Overview

This framework is designed to allow for the implementation of a Model Predictive Controller on the gantry crane system. The controller blocks (shown in yellow in Figure 1) are implemented inside MATLAB functions, which are then connected to a nonlinear model of the gantry crane. Your functions in this assignment have access to any MATLAB function provided by the Optimization toolbox, Model Predictive Control toolbox, Control toolbox and MATLAB itself.

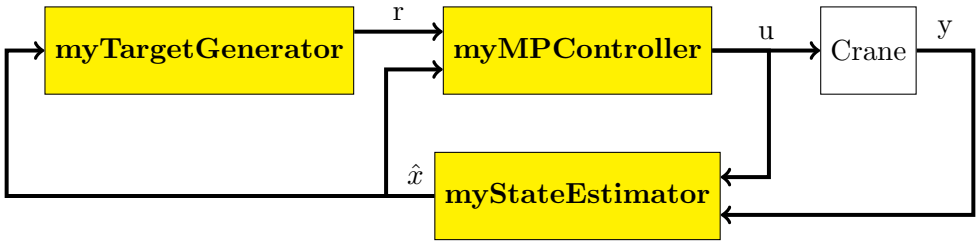

Figure 1: Block diagram of the complete system. Yellow blocks are the user-editable functions.

Files

The framework is composed of the following files provided on MATLAB Drive:

- **FunctionTemplate.m**
- **defaultCourse.m**
- **testMyDesign_Nonlinear.m**
- **HelperFunctions/analyzeCourse.p**
- **HelperFunctions/extractFunctions.m**
- **HelperFunctions/plotCourse.m**
- **Model/crane_nl_model.p**
- **Model/crane_nl_model_student.m**
- **Model/Crane_NominalParameters.mat**

The main design work must be done inside **FunctionTemplate.m**, which contains skeleton code for the four required functions. The course being tested against can be modified by changing the parameters inside the **defaultCourse.m** function to give new shapes/parameters to use. The functions inside the **HelperFunctions** and **Model** directories should not be modified.

Using the Framework

To begin using the framework, download the necessary files to your local computer and ensure that all the files are on your MATLAB path. To design the controllers, modify the functions provided in the **FunctionTemplate.m** file. Do not modify the input or output variable declarations of the functions, including the size of the variables. Only make changes to the functions inside the template file, since all other locations (e.g. inside the file with the function's name) will be overwritten every time you use the **testMyDesign_Nonlinear** script.

You can test your design on the nonlinear ODE model of the gantry crane using the **testMyDesign_nonlinear.m** script. This script will automatically extract your controller functions into

separate files, load the default course and the model parameters, run a simulation of the closed-loop system, and display the resulting crane path, constraints, and settling time information.

Your controller can be debugged by placing debug points inside the **FunctionTemplate.m** file on the appropriate line, and then using the MATLAB debugger. Note that the debugger will open the file with the function's name instead of the **FunctionTemplate.m** file when it is run.

Additional functions can be added to the template by simply adding their definition to the bottom of the file after the **myMPController** function. These functions will be automatically extracted to separate files in the working directory when the script is run, so they will always be on the MATLAB path.

Controller Functions

The controller is composed of four different functions:

- mySetup
- myStateEstimator
- myTargetGenerator
- myMPController

mySetup

This function is where you can implement any controller generation/design tasks (e.g. matrix generation, gain programming, constraint generation, etc.). This function gets 1 input variable that describe the shape being tested on: *shape*. This variable is a struct that contains the following fields:

- *constraints* - The rectangle and ellipses for the current course
- *eps_r* - The rate and input tolerance
- *eps_t* - The target point tolerance
- *target* - The target point
- *start* - The start point
- $Wmax$ - The maximum amount of work allowed to be done
- Tf - The final time for the controller

This function returns 1 variable, *param*. This variable is a MATLAB structure for which you define the fields. This structure will be passed to the 3 controller functions, so variables set inside this structure can be used inside your controller functions.

You may choose a sampling time for the controller functions to run at by setting the Ts variable inside the *params* struct before returning from this function to the desired sampling interval in seconds. If no Ts is given, the default sampling rate of 20 Hz will be used.

Controller Functions

You will be responsible for writing 3 controller functions that run inside the architecture shown in Figure 1. Every block will have a fixed size for the input and output vectors, which is described in the following sections.

Additionally, every block will have *param* as an input variable. This variable is a structure created by you inside the setup function and then passed into these functions. Note that due to the way MATLAB interacts with structs, you cannot modify *param* inside the 3 controller functions

and have the changes be preserved. Use persistent variables if you would like to keep values between the time steps of the controller.

myStateEstimator

If desired, you can implement a custom state estimator/observer inside this function block. This function takes two vectors as its input: the current measurements from the crane, and the inputs that were supplied to the crane at the last sample time. There is a single output from the estimator that contains the states of the crane (or whatever you desire).

The control input, u is a 2x1 vector that has the following components: $u = \begin{bmatrix} u_x & u_y \end{bmatrix}^T$. The input containing the measured system variables is an 8x1 vector that has the following components: $y = \begin{bmatrix} x_c & \dot{x}_c & y_c & \dot{y}_c & \theta & \dot{\theta} & \psi & \dot{\psi} \end{bmatrix}^T$.

The output from this function is 16x1 vector $\hat{x}$ which connects to the **myTargetGenerator** and **myMPController** blocks. By default, the measured system variables are passed through to the output in the same order, and the unused indices are set to 0.

myTargetGenerator

This block is designed to contain the target generator for your controller. The block has 1 input, which is the output from the state estimator, and 1 output to the controller block. The input is the $\hat{x}$ vector from the state estimator. The output is a vector r (that you specify the structure of) that connects to the **myMPController** block. By default, this block does not have anything in it and the output is all 0.

myMPController

This block is designed to contain your MPC implementation. The block has 2 inputs, the first is r, which is the vector from the target generator, and the second is $\hat{x}$, which is the output from the state estimator. The block has 1 output, which is the 2x1 control signal u that is given to the crane and to the state estimator in the next sample period. The control signal has the components: $u = \begin{bmatrix} u_x & u_y \end{bmatrix}^T$.

Submission

The entire **FunctionTemplate.m** file needs to be submitted on MATLAB Grader. Grader will perform 2 tests to check for the correctness of the input/output port sizes and to ensure that the controller is able to run on the linear model. No marking will be done on MATLAB Grader, and the tests on it have no impact on the final marks your controller will receive. Note that there is a limit of 60,000 characters for entering your controllers into the MATLAB Grader submission field.