Model Predictive Control of a Diesel Engine Air System

Rong Chen, Ph.D., MathWorks Development
Pete Maloney, MathWorks Consulting
Agenda

- Motivation for Model-Predictive Control In Diesel
- Diesel Engine Air System Control Problem
- Co-simulation Between Simulink and GT-POWER
- Model Predictive Control Design and Validation
  - Linear system identification
  - MPC design at each operating point
  - MPC scheduling
Motivation for Model Predictive Control Application

- Engine Manufacturers Must Meet Stringent Performance, Fuel Economy, and Emission Requirements

- Engines Are Being Down-Sized, Down-Speeded, and Boosted As a Result

- EGR Valve Controls NOx Emissions But Also Affects Fuel Economy and Performance

- Turbocharger Vanes Control Performance But Also Affects Fuel Economy and NOx

- Multi-Input/Multi-Output Control Is Needed!
Motivation for Model Predictive Control Application

- Model-Predictive Control Offers A Standard, Scalable Approach To M.I.M.O. Control Design

- Model-Predictive Control Examples in Diesel Application:
  - Diesel Air System Control – Our Example Today
  - Exhaust After-treatment Control
  - Fuel Control
Objective #1:

- Use “EGR position ($x_{EGR}$)” and “VGT position ($x_{VGT}$)” to control “Boost Pressure ($p_i$)” and “EGR Mass Flow ($W_{ci}$)” on their setpoints.
Diesel Engine Air System Control Problem

Objective #2

- The control system should perform well when engine is running at different operating points
  - The operating points are defined by “engine speed” and “torque demand”
  - The setpoints of “boost pressure” and “EGR mass flow” at each operating point are already optimized using Model Based Calibration Toolbox.
Co-simulation Between Simulink and GT-POWER

- The diesel engine model is a GT-POWER (Gamma Technologies) demo system using first principles

- GT-POWER model communicates with Simulink

- Co-simulation is used for
  - Generating data for system identification
  - Validating control design
Model Predictive Control (MPC) Design Workflow

Nonlinear Model → System Identification → Linear Model → MPC Design → MPC Controller → MPC Scheduling → Gain-scheduled MPC Controller
Generate Experiment Data

- At each operating point, the “EGR position”, “VGT position” and “Fuel Mass” inputs are perturbed with small random values.
- During the 200 second co-simulation, the “Boost Pressure” and “EGR Mass Flow” outputs are measured.
Identify Linear Model

- Based on the experiment data, a MIMO linear plant model is identified using the tools from the **System Identification Toolbox**
  - Pre-process i/o data (remove trends and outliers)
  - Determine the order of the plant model
  - Use “ssest” command to identify a MIMO state space LTI model
  - Use “compare” command to validate the model

- For example: at the following operating point: Speed = 2025 rpm and Torque = 160 Nm, a 4th order plant model is identified.
Validate Linear Model

- Another 200 second data set generated from co-simulation is used to validate the model. In this case, the linear model has a 88% fit.
MPC Design Workflow

Nonlinear Model → System Identification

Linear Model → MPC Design

MPC Controller → MPC Scheduling

Gain-scheduled MPC Controller
Why Use Model Predictive Controller

- Internal linear model predicts plant behavior in the future
- Handles constraints explicitly
- Provides both feedback and feed-forward control (e.g. treating “Fuel Mass” as a measured disturbance for feed-forward control)
Design MPC

- Design MPC controller using **Model Predictive Control Toolbox**
  - Process the plant model
    - Specify input/output types (manipulated variables, measured disturbances, measured outputs, etc.)
    - Scale inputs/outputs
    - Set the nominal values of inputs and outputs
    - Augment the plant model to reject unmeasured disturbances
  - Design the controller
    - Specify sampling time
    - Set prediction and control horizons
    - Specify hard and soft constraints
    - Tune weights
    - Adjust estimator gain
- Design MPC at command line or in the MPC Design app
Validate MPC (Using Linear Plant)

- Validate MPC tracking performance using linear model in MATLAB

At the operating point of Speed = 2025 rpm and Torque = 160 Nm:
Nonlinear GT-POWER Co-Simulation

- Validate MPC tracking performance using co-simulation

10% step change of boost pressure and EGR flow at 6 sec and 11 sec respectively

Boost pressure (kPa)  EGR mass flow (g/s)
MPC Design Workflow

1. Nonlinear Model
2. System Identification
3. Linear Model
4. MPC Design
5. MPC Controller
6. MPC Scheduling
7. Gain-scheduled MPC Controller
Define Operating Point Grid

- Scheduling is a popular way to solve a nonlinear control problem with linear controllers

- Nine operating points are defined for the diesel engine air system

<table>
<thead>
<tr>
<th></th>
<th>Low Torque (82 Nm)</th>
<th>Medium Torque (160 Nm)</th>
<th>High Torque (238 Nm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Speed (1290 rpm)</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Medium Speed (2025 rpm)</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>High Speed (2760 rpm)</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

- An MPC is designed and validated at each operating point
Build MPC Control System in Simulink
Configure Multiple MPC Controllers Block

You can use the Multiple MPC Controllers block to design and simulate a set of model predictive controllers. These controllers can be switched from one to another in real time to control a nonlinear plant with a wide operating range.

The scheduling signal connected to the switch input must be a scalar between 1 and N, where N is the number of MPC controllers being used.

While only the controller in action computes the optimal move, all controllers update their internal state observers to prepare for bumpless transitions.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>MPC Object</th>
<th>Initial States</th>
<th>Delete it?</th>
<th>Design it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPC #1</td>
<td>mPC0a1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPC #2</td>
<td>mPC0a2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPC #3</td>
<td>mPC0a3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPC #4</td>
<td>mPC0a4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Number of MPC controllers: 9

- Add
- Delete
- Design

### Optional Inputs

- Measured disturbance
- Input and output limits
- Externally supplied MV signal

### Signal Attributes and Block Sample Time

- Output data type: double
- Block uses inherited sample time (-1)

[Diagram of Multiple MPC Controllers block with parameters and connections]
Design MPC Scheduling Signal

<table>
<thead>
<tr>
<th>Speed Condition</th>
<th>MPC #1</th>
<th>MPC #2</th>
<th>MPC #3</th>
<th>MPC #4</th>
<th>MPC #5</th>
<th>MPC #6</th>
<th>MPC #7</th>
<th>MPC #8</th>
<th>MPC #9</th>
</tr>
</thead>
<tbody>
<tr>
<td>TQ &lt; 120 Nm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120 Nm &lt;= TQ &lt;= 200 Nm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 Nm &lt; TQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Speed < 1600 rpm**
  - **MPC #1**
  - **MPC #2**
  - **MPC #3**

- **1600 <= Speed <= 2400**
  - **MPC #4**
  - **MPC #5**
  - **MPC #6**

- **2400 rpm < Speed**
  - **MPC #7**
  - **MPC #8**
  - **MPC #9**

```
function signal = fcn(speed, torque)
    %#codegen
    if speed<1600
        if torque<120
            signal = 1;
        elseif torque<200
            signal = 2;
        else
            signal = 3;
        end
    elseif speed<=2400
        if torque<120
            signal = 4;
        elseif torque<200
            signal = 5;
        else
            signal = 6;
        end
    else
        if torque<120
            signal = 7;
        elseif torque<200
            signal = 8;
        else
            signal = 9;
        end
    end
end
```
Validate MPC Scheduling with GT-POWER (1)

- Engine speed: ramp from 1200 rpm to 2800 rpm in 5 seconds
- Torque demand: ramp from 70 Nm to 170 Nm in 5 seconds

<table>
<thead>
<tr>
<th>Speed</th>
<th>Low Torque (82 Nm)</th>
<th>Medium Torque (160 Nm)</th>
<th>High Torque (238 Nm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Speed (1290 rpm)</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Medium Speed (2025 rpm)</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>High Speed (2760 rpm)</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
Validate MPC Scheduling with GT-POWER (2)

Boost pressure tracking

EGR mass flow tracking
Validate MPC Scheduling with GT-POWER (3)

Manipulated Variables:

EGR Lift in (mm, in yellow)

VGT position (ratio, in magenta)
Code-Generation and Deployment

- After we validate the MPC control system with GT-POWER co-simulation, the next step is to deploy the controller to embedded system and test it against a real engine.

- MPC controller supports C code generation with Simulink Coder, Embedded Coder and other targets in single or double precisions.

- The MPC sample time can be as low as 1 millisecond for a typical use case. It does demand larger memory footprint compared with other types of controllers such as PID and LQR.