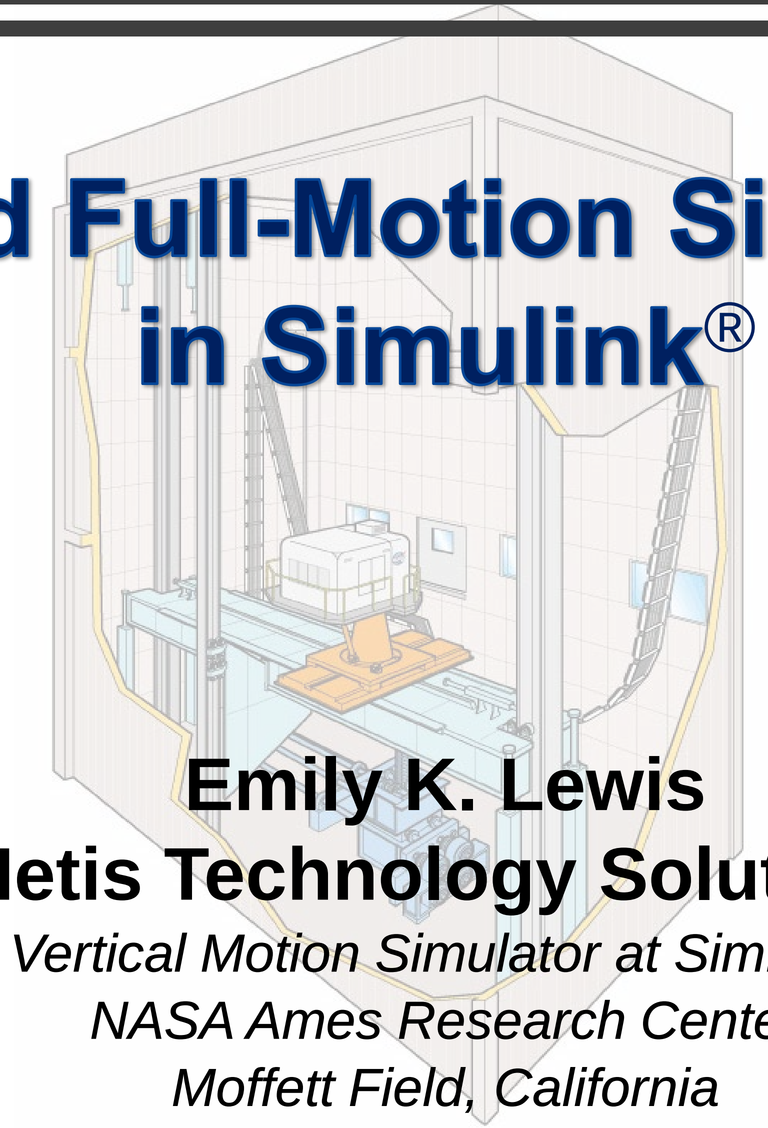


Piloted Full-Motion Simulation in Simulink®

Emily K. Lewis

Metis Technology Solutions

Vertical Motion Simulator at SimLabs

NASA Ames Research Center

Moffett Field, California

AIAA Science and Technology Forum, January 7 – 11, 2019, San Diego

Overview

- Background/Motivation
- Vertical Motion Simulator (VMS)
- Experiment Description
- Challenges
- Summary

Overview

- Background/Motivation
- Vertical Motion Simulator (VMS)
- Experiment Description
- Challenges
- Summary

Background

Traditionally, VMS simulation engineers programmed all math model changes.

- Error-prone and time consuming

Over past decade several VMS simulations integrated Simulink models.

- Standard practice: convert to C code using MATLAB's Simulink Coder®/Real-Time Workshop® (RTW)
- The conversion and integration process can be time consuming and cumbersome

VMS requires real-time.

- Simulink is not conducive to hard real-time

Motivation

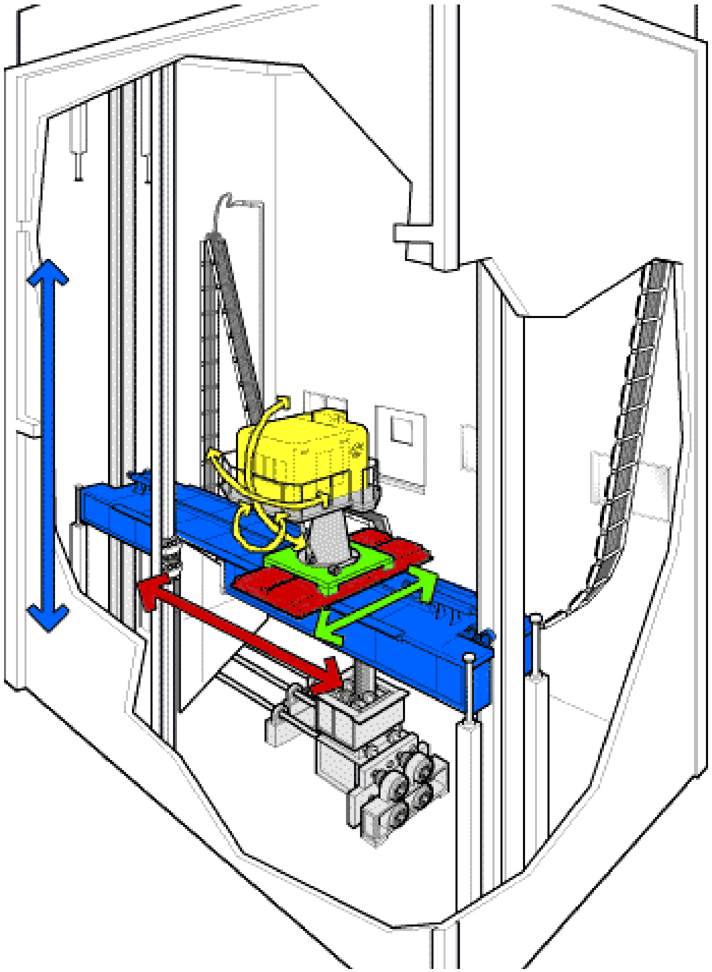
Advantages of running Simulink models in MATLAB® environment at VMS			
Greater efficiency in continued model development	Improved confidence with model integrity	Some Simulink blocks are not supported by Simulink Coder®	Increases VMS flexibility

Overview

- Background/Motivation
- Vertical Motion Simulator (VMS)
- Experiment Description
- Challenges
- Summary

Vertical Motion Simulator

- Six degree of freedom flight simulator
- Designed to provide high-fidelity motion for realistic pilot cueing
- Large displacement
  - ±30 ft vertical
  - ±20 ft lateral
  - ±4 ft longitudinal
- 1.0 g vertical acceleration capability
- Delivers high quality research data that translates well to flight



Vertical Motion Simulator

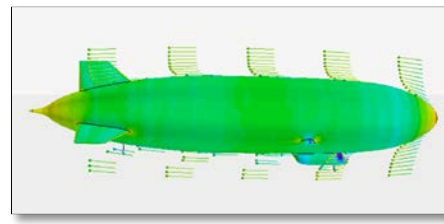
Simulated wide range of aerospace vehicles

F/A-18

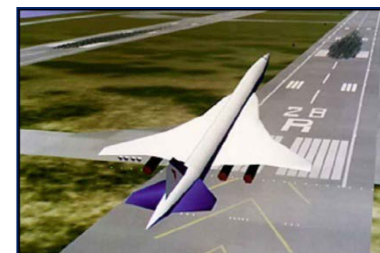
CH-47

Lunar Lander

FVL Coax Helicopter



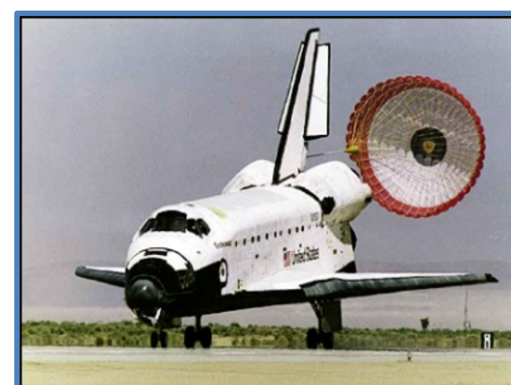
Airships



High Speed Civil Transport

UH-60

Large Civil Tilt Rotor



Space Shuttle

VMS System Overview

Host

Host Environment

- Real-time scheduling
- User interface
- Debugger

Executable Image

- Vehicle model
- Model interface
- Cab interface
- Display drivers
- Visual drivers
- Motion drivers

External Processors

- Image generator
- Flight instrument graphics
- Lab engineering display graphics
- Loader digital controller
- Motion control unit

Cab

- Visual scene monitors
- Instrument displays
- Pilot controls
- Motion platform

Lab

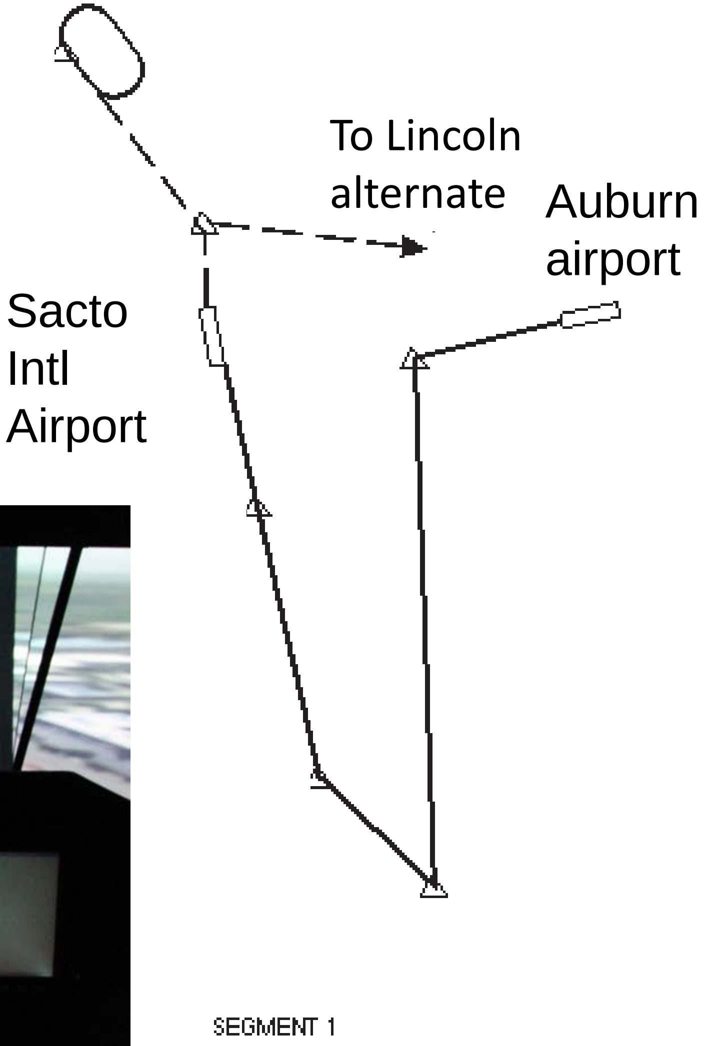
- Third person view
- Engineering displays
- Data collection

Overview

- Background/Motivation
- Vertical Motion Simulator (VMS)
- Experiment Description
- Challenges
- Summary

Experiment Overview

Objective: Evaluate Advanced Flight Control Systems (AFCS) in a real-world, high-workload environment with realistic, full-mission scenarios

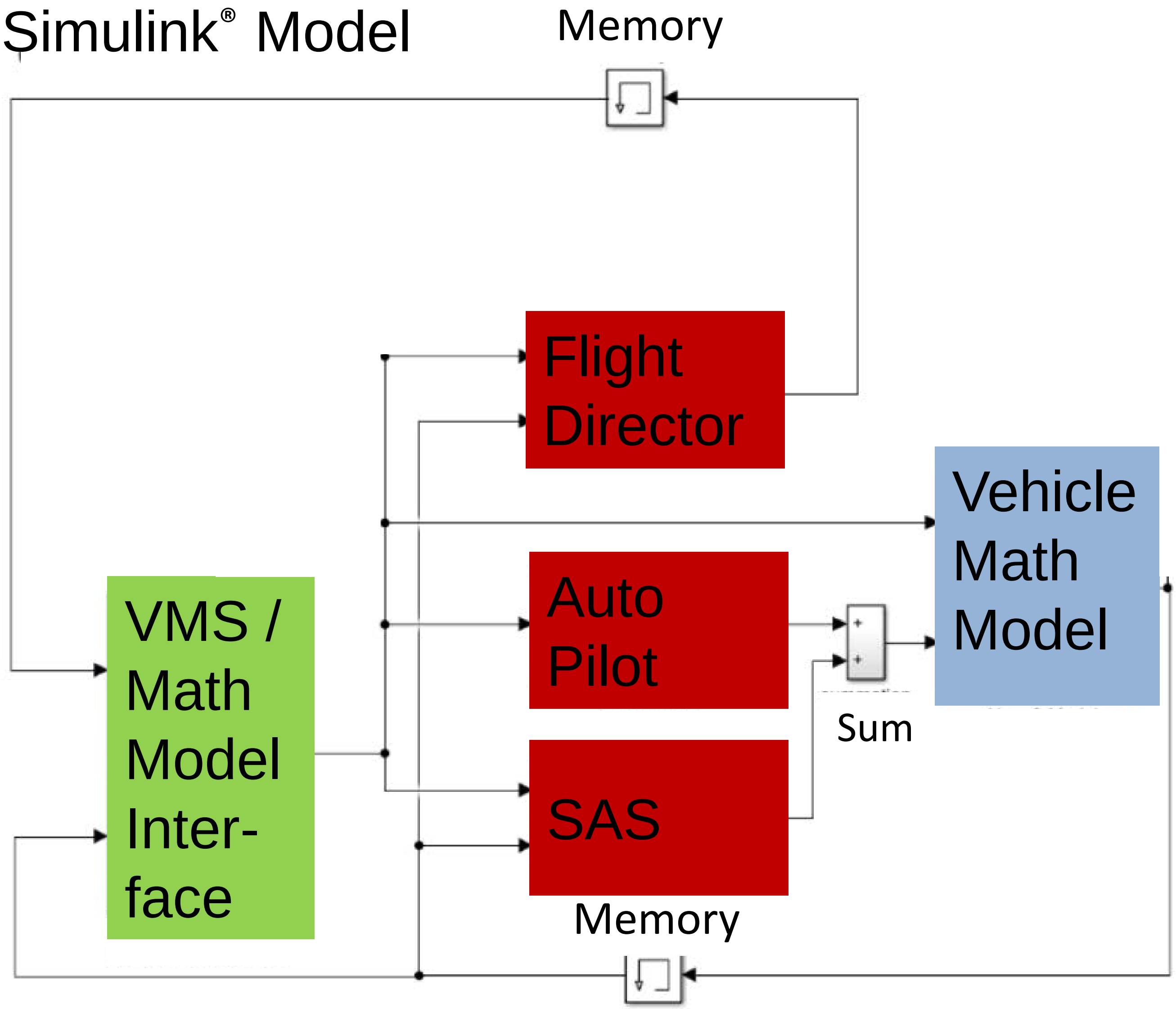


Experiment Components

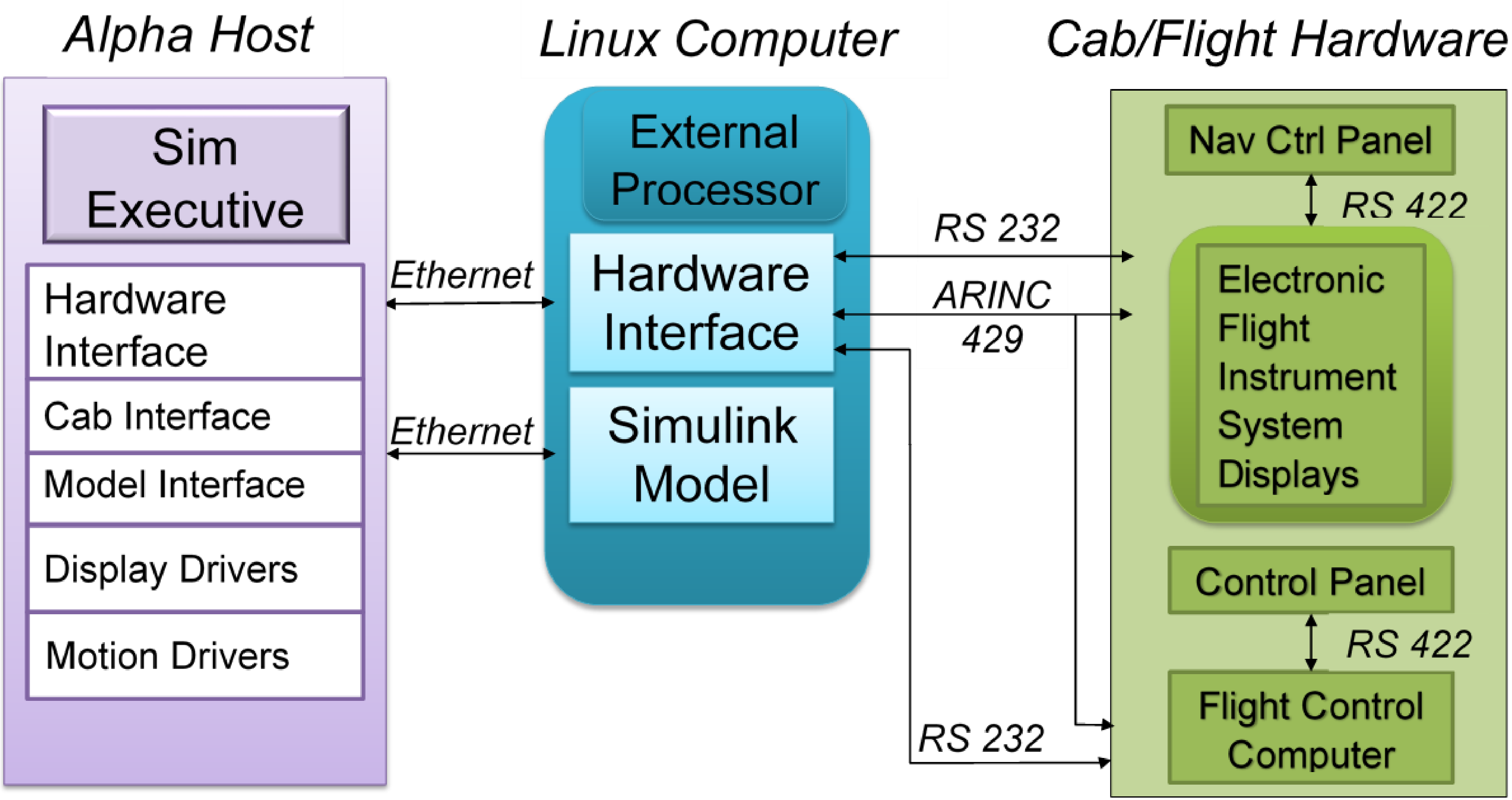
- Simulink® model of a mature helicopter simulation
  - Included Stability Augmentation Systems (SAS), auto-pilot, and flight director guidance
  - Ran in MATLAB® environment on an external device
- Flight hardware-in-the-loop
  - Electronic Flight Instrument System (EFIS) displays
  - Realistic control and pilot communication devices
- Special-to-VMS communication requirements
  - New-to-VMS data-transmission methods
  - New interface software

Math Model

Simplified Simulink® Model



Hardware & Software Schematic



VMS System Overview

Host

Host Environment

- Real-Time scheduling
- User interface
- Debugger

Executable Image

- ~~Vehicle model~~
- Model interface
- Cab interface
- Display drivers
- OTW drivers
- Motion drivers

External Processors

- Image generator
- Flight instrument graphics
- Lab engineering display graphics
- Loader digital controller
- Motion control unit
- **Vehicle Model**
- **Vehicle Interface**
- **Hardware Interface**

Cab

- Visual scene monitors
- Instrument displays
- Pilot controls
- Motion platform
- **Flight Hardware**

Lab

- Third person view
- Engineering displays
- Data collection

Overview

- Background/Motivation
- Vertical Motion Simulator (VMS)
- Experiment Description
- **Challenges**
- Summary

Challenges

Initial concern:

- ❑ Simulink® model may not run in real time

Mitigation:

- ➢ I/O communication occurs on fixed frame boundaries
- ➢ Real-time hardware clock controls communication between Simulink® model and host
  - Simulink® model is initiated when data packet is received from the host at start of frame
  - Model response is read by host on frame boundaries
  - Restriction that stops the simulation if a frame overrun occurs was disabled

Challenges

Initial concern:

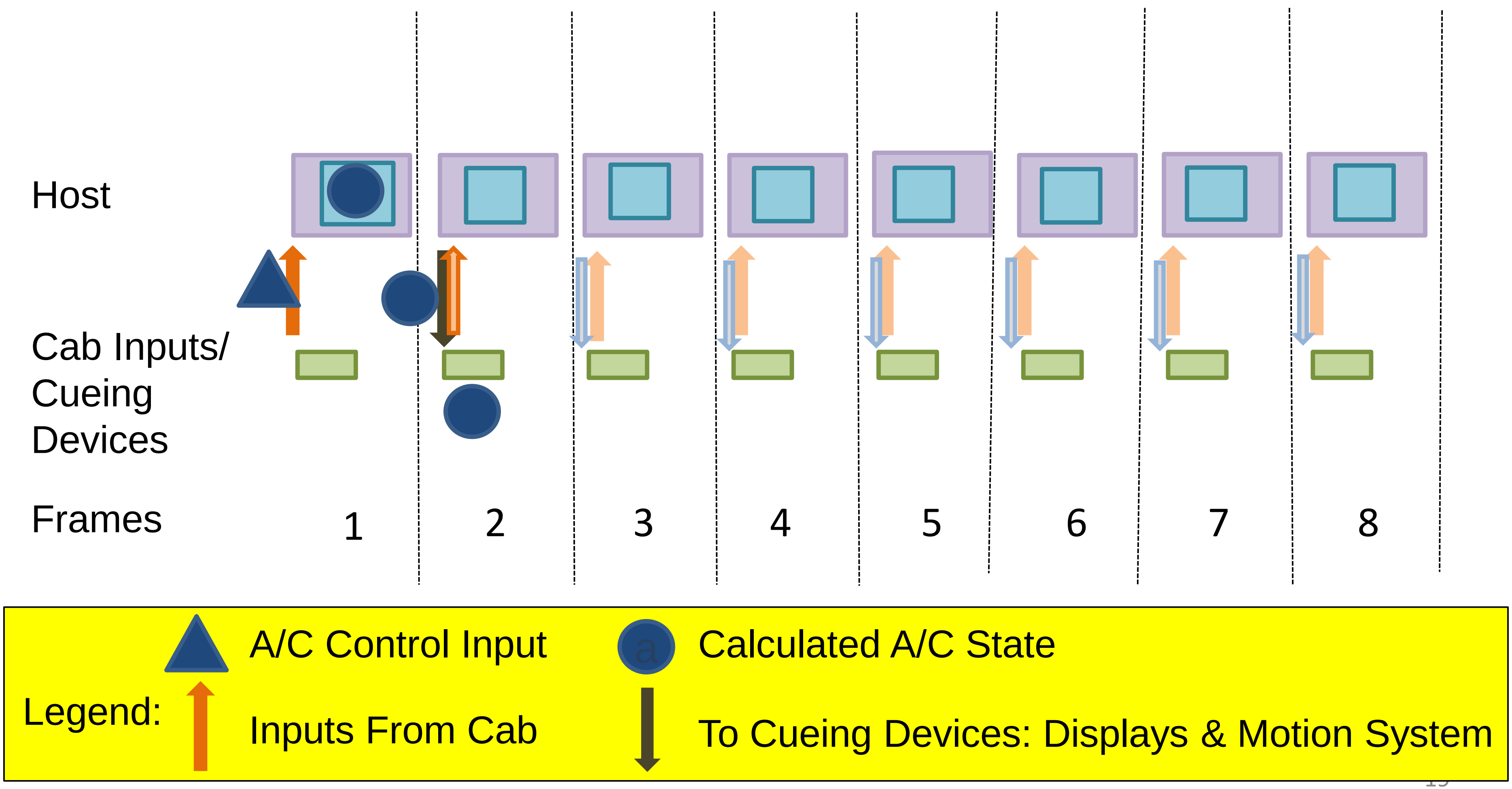
- ❑ Loss of synchronization could occur due to overhead on Linux box causing frame overruns.
  - Simulink model ran in the 1-9 ms range

Mitigation:

- ➢ Overruns reduced by
  - Minimizing the number of MATLAB® processes to reduce system overhead
  - Monitoring if overruns occurred

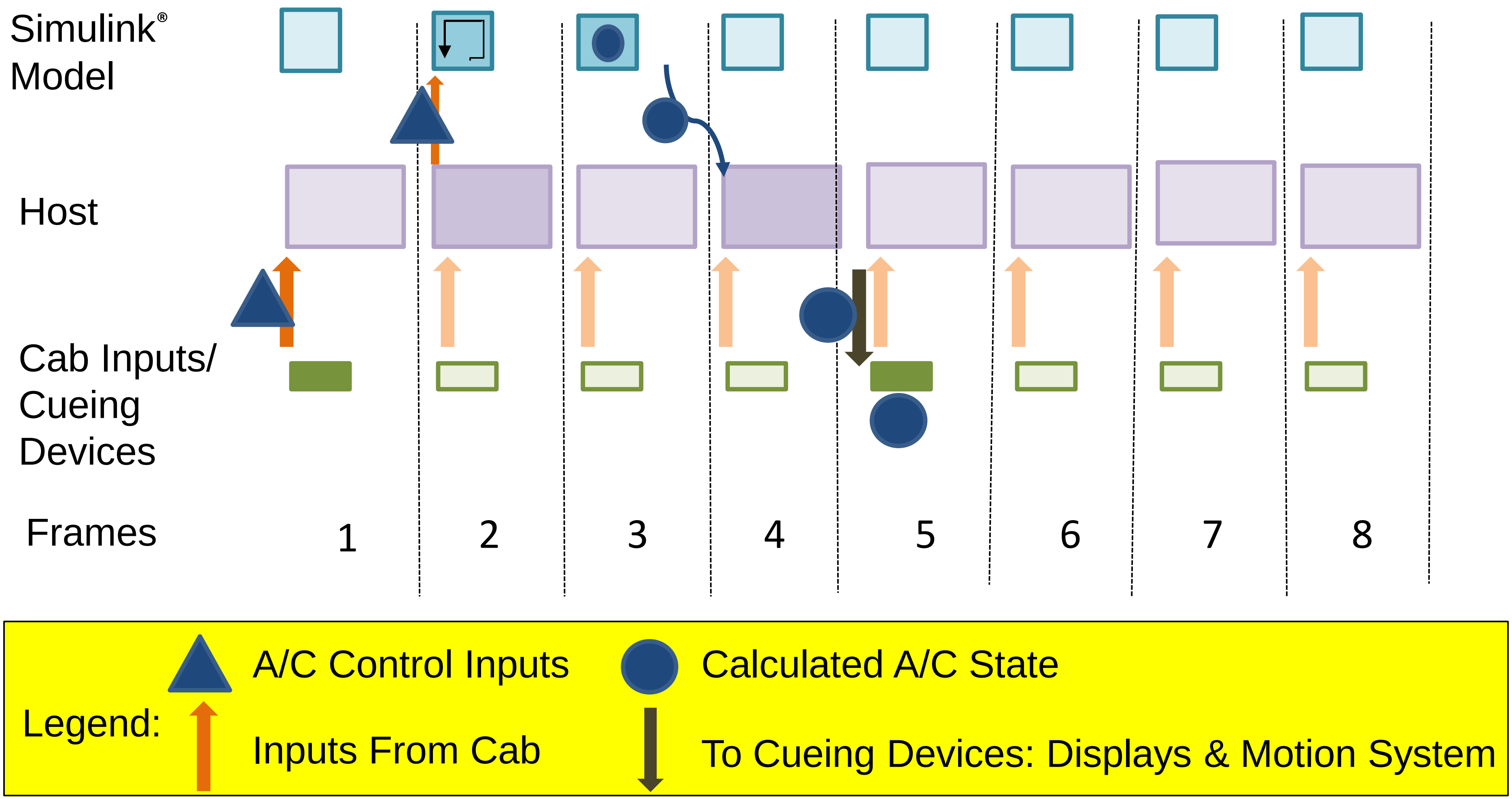
Execution Timeline

I/O Occurs on Hard Frame Boundaries



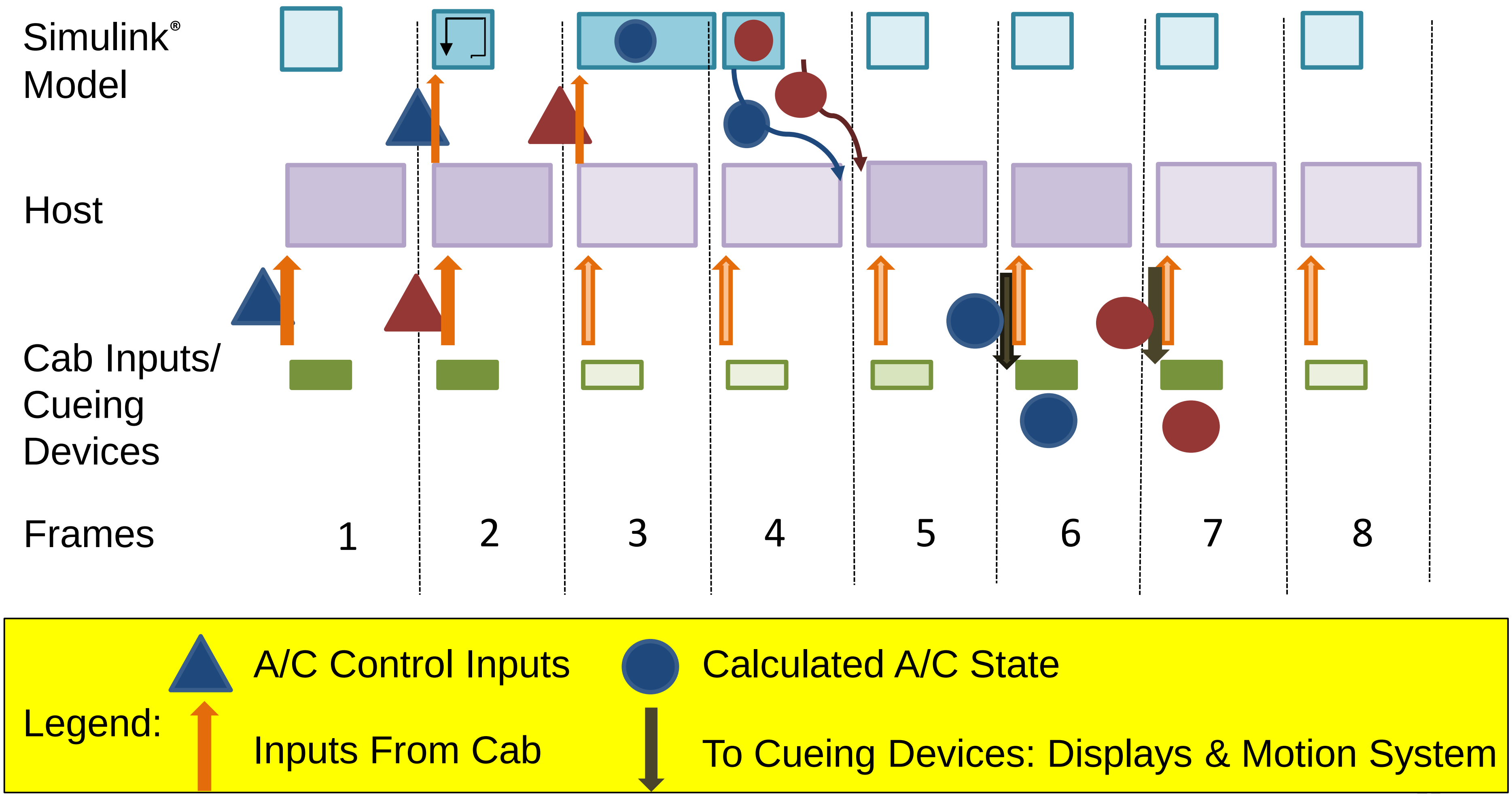
Delays

Three Frames of Delay Expected



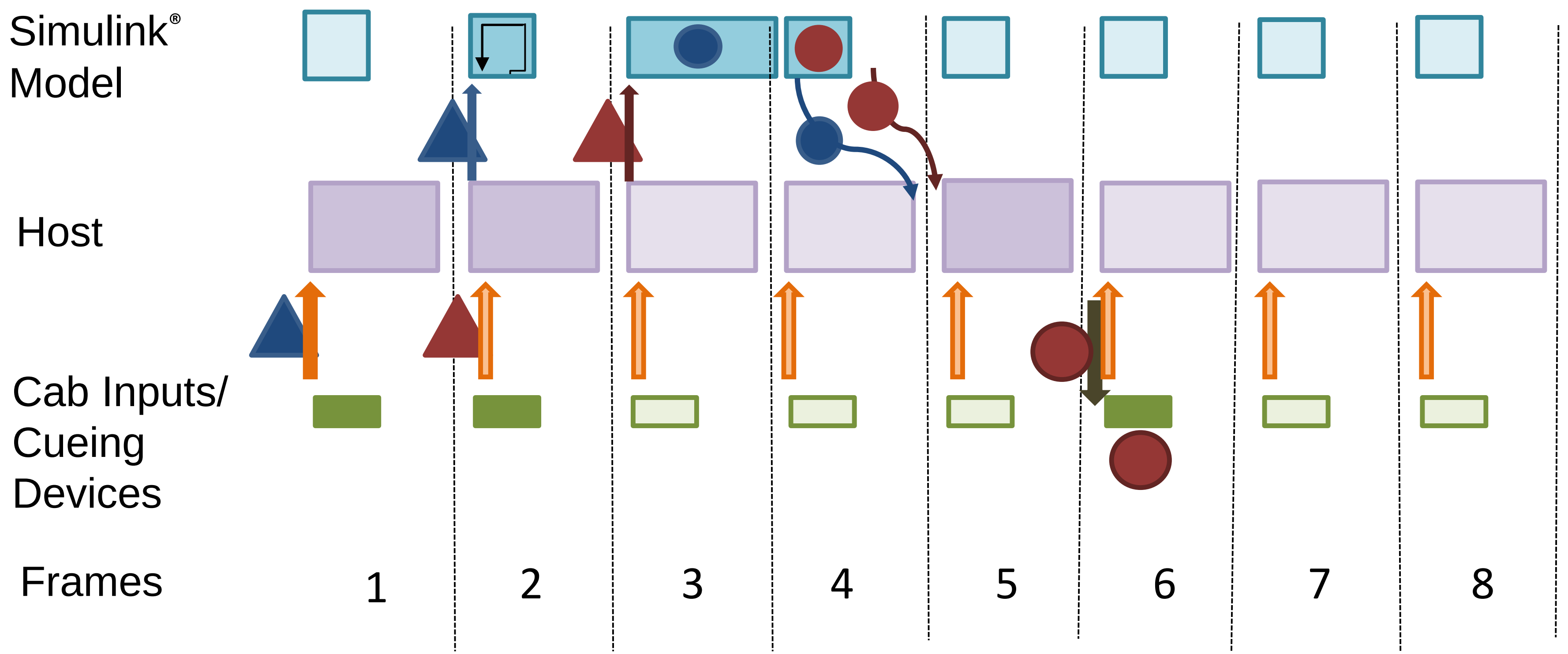
Delays

Frame Overrun



Delays

Frame Overrun

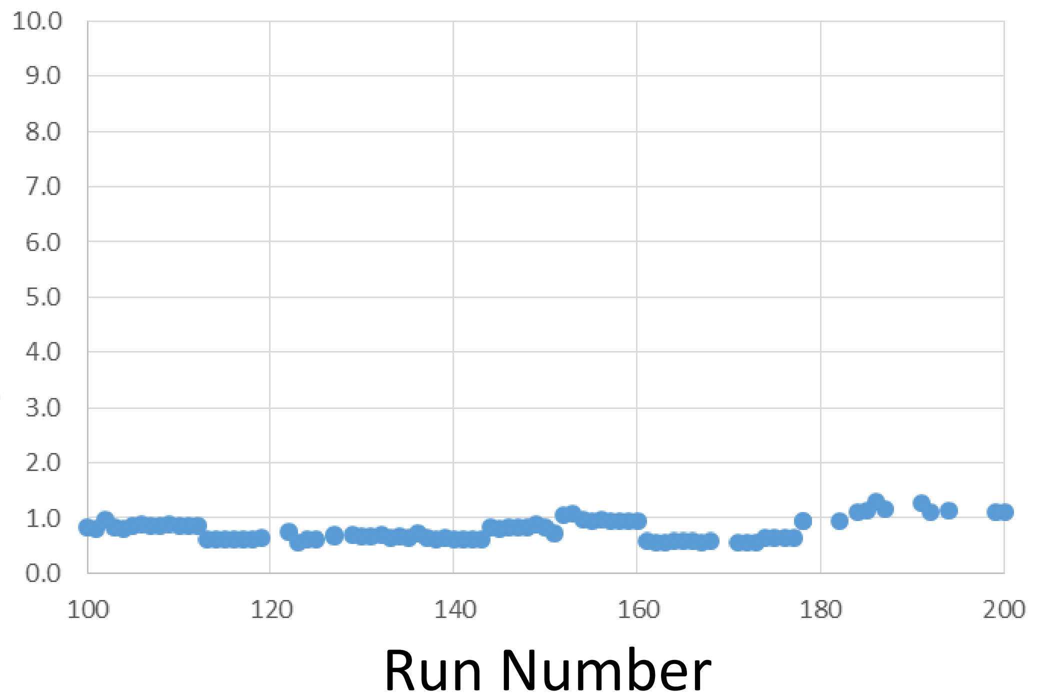


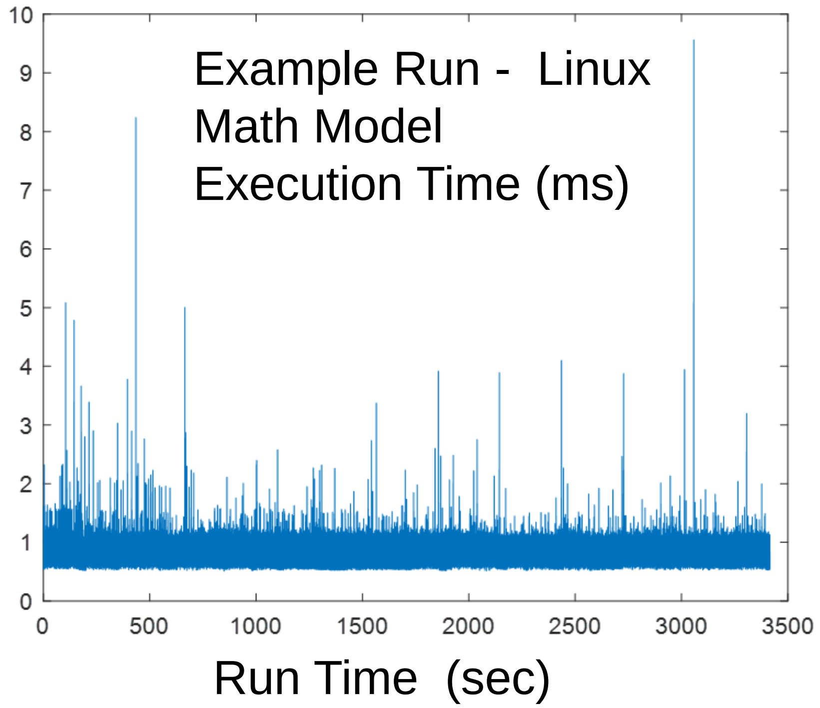
Legend:
- A/C Control Inputs
- Calculated A/C State
- Inputs From Cab
- To Cueing Devices: Displays & Motion System

Simulink® Model Execution

Ran in Simulink® environment on Linux System

Average Execution Time all runs (ms)





Overview

- Background/Motivation
- Vertical Motion Simulator (VMS)
- Experiment Description
- Challenges
- Summary

Summary

The piloted experiment was successful

- Solutions for latency problems provided stable model response
- Full test matrix completed
- Development work efficiency was significantly improved
- Confidence in model integrity

Future Work

Improvements Identified

Split Simulink® model interface S-function into 2 blocks
- Input, output blocks
- Reduce the delay by one frame

Upgrade Linux box to a faster computer
- Mitigate model execution time overruns

Upgrade host computer to include Simulink model
- Reduce delays due to I/O

Increase MATLAB® processes to high priority
- Ensure hard real-time

Takeaways

1

Successful piloted full-motion simulation with vehicle model running in Simulink®

- Researchers found method beneficial
- Full test matrix was completed and control system development work was accomplished

2

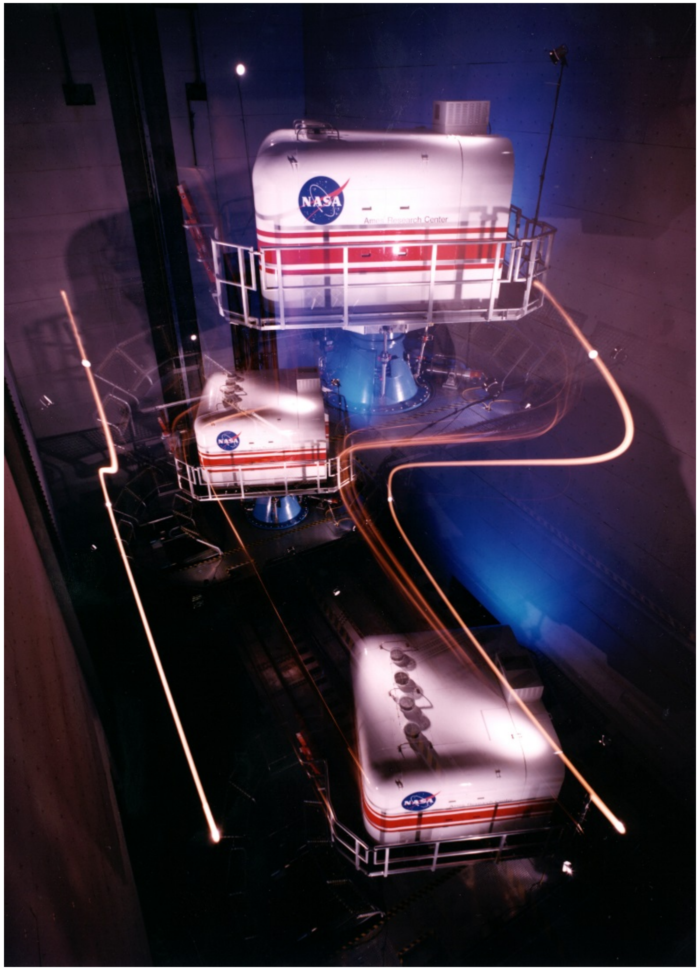
Problems were solved to reduce frame delays

- Host hardware clock used for I/O synchronization
- Overruns mitigated by minimizing MATLAB processes

3

Possible solutions were identified to allow hard real-time in the future

Questions?

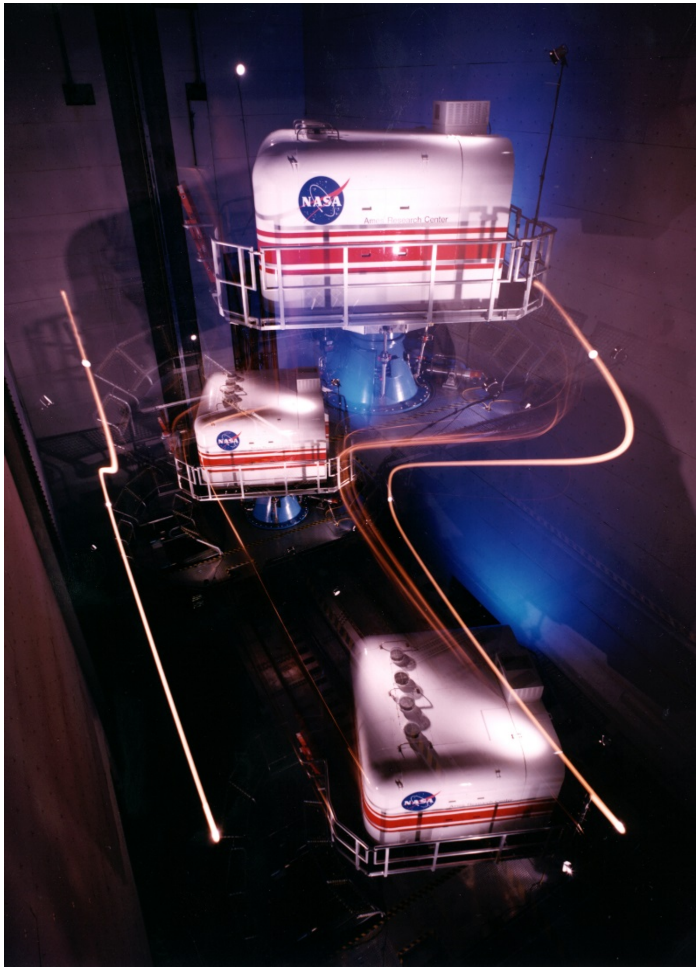


Presented at

AIAA Science and Technology Forum, San Diego, California

January 7 - 11, 2019

Backup Slides



Presented at

AIAA Science and Technology Forum, San Diego, California

January 7 - 11, 2019

Computer Speeds

Computer benchmark numbers:

Current Simulink Box

CPU Intel Xeon E5450@3.0GHz, Quad Core – **Passmark 4200**

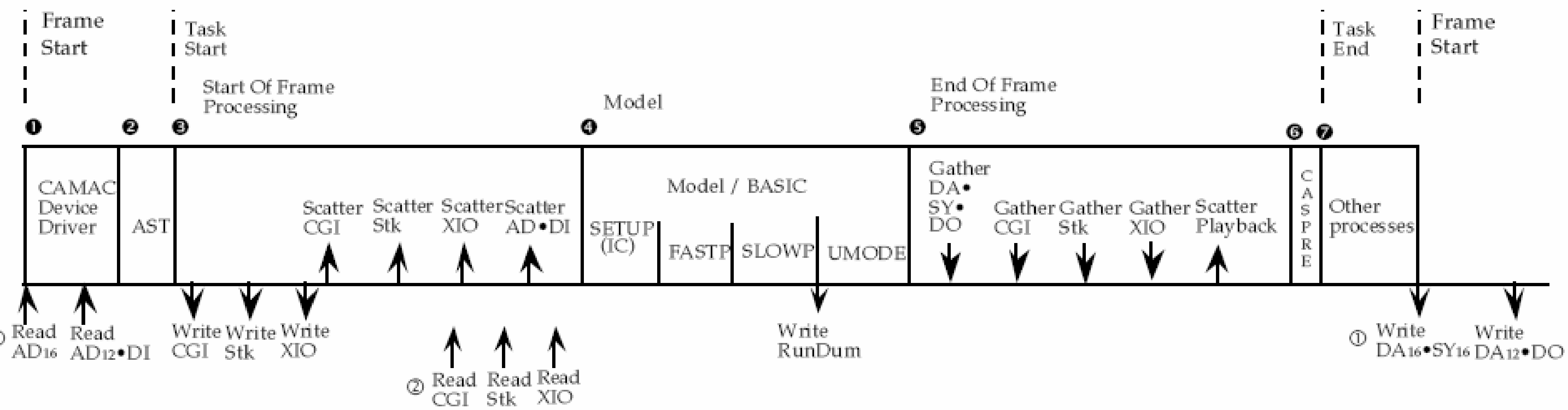
New Host Box

CPU Intel i7-8700K@3.7GHz, Six Core – **Passmark 15,961**

New IHawkT Box

CPU Intel i7-990x, Six Core – **Passmark 9086**

Frame Timeline



① 16-bit AD, DA and SY boards are converted on a clock tick via a hardware pulse to the boards.

② Data from CGI and XIO may arrive in any portion of the frame.

③ Timeline not drawn to scale.

Challenges

- Three times the model lost synchronization with the host
- Caused the simulation to freeze several seconds
- Host lost total communication with the Linux computer
- This was not due to a problem with Simulink
- Due to X-Server window going to sleep