Trip Optimizer
Development of a Driver Assistance System for Locomotives Using MATLAB

8 May 2017

MathWorks Automotive Conference | May 9th 2017
GE Proprietary
Agenda

1. Trip Optimizer Overview
2. Trip Optimizer and MATLAB
3. Integrating an external optimization library into MATLAB Code Generation Toolset
GE’s Trip Optimizer
“Fuel conscious cruise control for trains”

Train and driver variations result in:
• Less than optimal fuel use
• High emissions
• Trip variations
• Wear and tear

Trip Optimizer:
• Looks over the entire route for fuel savings opportunities
• Then controls the throttle to the plan
  – Saves fuel
  – Reduces emissions
  – Reduces equipment wear and tear
  – Consistent trips improve scheduling
Trip Optimizer Deployment/Operation

- 8,000 systems installed worldwide
- 60,000 miles of mapped track
- 216M miles of auto operation
- 73M miles of auto in 2016
- 1.7M auto miles per week
- 142,000 gallons of diesel fuel saved per week
Trip Optimizer and MATLAB
Converging Technologies

• Research and development into Trip Optimizer began close to the time MathWorks began rolling out automatic code generation from Simulink
• Trip Optimizer team leveraged this technology to quickly produce proof of concept simulations
• Automatic code generation for embedded targets allowed accelerated transition from simulation environment to on locomotive demos
• Simulink and MATLAB now embedded in the core elements of the product
Trip Optimizer Block Diagram

Customer & GE IT Systems

- GE Offboard

Train & Trip Information

Auxiliary Processing Node

- External Interface & Track Database
- Planner

Route & Plan Info

Control System Node

- HMI
- Core Loco Control
- Supervisor
- Location Determination
- Speed Regulator, Estimation, & Train Handling

GPS

Speed

Master Control

GE Proprietary

Trainlines

Throttle

GE Proprietary
Integrating an External Optimization Library into the MATLAB Code Generation Toolset

<table>
<thead>
<tr>
<th>Iter</th>
<th>Objective</th>
<th>Inf_pF</th>
<th>Inf_du</th>
<th>lg(Mu)</th>
<th>Primal</th>
<th>lg(Fy)</th>
<th>alpha_du</th>
<th>alpha_pF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.6109693e+01</td>
<td>1.12e+01</td>
<td>5.28e-01</td>
<td>0.0</td>
<td>0.00e+00</td>
<td>-0.00e+00</td>
<td>0.00e+00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.7410406e+01</td>
<td>7.49e-01</td>
<td>2.25e+01</td>
<td>-0.3</td>
<td>7.97e-01</td>
<td>-3.19e-01</td>
<td>1.00e+00</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.8001613e+01</td>
<td>7.52e-03</td>
<td>4.96e+00</td>
<td>-0.3</td>
<td>5.60e-02</td>
<td>2.0</td>
<td>9.97e-01</td>
<td>1.00e+00</td>
</tr>
<tr>
<td>3</td>
<td>1.71711</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.696</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.7003411e+01</td>
<td>2.16e-02</td>
<td>8.42e-03</td>
<td>-2.9</td>
<td>7.03e-02</td>
<td>-9.68e-01</td>
<td>1.00e+00</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.7013974e+01</td>
<td>2.03e-04</td>
<td>8.65e-05</td>
<td>-4.5</td>
<td>6.22e-03</td>
<td>-1.00e+00</td>
<td>1.00e+00</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.7028104e+01</td>
<td>2.76e-07</td>
<td>2.18e-07</td>
<td>-10.3</td>
<td>1.43e-04</td>
<td>-9.99e-01</td>
<td>1.00e+00</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.7014017e+01</td>
<td>2.13e-04</td>
<td>2.29e-04</td>
<td>-11.0</td>
<td>1.04e-07</td>
<td>-1.00e+00</td>
<td>1.00e+00</td>
<td></td>
</tr>
</tbody>
</table>

Number of Iterations: 8

<table>
<thead>
<tr>
<th></th>
<th>&lt;scaled&gt;</th>
<th>&lt;unscaled&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective</td>
<td>1.7014017140224134e+01</td>
<td>1.7014017140224134e+01</td>
</tr>
<tr>
<td>Dual infeasibility</td>
<td>2.2928101314633036e-014</td>
<td>2.2928101314633036e-014</td>
</tr>
<tr>
<td>Constraint violation</td>
<td>2.1316282072803006e-014</td>
<td>2.1316282072803006e-014</td>
</tr>
<tr>
<td>Complementarity</td>
<td>1.0023967333275279e-011</td>
<td>1.0023967333275279e-011</td>
</tr>
<tr>
<td>Overall NLP error</td>
<td>1.0023967333275279e-011</td>
<td>1.0023967333275279e-011</td>
</tr>
</tbody>
</table>
Trip Optimizer Planner

- Fuel optimal plan generated for entire route at time of trip initialization
- Trip plan adjusted to account for changes in conditions along the route
- Plan speed is reference for speed regulator
- Plan throttle is 100% feed forward term on speed regulator output

Diagram:

- Planner
- Regulator
- Throttle Command
- Train Speed
- Plan Speed
- Plan Throttle
- Grade
- Speed Limits
- Car Weights/Lengths
- Locomotive Characteristics
Why Integration?

- **Pre-Process**
  - m-code
  - AutoCode_EntryPoint_1()

- **Initial Guess**
  - m-code
  - AutoCode_EntryPoint_2()

- **Build Inputs**
  - m-code
  - Invoke external executable

- **Transform Output**
  - m-code
  - Transform Output
  - Hand Code

- **Post-Process**
  - m-code
  - AutoCode_EntryPoint_3()

**Single interface point with hand-code**

- **Do It**
  - m-code
  - AutoCode_EntryPoint()

**Desktop environment equivalent to embedded target**

Trip Optimizer: Development of a Driver Assistance System for Locomotives Using MATLAB

8 May 2017

GE Proprietary
The Optimization Library C++ Interface

```
SmartPtr<ProblemDefinition> my_problem = new MyProblemDefinition();
SmartPtr<OptimizationApplication> app = ApplicationFactory();
app->Initialize();
app->Optimize(my_problem);
```
How We Did It – ceval + minGW + addLinkObjects

```matlab
function [ProfileOut, errCode, cpuTime, nites] = CallOpt_ceval(OptSpec, Mesh, espec, ProfileOut)

    [OptData, Constants] = PopulateInitData_ceval(OptSpec, Mesh, espec, ProfileOut);

    x = zeros(1, OptData.nvars);
    nites = int32(0);
    errCode = int32(0);
    cpuTime = double(0);

    coder.cinclude('ceval_optimizer.h');
    coder.ceval('ceval_optimizer', coder.rref(OptData), coder.wref(x), coder.wref(errCode), coder.wref(nites), coder.wref(cpuTime));

    ProfileOut = Post_Process_ceval(Mesh, Constants, x, ProfileOut);
end
```

coder.updateBuildInfo('addLinkObjects','liboptimizer.a',icoptLibPath,'',true,true);
coder.updateBuildInfo('addLinkObjects','libfortran.a',mingwLibPath,'',true,true);
coder.updateBuildInfo('addLinkObjects','libquadmath.a',mingwLibPath,'',true,true);
coder.updateBuildInfo('addLinkObjects','libstdc++.a',mingwLibPath,'',true,true);
Algorithm development

Most robust debugging environment – breakpoints, figure, variable inspection

Logic errors
Example: Air brake effort is negative, I should be using max() to limit it

Code generation
Identify unsupported functions

Code flow errors
Example: Can’t assign a field to a structure after its been accessed

Memory allocation errors
Example: myvar = 1; myvar(2) = 2; Works in MATLAB, doesn’t work in generated code

Gross performance measures
Example: I could do this interpolation one time and pass it around instead of doing it over and over

Compare code outputs from target and MATLAB environment

Defects not expected
Each time the model is run on the target all inputs are written to a text file which can be read into MATLAB to recreate the scenario exactly

Failures from field can be brought back to MATLAB environment for debugging/algorithm enhancement

Planner - Improved Development Lifecycle

• Each time the model is run on the target all inputs are written to a text file which can be read into MATLAB to recreate the scenario exactly
• Failures from field can be brought back to MATLAB environment for debugging/algorithm enhancement
Conclusions

• Integrating external code into code generation process can enable parity between MATLAB development environment and embedded execution target
• Increased productivity – Development and debugging
• Defects found earlier in life cycle