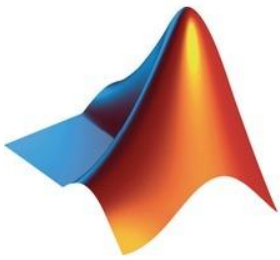

MathWorks®

Operation Manual
For
Engine Model Predictive Controller

Team 513:
Austin LaFever, Patrick Marlatt, Frederick Peterson, Jonathan Wozny

Table of Contents

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 transient 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

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)
- For automated creation of state-space models and MPC objects, download the following files from the GitHub:
    - CreateDiffMPC.zip:
        - RunMe.m
        - GenerateMPCDesigns.m
        - diffMAPParam.m
        - PlotFits.m

Adding or Replacing an MPC

- Open a new Dynamometer simulation by typing "autoblkSIDynamometerStart" into command window of MATLAB

- Note: 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.’

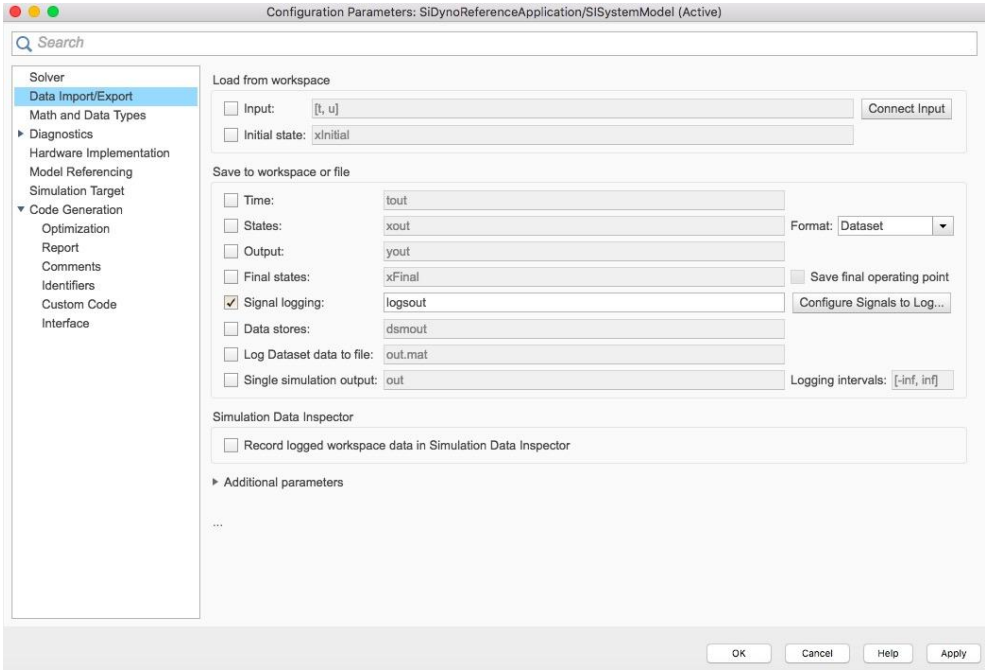

Figure1: Configuration Parameters

- 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.

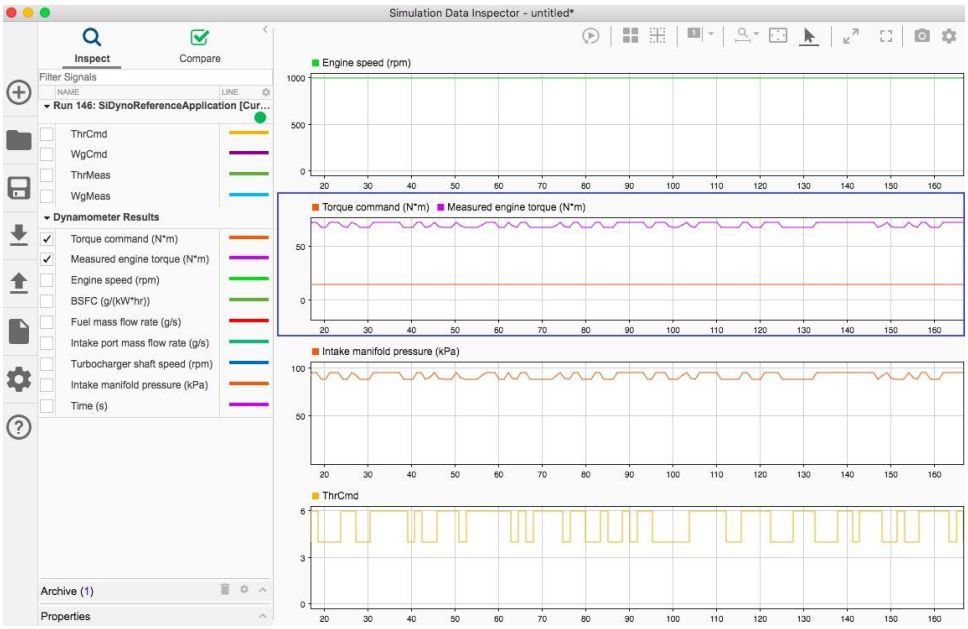

Figure 2: Data Inspector Tool

- Creating Linear State Space Models for System Identification

  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

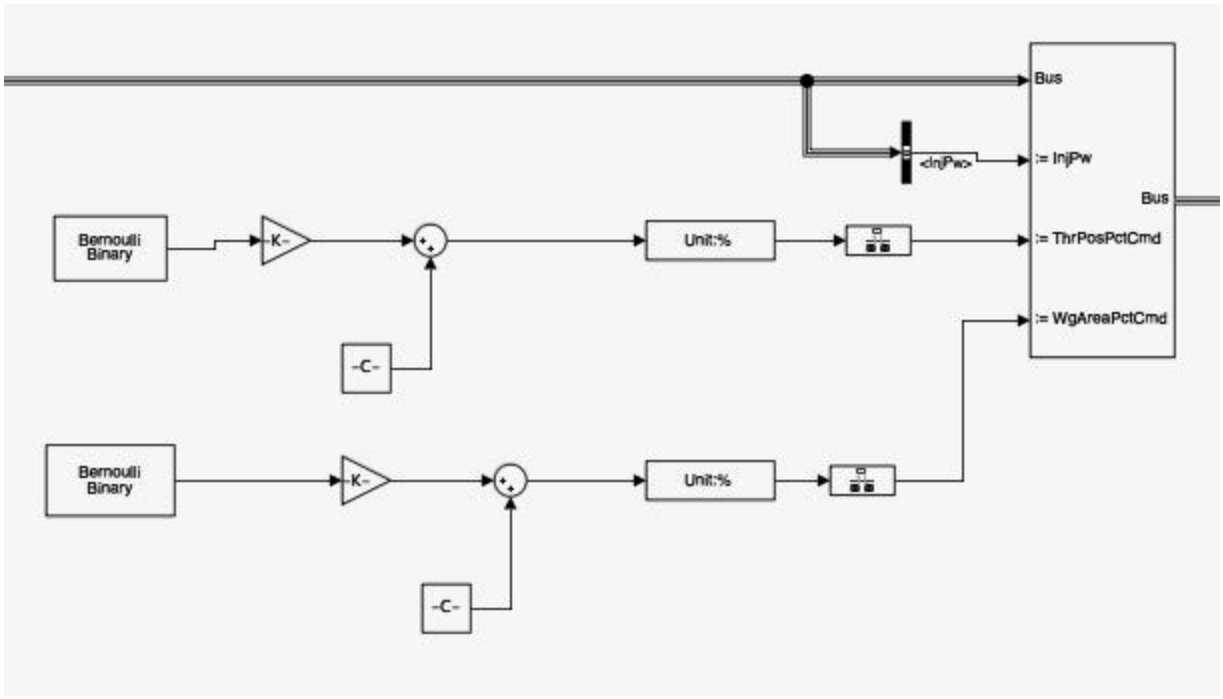

Figure 3: Bernoulli Binary Generator Implementation

- 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).
- The ‘Bernoulli Binary’ block generates a PRBS signal to be sent into the throttle or wastegate command. This specific block is not required to be the PRBS signal, it was just the one we used.
- Note: Figure 3 above appears to have two different PRBS signals, one going into the throttle command and one going into the wastegate command, but one should be set to wide open (100%) with no wiggle, while a PRBS will be sent into the other actuator. These values can be changed in the RunMe.m script (ThrB, ThrW, WgB, and WgW variables).

- 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 RunMe script from the GitHub will save the simulation data in the workspace in the order corresponding to entry into the mux block.

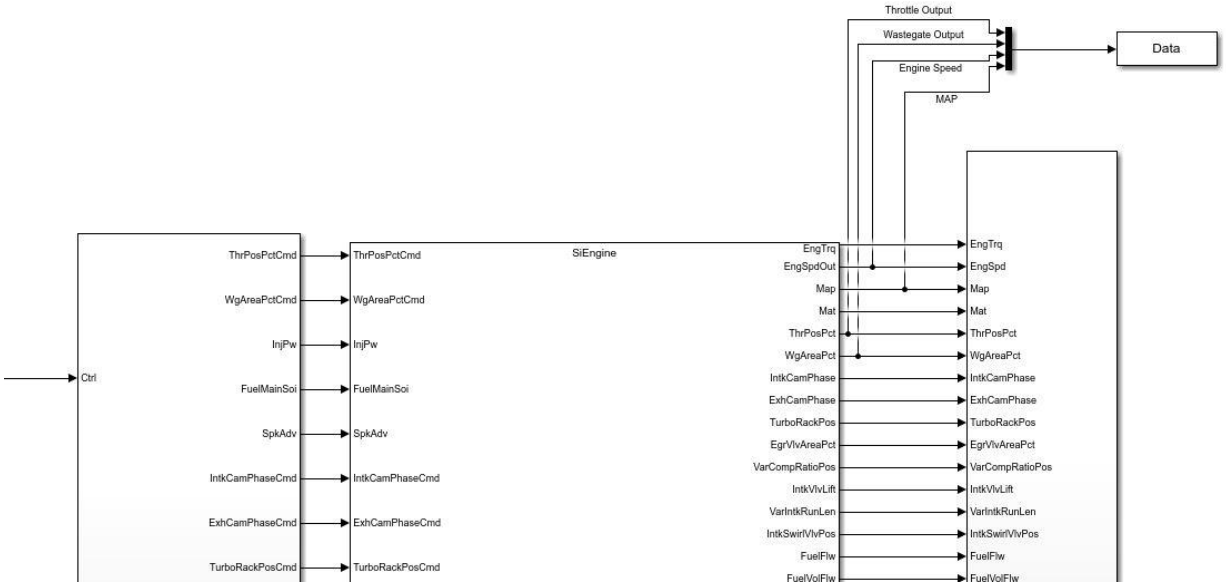

Figure 4: Data collection

- MPC Placement
  - Navigate through SIDynoReferenceApplication → Engine System → Engine Controller → SIEngineController
  - The MPC system will involve two separate sets of MPCs working together. One will use state-space models relating throttle-MAP, and the other will use state-space models relating wastegate-MAP. For lower MAP commands, only the throttle MPC will be needed. For higher MAP commands (over around 80 kPa), the wastegate MPC will be needed in addition.

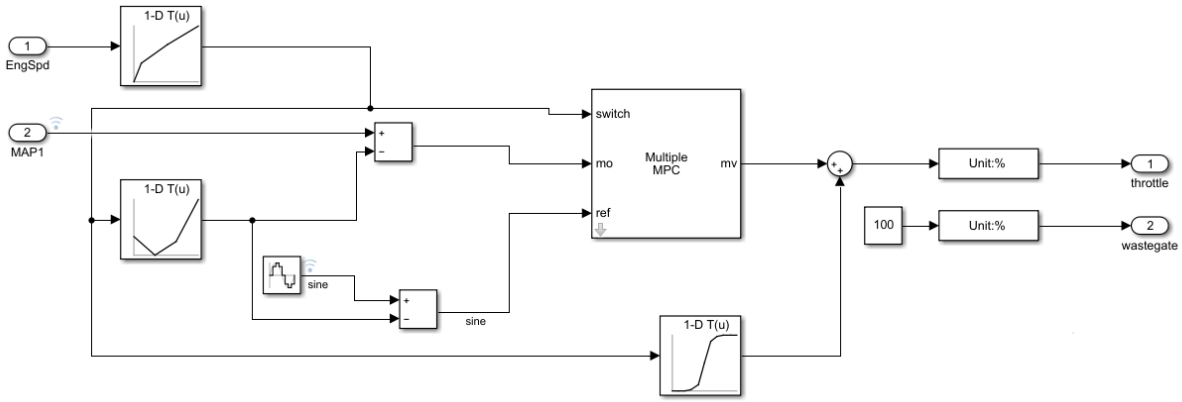

Figure 5: MPC Block Implementation

- Generating Models to be used for MPC
    - To create state space models and mpc objects to be placed into an MPC block, download the zip folder CreateDiffMPC. Open the RunMe file and the GenerateMPCDesigns files.
        - In RunMe.m, values can be manipulated to change the tested RPM, throttle value, and throttle manipulation and wastegate value and wastegate manipulation.
        - Do not make any changes to the GenerateMPCDesigns.m script. The only changes to be made would be when the data is unpackaged and set equal to ‘u’ (this happens twice: once for identification and once for validation). The value after signals will be either (:,1) for mpc objects made from throttle data or (:,2) for mpc objects made from wastegate data. For these actuator to MAP models, an order of 2 for the n4sid created models is best.
    - 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
    - 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
        - MPC Toolbox and helpful links
            - Model Predictive Control Toolbox - MATLAB
- Differential MPC Setup

    - Note that Figure 5 is for differential actuator-MAP models (see MATLAB's 'detrend' function', which is used in the GenerateMPCDesigns). (Actuator means either throttle or wastegate). The RunMe produces differential models.
    - To create an MPC setup for the differential models, simply take the MidMap and MidActuator variables produced by running the RunMe file. Use a sum block to subtract the MidMap from the MAP command going into the 'ref' and 'mo' ports of the 'Multiple MPC' block, and then use a sum block to add the MidActuator to the value coming out of the 'mo' of the 'Multiple MPC' block.
- 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.
        - 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 SAVED
    - Create a new file that opens the directory of the modified dyno