Linear Control Systems
25411

Instructor. Behzad Ahi **Assignment 1**

Fall 1402 **Due Date: 1402/8/1**

1 Inverse Laplace Transform

Find the inverse Laplace transform of the following functions

1. $T_1(s) = \dfrac{s+6}{s(s^2+4s+3)}$
2. $T_2(s) = \dfrac{5}{s(s^2+4s+5)}$
3. $T_3(s) = \dfrac{s^2+2s+3}{s^3+6s^2+12s+8}$

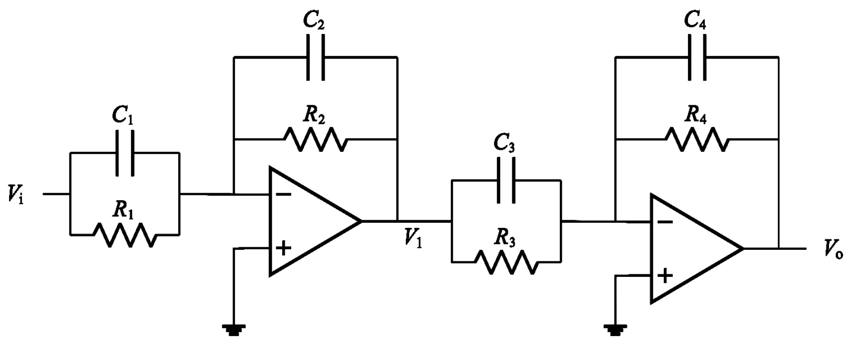
2 Derivative of Laplace Transform

Find $f(0^+)$, $f'(0^+)$ and $f''(0^+)$ for the function whose Laplace transform is given below

$$F(s) = \frac{2s+1}{s(s+1)(s+2)}$$

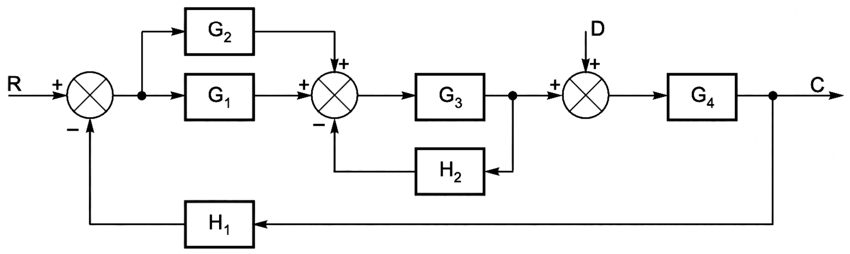
3 Transfer Function

Find the transfer function $G(s) = \dfrac{v_o(s)}{v_i(s)}$ in the following circuit



4 Block Diagram Reduction

Determine the $C(s)$ in the following block diagram. (Hint: Don't forget the disturbance $D(s)$.)



5 Signal Flow Graph

Represent the following set of equations by a signal flow graph and determine the overall gain $\frac{x_5}{x_1}$ using Mason's gain formula

$$\begin{aligned}
x_2 &= a_1x_1 + a_2x_2 + a_3x_4 \\
x_3 &= a_4x_2 + a_5x_4 + a_6x_5 \\
x_4 &= a_7x_3 + a_8x_5 \\
x_5 &= a_9x_4 + a_{10}x_2
\end{aligned}$$

6 Response of a Linear System

Given the transfer function $T(s)$ and the input signal $u(t)$, determine the output signal $y(t)$ for the time interval $0 \leq t \leq 10$.

$$T(s) = \frac{1}{(0.2s+1)(0.1s+1)}$$

$$u(t) = \begin{cases} 1 & 2 \leq t \leq 4 \\ -1 & 6 \leq t \leq 8 \\ 0 & O.W. \end{cases}$$

$$y(t) = ?$$

MATLAB Assignments

Note: you just need to do **Assignments**. Read extra explanations if you're having trouble figuring out the exercise.

7 Continues-Time Signals

Continuous-time signals are signals whose values are defined for every instant of time, such as analog signals. Discrete-time signals are signals whose values are defined only at discrete instants of time, such as digital signals. MATLAB is a discrete-time system, which means that it can only represent and manipulate discrete-time signals. Therefore, to represent continuous-time signals in MATLAB, you need to sample them at regular intervals and store the samples in a vector. Sampling is the process of obtaining a set of samples from a continuous-time signal by measuring its values at equally spaced instants of time. The sampling time or sampling period T_s is the duration between two consecutive samples. The sampling frequency or sampling rate f_s is the number of samples per unit time, and it is the reciprocal of the sampling time, i.e., $f_s = 1/T_s$.

To demonstrate how to represent continuous-time signals in MATLAB, let us consider an example of a sinusoidal signal $x(t) = \cos(2\pi f t)$, where f is the frequency of the signal in hertz. To sample this signal in MATLAB, we need to select a suitable sampling frequency f_s (or a suitable sampling time T_s). For instance, if $f = 10$Hz, we can select $f_s = 100$Hz (`fs = 100;` or `Ts = 0.01;`), which is 10 times higher than f. Then, we need to define a vector of time instants `t` that corresponds to the sampling points, using the colon operator (`:`). For example, if we want to sample the signal for 1 second, we can use `t = 0:1/fs:1;` This creates a vector of 101 elements from 0 to 1 with a step size of 0.01 (the sampling time $T_s = 0.01$). Next, we need to evaluate the signal $x(t)$ at these time instants and store the result in another vector `x`. For example, `x = cos(2*pi*f*t);` This creates a vector of 101 elements that contains the samples of $x(t)$ at `t`.

```
t_start = 0;
t_end = 1;
Ts = 0.01;
t = t_start:Ts:t_end;

f = 10;
x = cos(2*pi*f*t);
```

Assignment: Plot the graphs of the following signals for the time interval $0 \le t \le 10$, using a sampling time T_s that is appropriate for each signal. (Hint: You may need to use different values of T_s for different signals.)

1. $x(t) = \sin\left(\frac{t}{2}\right)$
2. $x(t) = \cos(2\pi t)$
3. $x(t) = \begin{cases} 1 & 4 \le t \le 7 \\ 0 & O.W. \end{cases}$
4. $x(t) = 1 - e^{-t}$
5. $x(t) = t\ln(t) - 2t$
6. $x(t) = \sqrt[3]{t^3 - 8t^2 + 13}$
7. $x(t) = \delta(t-3)$
8. $u(t)$ and $y(t)$ of problem number 6 (on the same axes)

8 Numerical Solution of Ordinary Differential Equations

To obtain a numerical solution of an ordinary differential equation (ODE) using MATLAB, you can use one of the built-in ODE solvers, such as `ode45`, `ode15s`, `ode23`, etc. To solve an ODE using the `ode45` function in MATLAB, you need to follow these steps

I) Define your ODE as a function that takes two inputs

1. the independent variable (usually time)
2. the dependent variable (usually a vector of state variables)

The function should return the value of the first derivative of the dependent variable. For example, if your ODE is $\dot{y}(t) = ty(t)$, you can define a function as follows

```
function dydt = myODE(t, y)
dydt = t*y;
end
```

II) Specify the initial conditions and the time span for the solution. For example, if your initial condition is $y(0) = 1$ and you want to solve the ODE from $t = 0$ to $t = 5$, you can use

```
y0 = 1;
tspan = [0 5];
```

III) Choose `ode45` as the ODE solver for your problem and pass the function name, the time span, and the initial condition as arguments. The solver will return two outputs

1. a vector of time points
2. a vector or matrix of solution values

For example

```
[t, y] = ode45(@myode, tspan, y0);
```

Assignment: Use the `ode45` function to obtain the numerical solution of the following ordinary differential equation with the specified initial conditions

$$\dot{y}(t) = y^2(t) - 3y(t)$$

1. $y(0) = 1$
2. $y(0) = 1.5$
3. $y(0) = 2$
4. $y(0) = 3$
5. $y(0) = 4$

9 Numerical Methods to Find the Response of a Linear System

Numerical methods are techniques for finding approximate solutions of linear systems. In this section, we will apply the Euler method, which is a numerical method for solving ordinary differential equations, and use the `lsim` function, which is a MATLAB function for simulating the response of a linear system to arbitrary inputs and initial conditions. The `lsim` function can help us analyze the behavior and performance of a system under different scenarios and conditions. The following steps describe how to use the `lsim` function in MATLAB

I) Define your system as a linear time-invariant (LTI) object, such as a transfer function, a state-space model, or a zero-pole-gain model. You can use the built-in functions such as `tf`, `ss`, or `zpk` to create these objects. For example, if your system is described by the transfer function $T(s) = 1/(s+1)$, you can use

```
sys = tf(1, [1, 1]);
```

II) Specify the input signal that you want to apply to your system. The input signal should be a vector or matrix that contains the values of the input at each time point. The length of the input signal should match the length of the time vector that you will use for the simulation. For example, if you want to use a unit step input for 10 seconds, you can use

```
t = 0:Ts:10;
u = ones(size(t));
```

III) **Optionally**, you can also specify the initial conditions of your system, if it is a **state-space** model. The initial conditions should be a vector that contains the values of the state variables at the initial time. The length of the initial conditions vector should match the number of states in your system. For example, if your system has two states and you want to use zero initial conditions, you can use

```
x0 = [0, 0];
```

IV) Pass the system object, the input signal, and the time vector as arguments to the function. If you have specified initial conditions, you need to pass them as well. The function will return the output signal `y`

```
y = lsim(sys,u,t);
```

To find the derivative of a function in the continuous-time domain, we have

$$\dot{y}(t) = \lim_{h\to 0} \frac{y(t+h)-y(t)}{h} \qquad \text{and} \qquad \dot{y}(t) = \lim_{h\to 0} \frac{y(t)-y(t-h)}{h}$$

In contrast, in the discrete-time domain, h is the sampling time T_s and there are many limitations to making it too small. Therefore, we have to approximate the derivative rule as follows

Forward Euler	Backward Euler
$\dot{y}(t) \approx \dfrac{y(t+h)-y(t)}{h}$	$\dot{y}(t) \approx \dfrac{y(t)-y(t-h)}{h}$

Second-order or higher-order derivatives are also achievable as follows

Forward Euler	Backward Euler
$\ddot{y}(t) \approx \dfrac{\dot{y}(t+h)-\dot{y}(t)}{h}$	$\ddot{y}(t) \approx \dfrac{\dot{y}(t)-\dot{y}(t-h)}{h}$
$\downarrow$	$\downarrow$
$\ddot{y}(t) \approx \dfrac{y(t+2h)-2y(t+h)+y(t)}{h^2}$	$\ddot{y}(t) \approx \dfrac{y(t)-2y(t-h)+y(t-2h)}{h^2}$

To find a numerical solution of an ODE using the backward Euler method iteratively in MATLAB, you need to follow these steps

I) Rewrite the differential equation in the following form using backward Euler method (green equations)

$$y[n] = f(n, y[n-1], y[n-2], \cdots, u[n], u[n-1], u[n-2], \cdots)$$

where discrete sample $y[n]$ approximately shows the continuous time signal $y(t)$ at time $t = nh$.

II) Initialize a vector of time instants `t` that corresponds to the sampling points, and a vector of output samples `Y` that stores the values of the output samples. If the system has initial conditions, you should assign them to the first elements of the `Y` vector.

```
t = 0:Ts:10;
Y = zeros(size(t));
Y(1) = y_0;
```

III) Initialize a vector of input samples `U` that stores the values of the input samples. For example if the input is a unit step signal, you can define the `U` array as follows

```
U = ones(size(t));
U(1:100) = 0;
```

IV) Use a `for` loop to iterate over the `t` array from the second element to the last element, and update the corresponding `Y` value using the Euler method

```
for i=2:length(t)
Y(i) = f(i, Y(i-1), ..., U(i), U(i-1), ...);
end
```

Assignment: Use the backward Euler method and the `lsim` function to find the response of the linear system in problem number 6. Plot and compare results.

10 Simulink

To find the response of a transfer function to an arbitrary input using Simulink, you need to follow these steps

I) Create a new Simulink model and add a `Transfer Fcn` block from the Continuous library. Double-click the block and enter the numerator and denominator coefficients of your transfer function in the corresponding fields.

II) Add a signal source block or combine them to provide the arbitrary input to your system. You can choose from different types of sources, such as Step, Ramp, Sine Wave, etc. from the Sources library. Double-click the block and adjust the parameters of the input signal as desired.

III) Add a Scope block from the Sinks library to display the output of your system.

IV) Connect the blocks!

V) Click the Run button on the toolbar to start the simulation. You can adjust the simulation time and other settings by clicking the Configuration Parameters button. You can also adjust the sampling time and change the solver.

VI) Double-click the Scope block to see the output of your system in response to the input signal. You can zoom, pan, or export the plot as needed

Assignment: Use Simulink to find the response of the linear system introduced in problem number 6. Compare results with those obtained in problems 6, 9 and 10.