

**Operation Manual**

**For**

**Engine Model Predictive Controller**

Team 513:

Austin LaFever, Patrick Marlatt, Frederick Peterson, Jonathan Wozny

**Table of Contents**

**Project Overview**  p.3

**Github Repository** p.3

**Adding or Replacing an MPC** p.3

**Saving, Uploading, and Sharing** p.9

**Hardware-in-the-Loop Validation** p.9

**Project Overview**

* Project summary
	+ The purpose of this project is to create a modular controller to control the throttle and wastegate in a simulated engine in the MathWork’s Powertrain blockset. This controller will be responsible for controlling these airflow components using a multi-input multi-output (MIMO) model predictive controller (MPC). Using this new control method will modernize the Powertrain Blockset as well as improve fuel efficiency and engine performance.
* Required software/Simulink toolboxes
	+ MATLAB
	+ For Project Simulation
		- Communications Toolbox (for Bernoulli Binary PRBS Signal)
			* DSP Systems Toolbox
			* Signal Processing Toolbox
		- Control System Toolbox
		- Powertrain BlockSet
		- Model Predictive Control Toolbox
		- Powertrain Blockset
		- Simscape
		- Simulink
		- Stateflow
		- System Identification Toolbox
* Required Hardware
	+ Arduino (instructions will specify mega, can be changed)
	+ Rotary encoder
	+ Servo motor

**Github Repository**

* All necessary files are in the GitHub repository linked below:<https://github.com/YorkPatty/T513---SIEngineDynamometer>
* To update the Simulink files in a new dynamometer with the files from the repository, simply download any .slx file from the GitHub, and replace the file with the same name in the SIDynamometer folder (The main reference application is under SIDynamometer → Main → Dyno)

**Adding or Replacing an MPC**

* Open a new Dynamometer simulation by typing “autoblkSIDynamometerStart” into command window of MATLAB
	+ Entering this into the command window will create a new project every time this command is used.
* Logging Data in SIDynoReferenceApplication for Data Inspector Tool
	+ To enable data logging (i.e. recording the data in the Data Inspector Tool) after simulation of the reference application, right click anywhere on the screen and select ‘Model Configuration Parameters.’
	+ In the pop-up menu, select the ‘Data Import/Export’ tab. (see Figure below)
	+ Under ‘Save to workspace or file’, check the ‘Signal Logging’ box.
	+ Select ‘Apply’ to the bottom right and click ‘OK.’



Figure 1: Configuration Parameters

* To log a desired signal, right click the signal arrow and select ‘Log Selected Signals.’
* Provide a name for the logged signal by double left-clicking the signal arrow and typing the desired name.
* After running the SI Dynamometer Reference Application, the Data Inspector window (shown in the figure below) will appear.
* Default logged signals will be shown under ‘Dynamometer Results’ and the manually logged signals will appear above all previous default signal logs. Past runs with manually logged signals can be accessed in the ‘Archive’ tab in the bottom left corner of the Data Inspector.



Figure 2: Data Inspector Tool

* Creating Linear State Space Models for System Identification
	+ **Overview:** The MPC system relies on linear state space models. These models can be developed by testing an engine (or engine simulation in this case) by stimulating or “wiggling” the inputs to the engine and measuring the responses. The SI Dynamometer Reference Application can be used to generate this artificial engine data. Engine operation can be divided into operating points (i.e. speed and torque of the engine). Stimulating the input and measuring the responses is performed at operating points. The Reference Application can be modified to send changes into the inputs (i.e. throttle and wastegate positions) and the responses will be measured.
	+ Stimulating Inputs using Pseudo Random Binary Generator
		- Generates semi random throttle positions for simulation
		- SiDynoReferenceApplication → Dynamometer Control → Steady State



Figure 3: Bernoulli Binary Generator Implementation

* + - Double click the Bus Assignment block (top right in the above figure).
		- In the pop-up box, select ThrPosPctCmd and WgPosPctCmd and click ‘Select>>’.
		- Click ‘Apply’ and ‘Ok.’
		- In the diagram above, the block with Unit:% is a Signal Specification block.
		- The block to its right is a Rate Transition block. Set the ‘Output port sample time’ parameter in this to -1 (will inherit the sample rate from the program).
	+ Collecting Simulation Data and Sending to Workspace
		- Navigate through SIDynoReferenceApplication → Engine System → Engine Plant
		- Drop in a To Workspace Block and set a desired workspace variable name.
		- Drop in a Mux bar. Set the desired number of inputs. Connect the desired i/o output signals.
		- Running the simulation (through ‘Run’ button or through GetIDData script (see github) will save the simulation data in the workspace in the order corresponding to entry into the mux block.



Figure 4: Data collection

* + MPC Placement
		- Navigate through SIDynoReferenceApplication → Engine System → Engine Controller → SIEngineController



Figure 5: MPC Block Implementation

* Generating Models to be used for MPC
	+ To create state space models and mpc objects to be placed into an MPC block, download the scripts GetIDData.m and GenerateMPCDesigns.m.
		- In GetIDData.m, values can be manipulated to change the tested RPM, throttle value, and throttle manipulation.
		- Do not make any changes to the GenerateMPCDesigns.m script. The only changes that would be made are the order of the state space model created. This can be done by changing the order parameter where the n4sid command is used. Do not create a model with an order larger than 5.
	+ For manual creation of state space and MPC object(s). Use GetIDData to obtain data. This data can be used to create a state space model in the System Identification Toolbox.
		- [Introduction to System Identification - Video - MATLAB](https://www.mathworks.com/videos/introduction-to-system-identification-81796.html)
	+ Once a state space model is created, it can be used in the MPC Designer to create an MPC object.
		- <https://www.mathworks.com/help/mpc/gs/introduction.html>
	+ MPC Help
		- What is MPC?
			* [What Is Model Predictive Control Toolbox? - Video](https://www.mathworks.com/videos/model-predictive-control-toolbox-overview-1516358143348.html?s_eid=PSM_15028)
		- MPC Toolbox and help links
			* [Model Predictive Control Toolbox - MATLAB](https://www.mathworks.com/products/model-predictive-control.html?s_eid=PSM_15028)
* Troubleshooting MPC implementation into Powertrain Blockset
	+ Controller inputs
		- Torque Command (reference)
		- Engine Torque
	+ Controller output
		- Throttle Position Command
		- Wastegate Position Command
	+ Possible Issues
		- Sampling time errors, make sure sampling time in MPC matches simulation sampling time
			* Fix this by with either a ‘Rate Transition’ block or by specifying sample time as inherited by inputting -1.
		- Log data in
		- Indexing issues for RPM testing
			* To test around a single operating point, replace the block in SIDynoReferenceApplication → Dynamometer Control → Steady State going into ‘EngSpdCmdPts’ with a constant, whatever RPM test value desired
			* To prevent the Dynamometer from continuing to index, go to SIDynoReferenceApplication → Dynamometer Control → Steady State → Select Operating Point and go to the furthest right block
				+ Replace Index = Index + 1; with Index = Index;
		- Array sizing errors in MPC blocks

**Saving, Uploading, and Sharing**

* To load a previously saved file, just click the SIDynoReferenceApplication.slx file path in the File Explorer window
	+ Typing “autoblkSIDynamometerStart” into the command window will create a new file, NO CHANGES MADE PREVIOUSLY WILL BE INCLUDED
	+ Create a file that opens the directory of the modified dyno

**Hardware-in-the-Loop Validation**

* For Hardware in Loop Testing
	+ Required Software/Add-ons
		- Embedded Coder
		- MATLAB Support Package for Arduino Hardware
		- MATLAB Coder
		- Simulink Coder
	+ Required hardware
		- Arduino (any model, as long as it has a serial port and can be wired to the other parts)
		- Rotary encoder
		- Servo motor
* HIL assembly instructions:
	+ Wire the rotary encoder to be read by the arduino
	+ Wire the servo motor to be commanded by the arduino
	+ Connect the servo motor to a shaft, and connect a flat disc to act as the ‘valve’ for the simulated throttle or wastegate

 

Figure 6: Hardware in Loop Diagram

* HIL troubleshooting
	+ Motor not spinning
		- Check wire connections
	+ Angle sensor not reading values
		- Check wire connections, ensure code is compiling correctly
* Wiring diagrams
	+ Wiring diagrams are included with purchased encoders and servo motors