Group assignment, Signals and Systems, 2024

Solving the assignment:
Students are required to solve the assignment in groups of maximal three students and *all members of the group must be able to answer all questions. Be prepared to show individual, complete solutions to answer theoretical questions.*

We recommend the use of Matlab for solving this assignment. The text below has been written assuming that Matlab is used. You should have some background in the use of Matlab from the course Beräkningsteknik 1 and this assignment will give you a good working knowledge of Matlab.[1] You are allowed to use Python instead of Matlab to solve the assignment. However, note that you in that case will have to find and include libraries that contain the necessary mathematics and signal processing functions, in particular functions for the design of Butterworth and Chebychev filters used in part 2 of the assignment. (According to last years students who did this, all necessary functions can be found.)

It is required that students solve all questions on their own, that is, without the help of other persons apart from the teachers in the course. To clarify, please observe the following rules, which must be strictly followed.

- You may work in groups of a maximum of three members.
- All group members should participate in all the assignments and subassignments.
- You may discuss your solutions and findings orally with others.
- You may not copy answers from anybody, including students in your class or students from previous years etc. This includes any derivations and any bits of Matlab code or other files.
- You may not share or copy your Matlab code with other students outside your group.
- You may not copy any external Matlab code.

Assessment:
The solution to this group assignment will be assessed by a demonstration (using a laptop) in front of one of the involved teachers at the end of the course.

You are required to be familiar with Matlab and your own code such that small manipulations of the code can be shown during the demonstration. This includes for instance the usual mathematical functions, matrix and vector calculations, plotting results with axis labels etc. You will probably see that some of the steps necessary to solve the assignment, were already necessary for the project in Transform methods, and you will find the assignment easier after having solved some suitable Matlab exercises from the course script.

[1] The student version of Matlab that you have access to includes the Signal Processing Toolbox and the Control Toolbox, so it includes all functions that will be used here. If you use some other version of Matlab, you might have to include these two toolboxes.

The teacher might ask follow-up questions, which require you to do small modifications in your code. *Hence, prepare your code in such a way that you can work with it in the most efficient and flexible way.*

The teacher examines your solutions, and more importantly, your understanding of the solutions and your ability to explain them in a clear and understandable way. Hence, you can fail the assignment if you cannot explain your solutions in a satisfactory way or have important misunderstandings despite having completely correct solutions! It is also possible to pass the assignment with not optimal solutions, if you show that you have a good understanding of the matter. *It is required that all questions of the assignment are solved. For instance, if it is required to plot something, the plot must be prepared when coming to the demonstration.*

Preparing the demonstration:
Students may sign up for groups and book dates and times for their demonstration on Studium. It is the students' responsibility to sign up for a group and book a time for demonstration in good time! All demonstrations will be scheduled at the end of the period, when the course is given. Later demonstration dates cannot be guaranteed.

From experience, we know that the majority of the students have access to private, individual laptops with Matlab. Hence, as long as nothing else was specifically arranged, we assume that the students will use their own laptop for the demonstration. If, in your group, no students has access to a suitable laptop, you are required to send all your Matlab files to the teacher the day before the demonstration, so that the teacher can prepare the files on his/her own laptop. Alternatively, the students can choose to demonstrate their answers in one of the computer labs, in which case the students must inform the teacher at least one week in advance such that a suitable room can be booked.

ALL student groups are required to upload their Matlab/Python code as a word document or searchable pdf file on their groups file-area on Studium on the day before their presentation.

Rest / Supplementation:
After your presentation, you may be asked to supplement one or more questions in the assignments. It is the students' responsibility to schedule a suitable date and time for the supplementation with one of the teachers in the course. Unless otherwise explicitly agreed with one of the teachers, supplementation must be completed at most 10 working days after the first presentation. Students have the right to supplement their solution *once*. Additional supplementations cannot be guaranteed!

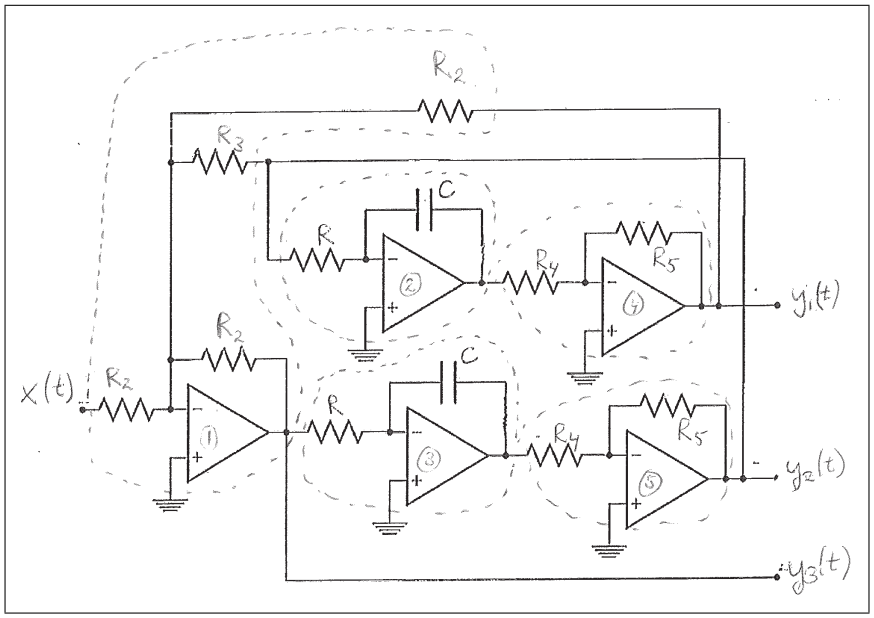
Students failing to book a time for their presentation, failing to show up for their presentation, failing to scheduling a time for the supplementation in time or not showing up for their supplementation, might fail the assignment.

1 Analysis of analog filters

In this exercise, the following Matlab functions will be helpful.

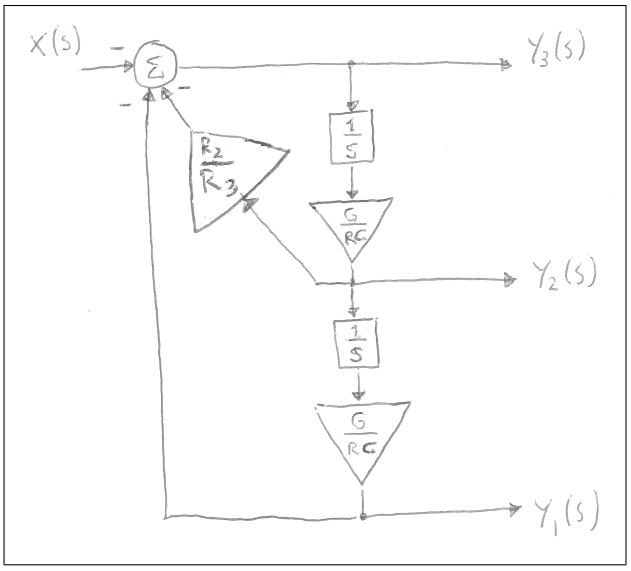
- `tf` creates a so called system object, which is needed to simulate an LTIC system.
- `lsim` simulates LTIC systems.
- `iopzplot` generates a pole-zero plot.
- `repmat` can be used to create a vector to represent a periodic function.
- `impulse` simulates the impulse response of an LTIC systems. See also the comments on the limitations of this function in the appendix.
- `bode` draws a Bode diagram.

Figure 1 below shows an active filter built of operational amplifiers, resistors and capacitors. The filter structure is interesting in the sense that it allows to use three different filter types depending on which output, $y_1(t)$, $y_2(t)$, or $y_3(t)$, is chosen.



Figur 1: Second order filter with three different transfer functions depending on the choice of output signal, $y_1(t)$, $y_2(t)$, or $y_3(t)$. The encircled components around operational amplifier (OP) 1 implement an (inverting) summer, the encircled components around OP 2 and 3 implement (inverting) integrators and the encircled components around OP4 and 5 implement (inverting) amplifiers. Note that the same parameter values, R and C, are used in both integrators and that the amplifiers (OP 4 and 5) yield the same amplification factor $-G_4 = -G_5 = -G = -R_5/R_4$. (We define G to be positive number in order to allow somewhat simpler calculations later.) Also note that the resistance R_2 is used several times in the circuit.

1. Show that the circuit in Figure 1 realises the flow diagram in Figure 2. For this, translate the electrical components into the corresponding system components. For instance, an integrator can be translated into a subsystem with transfer function $\frac{1}{RCs}$. Then, simplify the diagram successively.[2]



Figur 2: Flow diagram corresponding to the filter in Figure 1.

2. From the flow diagram in Figure 2, derive the transfer functions $H_1(s)$, $H_2(s)$ and $H_3(s)$, which describe the relationship between the input signal $X(s)$ and the three analog output signals $Y_1(s)$, $Y_2(s)$ and $Y_3(s)$, respectively. Write the transfer functions as functions of the parameter G, R, C, and the ratio R_2/R_3. Hint: Start with $H_3(s)$. Then, the two other transfer functions can be derived easily.

3. Write a Matlab script that allows you to

   (a) draw the pole-zero plots of all three transfer functions $H_1(s)$, $H_2(s)$ and $H_3(s)$ using `iopzplot`,

   (b) draw the impulse response of all three transfer functions $H_1(s)$, $H_2(s)$ and $H_3(s)$ using `impulse`[3],

   (c) draw the Bode diagrams of all three transfer functions $H_1(s)$, $H_2(s)$ and $H_3(s)$ using `bode`, and

   (d) change all parameter values in a simple and quick fashion.

4. The parameters G, R, C, and the ratio R_3/R_2 can be used to change key frequencies (for instance the cut-off frequency of an LP filter), whether the poles will be real or complex

[2] Note that in the lecture / video material on realisations of analog filters, an example was solved in the opposite direction, i.e., deriving the electrical realisation from the flow diagram.

[3] Considering the limitations of the `impulse` function.

conjugates, etc. Identify for yourselves and for the teacher which parameters influence what and how. [4] Use the Matlab script from above and/or analytical methods to examine the connections between the pole-zero plot, impulse response and frequency response of the three transfer functions. For this, vary the parameter values and examine its effect on the different system descriptions. Make sure that you understand and can explain those connections and parameter dependencies. For instance, can you see which of the frequency responses in the exercise above corresponds to which pole-zero plot and which impulse response?

5. Which filter types (e.g., LP, HP, BS, BP) are realised by H_1, H_2, or H_3, respectively?

6. Confirm your findings from Question 4 by doing the following:

   (a) Select one of H1 and H3 and derive an expression for the pole locations. What choice of parameters gives complex poles? What happens to the frequency response in the Bode plot if the imaginary part of the poles is made larger than the real part?

   (b) Set the parameters (and select the correct transfer function of H_1, H_2 or H_3) so that you receive an LP filter with a gain of about 10 dB at $f = 300Hz$ (called a resonance peak). The gain at lower frequencies should be 0 dB and at higher frequencies, the gain should decrease with increasing frequency. More precisely, the gain of the frequency response should monotonically decrease for frequencies above $300Hz$. (Note that the frequency axis in Matlab's Bode plot is scaled in rad/s.)

7. In Matlab, create a periodic signal for which you know the Fourier series expansion. You decide which! (Hint: The Mathematics Handbook lists a number of examples of Fourier expansions for a number of periodic functions that can be used. You may not use a pure sine wave instead the function must have a Fourier series expansion consisting of a main frequency components and multiple harmonics). Choose the fundamental frequency for the signal to be 50 Hz. Use the Matlab functions `tf` and `lsim` to simulate the output of the system with the periodic function as the input signal. Use the LP filter in the above task. In the simulation, you need to specify a vector of time for which the output is calculated. Make sure that these times are sufficiently close in time so that you do not get unnecessary problems due to aliasing. Show the input and the output signal together in a figure to clearly illustrate the system's impact. Explain in as much detail as you can, based on your selection of the input signal and its frequency content, why the output looks like it does.

2 Sampling

You have more freedom of choice on how to solve this exercise. You can use the Matlab functions as you want, as long as you can explain what is going on "under the hood".

The task is to design a measurement system to sample a continuous-time signal where the interesting information is represented by sine waves whose frequency and amplitude contain the desired information. We know the following about our measurement signal:

[4] In this task and task 6) below, it may be of help to express the transfer functions with the denominator on the standard form given on p.60 in the course compendium.

- The interesting sinusoidal signal components have frequencies up to 8 kHz and amplitude up to 1 V.
- In addition to the interesting signals, there is high frequency interference, also in the form of sine waves, which we assume to occur only one at a time. They occur from 11 kHz upwards and also their amplitudes are at most 1 V.
- It is known that there are no frequency components in the band between 8 kHz and 11 kHz.

Your task is to design a system that consists of an anti-aliasing filter that meets a specification, which you should set up based on the information given above, followed by a 12-bit AD converter that operates at a sampling frequency, which is also selected based on information above. The AD converter is capable of sampling at a frequency of 24 kHz, but not faster. To minimise the cost, we want the anti-aliasing filter to be of the lowest order possible, given the constraints above. The choice of other design parameters, however, is not critical.

The measuring system is considered to fulfil its purpose if we can measure the amplitude of the interesting sinusoidal signals in the band $0 - 8kHz$ using the sampled signal. At the same time, it is required that the interference will not influence the measurement in this band noticeably. A good guideline for "noticeably" here is that the amplitude of a disturbance does not exceed half a quantisation level, which in our case corresponds to $0.5/2^{11}V$. We accept a maximum ripple of 3 dB for sine waves located in the interesting range of $0-8$ kHz as we expect to be able to compensate for this distortion retrospectively using digital filters.

Take *all* the information given above into account, when solving the following tasks: (Note that the choices in (1) - (3) below are connected. You may need to reconsider your choices at different stages to find the best solution, ie, *the solution that provides a filter of minimum order.*)

1. Choose a suitable sampling frequency for your system.
2. Set up a specification for an anti-aliasing filter based on the reasoning above.
3. Select any of the standard filters (Butterworth, Chebychev I, etc) and design such a filter so that the specification is met. Justify your selection of the standard filter. You use the Matlab's built-in routines `butter`, `cheby1`, etc. as in e.g. Excercise 7.2 in Chapter 7. Read the help texts carefully so that you call them correctly. These help texts also give tips about other support functions that you may find useful. In particular, please note that the functions `butter`, `cheby1` etc. need an extra argument 's' to design an analog instead of digital filter.
4. Write a script that contains both the design of the anti-aliasing filters as well as the simulation of the entire sampling process. Simulate the continuous-time signal as a superposition of two sine waves of the same amplitude, with a frequency in the range of $0-8kHz$ and a frequency of 11 kHz and above. The analog filter is simulated using `lsim` as in the previous task and sampling is realised by simply reading the simulated output signal periodically at the times given by the sampling frequency. Select the time resolution of the time vector needed for the simulation of the analog signal "sufficiently high", i.e., as an integer multiple of your chosen sample rate. (Otherwise it will be difficult to read it periodically.)

5. Use the Discrete Fourier Transform. by using `fft`, to plot the amplitude spectrum of the discrete-time signal. In particular, ensure that the frequency axis is accurate and given in kHz. Additionally, make sure that the amplitude is normalized so that a sinus signal with amplitude of $1V$ corresponds to a peak in the spectrum with height 1, e.g. by investigating how the spectrum amplitude depends on the length of the signal.

NOTE 1: Some Matlab hints:

```
» % Create a sine signal of length 1 second, sample rate fs and frequency f:
» t = 0:1/fs:1; % time vector
» x = sin(2*pi*f*t); % notice that the input to sin() is in radians
»
» % Pick out every 10th sample of x:
» xs = x(1:10:end);
```

NOTE 2: Regarding the use of the `fft` in (5), assume that the vector x is N samples long. Then, calling `X = fft (x)` gives vector of the same length, where `X (1)` corresponds to the frequency 0 and `X (2)` corresponding to the frequency $\frac{fs}{N}$ where fs is the sampling frequency and so on. Hence, the second half of the vector `X` corresponds to frequencies above $\frac{fs}{2}$. Note that, due to the periodicity of the spectrum of a discrete-time signal, this is the same as for the negative frequencies from $-\frac{fs}{2}$ to 0. Some tips on how to plot the spectrum:

```
» N=length(x); % Number of samples in x.
» X=fft(x);
» fvector=(0:N-1)/N*fs; % fs is the sampling frequency.
» plot(fvector,abs(X)) % Plot the abslut value.
» xlim([0 fs/2]) % Limits the visiable part of the x axis in the
» % plot to the interval from 0 to fs/2.
```

3 Appendix: Limitations of the Matlab function `impulse`

The Matlab function `impulse` can be used to plot the impulse response of a LTIC system. Note that if the system has HP filter characteristics, parts of the impulse response will be missing. A rational system of HP character has a transfer function with polynomials of the same order in the numerator and in the denominator, ie it can be written as

$$H(s) = \frac{b_0 s^N + b_1 s^{N-1} + ... + b_N}{s^N + a_1 s^{N-1} + ... + a_N} \tag{1}$$

Such a rational function can always be rewritten as

$$H(s) = b_0 + \frac{c_0 s^{N-1} + c_1 s^{N-2} + ... + c_{N-1}}{s^N + a_1 s^{N-1} + ... + a_N}, \tag{2}$$

where the polynomial with coefficients $c_0, ..., c_{N-1}$ has a lower degree than the denominator polynomial $A(s)$.

The impulse response, $h(t)$, of the system $H(s)$ can be obtained by the inverse transformation of $H(s)$. From the above equation we see that the inverse transformation of $H(s)$ can be written as the sum of the inverse transformation of b_0, which is $b_0\delta(t)$, and the inverse transformation of $\frac{c_0 s^{N-1}+b_1 s^{N-2}+...+c_{N-1}}{s^N+a_1 s^{N-1}+...+a_N}$. The latter expression is so called "strictly proper" and can be treated like a well behaved, standard inverse transform.

The plot obtained from the Matlab function `impulse` is misleading for systems of the type in Eq. (1) because it does not show the contribution of Dirac function (which is perhaps not so surprising since it is difficult to illustrate together with well behaved functions.)