Learning objectives
Version: February 17, 2025

ISP / ADLS
Norman Juchler

zh aw Life Sciences and Facility Management
Institute of Computational Life Sciences

Part I.
Signal processing

Work in progress...

This document summarizes the learning objectives of the course. The document will be regularly updated.

Overall

- You know the difference between analog and digital signals
- You understand the role of sampling and the fundamental limitations it entails
- You know the meaning of the Fourier transform in signal theory
- You are able to create and interpret spectra
- You know how to process and visualize time series and image data in Python
- You know the effect of noise and other sources of error
- You know important filter operations and transformations

Introduction (SW01)

1. You can describe what a **signal** is, and what characteristics it may have.
2. You are able to differentiate between the following disciplines, describe them coarsely in two sentences each, and understand how they relate to signals.
   - Signal processing
   - Machine learning
   - Information theory
   - System theory
   - Control systems
   - Statistics / stochastics
3. You know at least three subdomains/specialties of signal processing (audio/image/video processing, etc.), you can name examples of applications for these subdomains.
4. You can name the main differences between **analog** and **digital** signal processing, and are able to roughly put these areas into historical perspective.

5. You can name the main characteristics of **continuous-time signals** and describe examples in which such signals occur either naturally or synthetically in our everyday life.
6. You can name the main characteristics of **discrete-time signals**, and again can name examples.
7. You understand how we can use **mathematical functions** to model continuous-time and discrete-time signals. In which situations do we prefer continuous-time models for signals? And when are we using a discrete representation of signals?
8. You understand that signals may be **multi-channel**, and can name examples for it.
9. Given an exemplary signal, you can identify its main **mathematical properties**:
    - Periodicity
    - Symmetry (odd, even)
    - Asymptotic behavior[1]
    - Structure of a function[2]
10. For common periodic functions (sinusoids, rectangular signals, etc.), you can provide a mathematical definition.
11. You know the main mathematical properties of sinusoidal signals are ($\sin(t)$, $\cos(t)$, re^{it})) and can apply those. You understand how sinusoids are parameterized.
12. You can describe what a beat (German: *Schwebung*) is, and understand, in which situations it occurs.

Sampling and interpolation (SW01)

13. You can describe and characterize the process of sampling.
    - You understand in which situations sampling plays a role.
    - You know the difference between **discretization** and **quantization** in the context of sampling.
    - You can describe the terms **oversampling** and **undersampling**.
    - You recognize **aliasing** as an adverse effect of sampling.
    - You understand that aliasing can be avoided effectively by **low-pass filtering** the signal.
    - You know how sampling rate and sampling period relate to each other.
14. You can formulate the **Nyquist-Shannon theorem** and recognize the implications it has.
15. You understand that interpolation is the reverse operation to sampling. You can name different methods and are able to apply basic interpolations in Python.

[1] Essentially: does function converge to some limit, or not?

[2] For example, you can recognize if the function results from a weighted summation of sub-functions, or if it was formed by chaining an inner function with an outer function, etc.

Fourier series (SW02 / SW03)

16. You are familiar with **complex numbers** and know how to perform basic calculations.
17. You know how a **sinc() function** is defined, what its function graph looks like, and how the function can be parameterized.
18. You understand the **superposition principle** and can name situations in which signals are naturally superimposed.
19. You know how magnitude and phase **spectra** are created and how to read them.
20. You can explain what a **Fourier series expansion** of a signal $x(t)$ is. You know how the series coefficients can be calculated in principle. And you can describe how to approximate the original signal $x(t)$ using a Fourier series.
21. You can explain how the **different notations** for Fourier series are linked to one another. You understand that they are equivalent.
22. When presented with the coefficients of a Fourier series, you know what they indicate. You know what effects **signal symmetries** have on the Fourier coefficients.
23. You can name the purpose of Fourier series and recognize the value of this type of signal decomposition.

Fourier transformation and friends (SW03 / SW04)

24. You know the definition of the **Fourier transform** and its **inverse**. You know the main purpose for using the Fourier transform (and its inverse).
25. You understand that the Fourier transform can be seen as a **generalization** of Fourier series to non-periodic signals.
26. You can name **properties** of the Fourier transform $F(\omega)$ of a time-continuous function $f(t)$. (They are similar to those of the Fourier series.)
27. You understand the **time-shift property** ($f(t - t_0) \leftrightarrow F(\omega) \cdot e^{i\omega t_0}$) can relate this to the observation that the time delay only influences the phase, but not the amplitude of the Fourier transform.
28. You have gained an overview understanding of "**Fourier's landscape**", how the different transformations came about, and that observations for one type of transform are likely to apply to the discrete/continuous, periodic/aperiodic counterpart.
29. Concretely, you know how the **Discrete Fourier Transform** (DFT), the **fast Fourier transform** (FFT) algorithm, the **Short-time Fourier Transform** (STFT) are related variants of the Fourier transform.

Convolution (SW03 / SW04)

30. You know how the **convolution operator** $h * x(t)$ is defined for two continuous or discrete functions $h(t), x(t)$ (or $h[n], x[n]$).
31. You understand that **filtering** and other signal processing operations can be modeled as a convolution of two functions.
32. You recognize that the **impulse response** $h(t)$ of a filter (or system) fully characterizes the filter's or system's effect on any input signal.
33. You recognize that the **convolution theorem** ($h * x(t) \leftrightarrow H(\omega) \cdot X(\omega)$) has great practical importance, as it allows us to efficiently filter signals in the frequency domain.
34. In particular, you can appreciate the computational efficiency (divide and conquer!) of the FFT algorithm and that, paired with the convolution theorem, marks a cornerstone of many signal processing applications.

Filtering (SW03 / SW04)

35. You can describe the most important **types of signal filters** and what they intend to achieve in the frequency domain.
36. You can describe how an **ideal low-pass filter** operates in the frequency domain, but you also understand that it is not realizable in practice.
37. You can describe how we have used **windowed filtering** to attenuate noise in the signal (low-pass filtering), and you understand that the choice of window function and window width is important for the trade-off between frequency domain characteristics and time domain characteristics (e.g., sharpness of the transition band).

Miscellaneous

38. You know the *sinc* **function**, and that it is the Fourier transform of a rectangular pulse.
39. You also know the **impulse function** $\delta(t)$ as an extreme case of a rectangular pulse, and that its Fourier transform is a constant.
40. You know that **cross-correlation** is a measure of similarity between two signals, and that it can be used to find the time delay between two signals.
41. Similarly, you know that the **autocorrelation** of a signal is a measure of the similarity of the signal with itself, and can be used to find periodicity in a signal.
42. You can describe how to model noise practically for time-discrete signals.

Coding experience

You are familiar with the following functions, expressions or packages. You can roughly describe their main purpose:

43. `np.arange()`, `np.linspace()`
44. `plt.stem()`
45. `array[::step]`, `array[start:stop:step]`
46. `scipy.interpolate.interp1d()`
47. `pd.Series.resample()`
48. `scipy.fft` and its main functions
49. `scipy.signal.convolve()`
50. `scipy.signal.window` (and a selection of three window functions)

Part II.
Image processing

Introduction and basics (SW07 / SW08)

1. You can line out the relevance of image processing, and name some **applications** in the life science domain.
2. You can name common tasks in image processing (e.g. filtering, segmentation, feature extraction, ...).
3. You can name at least five characteristics that a digital image processing system shares with the **human visual system**.
4. You can name the basic steps from information carrier to a digital image (sensing, sampling, quantization, coding).
5. You can list and explain at least 5 important **image quality** attributes. You can provide definitions or explanations for the following terms: resolution, quantization, contrast, dynamic range, exposure, noise, blur, artifacts, color fidelity.
6. You understand how the Shannon-Nyquist **sampling theorem** applies in the context of image processing, and relate it to the concepts of image resolution, sampling, aliasing, and moiré patterns.
7. You know different **noise** models (additive, multiplicative, Gaussian, ...) and recognize that noise shadows the information in an image. You know two metrics (SNR, CNR) to quantify the noise in an image or image quality.
8. You known how to display and write images in Python (e.g., using OpenCV, matplotlib, or Pillow).

Image representation (SW07 / SW08)

9. You understand how images can be **represented** in a computer, and specifically, in Python. You know the different common data types (e.g., uint8, uint16, float32) and how to convert between them.
10. You know how colors can be represented in a computer, explain why there are different **color spaces**, and know tools to convert between them. You understand the concept of a **color map**.
11. You know what **Bayer filters** are and how they are used in digital cameras.
12. You can compute the **memory footprint** for an image if you know the image dimensions, the number of channels, and the bit depth. You understand the trade-off between image quality and memory footprint.

Transformations (SW07 / SW08)

13. You know different types/classes of image transformations (e.g. linear, non-linear, point-wise, global (as opposed to local), geometric, intensity, color, ...)
14. You can describe various **point operations** and are in principle able to implement them in Python.
15. You know how to compute and interpret color or intensity **histograms**, and know how to adjust histograms by means of intensity-to-gray-value mappings (e.g., gamma correction, histogram equalization, inversion, histogram matching, etc.).
16. You understand how **geometric transformations** (such as scaling, rotation, translation, and shearing) can be implemented in Python (e.g., using OpenCV or Pillow).
17. You understand the **matrix representation** of these transformations, and can recognize different types of transformations (e.g., rotation, affine, perspective), and you know how to compute a **composite transformation** by means of matrix multiplication.
18. You can explain the difference between **interpolation methods** (nearest neighbor, bilinear, bicubic) and know how to apply them in Python (e.g., using OpenCV).

2D Fourier transform and convolution (SW08 / SW09)

19. You can expand your knowledge of Fourier transformations (FT, DFT, FFT, and their inverse operations) to 2D images, and understand how to apply it in Python (e.g., using OpenCV or NumPy). You know how to create, read and interpret the frequency spectrum of an image.
20. You know what sinusoidal **grating patterns** are and how to create them. You know how their Fourier transform looks like, and recognize the fact that by superimposing such gratings, we can create (almost) any image.
21. You can explain the concept of **convolution**, the convolution theorem, and how it is used in image processing. You can implement convolution in Python (e.g., using OpenCV or NumPy).
22. You are familiar with the concept of **filtering** and know different types of filters (e.g., low-pass, high-pass, band-pass, Gaussian, median, ...). You can implement filtering in Python (e.g., using OpenCV or NumPy).
23. You see how different **convolution kernels** can be used implement different mathematical operations efficiently (e.g., differentiation, smoothing, sharpening, ...).

Binary mask operations (SW09 / SW10)

24. You can perform basic **logical operations** on binary images (e.g., AND, OR, XOR, NOT) and know how to use them to combine or manipulate non-binary images.
25. You can enumerate and explain different types of **morphological operations** (e.g., dilation, erosion, opening, closing). You know when to use morphological operations (e.g., to remove noise, to fill holes, to separate objects, ...).
26. You know how to compute **connected components** in binary images and how to label them. You can compute the area and the centroid of connected components. Using OpenCV, you can compute the **contour** or the bounding box of connected components.
27. You know the **distance transform** for binary images and can compute it in Python (e.g., using OpenCV).

Content aware operations (SW09 / SW10)

28. You are familiar with the concept of **edge detection** and know different methods to detect edges in images (e.g., Sobel, Canny, ...). You can explain how these methods are linked to kernels and convolution.
29. You can explain what **segmentation** is and know different basic (non-AI) methods to segment images (e.g., thresholding, region growing, ...).

Coding experience

You are familiar with the following functions, expressions or packages. You can roughly describe their main purpose.

30. OpenCV, the computer vision library classic
    - `imread()` and `imwrite()` to read an write images
    - `cvtColor()` for color space conversions
    - `resize()` to resample an image
    - `warpAffine()`, `getRotationMatrix2D()`, `warpPerspective()`, and `getPerspectiveTransform()` to implement geometric transformations
    - `threshold()` and `adaptiveThreshold()` to binarize an image
    - `erode()`, `dilate()`, and `morphologyEx()`
    - `getStructuringElement()` to create special kernels
    - `connectedComponents()`, `findContours()` and `drawContours()`
    - `distanceTransform()`, `floodFill()`
    - `Sobel()`, `Canny()` for edge detection
    - `watershed()` as a binarization / segmentation tool
31. Pillow, a user-friendly for image processing
32. `plt.imshow()`
33. `scikit-learn` as resource for different image processing algorithms and functions.
34. NumPy
    - `np.ndarray()`, its slice operations, and the `astype()` method.
    - `@` operator (`A @ B`) for matrix multiplications.
    - `np.histogram()` to compute histograms.
    - `np.vstack()` to stack multiple image channels.
    - `np.meshgrid()` to create a grid of coordinates.
35. SciPy
    - `scipy.fft` and its functions to operate with 2D Fourier transforms.
    - `scipy.signal.convolve2d()` to compute 2D convolutions.
    - `scipy.ndimage.label()` to compute and label connected components.