MATLAB EXPO 2016

What’s New in Simulink
Release R2016a and R2016b

Mark Walker
What’s New in Simulink R2016a/b

Simulation Analysis and Performance
- Just-In-Time Acceleration Builds: Quickly build the top-level model for improved performance when running simulations in Accelerator mode
- Dataset Signal Plot View and analyze dataset signals directly from the MATLAB command line
- Multi-Status Image Dashboard Block: Display different images based on the signal value
- Simplified tasking mode setup
- Diagnostic: Suppressor: Suppress specific simulation warnings on particular blocks
- Diagnostic Viewer: Improved build diagnostics display

Simulink Editor
- Property Inspector: Edit parameters and properties of model elements using a single interface
- Edit-Time Checking: Detect and fix potential issues in your model at design time
- Finder: Search for model elements using improved interface
- Annotations in Libraries: Add annotations from libraries into models
- Library Browser: Expand or collapse libraries by default
- Library Browser API: Programmatically refresh the Library Browser
- Annotations: Click once to select annotation
- Default Model Font: Specify default font for model elements
- Simulink Preferences: Simplified and reorganized interface
- Simulink Editor Fonts: FreeType font engine replaces Windows GDI font engine

Component-Based Modeling
- Initialize and Terminate Function Blocks: Respond to events to model dynamic startup and shutdown behavior
- Variant Subsystem Condition Propagation: Automatically assign variant conditions to blocks outside the subsystem for improved performance
What’s New in Simulink R2016a/b

Your audience will need a lot of coffee to get to the end of this presentation.
Our Objectives with Simulink R2016b

- Provide immediate feedback – when it is most useful
- Make information more contextual
- Simplify options where possible
Q: What do these have in common?

- gray
- sievert
- katal
- slug
- kip
- steradian
- gee
- newt
- rod

A(1): They are all units

A(2): They are all units supported by Simulink
Mars Climate Orbiter

From Wikipedia, the free encyclopedia

The Mars Climate Orbiter (formerly the Mars Surveyor '98 Orbiter) was a 338-kilogram (745 lb) robotic space probe launched by NASA on December 11, 1998 to study the Martian climate, Martian atmosphere, and surface changes and to act as the communications relay in the Mars Surveyor '98 program for Mars Polar Lander. However, on September 23, 1999, communication with the spacecraft was lost as the spacecraft went into orbital insertion, due to ground-based computer software which produced output in non-SI units of pound-seconds (lbf s) instead of the SI units of newton-seconds (N s) specified in the contract between NASA and Lockheed. The spacecraft encountered Mars on a trajectory that brought it too close to the planet, causing it to pass through the upper atmosphere and disintegrate.\[1\][2]

Contents

1. Mission background
   1.1. History
   1.2. Spacecraft design
      1.2.1. Scientific instruments
2. Mission profile

Mission type: Mars orbiter
Operator: NASA / JPL
COSPAR ID: 1998-073A
Website: mars.jpl.nasa.gov/msp98/orbiter/
Mission duration: 286 days

However, on September 23, 1999, communication with the spacecraft was lost as the spacecraft went into orbital insertion, due to ground-based computer software which produced output in non-SI units of pound-seconds (lbf s) instead of the SI units of newton-seconds (N s) specified in the contract between NASA and Lockheed. The spacecraft encountered Mars on a trajectory that brought it too close to the planet, causing it to pass through the upper atmosphere and disintegrate.[1][2]
Unit Checking

1

requestedThrust_lbf*s
Unit Checking
Unit Checking

- We have seen:
  - Design information being captured in a way the tool can analyse
  - Automatic checking and conversion
  - Feedback in context
  - Simplified options leading to more detail
Staying in One Window
Staying in One Window

Source: Presentation by Chris Slack, Airbus, at MathWorks 2010 MBD Conference
# Staying in One Window

<table>
<thead>
<tr>
<th>Time</th>
<th>Session 1</th>
<th>Session 2</th>
<th>Session 3</th>
<th>Session 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:15</td>
<td>Break</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 15:45 | MATLAB Algorithm Development and Verification for Eurofighter Typhoon Praetorian  
Neil Brearley, Leonardo | Applying MathWorks Tools to Automotive Embedded Software Development  
Neil Robson, CHANGAN UK | Modelling Physical Systems in Simscape  
Steve Miller, MathWorks | Developing Robust MATLAB Code and Apps  
Paul Peeling, MathWorks |
| 16:15 | Modelling and Simulating RF Sensor Systems  
Marc Willerton, MathWorks | Verification of Automatically Generated Code  
Richard Anderson, MathWorks |                                                     |                                                     |
| 17:00 | End                                                 |                                                     |                                                     |                                                     |
Staying in One Window - Dashboards
Staying in One Window

Autopilot / DO-178 demo

Introduction to Simulink and Stateflow
2pm, Introductory Track
Staying in one window – Model Data
Model Data Editor

Configure model data properties using a table within the Simulink Editor

- Similar to information contained in Model Explorer
- Change the names of signals and mark which signals you want to test point, log, or stream
- When you select an item in the list, it gets highlighted in the model and vice versa
Staying in One Window – Property Inspector
Property Inspector

Edit parameters and properties of model elements using a single interface

- Open using View -> Property Explorer
- Similar to what you would see in the dialog windows for a block
- Undo any parameter edits using Ctrl-Z
Staying in One Window – Integrated Find
Integrated Find

Search through the model hierarchy without leaving the desktop

- Model searches more accessible
- Integrated across Simulink and Stateflow with contextual highlighting
- Configurable levels of detail
Edit Time Checking
Sisyphus

From Wikipedia, the free encyclopedia

For other uses, see Sisyphus (disambiguation).

In Greek mythology Sisyphus (/ˈsɪsfəs/; Greek: Σίσυφος, Sisyphos) was the king of Ephyra (now known as Corinth). He was punished for his self-aggrandizing craftiness and deceitfulness by being forced to roll an immense boulder up a hill, only to watch it come back to hit him, repeating this action for eternity. Through the classical influence on modern culture, tasks that are both laborious and futile are therefore described as Sisyphian (/ˈsɪsfən/).
Edit Time Checking - Simulink
Edit Time Checking - Stateflow
Edit Time Checking

Enables “fix-as-you-go” during model development

- Highlights errors and warnings
- Integrated across Simulink and Stateflow with contextual highlighting
- Compile errors and Model Advisor warnings
Our Objectives with Simulink R2016b

- Provide immediate feedback – when it is most useful
- Make information more contextual
- Simplify options where possible
Our Objectives with Simulink R2016b

- Units
- Dashboards
- Model Data View
- Property Inspector
- Integrated Find
- Edit Time Checking
What’s New in Simulink R2016a/b

- **Simulation Analysis and Performance**
  - Just-In-Time Acceleration Builds: Quickly build the top-level model for improved performance when running simulations in Accelerator mode
  - Dataset Signal Pkt. View and analyze dataset signals directly from the MATLAB command line
  - Multi-Stats Image Dashboard Block: Display different images based on the signal value
  - Simplified tasking mode setup
  - Diagnoic Suppressor: Suppress specific simulation warnings on particular blocks
  - Diagnostic Viewer: Improved build diagnostics display

- **Simulink Editor**
  - Property Inspector: Edit parameters and properties of model elements using a single interface
  - Edit-Time Checking: Detect and fix potential issues in your model at design time
  - Finder: Search for model elements using improved interface
  - Annotations in Libraries: Add annotations from libraries into models
  - Library Browser: Expand or collapse libraries by default
  - Library Browser API: Programmatically refresh the Library Browser
  - Annotations: Click once to select annotation
  - Default Model Font: Specify default font for model elements
  - Simulink Preferences: Simplified and organized interface
  - Simulink Editor Fonts: TrueType font engine replaces Windows GDI font engine

- **Component-Based Modeling**
  - Initialize and Terminate Function Blocks: Respond to events to model dynamic startup and shutdown behavior
  - Variant Subsystem Condition Propagation: Automatically assign variant conditions to blocks outside the subsystem for improved performance

What’s New in Simulink R2016a/b

Thank You!
Reference slides follow
JIT / Accelerator

% JIT Accelerator test script
% Start in a clean state
ｂclose all
clear all

v = version
open_system('serblok_ML20_noVRnoGausses')
disp('Running normal simulation to establish clean state for profiling')
sim('serblok_ML20_noVRnoGausses')
% remove any accelerator artefacts
clear mex
try
tmp = system('erase /Q a1prj');
catch
end
if exist('serblok_ML20_noVRnoGausses_aoo.mxx', 'file')
delete serblok_ML20_noVRnoGausses_aoo.mxx
end
% Set model's simulation mode to Accelerator
set_param('serblok_ML20_noVRnoGausses', 'SimulationMode', 'accelerator')
tic
sim('serblok_ML20_noVRnoGausses')
ct = toc;
disp(['Elapsed time for first accelerator sim = ', num2str(ct), ' seconds.']);
tic;
sim('serblok_ML20_noVRnoGausses')
ct = toc;
disp(['Elapsed time for second accelerator sim = ', num2str(ct), ' seconds.']);
close_system('serblok_ML20_noVRnoGausses')
clear mex

Get model loaded into memory
Baseline simulation in Normal mode
Tidy up artefacts
Go into accelerator mode
First simulation
Second simulation
JIT / Accelerator

- In R2015b:

```matlab
>> jittest

v =

8.6.0.267246 (R2015b)

Running normal simulation to establish clean state for profiling
### Building the Accelerator target for model: aeroblk_HL20_noVRnoGauges
### Successfully built the Accelerator target for model: aeroblk_HL20_noVRnoGauges
Elapsed time for first accelerator sim = 16.9316 seconds.
Elapsed time for second accelerator sim = 0.95891 seconds.
```
JIT / Accelerator

- In R2016b:

```matlab
>> jittest

v =

9.1.0.441655 (R2016b)
```

Running normal simulation to establish clean state for profiling
Elapsed time for first accelerator sim = 1.5615 seconds.
Elapsed time for second accelerator sim = 1.1001 seconds.

No build stage

JIT compilation happens during edit
Additional SimStruct Functions to Specify Units for Input and Output Ports

Simulink.BusElement: SamplingMode property removed to support having blocks specify whether to treat inputs as frame-based signals.

The SamplingMode property of Simulink.BusElement objects has been removed in R2016b. Specify the sampling mode (sample-based and frame-based) of input signals at the block level instead of at the signal level.

Compatibility Considerations

Scripts that use the SamplingMode property of Simulink.BusElement objects continue to work in R2016b. However, support for SamplingMode will be removed in a future release.

Export functions allow periodic function calls

Variant Refresh: Improved performance with removal of live refresh

Variant Subsystem: Convert Subsystems with physical ports to Variant

Variant Reducer: Additional model reduction modes in Variant Reducer (requires SLDV product license)

Enhanced find_mdlrefs function: Keep models loaded that the function loads

Subsystem conversion to referenced models: Automatic subsystem wrapper and improved Goto and From block handling

Disallow multiple iterations of root Inport function-call with discrete sample time

Data Management

Model Data Editor: Configure model data properties using a table within the Simulink Editor

Output Logging: Log data incrementally, with support for rapid accelerator mode and variant conditions

Logging Inside For Each Subsystem: Log signals inside a For Each subsystem by marking lines with antennas

Logged Dataset Data Analysis: Call same function for all timeseries objects in logged Dataset data

Scalar expansion of initial value for data store

Technique to determine whether signal has variable size

View your model configuration parameters as a group on the All Parameters tab

Enhanced error reporting and extended syntax for specifying argument dimensions for function specifications in Legacy Code Tool

Class to package and share breakpoint and table data for lookup tables

Option to disable resolution of signals and states to Simulink.Signal objects

Help fixing configuration errors from Diagnostic Viewer
Stateflow

- Immediate feedback – stay in the canvas
  - Property editor – show highlighting to diagram
  - edit time checking – show by adding default transition – show long list of features
- State transition table debugging – consider?
- Messages?
Startup and Shutdown

- Simulink for designing a system
  - Build a big model
  - Test it in simulation
  - Generate code from components one by one
  - Assemble them in another environment
  - Verify that the assembled system does the same as the original model

- Why?
  - System has startup and shutdown behaviours that are hard to model
Startup and Shutdown

- <New appearances on the Simulink canvas>
- Init, event, terminate
- Simulink Functions
- …more to do in this area, please tell us

- Moves on what you can do with top level models
- Progression of scheduling constructs
  - Sample time
  - Enable / trigger
  - Function call
  - Simulink Function
  - Event listener
Initialize and Terminate Function Blocks

Respond to events to model dynamic startup and shutdown behavior

- Model functions that embedded systems use to start up and initialize themselves, as well as those functions to terminate and shut down
- Initialize and terminate functions are allowed to be customized and aggregated in generated code
- Important new modeling workflow enabled by two features:
  - Initialize, Reset and Terminate Functions
  - State Reader and Writer Blocks
State Reader and Writer Blocks

Reset and record states during model execution

- Support explicit state reading, resetting and initialization behavior tied to ECU events
- Make state read/write much easier (can now be done via merge blocks and/or data store memory blocks, but it is cumbersome)
- Important new modeling workflow enabled by two features:
  - Initialize, Reset and Terminate Functions
  - State Reader and Writer Blocks
Export Function Rules

- <Must follow these>
- Calls, ports and subsystems / model blocks at top level only
- Sample time independent
- Fixed step discrete