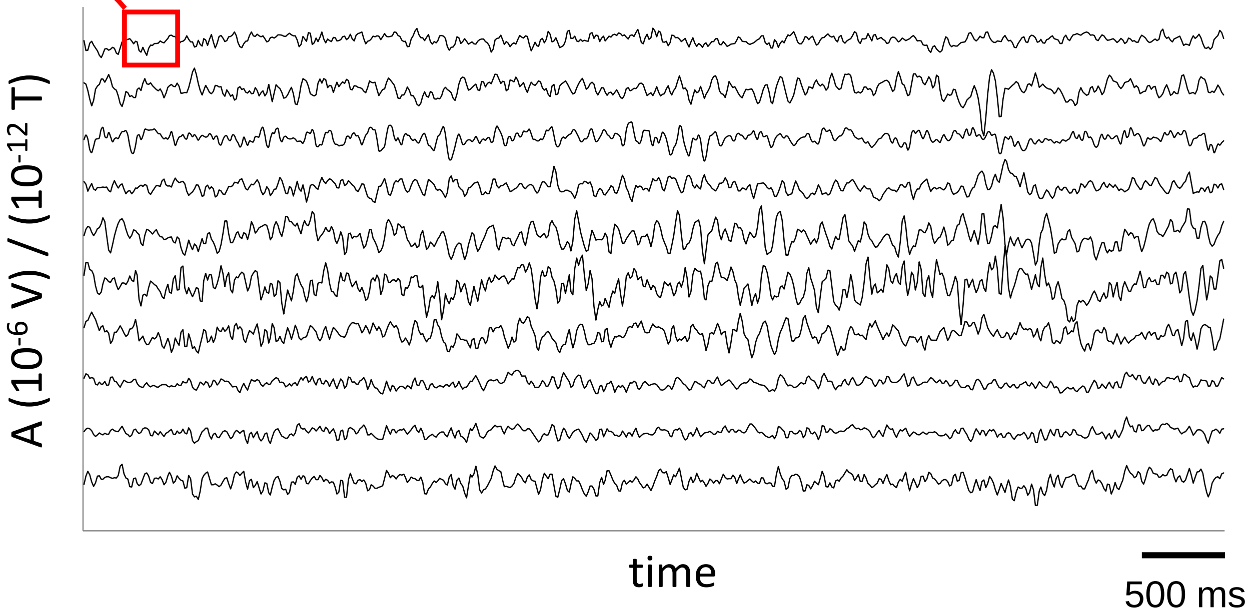




Magnetic field

Electric field



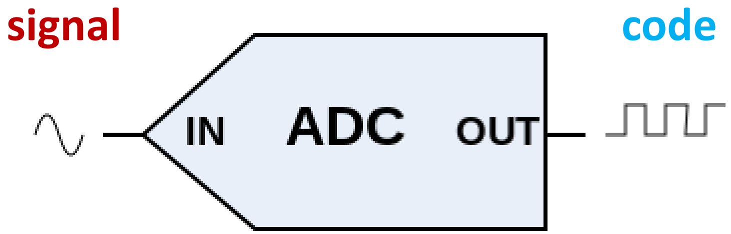


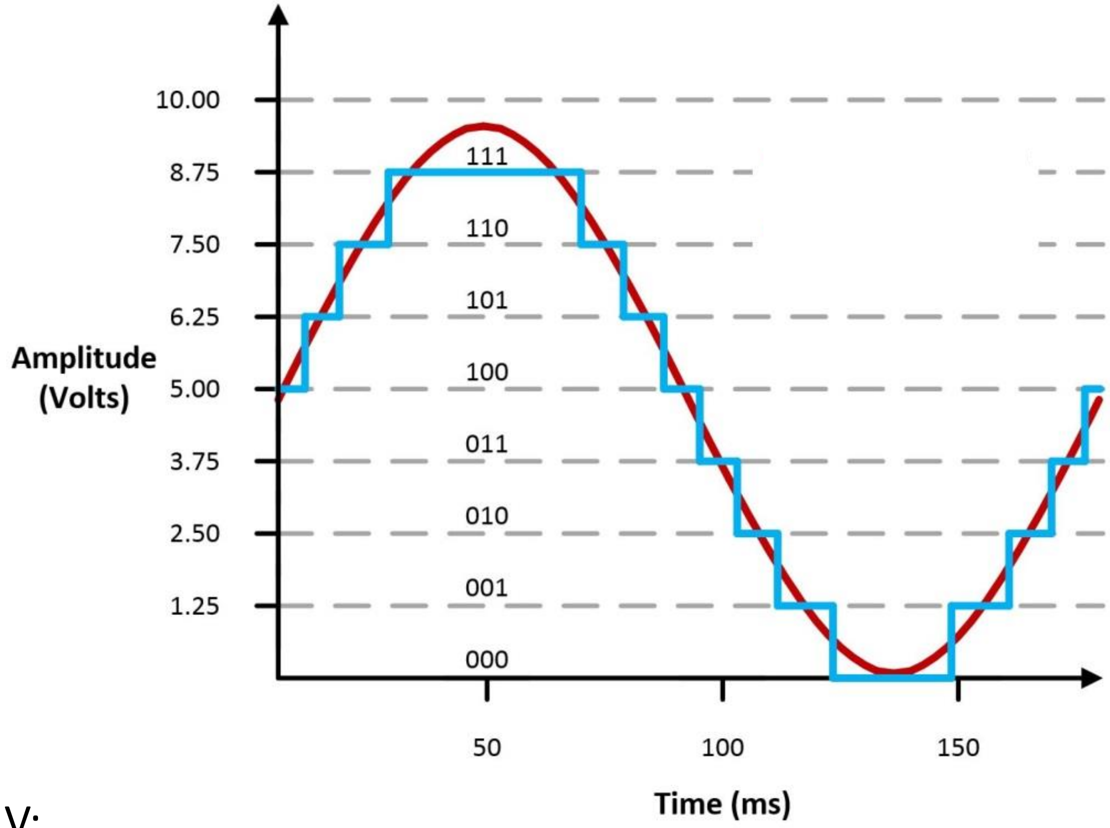
Section 2. Recording of time series

Type of signals

- **Analog** – continuous in time and amplitude
- **Digital** – discrete in time and amplitude

Analog-to-digital converter



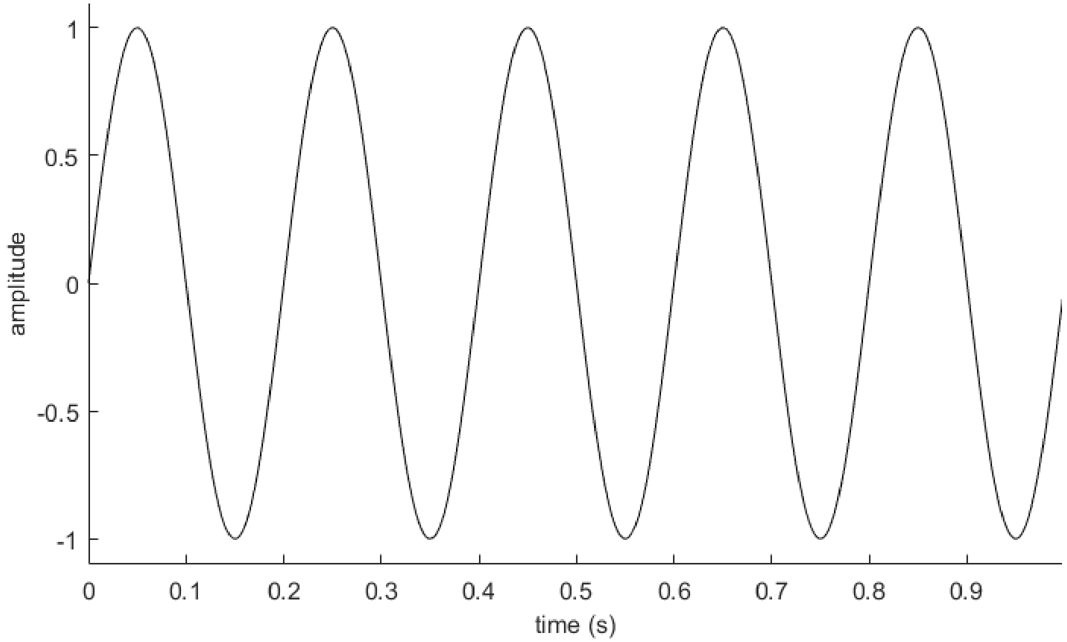


Key parameters of ADC

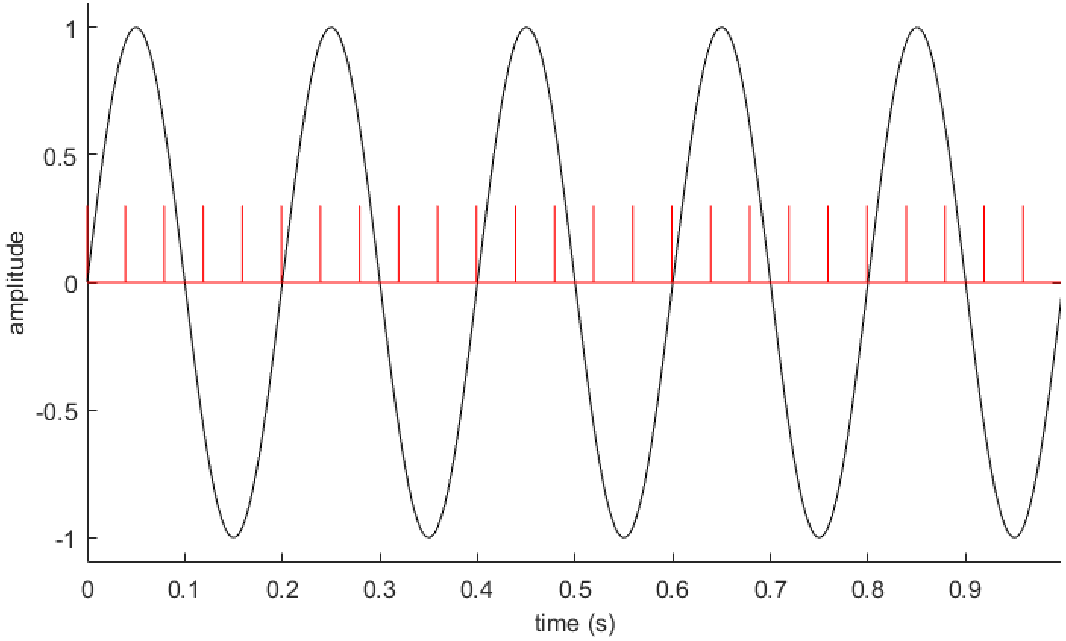
- **Resolution**, in bits. For instance, R = 10 bit; V_{ref} = [0, 2] V;
  $V_{min} = V_{ref} / R = (2 - 0) / 2 ^\wedge 10 = 0.002$ V
- **Sampling rate**, in Hz. For instance, 1000 Hz.

Principle of analog-to-digital conversion (1/2)

Original signal (continuous)

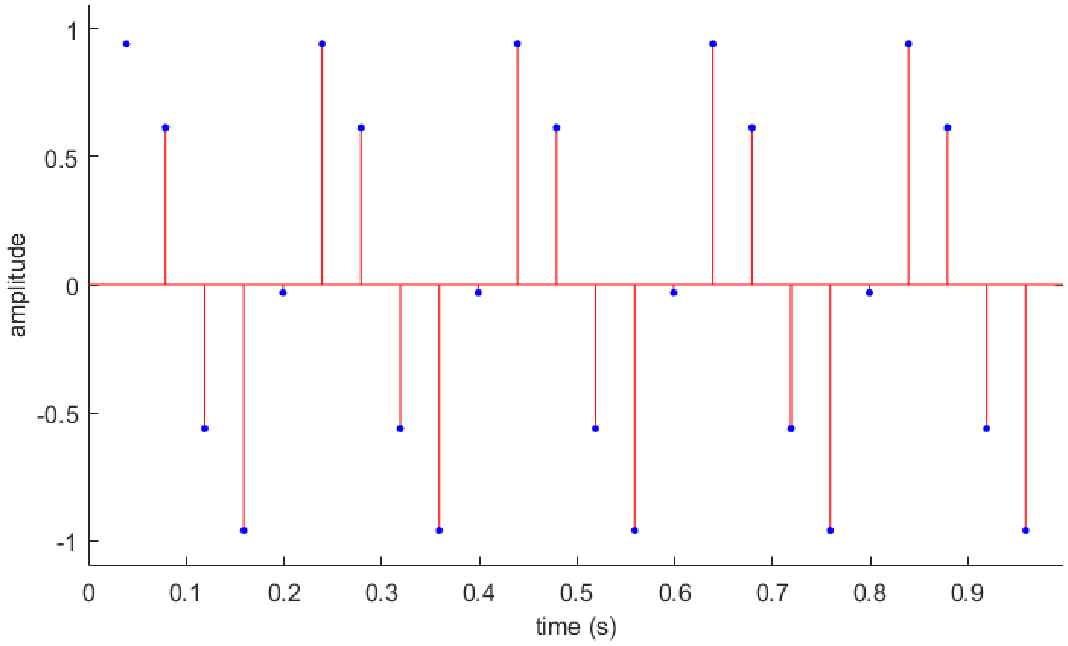


Sampling sequence

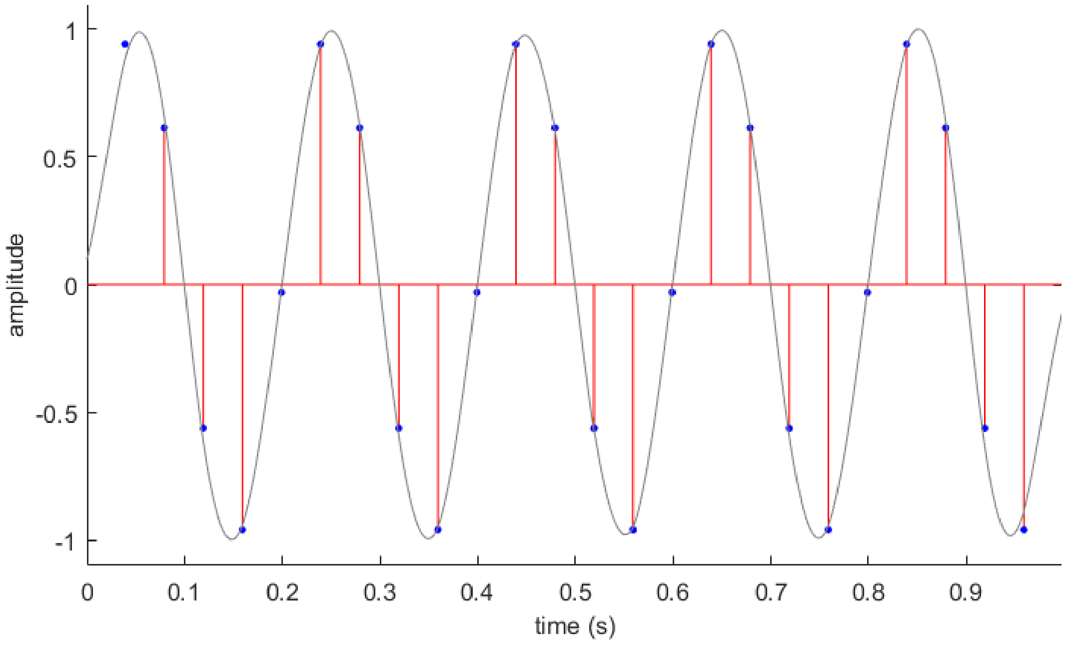


Principle of analog-to-digital conversion (2/2)

Sampled signal (discrete)

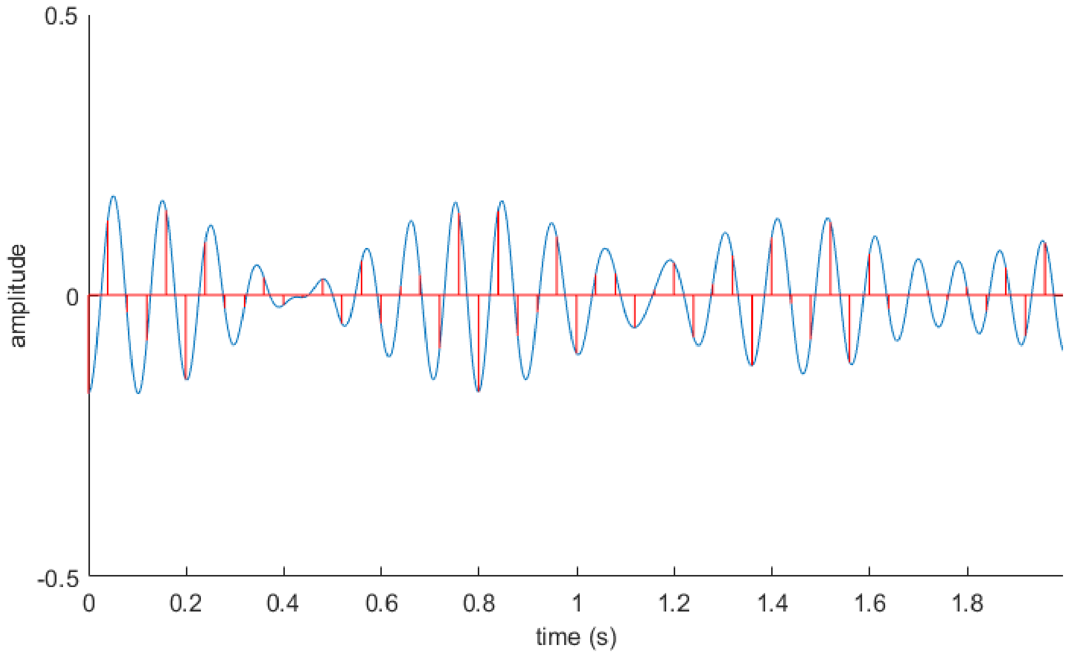


Reconstructed signal

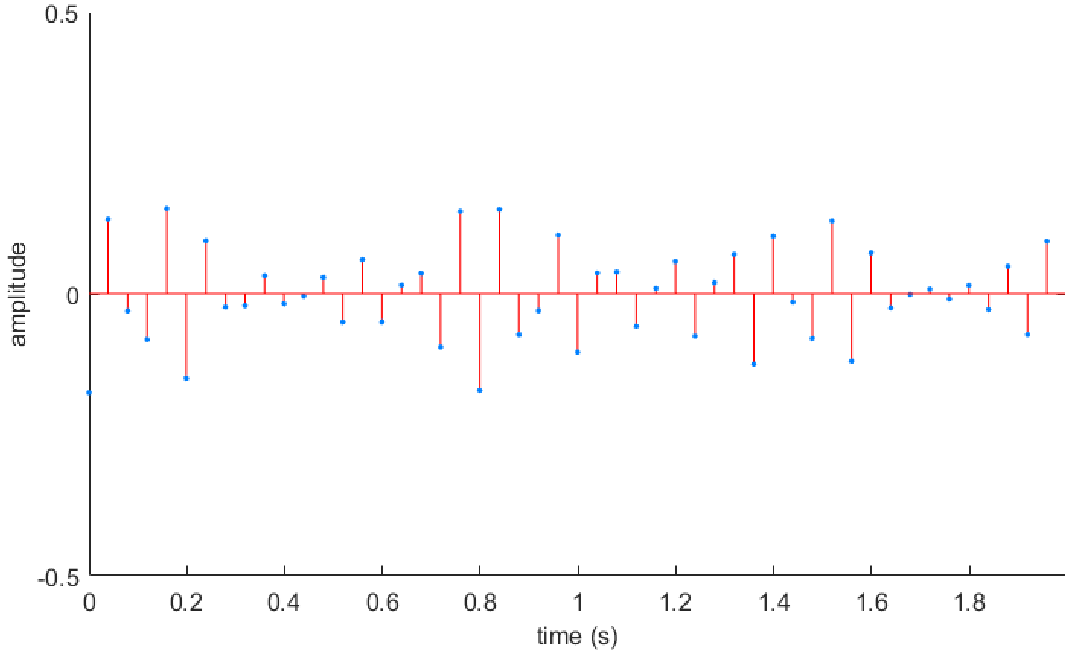


Temporal dependencies between samples – autocorrelation (1/3)

LOW sampling rate

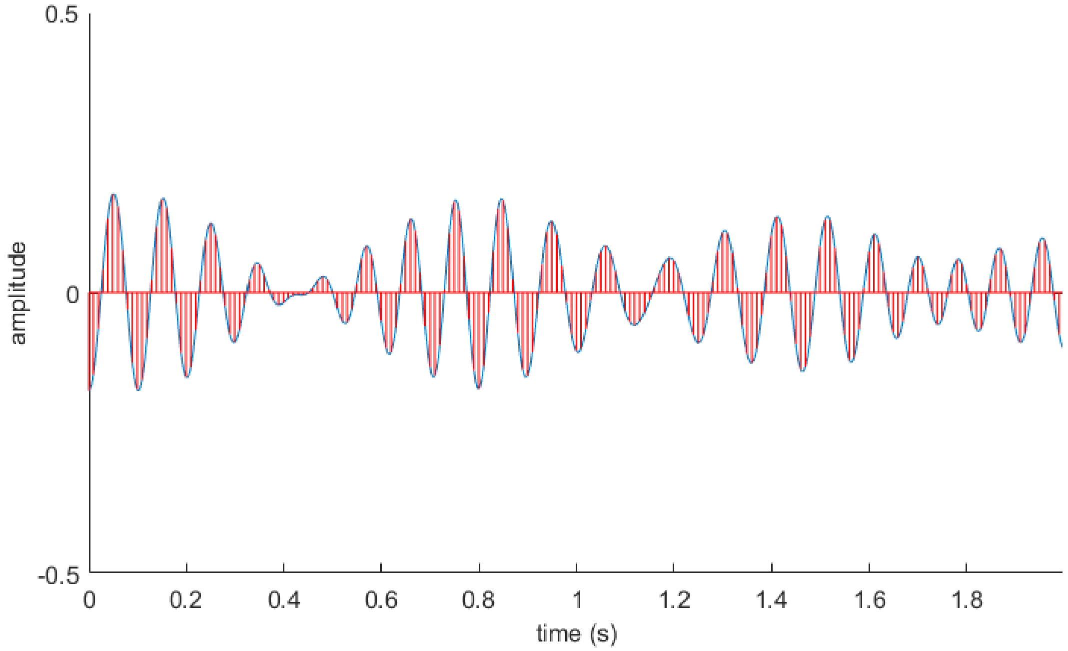


Samples

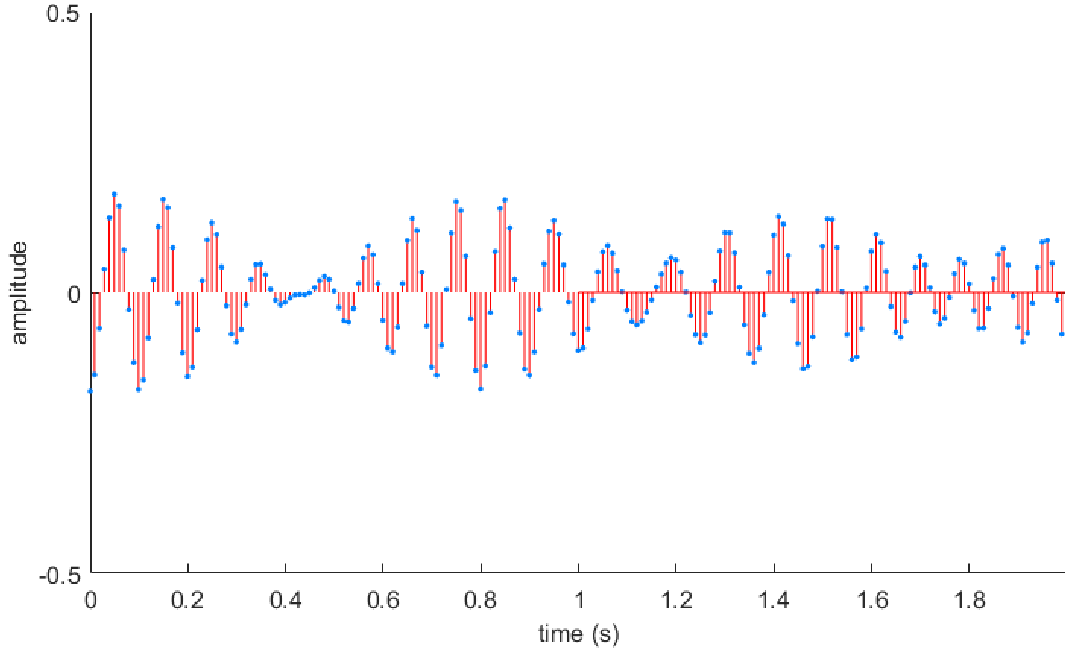


Temporal dependencies between samples – autocorrelation (2/3)

HIGH sampling rate

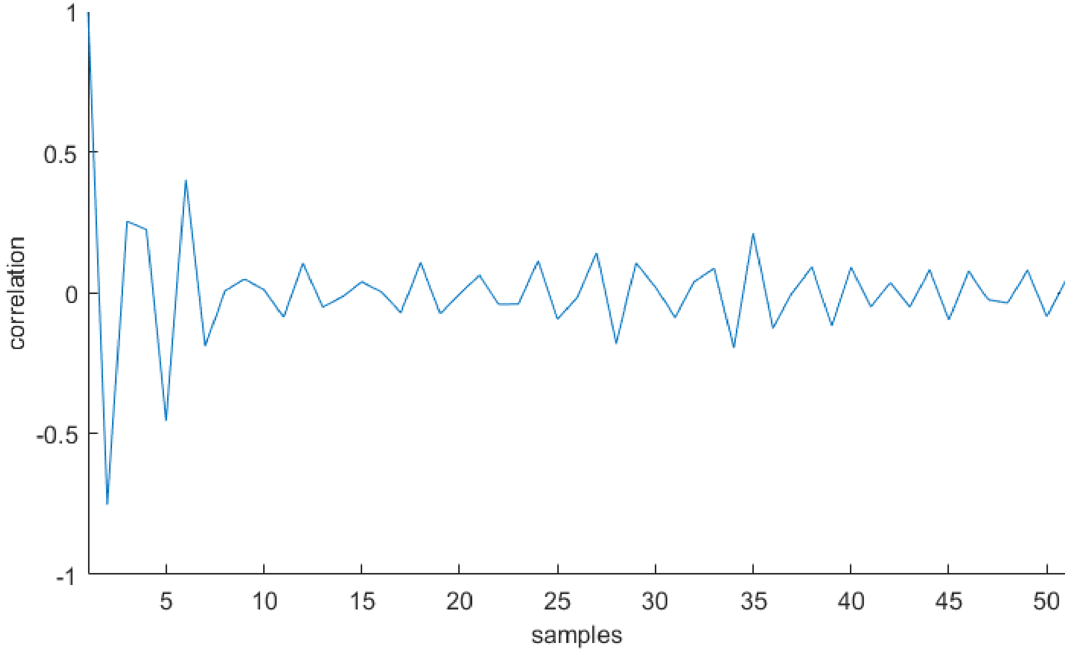


Samples

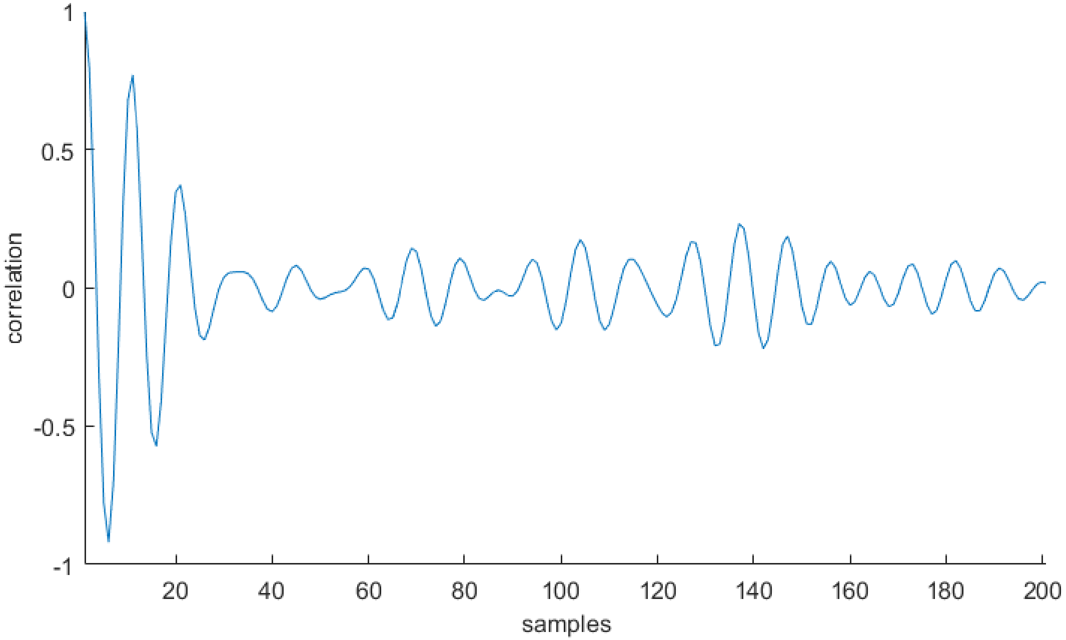


Temporal dependencies between samples – autocorrelation (3/3)

LOW sampling rate



HIGH sampling rate



Types of data

- Integer (1, 2, 4, 8 bytes)
- Float (4, 8 bytes)

Sign of data

- Signed (e.g., int8, range [-128, 127])
- Unsigned (e.g., uint8, range [0, 255])

Pitfalls

- Type cast. For instance, int8(1.49) = 1; double(2) = 2.0;
- Overflow. For instance, int8 n = 300 (?);

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	0	0	0	0	0	0	0	0
sign	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1

$$n = b_1 * 2^0 + b_2 * 2^1 + \ldots + b_7 * 2^6$$

0	0	0	0	0	1	0	1

$$n = 1 * 2^0 + 0 * 2^1 + 1 * 2^2 \ldots = 5$$

Section 3. Programming languages

What is programming language?

"A programming language is a formal constructed language designed to communicate instructions to a machine, particularly a computer."

wikipedia.org

Programming language is a limited set of instructions or "words" to describe the sequence of arithmetical/logical operations.

Example 1 (easy to compute without computer),

```
a = 2
b = 3
c = a + b
```

Example 2 (less easy to compute, more chances to make a mistake),

```
a = 2.45125
b = 3.54875
c = (a * b) / (a ^ b)
```

Python and MATLAB

In this course, we will use **Python** as a replacement for **MATLAB**. We will not learn Python in depth.

Why do we use Python?

- free compared to MATLAB
- numerous libraries for scientific computing, hardware communication, data visualization
- growing number of Python toolboxes in neuroscience
- advanced mechanisms (e.g., multithreading) compared to MATLAB

What do we need from Python?

- libraries for signal processing (**scipy**)
- data visualization (**matplotlib**)
- numerical analysis (**numpy**)
- machine learning (**scikit-learn**)

What is the difference in syntax between Python and MATLAB? (1/2)

Python

```python
import numpy as np
import matplotlib.pyplot as plt

fs = [1, 2, 4]
all_time = np.linspace(0, 2, 200)
t = all_time[:100]

for f in fs:
    y = np.sin(2 * np.pi * f * t)
    plt.plot(t, y, label='{} Hz'.format(f))

plt.legend()
plt.savefig('basics_python.pdf')
```

MATLAB®

```matlab



fs = [1 2 4];
allTime = linspace(0, 2, 200);
t = allTime(1:100);
hold('on')
for f = fs
    y = sin(2 * pi * f * t);
    plot(t, y, 'DisplayName', sprintf('%d Hz', f));
end
legend('show')
saveas(gcf, 'basics_matlab.pdf');
```

What is the difference in syntax between Python and MATLAB? (2/2)

	Python	MATLAB
1. Indexing	0	1

2. Block indent	`for n in range(0, 10):`
	`    print("%d\n" % (n))`
		`for n = 0:9`
		`    fprintf(1, "%d\n", n);`
		`end`

3. Comment	#	%
4. Arrays	`p = np.zeros(10)` `p[0] = 1`	`p = zeros(1, 10);` `p(1) = 1`

1. Functions	`def example():`
	`    # body`
		`function example()`
		`    % body`
		`end`

2. Script extension	.py	.m

Section A1. Installing Python

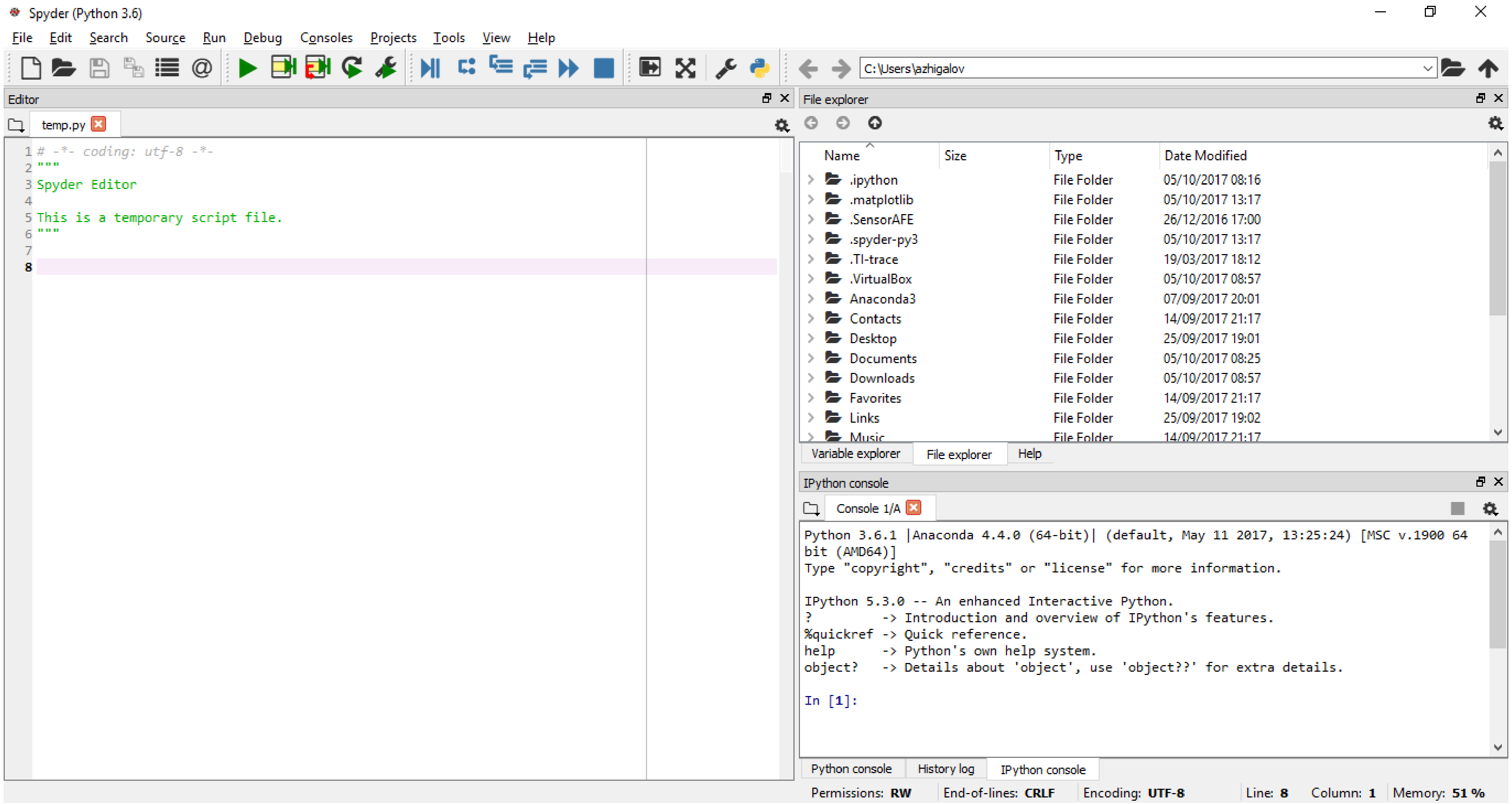
How to install Python?

- Download Anaconda package (**Python 3.6 version**) for Windows, Linux or macOS (https://www.anaconda.com/download/). It includes all necessary libraries.

- Install with default settings.

- Run **Spyder**.

How to start with Python?

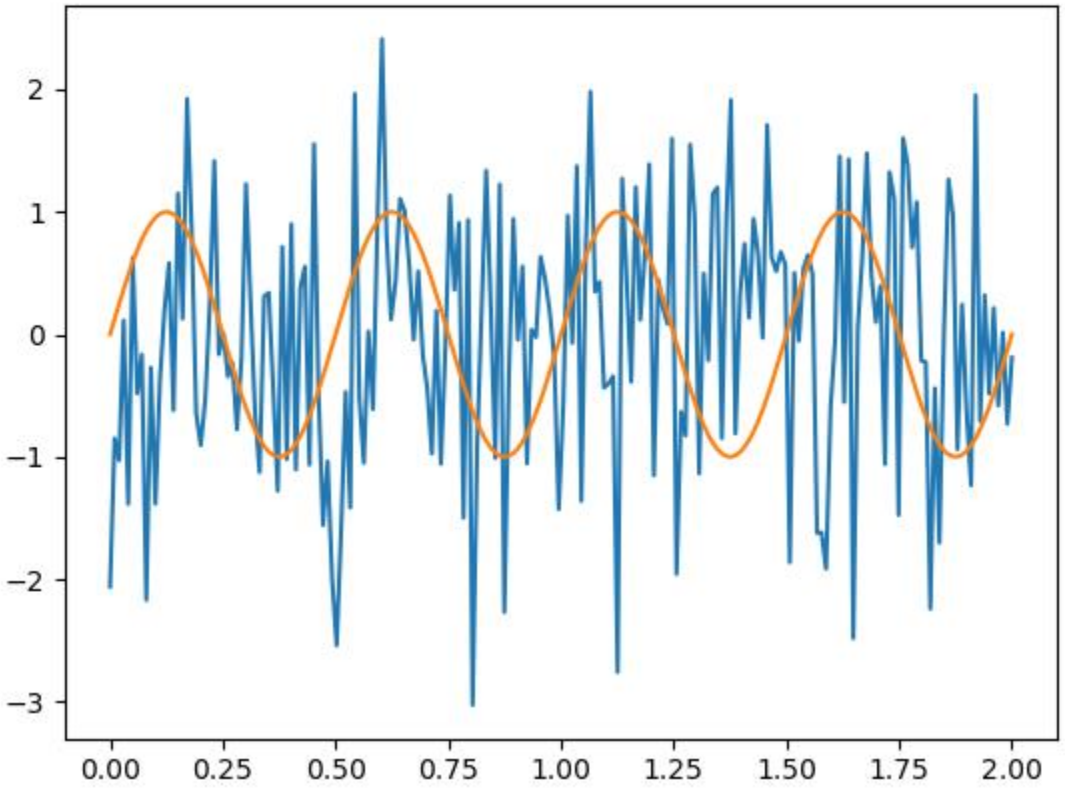


Section A2. Basic libraries and examples

Import libraries and initialize variables

```
# -*- coding: utf-8 -*-
"""
L01: basic libraries and examples
"""

import numpy as np
import matplotlib.pyplot as plt

# time variable
t = np.linspace(0, 2, 200)
N = np.shape(t)[0] # size() MATLAB

# frequency
f = 2

# signal
y = np.sin(2 * np.pi * f * t)
u = np.random.randn(N)

# plot
plt.plot(t, u)
plt.plot(t, y)
```

Print and visualize variables



Literature

- **Python/MATLAB programming language**
- http://se.mathworks.com/academia/student_center/tutorials/launchpad.html (tutorial, video, user guide)
- **MATLAB to Python migration guide**, “Enthought-MATLAB-to-Python-White-Paper.pdf“ (https://www.enthought.com/wp-content/uploads/Enthought-MATLAB-to-Python-White-Paper.pdf)
- Cohen M., “**Analyzing Neural Time Series Data: Theory and Practice**”
- Bressert E., “**SciPy and NumPy: An Overview for Developers**”

- **Electrophysiology**
- Buzsaki G., “**Rhythms of the Brain**”
- Schomer and Lopes da Silva, “**Niedermeyer's Electroencephalography: Basic Principles, Clinical Applications, and Related Fields**” (6th edition)

- **Signal processing** (advanced)
- van Drongelen W., “**Signal Processing for Neuroscientists: An Introduction to the Analysis of Physiological Signals**”