ME 417 Control of Mechanical Systems

Summer 2020 - Numerical Assignment #2

Complete the numbered tasks and submit your work as a MATLAB Livescript in addition to an exported PDF report.

Complete the numbered tasks and submit your work according to the class assignment guideline document.

There are three parts to this assignment. In the first part you will continue to model and simulate the response of systems, in addition to calculating performance specifications. In the second part, you will follow a tutorial on how to implement a PID controller numerically. In the third part, you will begin to design a controller for given systems by tuning the gains of a PID controller until you achieve a desired performance.

PART I: TIME RESPONSE ANALYSIS

Problem A. Disk Drive Time Response Anlaysis

A disk drive read head can be modeled as a pair of rotating masses connected by a flexible shaft. As shown on Figure 1. I_m represents the motor's inertia, while I_h represents the head's inertia. The flexible shaft has a stifness K and damping coefficient b. M_m is the torque applied by the motor while M_D is the disturbance torque.

Disk Drive Parameters: $I_m = 0.1\text{kg} \cdot m^2, I_h = 0.04\text{kg} \cdot m^2, K = 0.07N \cdot \frac{m}{\text{rad}}, M_D(t) = 0, b = 0.1N \cdot s \cdot \frac{m}{\text{rad}}$

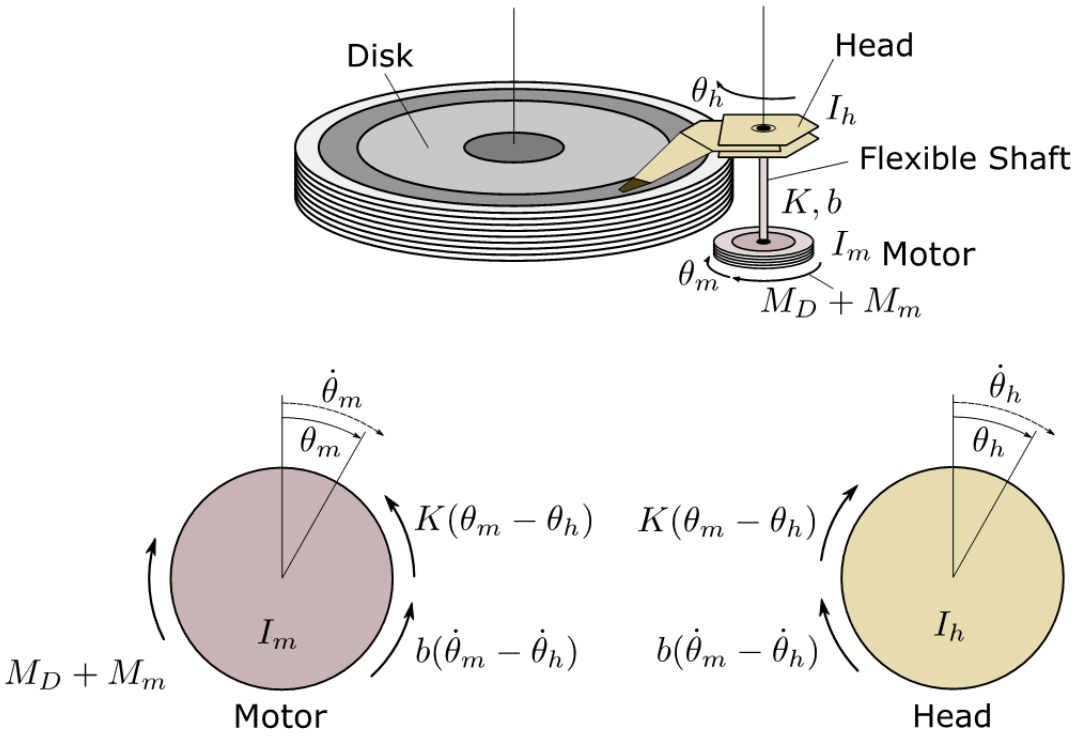

Figure 1 - Disk Drive Model

Task 1. Using the impedance method, find the equations of motion for the system in the Laplace Domain.

Task 2. Derive the transfer functions $G_1 = \frac{\ddot{\theta}_m}{M_m(s)}$, $G_2 = \frac{\ddot{\theta}_h}{M_m(s)}$**, symbolically. Hint: solve for a linear system of equations or use Cramer's rule. These functions may be useful:** ***numden(), sym2poly(), tf()***

Task 3. Simulate the response to a step input using MATLAB ***step()*** **command, then call** ***stepinfo()*** **command on the same system to get the performance specifications of this system.**

Task 4. Use basic numerical integration to simulate the response of the system to a step input $M_m(t) = 2$**, note that this is a 2DOF system, so the numerical integration vector would be** $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \theta_m \\ \dot{\theta_m} \\ \theta_h \\ \dot{\theta_h} \end{bmatrix}$**. But you will need to keep track of acceleration as well. So, either create a 6xn state vector to store** $\ddot{\theta}_m$ **and** $\ddot{\theta}_h$ **or keep a separate array for the accelerations.**

- **Plot the angular accelerationson one sublot, and the angular velocities on a second subplot.**
- **Hint: You already have the EOM from task 1, construct the numerical integration xdot function from them.**

Task 5. From the response in the previous task, numerically compute the following performance values for the acceleration response $\ddot{\theta}_h$**: Rise Time** T_r**, Peak Time** T_p**, Settling Time** T_s **and percet overshoot %OS. And display the computed results of these performance specifications on the figure. Compare the values you obtained from** ***stepinfo()*** **to the values you numerically computed.**

Hints:

- **For Rise Time: You need to compute the time it takes to go from 10% to 90% of the final output**
- **For %OS, find the** ***max()*** **value in the response and compare with the final value x(end)**
- **For Peak Time: use the index of** ***max()*** **to find the corresponding time.**
- **For Settling Time: This is more involved. You need to capture the instance when the response reaches AND stays within 2% of the final value. One way to do this is to**

set the settling time whenever the response reaches 2% of the final value and clear it (set $T_s = -1$ for instance) if the response leaves the 2% error band.

```
clear all
% Remember to call figure() every time you want a new figure generated
```

Problem B. DC Motor Parameter Identification

You are given a real brushed DC Motor, but you don't have its full specifications. You do; however, have a simple model for a DC motor and you are given three sets of experimental data for a motor:

- The angular velocity (rad/s) of the motor in response to a step voltage input of 10V
- The torque-speed curve for the motor at a set input voltage of 10V.

Each of the above data sets are provided in a separate and labeled csv file. The below code snippet shows you how to read the data into MATLAB arrays. Make sure the *.csv file is placed in the same folder as the MATLAB script file.

```
clear all
Data = csvread('Part_I_Problem_B_Data_Step_Response.csv');
t = Data(:,1);
x = Data(:,2);
```

The simple DC Motor Transfer function is given as

$$\frac{\Theta_m(s)}{E_m(s)} = \frac{K_t/(R_a J_m)}{s\left[s + \frac{1}{J_m}\left(D_m + \frac{K_t K_B}{R_a}\right)\right]} \qquad \text{Equation 1}$$

which is equivalent to $\frac{\Theta_m(s)}{E_m(s)} = \frac{K}{s(s+\alpha)}$, if all the constants are lumped. Note that the data set for the step response is given for angular velocity, so we need $\frac{\dot{\Theta}_m(s)}{E_m(s)}$. This can be easily derived by differentiating the angular position, by multiplication with a differentiator $\Theta(s)$, which gives the first order transfer function:

$$\frac{\dot{\Theta}_m(s)}{E_m(s)} = \frac{K}{(s+\alpha)} \qquad \text{Equation 2}$$

And remember, for the torque-speed curve, we use the time domain relationship that relate input voltage, angular velocity and torque:

$$T_m(t) = -\frac{K_B K_t}{R_a}\omega_m(t) + \frac{K_t}{R_a}e_a(t) \quad \text{Equation 3}$$

Task 6. Obtain, from the step response data set, the values of K and α in Equation 2.

- **The response is first order, if you find the time constant and scale the response you can find the two values K and α. Hint: Apply the Final Value Theorem to find K**

Task 7. Obtain, from the torque-speed curve data set, the value of the coefficients in Equation 3. Use $R_a = 8$

Task 8. From the previous two tasks, compute all the remaining coefficients, $J_m \,\&\, D_m$**, in Equation 1**

- **Also covered in class.**

Task 9. Now that you can redefine your transfer function, simulate it to a step input using the *step()* command and compare it to the step response data in the provided data set. (Remember to scale by 10 to account for the value of voltage input of 10V and to use the transfer function $\frac{\dot{\Theta}_m(s)}{E_m(s)}$**, which is equal to** $\frac{s\Theta_m(s)}{E_m(s)}$**.**

PART II: INTRODUCTION TO FEEDBACK CONTROL - NUMERICAL IMPLEMENTATION

This part is meant to serve as a tutorial on how to numerically implement a PID controller in a simulation. A MATLAB template script is provided to get you started. Follow through and complete the tasks.

A PID controller takes the form shown on Figure 2. Using the transfer function form, we can come up with a reduced equivalent closed-loop form and simulate the response of the feedback system to an impulse, step or ramp response directly, and this is useful in a number of control design scenarios. The reduced form is shown on Figure 3

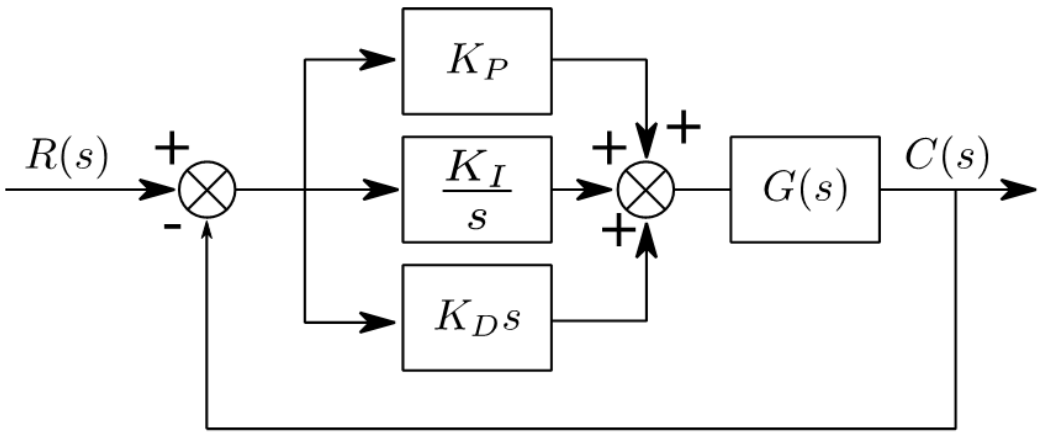

Figure 2 - PID Controller: Transfer Function Form

$$R(s) \quad \boxed{G_{cl} = \frac{(K_D s^2 + K_P s + K_I) G_{ol}}{s + (K_D s^2 + K_P s + K_I) G_{ol}}} \quad C(s)$$

Figure 3 - PID Controller: Equivalent Reduced Closed-Loop Form

But as we discussed in the previous assignment, implementing the simulation using numerical integration is a much more flexible design platform. Let's first get a sense on how to implement the PID Controller using transfer functions and using the reduced closed-loop form.

Task 10. Given the system $G_p = \frac{1}{(s^2 + 5s + 50)}$, and the gain values: $K_P = 300,\ K_D = 1, K_I = 300$.

Simulate the closed-loop response of the system with the PID controller, to a step input using *step()*.

- **Substitute the values into the closed-loop form and simulate.**

```
%% PID Using Step Response
clear all;
s = tf('s')
Gp = 2 / (s^2+8*s+25)
```

To implement the PID Controller, or any controller, numerically, we must understand its logic in the time domain. Figure 4shows the PID feedback controller expressed in the time domain.

The PID controller has three components: Proportional to Error, Proportional to Error Integral, Proportional to Error Derivative. So if we calculate the error, its integral and its derivative at every time sample we can compute the control law $u_{PID}(t)$ at every time sample (we assume a small enough simulation Δt , otherwise we have to discuss digital controllers, which is outside the scope of this assignment).

$$u_{PID}(t) = K_P e(t) + K_I \int e(t) + K_D \frac{de(t)}{dt} \qquad \text{Equation 4}$$

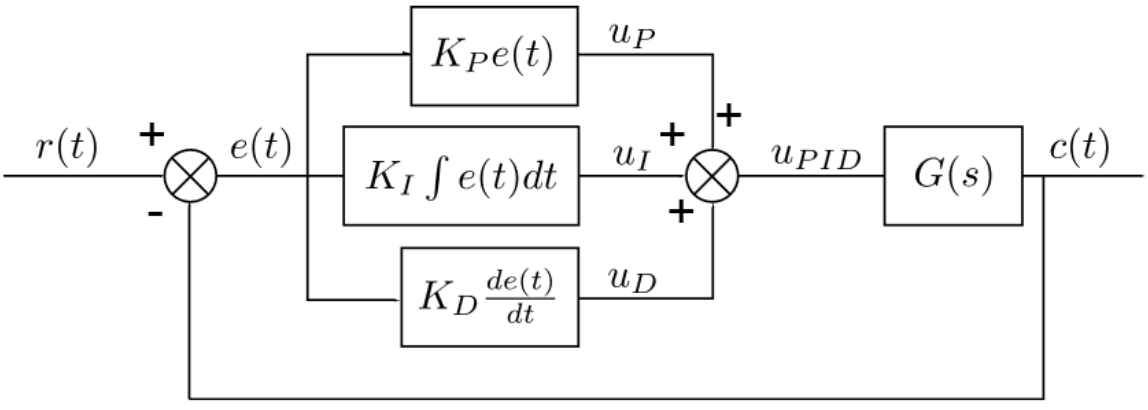

Figure 4 - PID Controller: Time Domain Form

```
% Task 10
```

Task 11. Use the provided script template, complete the algorithm definitions for

1. **The first error value and first control law before the integration loop**
2. **The next error value in the integration loop:** $e = r - x$
3. **The next error integral in the integration loop:** $\int e = (\int e)_{previous} + e \cdot dt$
4. **The next error derivative in the integration loop:** $\frac{de}{dt} = (e - e_{previous})/dt$
5. **The next control output in the integration loop (Use Equation 4)**

Then test your implementation of the numerical PID controller and compare the simulation output against the output you obtained using the closed-form method above. To troubleshoot your code, start first by applying the proportional gain to both methods (set $K_I = K_D = 0$**), then add the integral gain (set** $K_D = 0$**), then add the derivative gain instead (set** $K_I = 0$**) , then add all the gains.**

The expected responses, if implemented correctly with matching gains, should match just like the responses shown on Figure 5 below (actual values may vary).

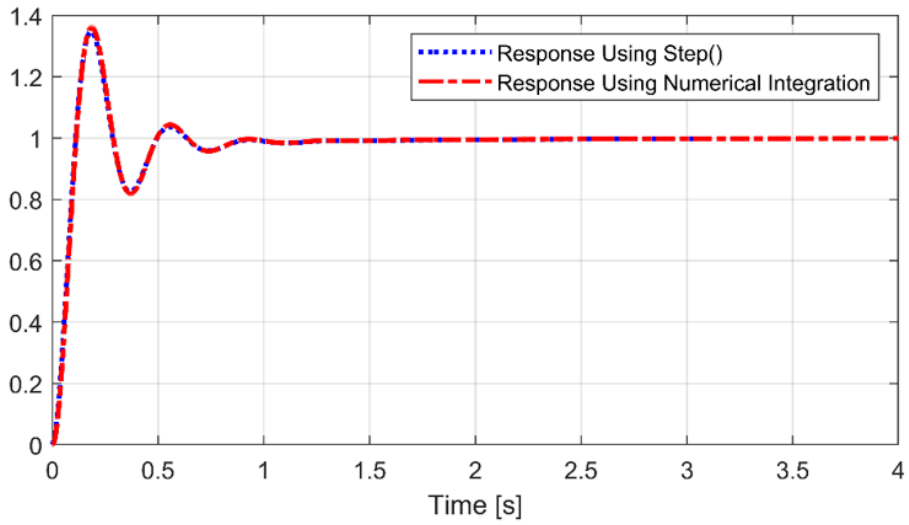

Figure 5 - PID Controller Feedback System Response Using Two Methods

```
%% PID Using Numerical Integration
% funtion for xdot vector - using an anonymous function. You can also
% define a separate function in a separate file or at the end of the
% script
dxdt = @(t,x,u) [x(2);
    2*u - 8 * x(2) - 25 * x(1)
    ];
% Integration Time step

dt = 0.001;
```

```
% Time vector
t_sim = 0:dt:4;

% Initialize x
x_sim = zeros(2,length(t_sim)); % Empty 2xn array
u = zeros(1,length(t_sim)); % Empty 1xn vector
e = zeros(1,length(t_sim)); % Empty 1xn vector
e_int = zeros(1,length(t_sim));
e_dot = zeros(1,length(t_sim));

% Set Initial Conditions
x0 = [0; 0];
x_sim(:,1)= x0;

% Reference input
r = 1;
% Compute First Error
e(1,1) = 0;
% Compute First Control Output
u(1,1) = 0;

for ix = 1:length(t_sim)-1

    % Simulate Response
    xdot = dxdt(t_sim(ix), x_sim(:,ix), u(1,ix)); % Grab the derivative vector
    x_sim(1, ix+1) = x_sim(1, ix) + xdot(1) * dt; % Integrate x1
    x_sim(2, ix+1) = x_sim(2, ix) + xdot(2) * dt; % Integrate x2

    % Compute Next Error
    e(1, ix + 1) = 0;
    % Compute Next Error Integral - Forward Integration
    e_int(1, ix + 1) = 0;
    % Compute Next Error Derivative
    e_dot(1, ix + 1) = 0;

    % Compute Next Control Law
    u(1,ix + 1) = 0;

end

hold on
plot(t_sim,x_sim(1,:))
xlabel('Time [s]')
% Task
```

PART III: FEEDBACK CONTROL - VIA GAIN TUNING

Problem A. DC Motor PID Speed and Torque Control - MATLAB Control Toolbox Commands

In **Part I Question B** you retrieved the model parameters for a DC motor, you will attempt to implement a feedback controller on this system to

a. control the speed of the motor to a given step set-point,

b. control the torque of the motor to a given step set-point.

For this question, implement the feedback controller using the transfer function method explained in **Part II.** Then tune the gains to achieve the required performance specifications. *Plot the responses and the performance specifications.*

Note: With the parameters retrieved in **Part I Problem B**. In addition, use the inductance term here L_a to achieve a more realistic transient response, with $\boldsymbol{L_a = 5H}$**.** The transfer functions are given.

Task 12. Using a PID Controller, implement it in a feedback loop to control the speed of a motor given the torque input. Attain the following performance specifications: $r(t) = 10rad/s$ **,** $T_r < 0.2s$**,** $T_p < 0.5s$**,** $\%OS < 12\%$**,** $T_s < 0.6s$**.Plot the responses**

$$\frac{\dot{\Theta}_m(s)}{E_a(s)} = \frac{K_t s}{\left(\left(Js^2 + D_m s\right)(R_a + L_a s) + K_t K_b s\right)}$$ Equation 5

```
% Task 12
clear all;
```

Task 13. In class, we discussed how we can represent the transfer function relating voltage to output torque. Use a PID controller in a feedback loop to control the Torque of the same motor. Attain the following performance specifications: $r(t) = 2N \cdot m$ **,** $T_r < 0.1s$**,** $T_p < 1s$**,** $\%OS < 0.1\%$**,** $T_s < 0.4s$**.Plot the responses**

$$\frac{T_m(s)}{E(s)} = \frac{K_t\left(Js^2 + D_m s\right)}{\left(Js^2 + D_m s\right)\left(R_a + L_a s\right) + K_t K_b s}$$ Equation 6

This problem should not consume much time if you have completed Part II: Use the right transfer function for the system and tune the gains through trial and error and your understanding of what each term of the PID does to achieve the required performance. Use this exercise to gain an appreciation for how each PID controller term affects the response.

Problem B. Disk Drive Position Control - Numerical Integration

In **Part I Problem A**, you were able to get a stable response for controlling the **acceleration** of the disk drive motor (The system $G_1(s) = \frac{\ddot{\Theta}_m(s)}{M_m(s)}$ is stable), if we attempt to get the speed or position response with a step input and without feedback, the system then is unstable (The systems $G_2(s) = \frac{\dot{\Theta}_m(s)}{M_m(s)}$ and $G_3(s) = \frac{\Theta_m(s)}{M_m(s)}$ are unstable). We can use feedback control to stabilize the systems G_2 or G_3.

Task 14. Use a PID Controller to achieve a stable position control response for the motor and attain the following performance specifications: $r(t) = 1rad$, $T_r < 0.15s$, $T_p < 2s$, $\%OS < 13\%$, $T_s < 2s$.

Plote the responses as follows

- **Subplot 1:** $\theta_m(t)$ **and** $\theta_h(t)$**, the motor angular position and the head angular position**
- **Subplot 2:** $\dot{\theta}_m(t)$ **and** $\dot{\theta}_h(t)$**, the motor angular velocity and the head angular velocity**
- **Subplot 3:** $e(t), \int e(t)dt$, $\dot{e}(t)$**, the motor angular position error, its integral and its derivative**
- **Subplot 4:** $u_{pid}(t)$**, the PID control output.**

To implement a PID controller using numerical integration, use the template you completed in **Part II** and replace the system model with the one in Part I Problem A, and adjust the code as needed. You should already have the model implemented in Part I Problem A, and completed Part II, you just need to combine them together and note that for calculating the error, you will be using Θ_m, which you already have calculated in the simulation.

Note: Use low values for the gains (start with values lower than 1)

```
% PID Control structure using basic numerical integration
clear all
```