A modular framework for modeling hardware elements in distributed engine control systems

Alicia M. Zinnecker
N&R Engineering

Dennis E. Culley
Eliot D. Aretskin-Hariton
NASA Glenn Research Center

Intelligent Control and Autonomy Branch
NASA Glenn Research Center

AIAA Propulsion and Energy 2014
July 28, 2014
Outline

1. Introduction

2. Modeling Framework
   - Baseline controller model
   - Distributed controller model
   - Summary of this modeling approach

3. Demonstrating the framework through simulation

4. Summary and Conclusion
Introduction

- Distributed control architecture has been slow to transition into aerospace applications because *challenges* perceived to outweigh *benefits*

**Benefits**
- Computational effort spread across the control system
- Engine control unit (ECU) not responsible for input/output conditioning
- **Digital** network replaces **analog** wiring, reducing complexity and weight of connections
- **Modularity** allows for easy replacement, upgrading, or maintenance of parts

**Challenges**
- Electronics needed to withstand **harsh engine environment**
- Specification and testing of **reliable controller network** must be done
- Collaboration to advance technology must **protect intellectual properties** of participants
- Testing of new hardware, control architectures is **limited** within present design process
Introduction

- A hardware-in-the-loop (HIL) system is under development at NASA that will allow for testing hardware models and prototypes in various control configurations without the need for a physical engine.
  - Control and engine design can proceed in parallel.
  - Lowers the cost for hardware, controller testing.
  - Simulation of conditions too extreme for test cells.
  - Requires high-fidelity hardware and network models so simulations accurately represent tests on actual hardware.

- Interfaces between elements of the control system, important in distributed architectures, can be leveraged to develop a modeling framework.
Baseline controller model

- Development of the system is around a baseline model: C-MAPSS40k (the ‘unstructured’ model)
  - Commercial Modular Aero-Propulsion System Simulation, 40,000 lb$_f$-thrust
  - Zero-dimensional simulation of a twin-spool turbofan engine
  - Controller contains simple sensor and actuator models along with setpoint controller and limiters

- Structure introduced, defining clear separation between engine and controller models
Baseline controller model

- Two sets of interfaces exist in this baseline system
  - Between controller, engine and wrapper models
  - Within controller model
  - May define a third interface: Connections between components on individual sensors, actuators
Distributed controller model includes **data conditioning, conversion, and processing** on the sensors and actuators, and a **controller network**.

Higher fidelity computational models expected to **more closely match** results from tests with **real** hardware communicating over a **real** network.
Distributed controller model

- Network represents **physical decoupling** of sensors, actuators, and the controller in an engine controller system

- **Data transfer effects** need to be modeled to understand how these affect reliability and performance of closed-loop system

- Presently modeled as a **delay** and **packet loss** (stochastically)

- If higher fidelity is required, packet-level models may be constructed
Distributed controller model

- **Smart transducers** contain sensor or actuator hardware with local data conditioning and processing functionality
  - Simulink® library under development containing building blocks for modularly creating models of smart transducers
  - Library follows the **IEEE 1451 standard** for smart transducers
    - **Smart Transducer Interface Module (STIM)** contains transducer, signal conditioning and conversion hardware (analog signals)
    - **Network Capable Application Processor (NCAP)** contains microprocessor and network adapter (digital signals)
    - **Transducer Electronic Data Sheet (TEDS)**, stored on STIM, contains calibration and manufacturer information
Summary of this approach

- **Modularity** imposed at each level of the framework:
  1. Between controller, engine, and wrapper models
  2. Between control hardware and control algorithm
  3. Within each smart transducer
Imposing this framework on C-MAPSS40k

To demonstrate how framework affects simulation results, the C-MAPSS40k controller model was modified to follow it

- **Replace sensor models** with smart sensor models (sensor, signal conditioning filter, analog-to-digital conversion and averaging blocks from Smart Sensor Library)
- **Replace actuator models** with smart actuator models (extrapolation, digital-to-analog conversion, signal conditioning filter, and actuator library blocks)
- **Add feedback sensors** for local loop closure on two actuators
- Place **network block** on output of each sensor, input of each actuator

**Three models** considered for comparison

1. **Unstructured** model (baseline C-MAPSS40k controller)
2. **Distributed** model (smart transducer models, no network)
3. **Networked** model (smart transducer and network models)
Imposing this framework on C-MAPSS40k

- Sensors and actuator configured using information from C-MAPSS40k for bandwidths, ranges; generic data sheets for conditioning, processing components

- Network model configured to **exaggerate** time delay, packet loss probability to better demonstrate effects of element

<table>
<thead>
<tr>
<th>Sensor model configuration</th>
<th>Network cable model configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor input range (psi)</td>
<td>Average delay (s)</td>
</tr>
<tr>
<td>Sensor output range (V)</td>
<td>Delay standard deviation (s)</td>
</tr>
<tr>
<td>Sensor rise time (s)</td>
<td>Packet-drop probability (%)</td>
</tr>
<tr>
<td>ADC range (V)</td>
<td></td>
</tr>
<tr>
<td>ADC resolution (bits)</td>
<td></td>
</tr>
<tr>
<td>Averaging window (sample)</td>
<td></td>
</tr>
</tbody>
</table>
Imposing this framework on C-MAPSS40k

- Controller model further modified to allow for **multiple update rates** within simulation
  - **Baseline** model updates at a **fixed time-step** equal to the controller update rate
  - In **physical system**, each element operates **asynchronously** at its **own rate**
  - **Different (fixed) update rates** assigned to sensors, actuators, control law to improve realism of model

- Model can be viewed as collection of functions **accessing network at different rates**

![Diagram](image-url)
Comparing simulation results

- Provided a 60-second multi-step throttle command
- Tracking and thrust responses not significantly different, despite more-detailed hardware models, presence of a network model
Comparing simulation results

- Biggest difference between simulation results seen by comparing outputs of actuators and sensors (here, fuel flow actuator and $P_{50}$ sensor)
- Exaggerated network model does not have much effect on results
In addition to comparing simulation results, it is also important to verify that **real-time simulation** is possible.

Each model simulated 200 times, recording total run time.

- Variations likely due to processor demands during simulation.
- **Increased average time** for distributed and networked models due to added complexity.
- On average, distributed (3.06 times) and networked (2.35 times) models run **faster than real-time**, suggesting the model may be run with hardware in the loop.
Summary and Conclusion

- Framework presented for developing models for hardware-in-the-loop systems, based on interfaces present in the system:
  - Between engine, controller, and user input source
  - Between control hardware and control law (over a network)
  - Within each individual piece of hardware
- Approach introduces modularity, enabling independent development of control algorithm, sensor, actuator, and engine models compatible with framework
- Simulink library, based on the IEEE 1451 framework, simplifies creation of smart transducer hardware models
Summary and Conclusion

**Trade-offs** of this design choice must be weighed

**Benefits**

- Decoupled systems enable **collaboration, independent development** of models
- Protection of intellectual property by using **compiled code** in place of Simulink library blocks
- Use of Simulink library allows similar models with **varying fidelity** to be developed, interchanged easily

**Drawbacks**

- **Limited flexibility** of independent development at higher levels
- Models may relay information unnecessary for control algorithm, but needed for analysis, **adding complexity**
- More accurate models **increase computational cost**, real-time operation no longer guaranteed
- At this time, hardware and network models **not yet validated**, so simulations only act as proof-of-concept
Summary and Conclusion

- Simulation of C-MAPSS40k using this framework shows quantization effects in tracking
  - Overall results otherwise differ little from baseline
  - Simulation (on average) runs faster than real-time

- Future investigation may involve:
  - Validation of network model against physical network
  - Testing of framework in simulation with hardware in loop to verify accuracy of models in predicting actual system behavior
Thanks.
Questions?
References


References


