Guidelines

- Write your name and matriculation number on each sheet of paper.
- Only clearly readable exercise-elaborations are evaluated.
- Results have to be provided together with an evident way of calculation.
- Keep textual answers short and concise. Lengthy or vague statements won't gain points.
- Unless otherwise mentioned, all Python problems have to be solved using only Python standard libraries, `numpy` and `matplotlib`.

Problem 2.1 (0.5 points)

Consider systems with a transfer function of the form

$$H(e^{j\Omega}) = A(e^{j\Omega})e^{-j\alpha\Omega+j\beta}, \tag{2.1.1}$$

where $A(e^{j\Omega})$ is real valued.

2.1.1 Determine the phase delay and the group delay of such systems. What does the group delay indicate?

2.1.2 Show that systems with a transfer function as in (2.1.1) have linear phase? What is the advantage of systems with linear phase?

In literature, we distinguish between four different types of FIR filters with linear phase:

(I) $h_n = h_{M-n} \quad 0 \leq n \leq M \quad M$ even,

(II) $h_n = h_{M-n} \quad 0 \leq n \leq M \quad M$ odd,

(III) $h_n = -h_{M-n} \quad 0 \leq n \leq M \quad M$ even,

(IV) $h_n = -h_{M-n} \quad 0 \leq n \leq M \quad M$ odd.

2.1.3 For each of the four systems, sketch an exemplary impulse response (using $M = 4$ and $M = 5$, respectively).

2.1.4 Now calculate the transfer functions $H_I(e^{j\Omega})$, $H_{II}(e^{j\Omega})$, $H_{III}(e^{j\Omega})$, and $H_{IV}(e^{j\Omega})$ for arbitrary h_n for each of the four types of filter with linear phase.

2.1.5 Determine the phase delay and the group delay for each of the four filter types.

Problem 2.2 (0.5 points)

Consider a *causal* and *stable* LTI system with *real-valued* impulse response h_n. The real part of the corresponding transfer function is given as:

$$H_R(e^{j\Omega}) = \mathrm{Re}\{H(e^{j\Omega})\} = \frac{1 + a\cos(\Omega)}{1 + 2a\cos(\Omega) + a^2}, \qquad \text{with} \qquad |a| < 1 \tag{2.2.1}$$

2.2.1 Show that even part $h_n^{(e)}$ of the impulse response h_n can be expressed as:

$$h_n^{(e)} = \frac{1}{2}\delta_n + \frac{1}{2}(-a)^{|n|} \tag{2.2.2}$$

Hint: Use following properties:

$$x_{n+n_0} \leftrightarrow z^{n_0} X(z) \tag{2.2.3}$$

$$\delta_n \leftrightarrow 1 \qquad \forall z \tag{2.2.4}$$

$$\alpha^n \sigma_n \leftrightarrow \frac{z}{z-\alpha} \qquad |z| > |\alpha| \tag{2.2.5}$$

$$-\alpha^n \sigma_{-n-1} \leftrightarrow \frac{z}{z-\alpha} \qquad |z| < |\alpha| \tag{2.2.6}$$

where σ_n is the unit step function.

2.2.2 From $h_n^{(e)}$, determine the odd part $h_n^{(o)}$ of the impulse response.

2.2.3 Calculate $H(e^{j\Omega})$.

Problem 2.3 (0.5 points)

The bilinear transform is widely used for designing a digital filter $H(z)$ from an analog filter $H(s)$ and may be interpreted as a mathematical transformation from the s-domain (Laplace transform) to the z-domain (Z-transform). The bilinear transform is defined as

$$H(z) = H(s)\Big|_{s=\frac{2}{T}\frac{z-1}{z+1}}.$$

2.3.1 What does T represent in the bilinear transformation?

Let us now investigate how certain regions of the s-plane are mapped onto the z-plane.

2.3.2 Express r of the term $z = re^{j\phi}$ as a function of the real part σ and the imaginary part ω of $s = \sigma + j\omega$.

2.3.3 Using the result from Task 2.3.2, determine the corresponding mappings of general values $s = \sigma + j\omega$ from the s-plane (Laplace transform) to the z-plane (Z-transform) for

1. $\sigma > 0$,
2. $\sigma = 0$,
3. $\sigma < 0$.

Which regions of the s-plane are mapped onto which regions of the z-plane? Display your findings using an appropriate sketch of the z-plane that includes the unit circle.

2.3.4 Where do the poles of a stable analog system lie on the s-plane (Laplace transform)?

2.3.5 Where do the poles of a stable digital system lie on the z-plane (Z-transform)?

2.3.6 Using the results from Tasks 2.3.3–2.3.5, determine if the bilinear transform preserves stability.

Now that we are familiar with the bilinear transform, let's design a stable and causal digital filter based on the transfer function of an analog filter

$$H(s) = \frac{s+2}{(s+1)(s+3)}.$$

2.3.7 Employ a partial fraction expansion to get a representation of $H(s)$ that is a linear combination of two first-oder low-pass filters $H_{\mathrm{LP}_i}(s) = \frac{1}{1+\frac{1}{s_i}s}$, with $i = 1{,}2$.

2.3.8 Determine the corresponding transfer function $H(z)$ after the bilinear transform.

2.3.9 Determine the discrete-time impulse response of the stable digital filter $h[n]$.

Problem 2.4 (Python, 1.0 point)

In this exercise, you will learn how to perform image down- and up-sampling. In other words, how to resize an image. Images are nothing more than 2-D signals, and therefore, your knowledge regarding 1-D signal processing carries on to here, including what you have learned about the sampling theorem.
We adopt the following notation: $A^{h\times w}$ is a matrix representing an image of height h and width w. The element $A_{n,m}$ is the pixel colour at the n^{th} row and m^{th} column of A, respectively. We deal exclusively with gray-scale images for simplicity, and thus $A_{n,m}$ represents a shade of gray (luminance). For this Python exercise, you need to import the library `scipy.signal`.

(Downsampling)
Let us start with downsampling. Given an image $A^{h_1\times w_1}$, we would like to downscale it into a new image $B^{h_2\times w_2}$, such that $h_2 \leq h_1, w_2 \leq w_1$. For simplicity, we focus here on the case where the image is downsampled in both dimensions by an integer factor d, so that $h_2 = h_1/d, w_2 = w_1/d$. The simplest way to downscale an image is through decimation, i.e., keep only every d^{th} pixel in each row and column. This is achieved through the following

$$B_{n,m} = A_{nd,md}\,, \quad n = 0,1,\ldots,h_1/d-1\,, m = 0,1,\ldots,w_1/d-1\,. \tag{2.4.1}$$

Decimation by a factor d reduces the number of samples representing an image by a factor of d. In other words, the sampling rate is reduced by a factor of d. If the image contains high-levels of detail (high frequency components), then the resultant sampling rate might no longer satisfy the Nyquist criterion, and the image will be distorted.
To overcome this issue, we first apply a low-pass filter to the image such that the high-frequency components are reduced, and then we decimate it as in (2.4.1). A typical filter is a square 2-D Gaussian filter of size $(p+1)\times(p+1)$ with a filter matrix given by

$$G_{n,m} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(n-p/2)^2+(m-p/2)^2}{2\sigma^2}\right), \quad n,m = 0,1,\ldots,p\,, \tag{2.4.2}$$

where σ is the standard deviation of the 2-D Gaussian filter. Notice how the 2-D impulse response is shifted such that the filter is causal and symmetric.

2.4.1 Create the function `my_deci(A,d)`, which decimates the matrix (image) A by a factor d as in (2.4.1).

2.4.2 Create the function `my_gauss2D(p,sigma)`, which produces a symmetric 2D Gaussian filter as in (2.4.2).

2.4.3 From the exercise page at TUWEL, download the folder `testimages`. Load the image `'moire_pattern.png'` into a matrix A using the command `imread`. Set $d = 2, p = 6, \sigma = 1$. Using your functions, perform the following

a) First, generate the image B_1 by direct decimation of A.

b) Now, generate the image B_2 by first convolving the image A with the Gaussian filter G using the command `convolve2d(A, G, mode='same')`, and then decimating the filtered image.

c) Plot the full-size image A and the two downscaled images B_1 and B_2 using the command `imshow(A,cmap='gray',interpolation='lanczos')`. The image B_1 suffers from an artifact that you know from sampling theory. What is it called?

(Upsampling)

Now consider the operation of upsampling, in which an image $A^{h_1 \times w_1}$ is enlarged into $B^{h_2 \times w_2}$, such that $h_2 \geq h_1, w_2 \geq w_1$. Again, we assume that the upsampling is by an integer factor d. Therefore, $h_2 = h_1 d, w_2 = w_1 d$. The simplest way of upsampling is to repeat every row of A $d-1$ times, and then every column of the resultant image is repeated $d-1$ times. This is known as nearest-neighbour interpolation (NNI). The problem with NNI is that it does not exploit the spatial correlations from all directions, and therefore can result in a pixelated image. A more general approach first performs zero padding to the image, and then applies an interpolation filter. The zero padding is achieved by inserting $d-1$ zeros between the rows of A, followed by inserting $d-1$ zeros between the columns, as shown in Figure 2.4.1 below. The zero padded image is then convolved with

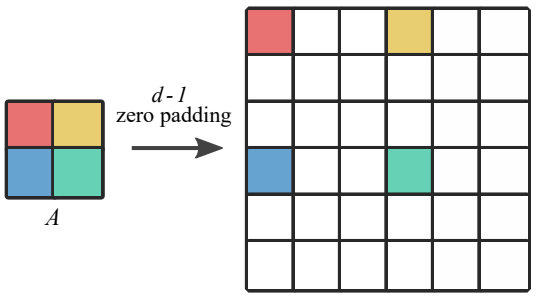

Figure 2.4.1: Zero padding of an image. $d = 3$ here.

an interpolation filter. The filter could be linear (bilinear interpolation), cubic (bicubic), etc. In this exercise, we will use our 2-D Gaussian filter that we created in the first part.

2.4.4 Create the function `my_nni(A,d)` that performs NNI upsampling.

2.4.5 Create the function `my_gauss2DUp(A,d,p,sigma)` that first performs zero-padding as in Figure 2.4.1, followed by a convolution with the 2-D Gaussian filter of (2.4.2).

2.4.6 Load the image `'cameraman.png'` into matrix A. Set $d = 3, p = 20, \sigma = 1.6$. Perform the following

a) Generate the image B_1 by NNI upsampling.

b) Generate the image B_2 by Gaussian interpolation upsampling.

c) Plot the original image A and the two enlarged images B_1 and B_2. How do the enlarged images compare in terms of smoothness?