Lab Verification Test

SCIPER number: _____

This computer-based test requires the following equipment:

- Matlab **(2021b)**

The test has the duration of three hours. It consists of two parts: Signal Processing, worth 60 points, and Systems Theory, worth 60 points as well. The maximum score you can get is 120 points; you will be awarded the maximum grade if you obtain 100 or more points; your potential bonus of points above 100 will be integrated in the overall weighted sum for the course grade.

The test is open book, i.e. all printed/written material is allowed (e.g., books, lecture notes, exercises, solutions, personal notes). In addition, you will use a computer from the computer room running Ubuntu, as you did in the labs. You are **not allowed** to use your personal computers. You can use internet to look for the necessary information, but **never to communicate with other students or anyone outside the computer room and never use on-line advanced AI tools (e.g., ChatGPT) for solving questions. Note that we will log all the communications and connections that your computer will create. Please, before starting, log out and close all mail and messaging services (Gmail, Yahoo, EPFL mail, Facebook, etc.).**

This intermediate test is available in English only and must be answered in English. Note that we do not grade on grammar as long as the meaning of the sentence is clear.

Getting Started

First start **Matlab 2021b** and apply the following settings for the keyboard:

`Preferences -> Keyboard -> Shortcuts -> Active settings -> Windows Default Set` (instead of `Emac Default Set`)

To start with this test, you will need to **download** the material available on Moodle. Download `material.zip` and extract it in your home directory. The uncompressed folder `material` has the following contents:

- *part1*
  - *convolution_fourier*
    - ***part1_convolution_fourier.m****:* Implement your code here.
    - ***plot_fourier.m***
  - *filtering*
    - ***analyze_signal.m***
    - ***filter_data.m***
    - ***signal.mat***
    - ***filtering.m***
- *part2*
  - ***part2.m****:* Implement your code here.

You will also need to **download** the empty answer sheet *SIS_23-24_test_answer_sheet.odt* and fill your answers into this file.

Submitting your answers

Answers must be written on the provided answer sheets file (*SIS_23-24_test_answer_sheet.odt*), and the code implemented in the files of `material.zip` as required. Submit these two files and rename them as:

- *SIS_23-24_test_answer_sheet_yourfirstname_yourlastname.odt*
- *material_yourfirstname_yourlastname.zip*

Both the answer sheet and zip file must be submitted via Moodle by the end of the test. Please **do not change** the folder structure and **do not copy your code into the answer sheet if not explicitly asked for.**

Make sure that all the code that you submit **compiles/runs (warnings are acceptable)**. Otherwise, there will be a **significant penalty**.

The questions have the following notation, similar to your lab exercises:

- **S**: The question can be solved using only additional simulation or simple operation with the computer.
- **Q**: The question can be answered theoretically, without any simulation or computer use.
- **I**: The problem has to be solved by implementing and/or compiling some code and running it for validation.

Please do not go beyond this page before the test start time.

This page is intentionally left blank.

Please do not go beyond this page before the test start time.

Part I: Signal Processing (60 points)

Convolution (15pts)

Before answering the following questions, you will need to go to the folder *part1/convolution_fourier*. Solve all implementation questions of this part in *part1_convolution_fourier.m*.

Consider the following two signals:

$$x[n] = sin(0.1\pi * n) + \cos(0.3\pi * n + 0.5) \qquad h[n] = e^{-\frac{n^2}{3}}$$

1. **(I)** (8pts): Plot both signals in Matlab between $n = [-50{:}\,50]$, without interpolation. Insert your plots in the answer sheets.

2. **(I)** (7pts): Using Matlab, compute the discrete convolution of $h[n]$ and $x[n]$, i.e., $y[n] = (h * x)[n]$. Report your results ($y[n]$) by plotting the result for $n = [-50; 50]$ and insert the plot in the answer sheet.

Fourier transform and signal reconstruction (21pts)

Before answering the following questions, you will need to go to the folder *part1/convolution_fourier*. Solve all implementation questions in *part1_convolution_fourier.m*.

Consider the signal x expressed in continuous time domain:

$$x(t) = \sin(\pi * t) + \cos(3\pi * t + 0.5)$$

3. **(Q)** (2pts): From the expression for `x(t)`, how many main frequencies does the signal have? Write down these main frequencies `[f1, f2, …, fn]`.

4. **(Q)** (3pts): The signal `x[n]` used in Questions 1 and 2 corresponds to a discretized version of x(t). What sampling frequency `Fs` has been used? **Justify your answer.**

5. **(I)** (4pts): Compute the FFT of both `x[n]` and `h[n]` using the `fft()` function, using $n = [-50; 50]$. Plot the amplitudes of the computed FFT (double-sided amplitude spectrum) using the provided function `plot_fourier(f[w], Fs)`. Insert your **2 plots** in the answer sheet.

6. **(Q)** (4pts): For each main frequency `[f1, f2, …, fn]` of `x(t)` that you identified in Question 3, what is the corresponding phase of the Fourier transform `[Φ1, Φ2, …, Φn]`. **Justify your answer.**

7. **(I)** (8pts): Compute the inverse FFT of `z[ω] = x[ω] • h[ω]` (element-wise multiplication) using the `ifft()` function. Insert the plot in the answer sheet and comment on it (compare it to the one obtained in Question 2).

Filtering (24pts)

In Lab 4, you learned about filters and how to design them using the `filterDesigner`. In the following questions, you will have to load and analyze a signal, and filter it using `filterDesigner`. In folder the *part1/filtering* you will find the following files:

- *signal.mat* – Matlab file containing a variable *signal* corresponding to the signal, which is sampled at 8 kHz.
- *analyze_signal.m* – Matlab function similar to *analyzeSoundSignal.m* (Lab 4), that plots the signal and its Fourier transform. *Note: here, the frequency domain plots only show half of the frequency spectrum (first quadrant), and the frequencies are properly scaled.*
- *filter_data.m* – Matlab function to filter the data using the filter you will design.
- *Filtering.m* – Matlab file for you to implement anything needed.

8. (Q) (6pts): You recorded a seismic activity as an audio signal, but unfortunately, during your recording, your neighbor's washing machine has been running, which ruined your recording due to the vibrations. Luckily, the vibrations caused were limited to two frequencies only, which can be filtered out.
Go to folder *part1/filtering,* load *signal.mat* and the variable `signal` will contain the ruined audio signal and analyze the signal using the function *analyze_signal.m*. You should clearly see the two frequencies caused by the washing machine. What type of filter would you implement (low pass, high pass, band pass or band stop) to remove them? **Justify your answer.**

9. (I) (12pts): Use `filterDesigner` to implement the filter of Question 8. Set the following parameters:

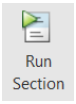
- *Response Type:* according to your answer in Question 8
- *Design Method:* FIR – Equiripple.
- *Filter Order:* Minimum order
- *Options:* Density Factor: 20
- *Magnitude Specifications:* leave it as default.

The part left for your design is then defining the *Frequency Specifications:* `Fs`, `Fpass1`, `Fstop1` `Fstop2`, and `Fpass2`. Please report the values you choose in the answer sheet. **Justify your answer.**

10. (I) (6pts): Now you can filter the signal using the filter coefficients obtained in Question 9 in `filterDesigner` you can export the coefficients of the filter which you just designed by clicking on `File → Export`. When you click the `Export` button, the coefficients will be stored in your workspace in the variable `Num`. Use *filter_data.m* to filter the signal and generate the time domain and frequency domain plots of the filtered signal. Paste the plot in your answer sheet.
Note: You don't need to submit code for this question.

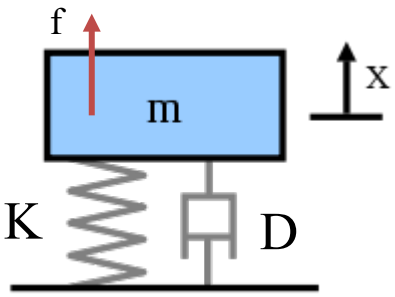
Part II: System Theory (60 points)

Solve all implementation questions in the file *part2.m*. This script contains sections, each associated to one of the questions of this assignment. You will need to run each section individually as you progress through this part. To do this, once you placed the cursor within a section, press `Ctrl+Enter` to run only the current section. Alternatively, you can use the button when the cursor is in the desired section:



In this part, we will consider the mass-spring-damper system illustrated in the figure below. The output of the system is the position $x(t)$ of a mass m, to which a force $f(t)$ can be applied. The spring has a constant K and the damper a constant D. We do not consider gravity. The dynamic equation of the system is therefore given by:

$$m\ddot{x} = -Kx - D\dot{x} + f \tag{1}$$



11. **(Q)** (2pts) Write the Laplace transform of the dynamic equation (1) provided above.
Hint: You can use the tables for the Laplace transform.

12. **(Q)** (2pts) Using the Laplace transform from the previous question, rewrite it as a transfer function $H(s) = \frac{X(s)}{F(s)}$, where $X(s) = \mathcal{L}(x(t))$ is the **output** and $F(s) = \mathcal{L}(f(t))$ is the **input** of the system.

Run the **`Setup`** section of *part2.m* to generate the required variables to start solving this part. This will create three symbolic variables `t`, `s`, and `z`, as well as define the numerical constants of the mass-spring-damper system m, K and D.

13. **(I)** (3pts) Implement the transfer function in the Laplace domain **symbolically** in Matlab, using the relevant symbolic and numerical variables mentioned above. Store the function in the variable `H`. Report the symbolic plot showing the transfer function.

14. **(I)** (8pts) We want to study the impulse and step response of the system using **symbolic** computations. To this end, we will define a symbolic impulse (use `dirac(t)`) and step (use `heaviside(t)`) function as input and observe the system's response for each case. Apply these inputs to the symbolic transfer function in the Laplace domain obtained in the previous question. Store the resulting system's response **in time domain** for each input function in the symbolic variables `yi` (impulse case) and `ys` (step case) respectively and report the obtained plot.
Hint: You may need to use the functions `laplace` and `ilaplace`.

15. (Q) (2pts) Did you need to use the `dirac(t)` function in the previous question to compute the impulse response? **Justify your answer**

16. (S) (2pts) We will now leverage Matlab's control toolbox and use the `tf()` function to express the transfer function in the Laplace domain. In this section, we store the output of `tf()` into the `sys` variable for you. How many arguments did we give to `tf` and what do they represent?

17. (I) (6pts) Implement the impulse and step responses using the system toolbox functions `impulse()` and `step()`. Both these functions automatically generate a plot if their output is ignored. Instead, we want to store their output in the respective pairs `[yImp,tImp]` (for the impulse response) and `[yStp,tStp]` (for the step response). Compute the outputs for a duration of 40 seconds. Do you notice any difference with the results from the symbolic computations you performed earlier in Question 14? Report the plot in your answer sheet.
Hint: Check the functions' documentation if you are unsure what these outputs represent.

18. (S) (11pts) Run the corresponding section to generate the system's Bode plot. Identify three different regimes of the system, indicate their rough frequency ranges (in rad/s), and briefly explain what is happening in each of them (amplitude, phase, underlying physical intuition). How can you characterize this system in terms of filter type (e.g., low-pass, high-pass, band-pass, …)? **Justify your answer**.

19. (I) (4pts) You will now observe these three regimes by providing a sine wave as input to the system and observe the output. Select a representative frequency `wi` for `i` $\in$ [1,2,3] in rad/s for each regime and a suitable title for each plot. The signals and plots generation are implemented for you. Do these plots match the observations you made on the Bode plot in the previous question? Report the plot in your answer sheet.

So far, you were using the continuous representation of the system. Even though Matlab can handle continuous computations, it only emulates them and performs discrete computations in the background. We will now discretize the system and compare the results with the continuous case. The sampling period is defined by the variable `Ts = 0.2` s and will be kept the same for the rest of this part.

20. (I) (3pts) Use Matlab's `c2d()` function to discretize the system defined earlier with `tf()` in Question 16. In other words, it converts the transfer function expressed in the Laplace domain into the Z-domain. Store the resulting discrete system in the `sysd` variable. Report the generated plot. Is the discrete time response as accurate as the continuous one?
Hint: The `c2d` function takes the following arguments:
- *continuous system (`sys` variable in our case)*
- *sampling period (`Ts` variable in our case)*

Matlab's `c2d` function performs several optimizations and makes the discretization work as a black box. We will look more in detail into the discretization process. A common approach to discretize a system is to apply the backward Euler method to the differential equation. The zeroth, first and second derivatives are computed as follows:

$$x[n] = x[nT_s] \quad \text{(position)}$$
$$dx[n] = \frac{1}{T_s}(x[n] - x[n-1]) \quad \text{(velocity)}$$
$$ddx[n] = \frac{1}{T_s^2}(x[n] - 2x[n-1] + x[n-2]) \quad \text{(acceleration)}$$

Pay attention to the **squared** sampling period T_s in the acceleration definition!

21. (I) (8pts) We will discretize symbolically the differential equation in (1) using the backward Euler method described above. For this, complete the function `discretize_system(T,m,K,D)`, which you will find towards the end of the script. The idea is to rewrite the differential equation by replacing the time derivatives with their discrete definitions given by the Backward Euler method, hence:

$$m\ddot{x} = -Kx - D\dot{x} + f \quad \Rightarrow \quad eqn = m\, ddx[n] + K\, x[n] + D\, dx[n] - f[n] = 0$$

Define a symbolic expression for each discrete derivative and then store the discrete differential equation in the form above into the variable `eqn`. Report the discrete equation you obtain (displayed in the command line window when executing the section).

Hint: In the function you must complete, the symbolic variable `x` is a function of the symbolic variable `n`. For instance, if you want to declare a function $q[n] = x[n-2] + x[n-1] + x[n]$, you can declare it in Matlab as: `q = x(n-2) + x(n-1) + x(n)`. We provide you with the zeroth derivative (position) as an example, and leave the velocity and acceleration for you to complete.

We used the backward Euler method in the previous question to discretize the system, but one can also use the equivalent Forward Euler method. By applying the forward method to our system, we obtain the following difference equation:

$$37.5x[n+2] - 72.5x[n+1] + 36.0x[n] - f[n] = 0 \quad (2)$$

If you managed to solve the previous question, you will see that both equations are quite similar but present some key differences.

22. (Q) (4pts) Apply the Z-transform by hand to the discrete difference equation (2) (obtained with the **Forward** Euler method, hence **not** the one you computed) and report its Z-transform. Then compute by hand the transfer function $H(z) = \frac{X(z)}{F(z)}$. Report it in the answer sheet as well.

23. (I) (5pts) Implement the discrete transfer function using `tf()`. How does the Forward Euler method results compare with the ones obtained with the `c2d` function (see generated plot)? Is the Forward Euler method accurate? What could you do to improve the results? Report the plot in your answer sheet.