Introduction to
Digitial Signal Processing
Edited: Thursday 30th October, 2025

fabian.brinkmann@tu-berlin.de Winter 2024/25

1. Homework

Total score: 27 points

Topics: Audio signals, system theory, convolution, discrete fourier transform, aliasing.
Material: `speech.wav`, `impulse_response.wav`.

Submission

Submit the homework by **Sunday 30. November 2025 (23:55)** via the ISIS portal. Late submissions or submissions that are not made via the ISIS portal will not be considered. Only one submission must be uploaded per group.

The homework must be submitted as a single **iPython Notebook** without any attachments. The notebook can include code, plots, text, images, and equations. Equations can typeset with LaTex or handwritten. In the latter case, they should be included as in image. Please **run the Notebook** before the submission to make sure all plots and outputs are included.

Include the **names and matriculation numbers** of all group members in the iPython notebook, comment your code, and submit the notebook as a **single file** named for example **homework_01_group_A.ipynb**.

The following Python packages might help you to solve the tasks: `numpy`, `scipy`, `matplotlib`, `sounddevice`, `timeit`, `pyfar`. pyfar can **only** be used if it is explicitly allowed.

1 Import and display audio files

Import the audio files `speech.wav` and `impulse_response.wav` and store the signals and sampling rates in variables. If necessary, reduce stereo signals to the first channel. Normalize the signals to an absolute amplitude of 1 and plot the signals using the time in seconds on the x-axis and the name of the signal for the title.

2 Points: Import and normalization (1), Plot (1).

2 System properties

Formally test the systems given by the difference equations below for linearity, time invariance, recursivity, causality, memory freedom.

$$y_1[n] = 2x[n^2 - 1]$$
$$y_2[n] = \sqrt{3}x[n] + x[n + 2]$$

5 Points: system properties (5).

3 Convolution

Given is the non-causal system defined by the following difference equation

$$y[n] = 0.2(x[n + 2] + x[n + 1] + x[n] + x[n - 1] + x[n - 2])$$

a) Make the system causal and determine the impulse response $h[n]$

b) Implement a function `convolve(a, b)` that convolves the signals a and b without using third party function such as `numpy.convolve`. Verify that the output for `a=[1, -1]` and `b=[2, 0, 0, 1]` is `a=[2, -2, 0, 1, -1]`.

c) Generate sine signals $x_1[n]$ and $x_2[n]$ with a duration of 16 samples, a sampling rate of 16 kHz, and frequencies of 1 and 4 kHz. Convolve the signals with the impulse response from a) (use `numpy.convolve` if you did not finish b), and plot the results.

d) Describe the behavior of the system. How does it affect the time signals?

e) Show the following by calculating and plotting the corresponding signals: $x_1[n] * h[n] = h[n] * x_1[n]$ and $(x_1[n] * h[n]) + (x_2[n] * h[n]) = (x_1[n] + x_2[n]) * h[n]$.

11 Points: a (1), b (3), c (2), d (3), e (2).

4 Discrete Fourier Transform Implementation

Implement a function `X, f = rdft(x, fs)` that calculates the single sided spectrum `X` (that is, the spectrum up to the point of symmetry) and the corresponding frequency vector `f` using the real valued time signal `x` and the sampling frequency `fs` in Hz. Hint: The length of the single sided spectrum is `N//2 + 1`; your function can not use third party FFT modules such as `numpy.fft` or `scipy.fft`.

Compare your implementation against `numpy.fft.rfft` and `numpy.fft.rfftfreq`:

a) Check if the results are identical using `numpy.testing.assert_allclose` (This should work with the default parameters for `rtol` and `atol`).

b) Compare the computation times with `timeit.timeit` using five repetitions (`number=5`). Hint: You have to use the parameter `globals` to pass the required variables and function to timeit, e.g., `globals={"rdft": rdft, "x": x, "fs": fs}`.

Comparisons a) and b) shall be performed using noise signals with a length of `N` samples and the six values `N=[1024, 1025, 1026, 1027, 4096, 8192]`. The computation times returned by `timeit` shall be printed.

6 Points: 3 for the DFT implementation and 3 for the comparison

5 Temporal aliasing

Temporal aliasing denotes the fact that the frequency of a pure tone can change due to sampling (discretizing the signal in time). To understand aliasing, plot the first millisecond of the two sine signals with a frequency of 15 kHz and sampling rates of 16 and 192 kHz on a common time axis in milliseconds. Use `matplotlib.pyplot.plot` to plot the signal with the higher sampling rate and `matplotlib.pyplot.stem` to plot the signal with the lower sampling rate on top. Explain why temporal aliasing occurs in this case. It is not allowed to use pyfar.

3 points: Signal generation (1), plot (1), explanation (1)