MTHE 393 Project Description

The files required for this project can be downloaded from the MTHE 393 onQ page (Content top tab > Projects side tab). Each group has been assigned a black box system for which they will design a controller for - see the "Group System Assignments" document. Every system has a set of performance parameters for which the controller they design must meet - see the "System Targets" document.

All of the files mentioned above are relevant to the *technical problem* which this project addresses. In addition to this technical problem, your group must find a suitable application for the controller you are designing. This will be referred to as the *application problem*. At first, these problems will be considered separately; however, as your project progresses, the solutions to each problem should converge into one narrative which culminates in your final report and presentation. We must make one critical assumption to facilitate this narrative: The dynamics of your black box is representative of the system from your application. You are also welcome to make other assumptions over the course of the project so that your project is cohesive; however, it is critical that you make these assumptions explicitly clear to the reader. Failure to do this will result in the project receiving a lower mark.

The main purpose of this document is to outline the tasks for each week of this seven week project. This document should be considered as "living" meaning that it will be updated on a weekly basis with next week's tasks. Observe the "VXX" at the end of the documents title where "XX" is the version number.

Week 6 - Is it LTI?

***IMPORTANT NOTE* In order for HowToScriptGUI.m to work properly, you must first install the blackBox app using the installer in your system folder, and then open the app in the `APPS` header in MATLAB**.

The scope of this course is restricted to linear time-invariant (LTI) systems and so we first need to check that your black box indeed behaves as an LTI system. Consider an LTI system with $y(t)$ as the output and $u(t)$ as the input. Recall that the following properties must hold:

1. Two conditions must hold for linearity:
   (a) **Homogeneity** Let $a \in \mathbb{R}$. The system output to the input, $au(t)$, is $ay(t)$.
   (b) **Superposition** Let $\tilde{y}(t)$ be the output of the same system subjected to $\tilde{u}(t)$ as an input. When this system is subject to $u(t) + \tilde{u}(t)$ as the input, the output must be $y(t) + \tilde{y}(t)$.
2. **Time-Invariant** If the input is shifted by T seconds (i.e., $u(t - T)$), then the output must also be shifted by the same amount of time (i.e, $y(t - T)$).

Check that the above conditions hold for your black box. After you have checked that your black box is an LTI system, answer the following questions:

1. Explain mathematically what it means for a system to be LTI;
2. Why is it important that the black box is LTI?
3. Do your graphs from 1a and 1b overlay with each other? Can you quantify how similar they are or how much error there is between them?

Begin researching potential applications for your controller. It's good to consider big messy problems that your controller may solve, but keep in mind that the final application you choose must have one input and one output.

Week 8 - System ID using Frequency Response

Now that you have confirmed t hat a ssuming y our b lack b ox b ehaves a s a n L TI, y ou can move on to generating a model of your system. During the labs, you saw two ways to model a system:

1. **Time domain approach** Assume first-order dynamics. Step the input and then use the output response to compute the time constant and gain of the system.
2. **Frequency domain approach** Make a Bode plot of the system which captures how the magnitude and phase of the output changes with the frequency of a sine wave inputted to the system.

When you step your black box, you will likely observe that the assumption of first-order dynamics is not appropriate and so you will need to generate a Bode plot instead. The connection between the Bode plot and a system's transfer function will be addressed in this week's lecture. Here are some points to keep in mind as you build your Bode plot:

- Use approximately 10^{-4} to 10^{4} rad/s as your frequency range and assess approximately 50 "evenly" spaced points across the range. This process could be done either manually or automated through the use of a script.
- As you will see in the lecture, it is important to compute the amplitude ratio and phase shift once the initial transient behaviour has died off. You will need to adjust the length of your simulation to achieve this. Generally speaking:
  - Lower frequencies require longer run times (and larger time steps)
  - Higher frequencies require shorter run times (and smaller time steps)
- The output of the black box is clearly corrupted by noise. Consider using filtering to to minimize the impact of noise on your final Bode plot.

Once you have completed your Bode plots, answer the following questions:

1. Why is it important to create a Bode Plot for your black box?
2. How did you make your Bode Plot for your system? Explain the techniques you used to deal with noise, how you calculated the phase shift, etc.

In addition to working on the technical problem, continue exploring different applications for the controller. Also, do not lose sight of the upcoming deliverables:

- Status Report 1 - due Friday February 28, 2025 at 23:59 to onQ
- Interim Report - due Friday March 14, 2025 at 23:59 to onQ

Week 9 - System ID using Frequency Response - part 2

Keep working on generating your Bode Plots while not losing sight of the Interim report due on Friday March 14, 2025 at 23:59.

Week 10 - Deriving your model

Derive a transfer function model of your black box using the Bode plot you have generated. The lecture notes and MATLAB Live script used in the Week 8 and 9 lecture are good resources for how to do this. Make sure that you use your magnitude plot and not the phase plot for this task. You will see in this week's lecture that if a plant has RHP zeros, the rules that your have learned on how to sketch Bode plots break down.

Once you have determined your transfer function model, you need to verify that it is a good fit. There are multiple ways this could be d one. You could look at it through the lens of the time domain and compare the step response of your transfer function model with the black box. Alternatively, you could take a frequency domain based approach by comparing the Bode plot of your transfer function model with the empirical Bode plot you generated from your black box. You are welcome to tune your initial transfer function model to obtain a better fit.

Once you have your transfer function model, answer the following questions:

1. Does your transfer function give you any hints about the stability of your system? Look at the location of the poles and the zeroes of your transfer function.
2. Now, you can check whether the transfer function you created is a good representation of your black box system. Plot the step response of the black box and your transfer function, and look at the difference between the two. You can justify this quantifiably using the MATLAB plots.
   - Hint: you can make you the `step(sys)` command in MATLAB after creating a transfer function variable `sys = tf([],[])` and putting in the coefficients for the transfer function's numerator and denominator. See the appendix of the lab manual if you don't remember how to do this.

Group Evaluation: during Week 10 studio session

Week 11 - Controller design - coarse tuning

The reason for finding a transfer function model of the black box is so that a control system can be designed for it. You may be asking why couldn't we have just have designed the controller using the black box? In this project, the black box is taking the place of a physical system. You will discover that many controller tunings can make the system go unstable which in the case of a physical system would lead to potentially dangerous operating conditions as well as very expensive repairs.

This week you are going to make your first pass at designing a controller using your transfer function model in place of your black box:

1. Open a blank Simulink model.
2. Create a block for your transfer function model.
3. Drag another continuous transfer function block into the Simulink workspace and configure it as follows:
   (a) Double-click on the block to view the Block Parameters window.
   (b) Enter `[Kd Kp Ki]` as the numerator vector.
   (c) Enter `[tauf 1 0]` as the denominator vector.
   (d) Click "OK" to lock in these parameters.

   You have just made a commercial PID controller in you Simulink model. The K_p, K_i and K_d are the tuning constants for the proportional, integral and derivative actions, respectively, and the τ_f is the filter constant. These parameters will need to be defined in your MATLAB workspace prior to running the simulation. Alternatively, you could replace these variables with actual numbers.

4. Build the remainder of the simulation as shown in Figure 1. Note the following points:
   (a) This figure does not include the blocks required to capture the signals for analysis. You will need to figure this out on your own but recall that you had to complete a similar task during the labs for this course
   (b) The sum block **subtracts** the model output from the setpoint signal.
   (c) The transfer function-based controller defined above is one way to implement a PID controller in Simulink. If have a closer look in the Simulink library, you will find that there are other PID controller blocks. If you opt to use one of these built-in blocks, be careful as their parameterization and filter design (required when using derivative action) are likely different than what was discussed in lecture.
5. Tune your controller by experimenting with different values of K_p, K_i, K_d and tau_f. The"System Targets" and "System Targets Graphic" documents posted to onQ (Content top tab > Projects (Weeks 6-12) side tab) captures the performance you should be aiming for.

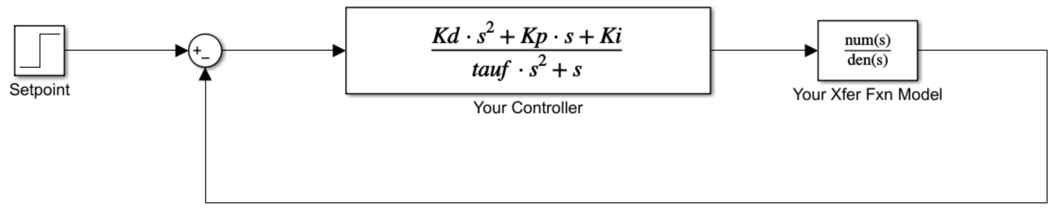


Figure 1: Linear feedback control of transfer function model

In addition to working on the technical problem, do not lose sight of the upcoming deliverables:

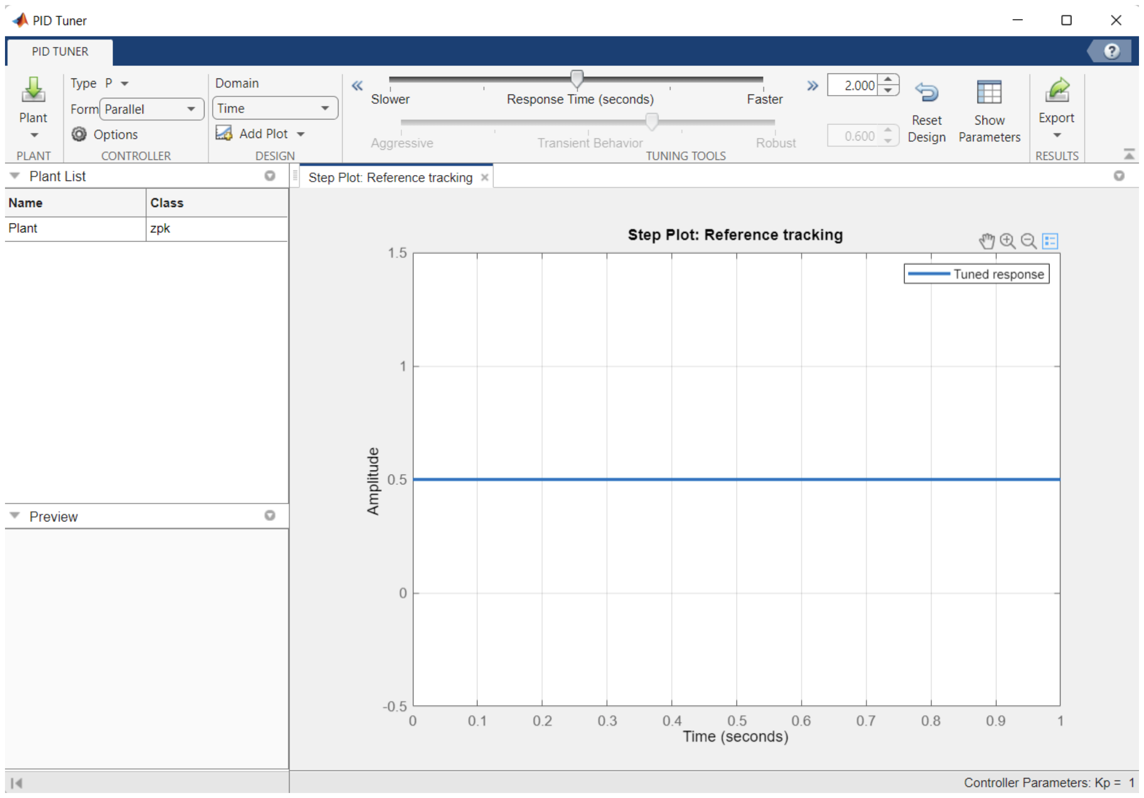
- Status Report 2 - due FridayMarch 28, 2025 at 23:59 to onQ
- Final Presentation - due Monday April 7,2025
- Final Report - due Tuesday April 8, 2025 at 23:59 to onQ

PID Tuner MATLAB

MATLAB has a useful built in app called the PID Tuner. This can be accessed through the `Apps` tab in the MATLAB navigation bar. To use the app, carry out the following steps:

1. Create the system in your MATLAB workspace. One way to do this is using the `zpk()` function of the control system toolbox. Refer to the appropriate documentation on how to use this.

2. To import the function into the app click the `Plant` button in the top left corner of the application shown in Figure 2. This will automatically detect the system in the workspace. Click Import.



Type P

Form Parallel

Domain Time

Response Time (seconds) 2.000

Transient Behavior 0.600

Name	Class
Plant	zpk

Controller Parameters: Kp = 1

Figure 2: PID Tuner MATLAB app

3. Change the type drop down to `PID` or your preferred method.

4. Now you will be able to drag the `Response Time` and `Transient Behavior` sliders to adjust the characteristics of your controller.

5. Once satisfied click the `Show Parameters` button which will open a popup containing the values of K, KI, and KP as well as other information such as rise time and settling time.

Use the parameters from the PID tuner as a starting point for tuning the black box system. If you are using the PID tuner, it may be helpful to use one of the PID blocks from the Library Browser.

- Status Report 2 - due **Friday** March 28, 2025 at 23:59 to onQ

Week 12 - Controller design - fine tuning

Last week, you developed a preliminary controller design using a model of your black box. This week you will you will implement this controller on you black box and see how it performs using the following steps:

1. Make a copy of last week's simulation and place it in the directory which holds all of black box files you have been using. We will refer to this simulation as `bbcontrol.mdl`
2. Open `bbcontrol.mdl` and copy the model.
3. Open your system's `stateSpace.mdl` and paste the `bbcontrol.mdl*` blocks into `stateSpace.mdl`. Delete the model transfer function block.
4. Route the signals so that your controller is now using the `csfunc*` block as the plant.
5. Run the simulation to assess the baseline performance of your controller. The noise in the plant and the error in you model means that controller will very likely have worse performance than what you saw last week.
6. Adjust the tuning of your controller to achieve the desired level of performance. This can be done using the `PID Tuner` App. Refer to The "System Targets" and "System Targets Graphic" documents posted to onQ (Content top tab > Projects (Weeks 6-12) side tab) captures the performance you should be aiming for.

In addition to working on the technical problem, do not lose sight of the upcoming deliverables:

- Final Presentation - due on Monday April 7, 2025
- Final Report - due Tuesday April 8, 2025 at 23:59 to onQ