MATLAB EXPO 2021

Integrating external simulation components with Simulink

Brad Hieb

MathWorks®

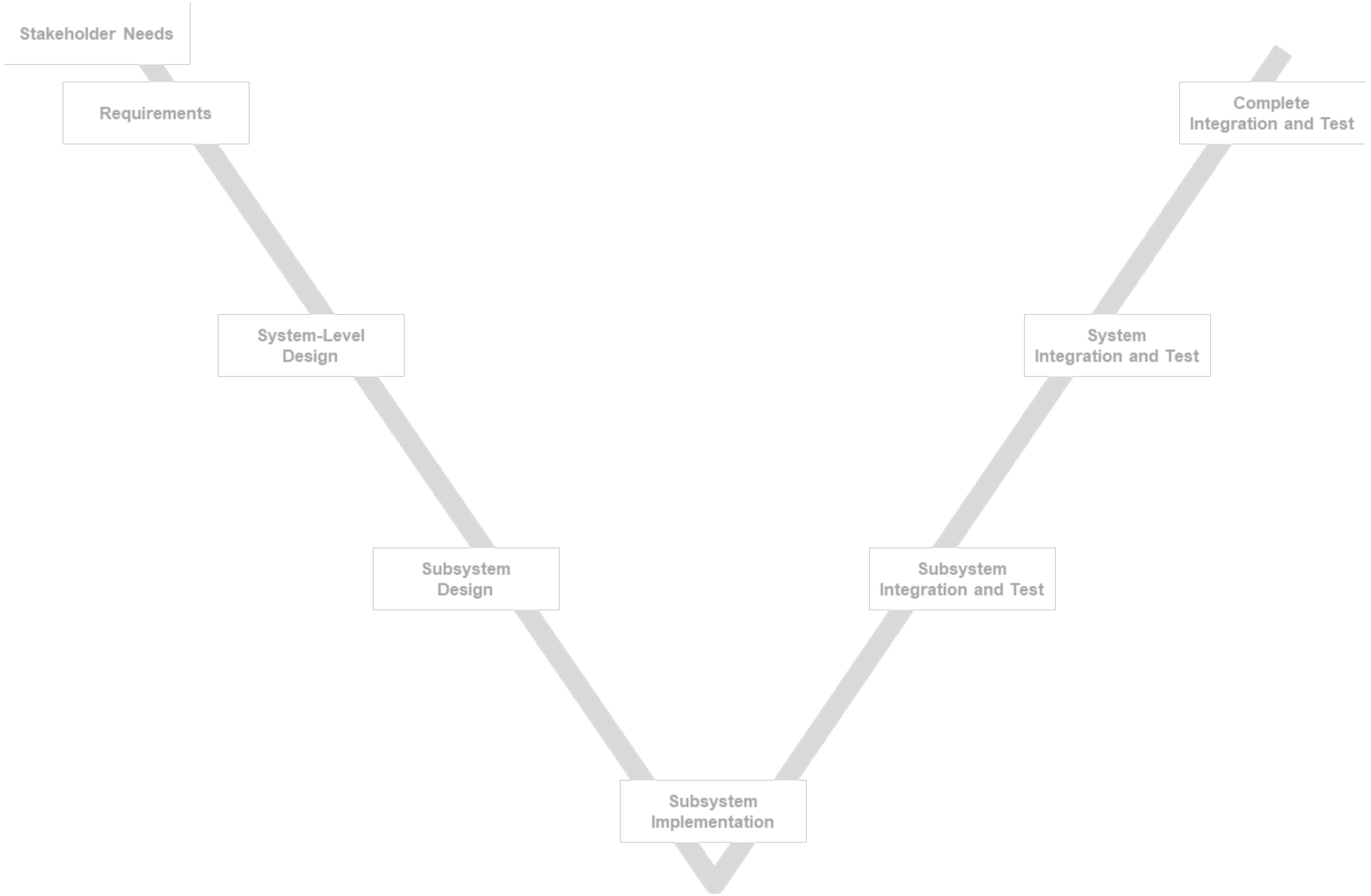
Key Takeaways

Simulink is an integration platform for simulating your complex, heterogeneous, and multi-domain systems:

- Standard-based interfaces to integrate 3rd party simulation tools/models
- Co-simulation numeric robustness with automatic signal compensation
- Bringing in custom C/C++ code made easy
- Utilizing parallel simulation capabilities to speed up system-level simulations

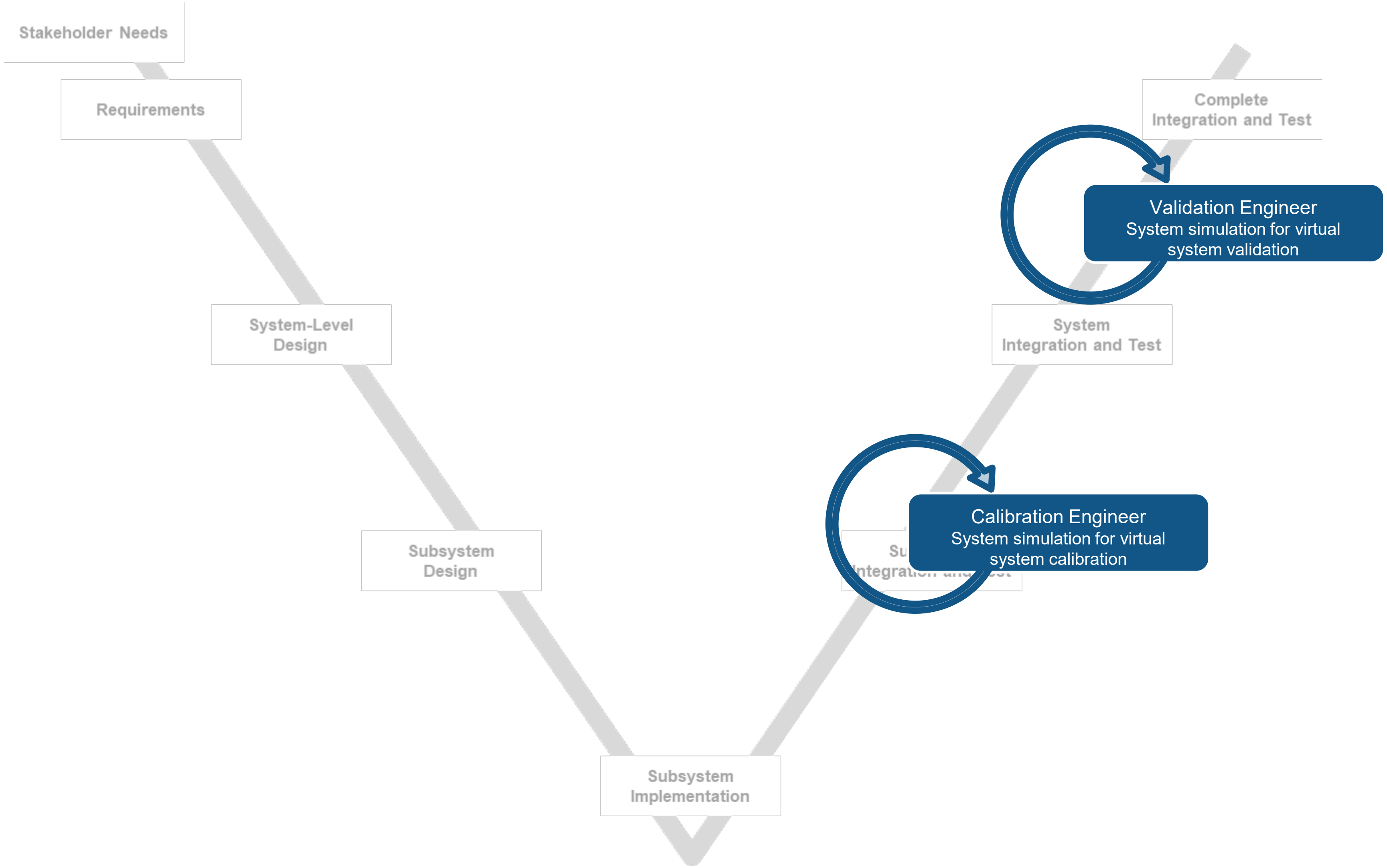
Motivation

System-level simulation is becoming pervasive in your product development cycle



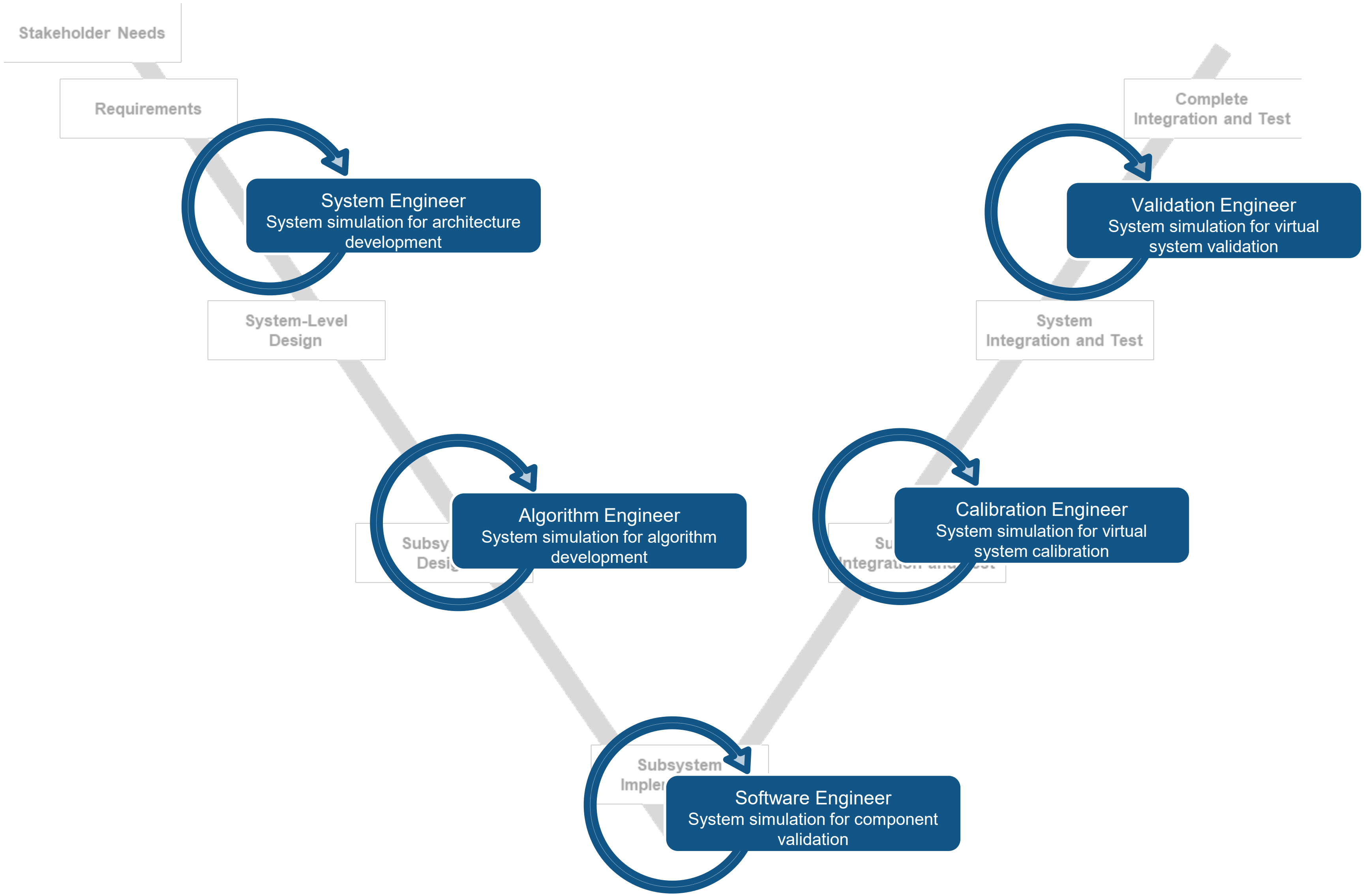
Motivation

System-level simulation is becoming pervasive in your product development cycle



Motivation

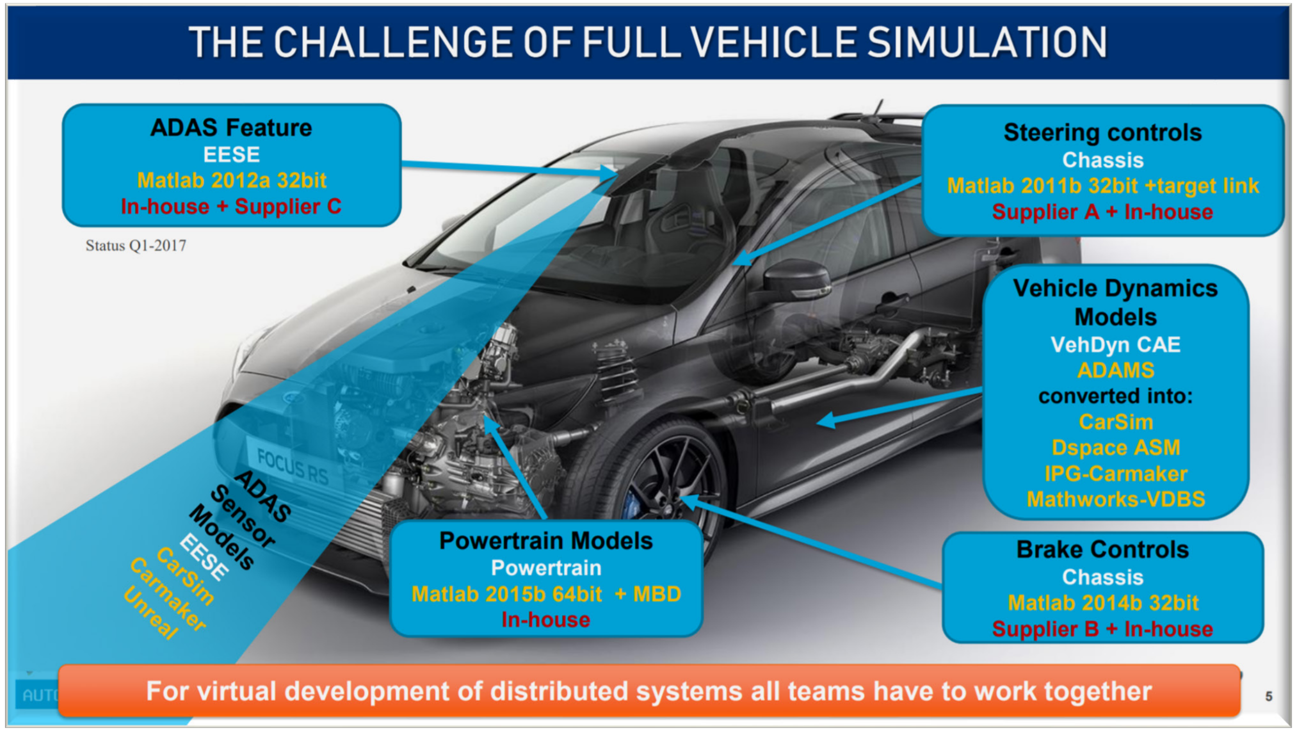
System-level simulation is becoming pervasive in your product development cycle



Motivation

Increasing challenges when simulating complex systems

- Multi-domain, inter-disciplinary design
- Model re-use among suppliers, clients and collaborators while hiding design details
- Performance: the need to speed up simulations for quick insights



Model-Based Agility with Ford Automated System Simulation Toolchain (FASST)

MathWorks Automotive Conference 2020

Agenda

- Interfaces to external simulation tools
- Robust co-simulation
- Bringing in custom C/C++ code
- Scale up system-level simulations

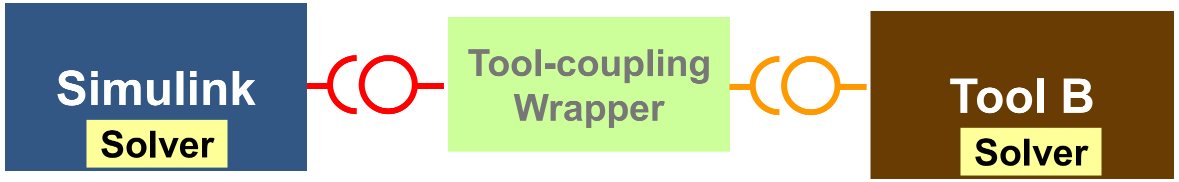
Interfaces to external simulation tools

SIMULINK
Enabled
MathWorks Partner

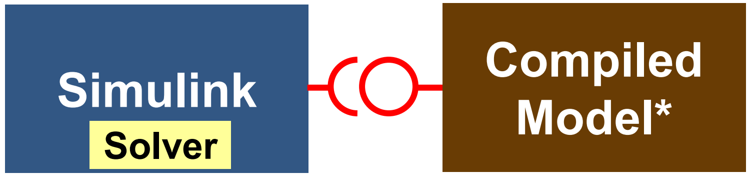
fmi Functional Mock-Up Interface

White-box and Black-box Integration

- White-box (tool-coupling)
  - Both Simulink and the external tool are running during simulation



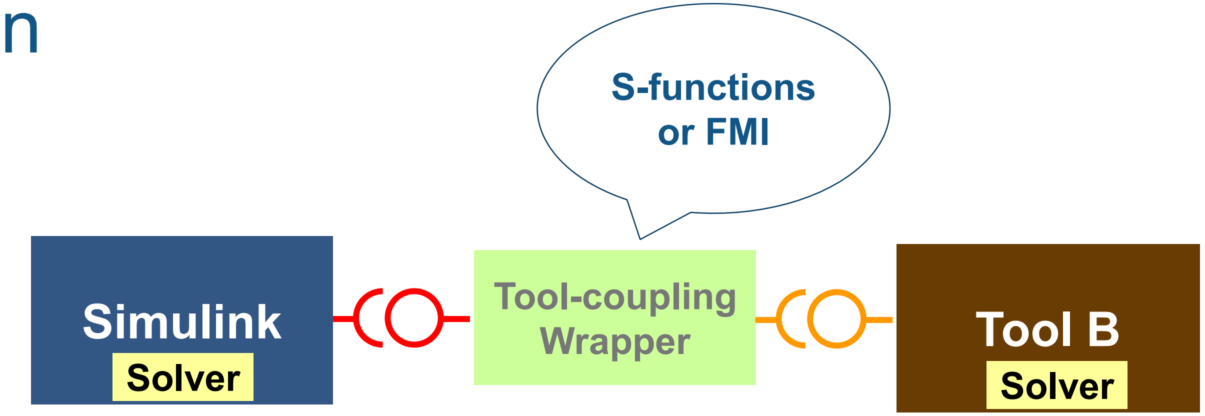
- Black-box (compiled model)
  - Only Simulink is running during simulation
  - The 3rd party model is a component inside Simulink



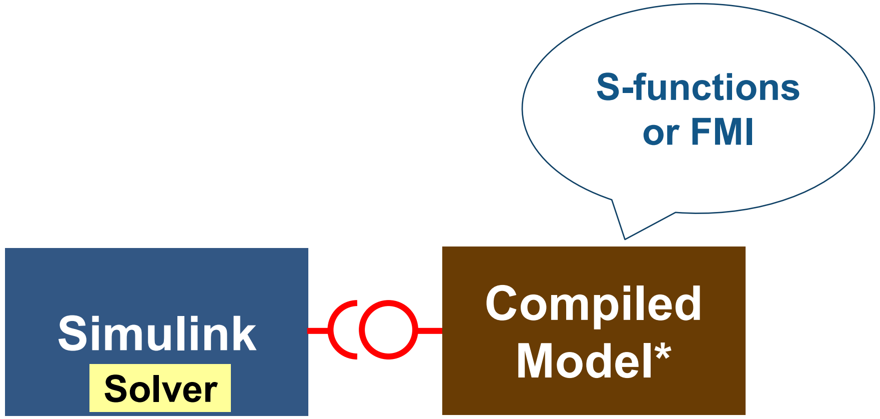
* With or without solver

White-box and Black-box Integration

- White-box (tool-coupling)
  - Both Simulink and the external tool are running during simulation



- Black-box (compiled model)
  - Only Simulink is running during simulation
  - The 3rd party model is a component inside Simulink

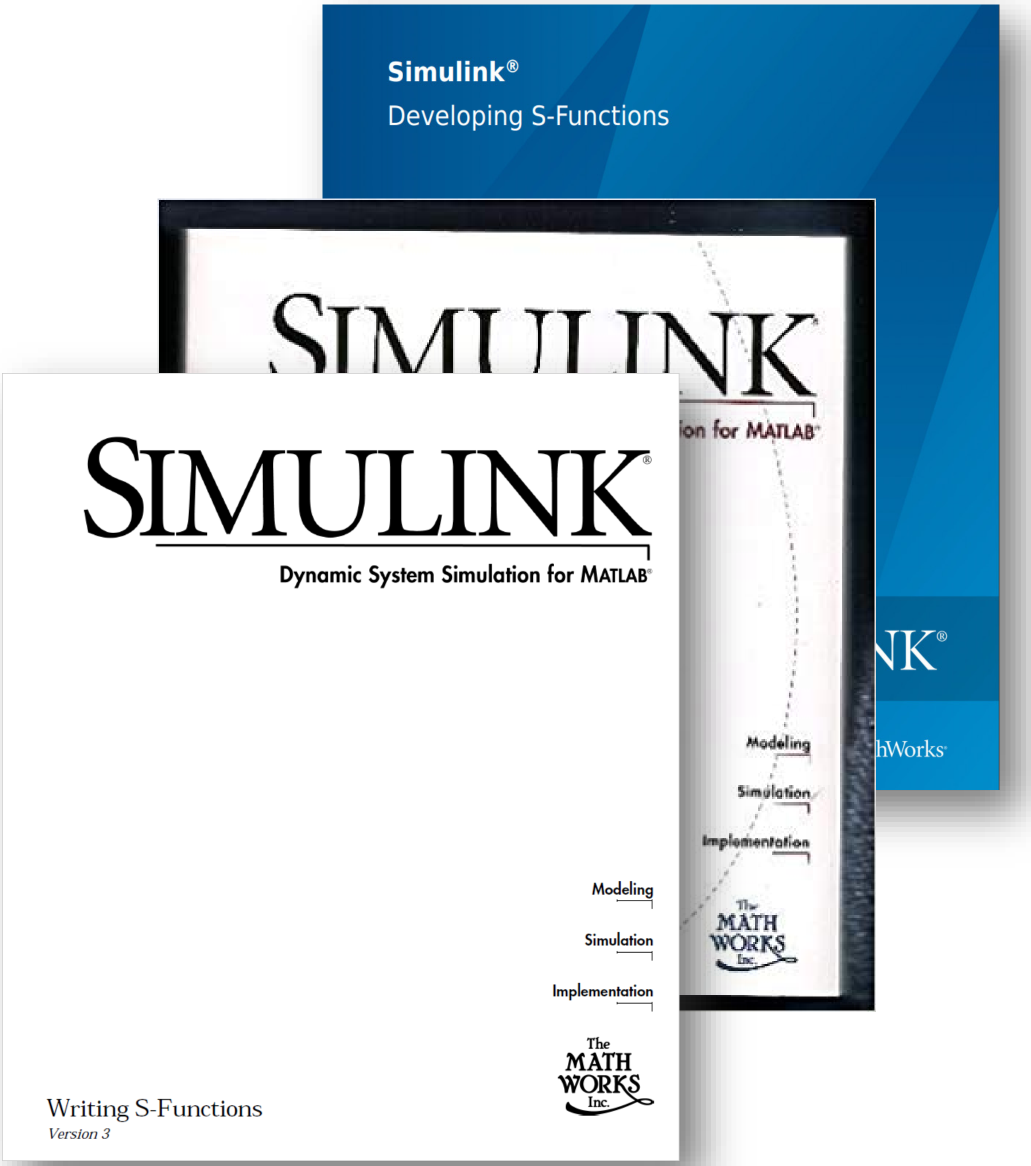


* With or without solver

- Standard-based interfaces used for both styles
  - S-functions
  - Functional Mockup Interface (FMI)

S-functions Interface

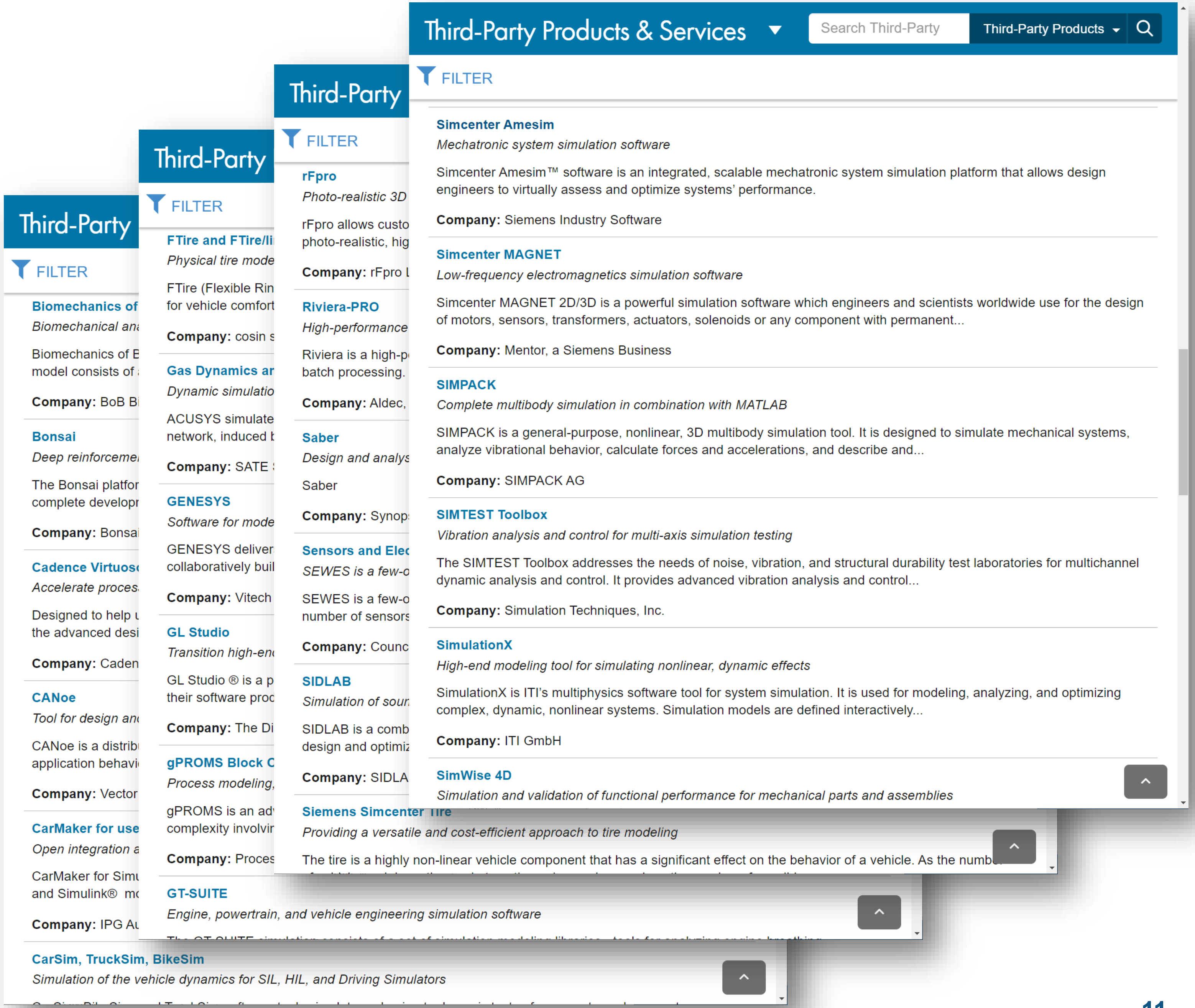
- Build custom dynamic systems in MATLAB, or C/C++
- Supports all Simulink semantics
- Well validated by industry for 20+ years
- The de facto standard to couple external tools with Simulink



S-functions Interface

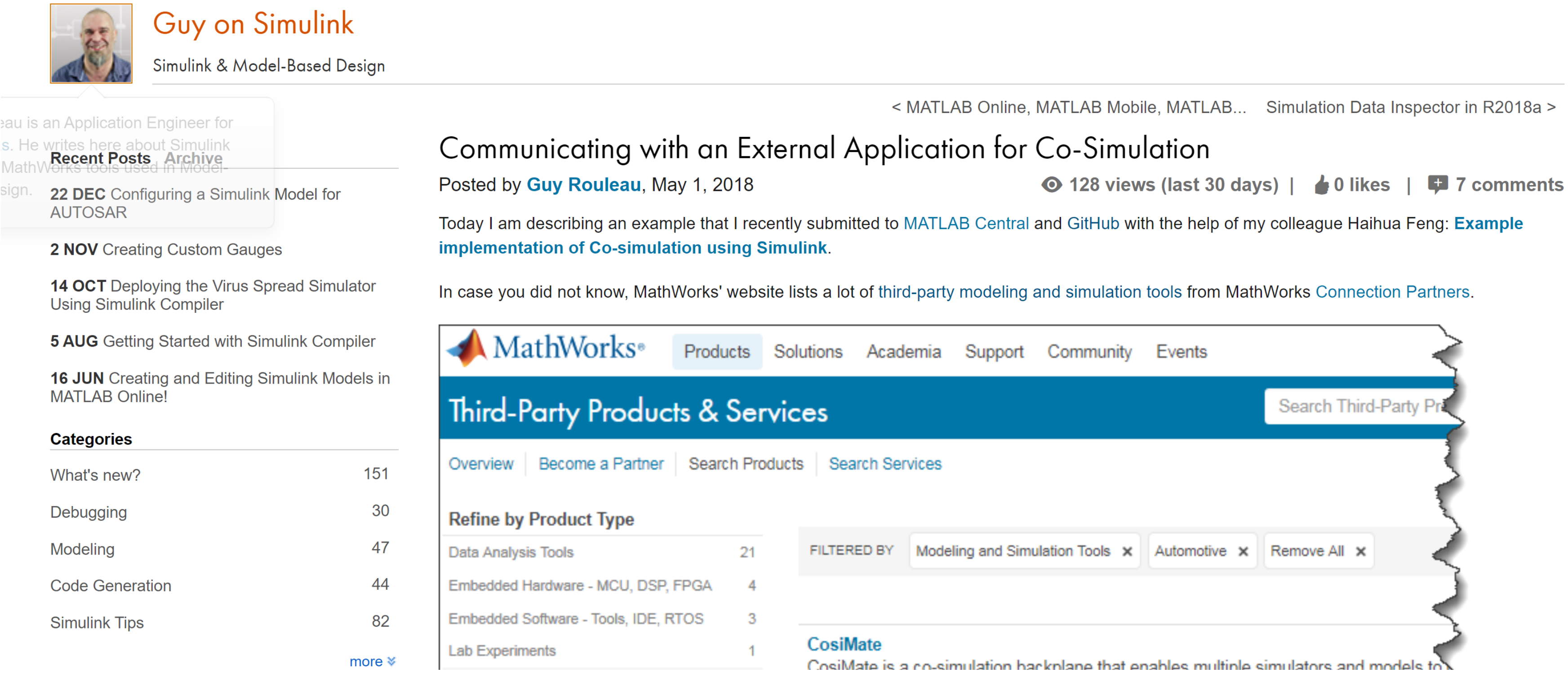
- Many of the MathWorks' Connection Partner simulation tools (150+) provide prebuilt co-simulation interfaces using S-functions

- S-functions based co-simulation interface is also available with some non-partner's tools



To learn more to use S-functions to communicate with an external application

- Example template to use S-functions as the tool-coupling interface
  - Available on "Guy on Simulink"



Functional Mock-up Interface (FMI)

- FMI is a tool independent specification to support dynamic system simulation
  - A FMU is a ZIP file packaging a model exported in FMI format
- A growing list of tools of supporting FMU export or / and FMU import
- Simulink can import both co-simulation and model-exchange FMUs for both FMI 1.0 and FMI 2.0


fmi Home Downloads Tools Related Development Literature History FAQ

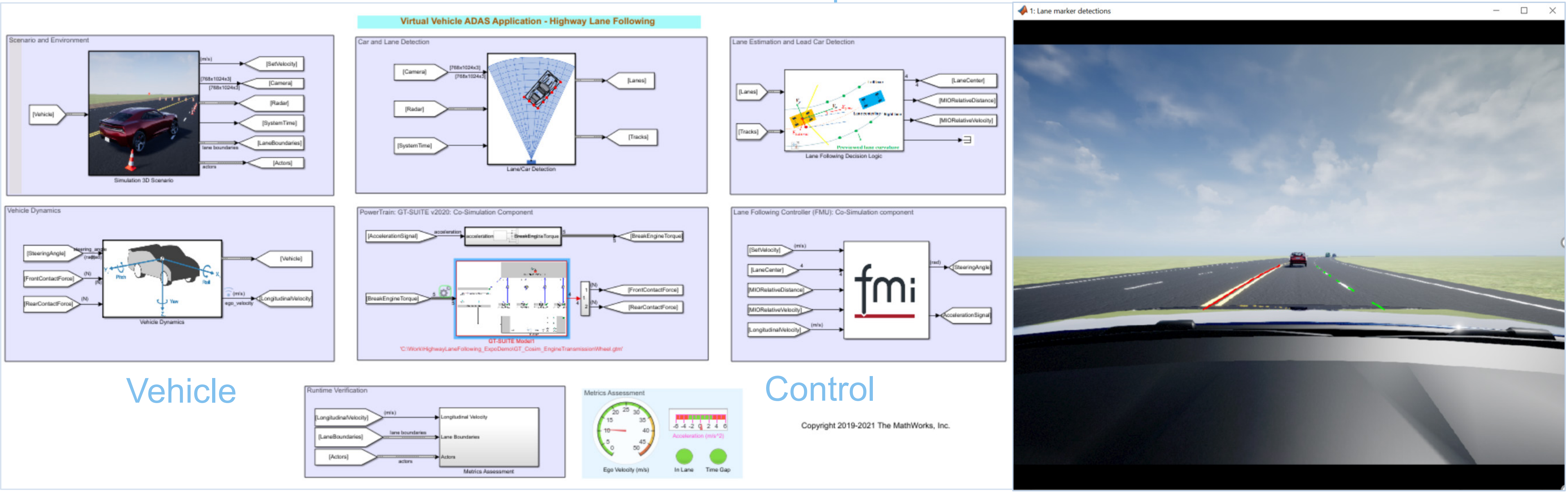
			FMU Export		FMU Import	
Name	License	Platforms	Co-Simulation	Model Exchange	Co-Simulation	Model Exchange
MATLAB® Simulink®	$		2.0		1.0 2.0	1.0 2.0

Demo - Virtual Vehicle ADAS Applications

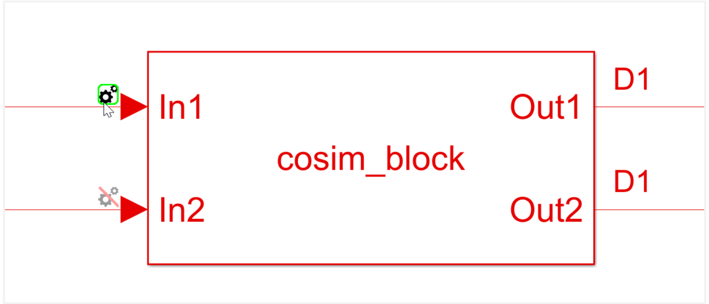
- Integrating external components using S-functions and FMU

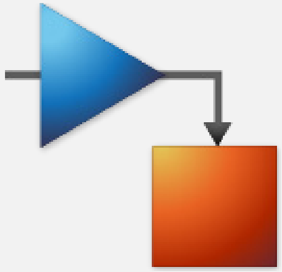
Environment

Perception



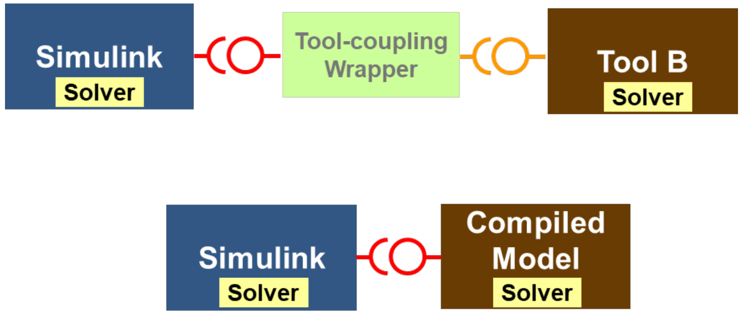
Robust Co-simulation



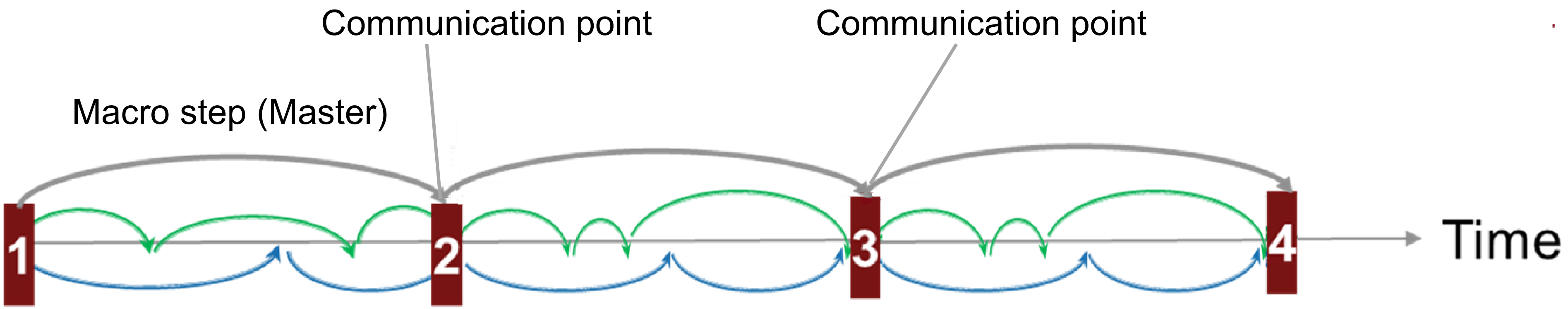


Co-simulation

- A frequently used method to bring models of external tools into Simulink
  - Each co-simulation component has its own solver
  - Can be implemented either white-box or black-box style

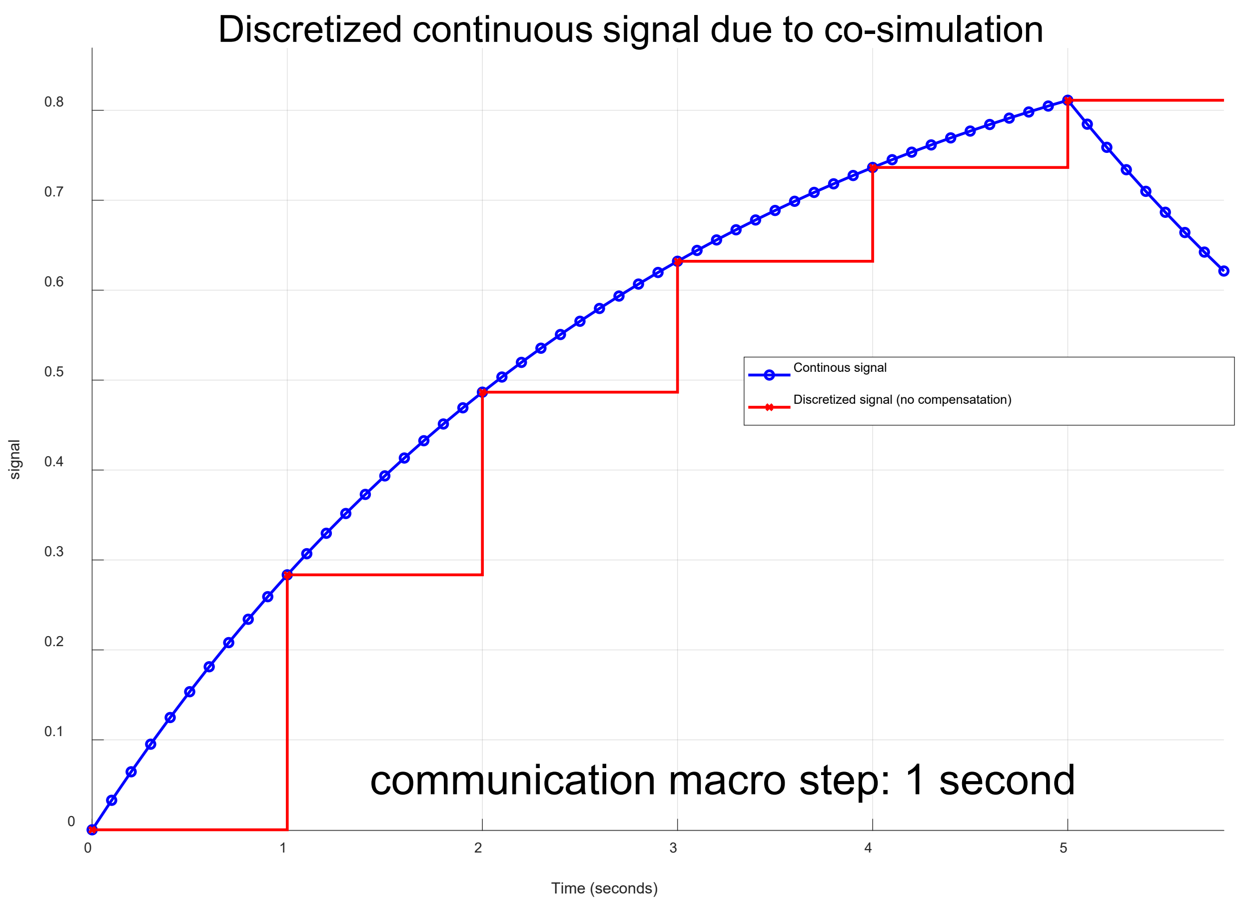


- Co-simulation components can run in parallel freely between communication macro steps



Co-simulation numeric behavior

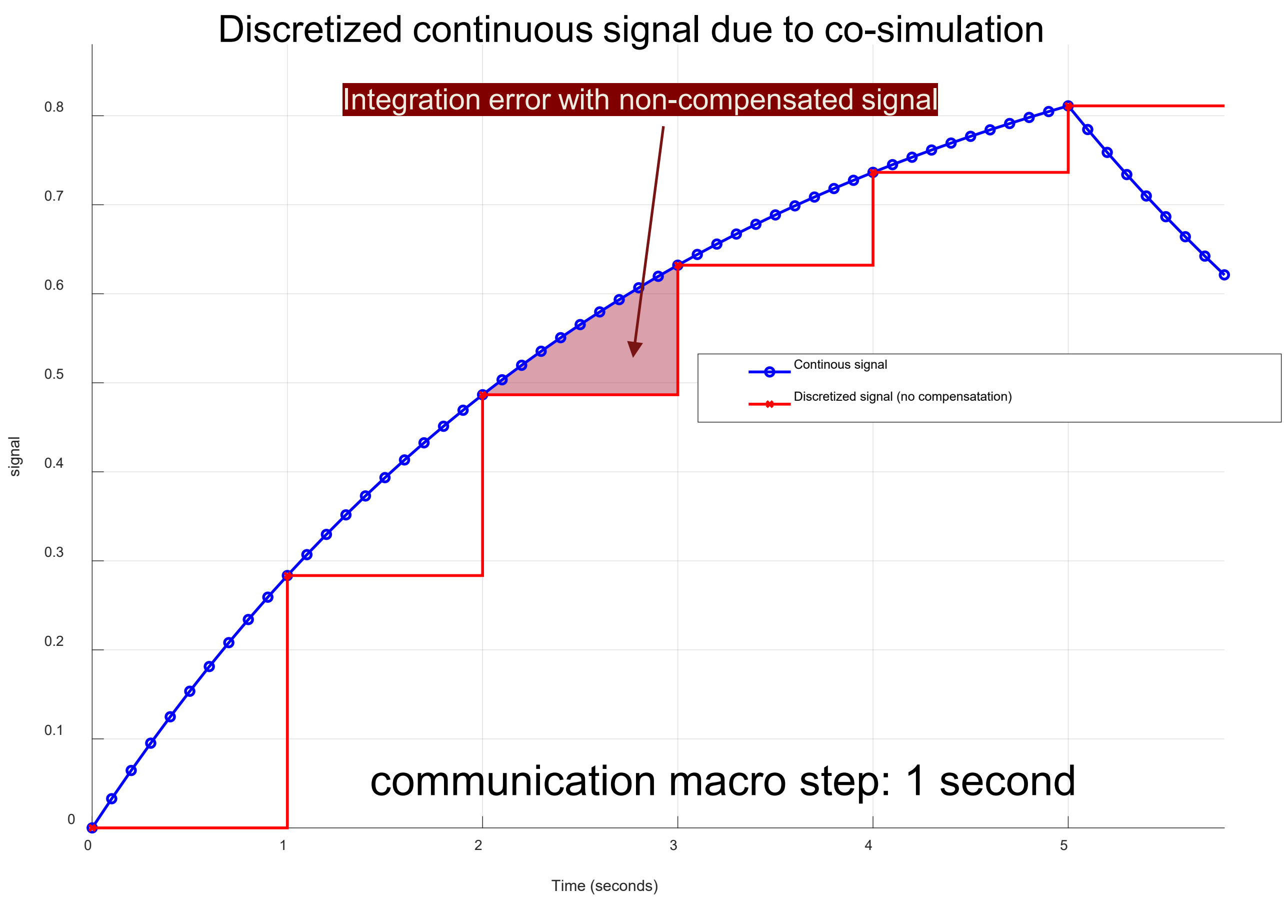
- Model integration is more than coupling the signals
- Potential error when coupling continuous signals
  - Discretized and delayed crossing co-simulation boundary
  - Non-compensated signals could lead to accuracy loss or even system instability



The un-compensated signal (red line) deviates from the ideal, continuous signal (blue line) due to discretization

Co-simulation numeric behavior

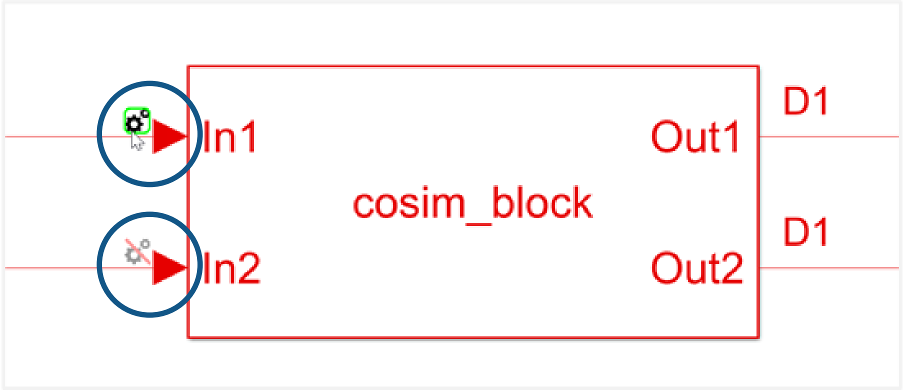
- Model integration is more than coupling the signals
- Potential error when coupling continuous signals
  - Discretized and delayed crossing co-simulation boundary
  - Non-compensated signals could lead to accuracy loss or even system instability

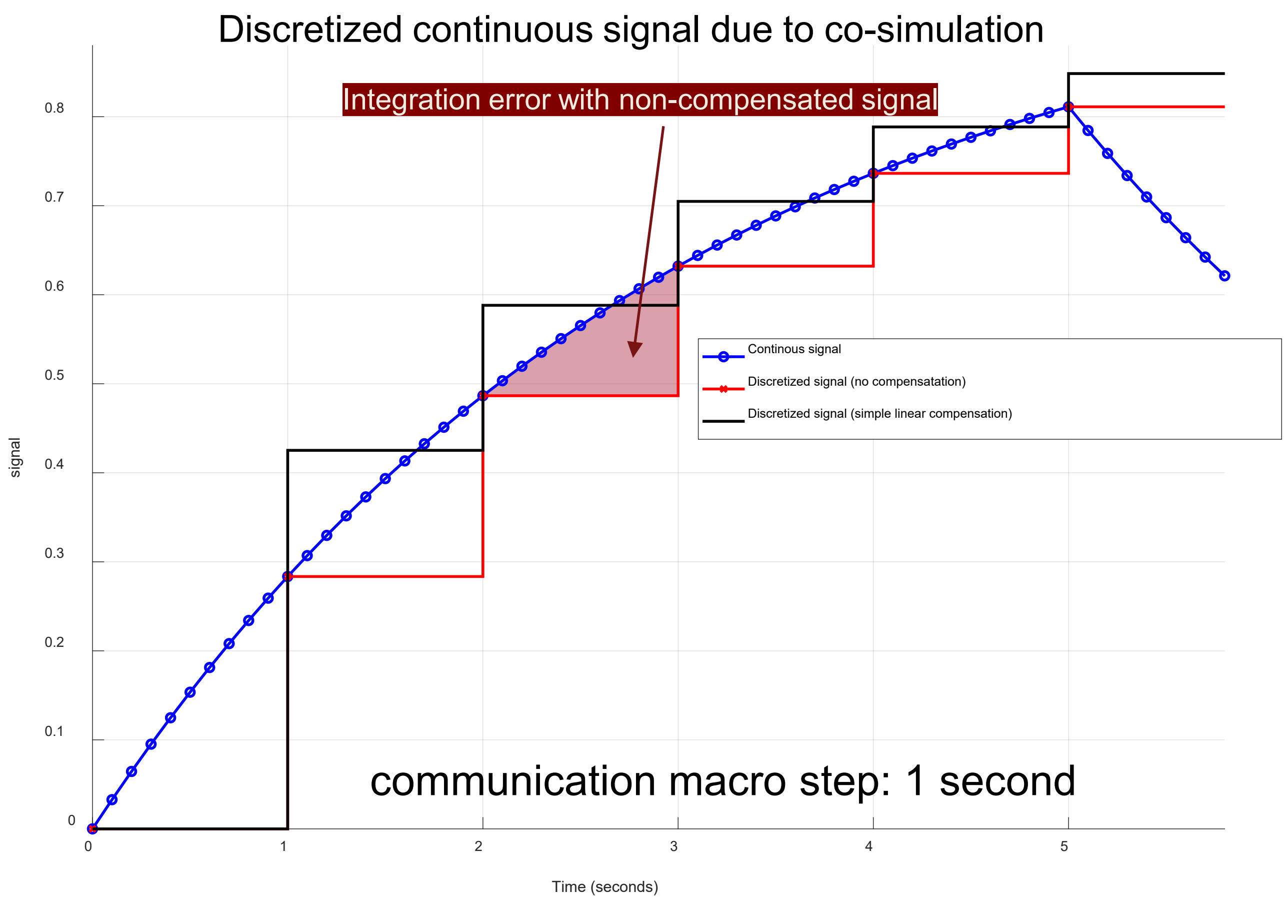


The un-compensated signal (red line) deviates from the ideal, continuous signal (blue line) due to discretization

Robust co-simulation

- Automatic and manual mechanism to compensate the discretized continuous signals
  - Choice of linear or high order extrapolation compensation methods
- More robust co-simulation results compared to un-compensated co-simulation

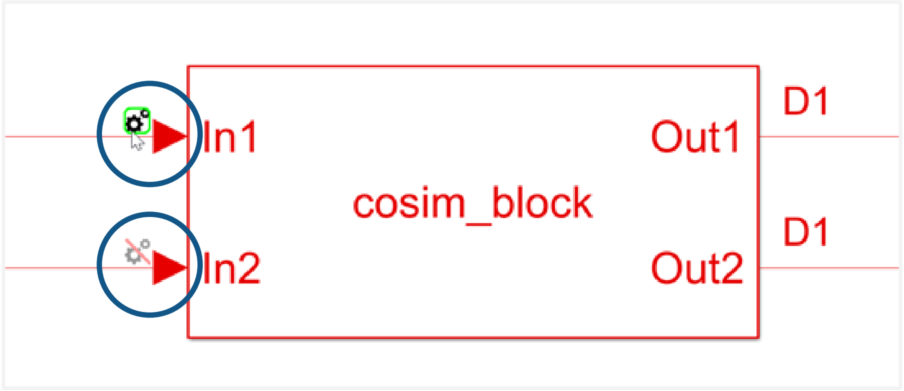


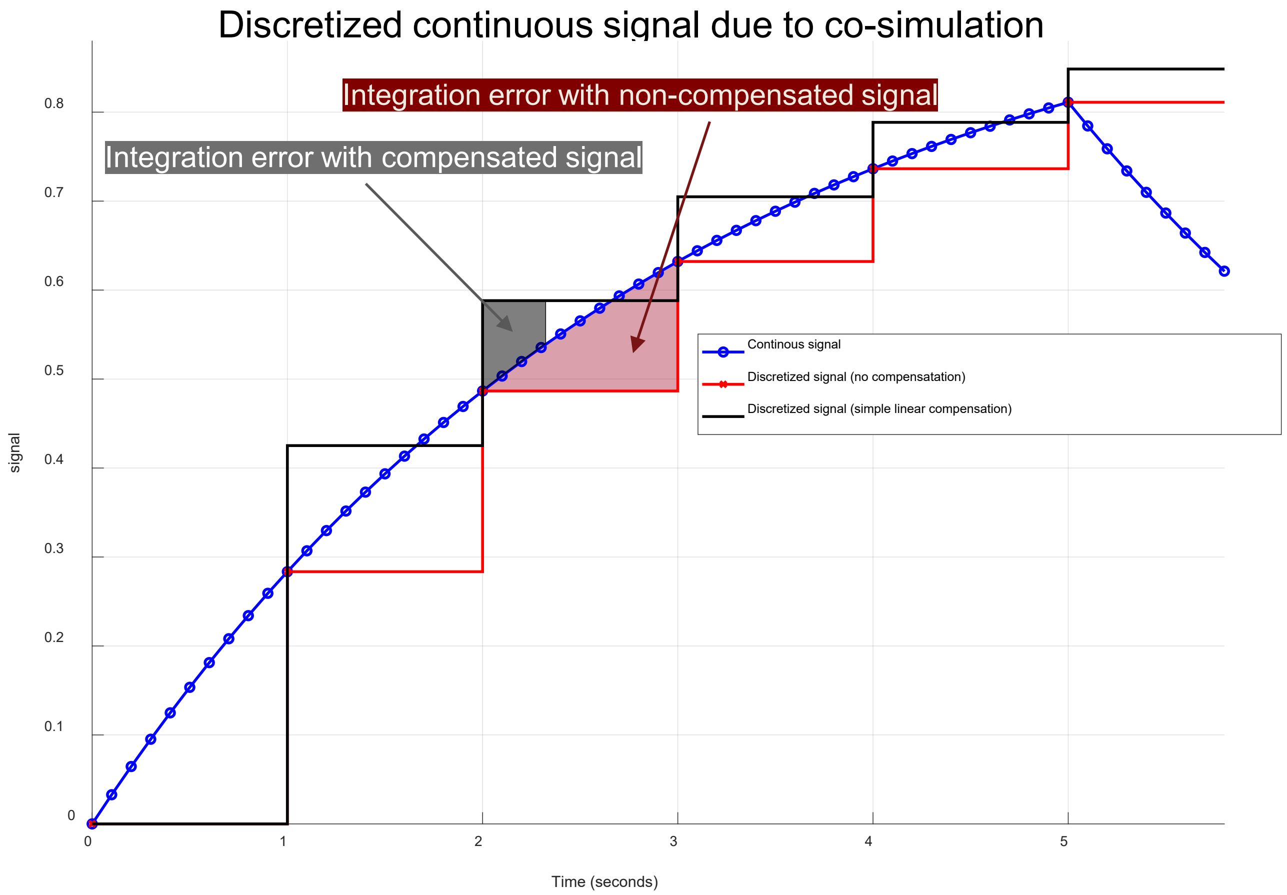


The compensated signal (black line) with simple linear extrapolation is closer to the ideal, continuous signal (blue line) than the uncompensated, discretized signal (red line)

Robust co-simulation

- Automatic and manual mechanism to compensate the discretized continuous signals
  - Choice of linear or high order extrapolation compensation methods
- More robust co-simulation results compared to un-compensated co-simulation





The compensated signal (black line) with simple linear extrapolation is closer to the ideal, continuous signal (blue line) than the uncompensated, discretized signal (red line)

Custom Code Integration

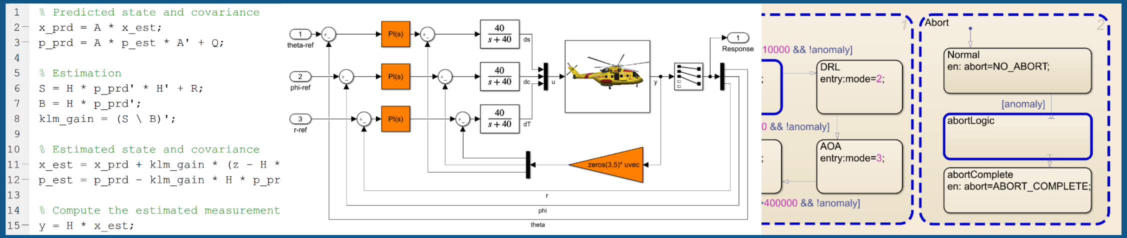
```
template<class InputString, class OutputString>
bool unhexlify(const InputString& input, OutputString& output) {
  if (input.size() % 2 != 0) {
    return false;
  }
  output.resize(input.size() / 2);
  int j = 0;
  auto unhex = [](char c) -> int {
    return c >= '0' && c <= '9' ? c - '0' :
           c >= 'A' && c <= 'F' ? c - 'A' + 10 :
           c >= 'a' && c <= 'f' ? c - 'a' + 10 :
           -1;
  };

  for (size_t i = 0; i < input.size(); i += 2) {
                                    ]);
                                    + 1]);
                                    < 0) {

                                    ) + lowBi
```

Internal Libraries

Vendor Libraries

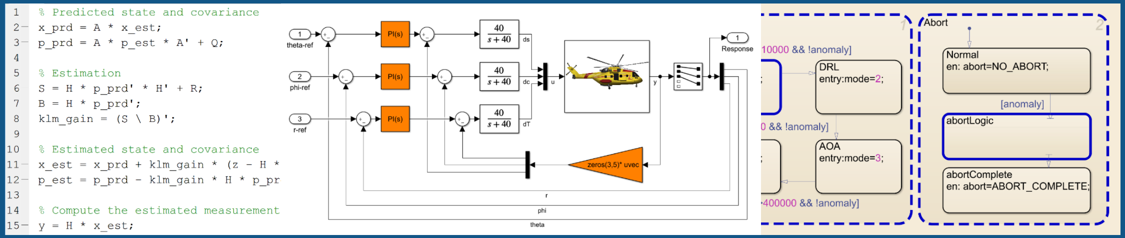
Custom Code Integration

Model-Based Design



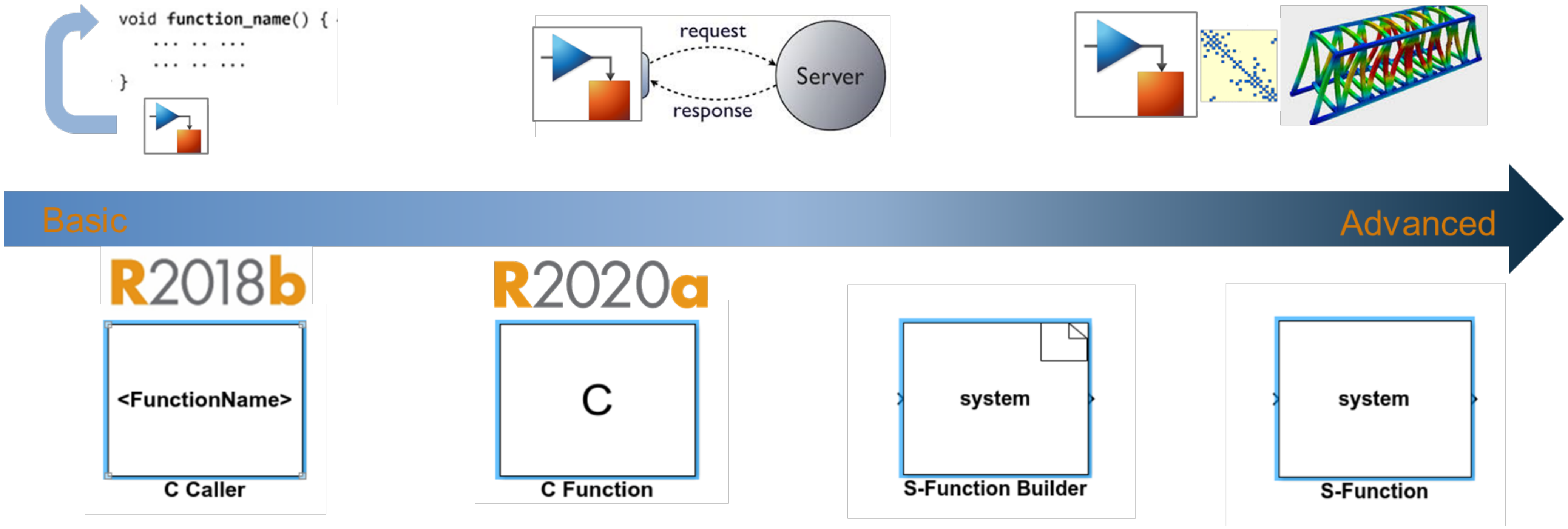
Custom Code Integration

Model-Based Design



C/C++ Libraries

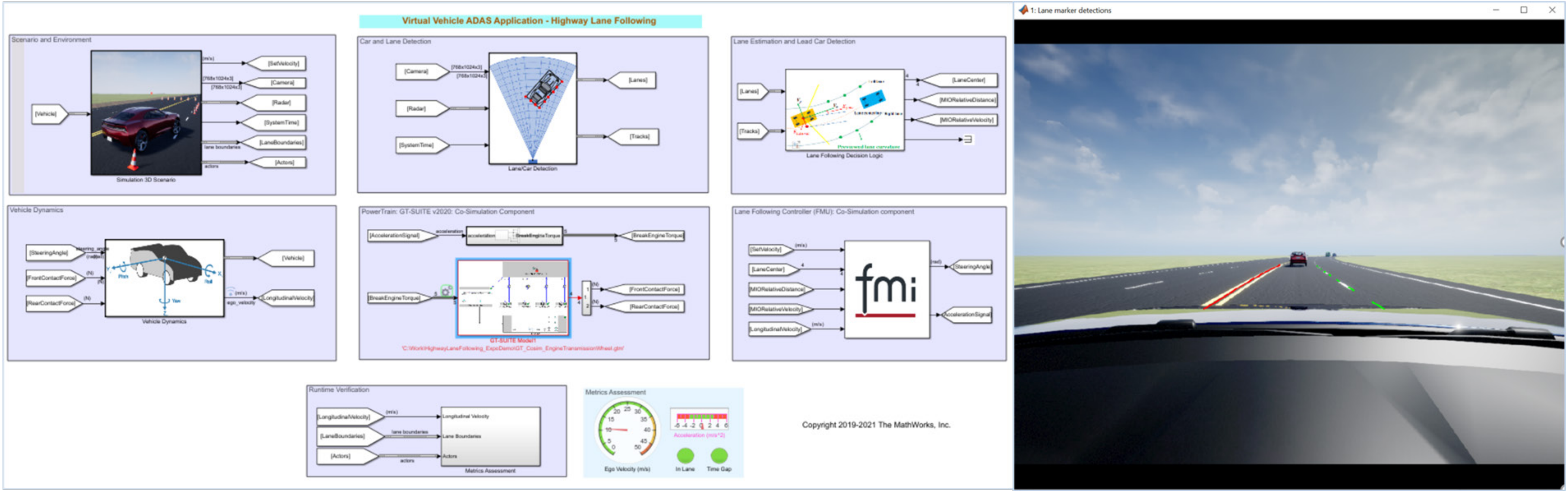
Custom Code Integration



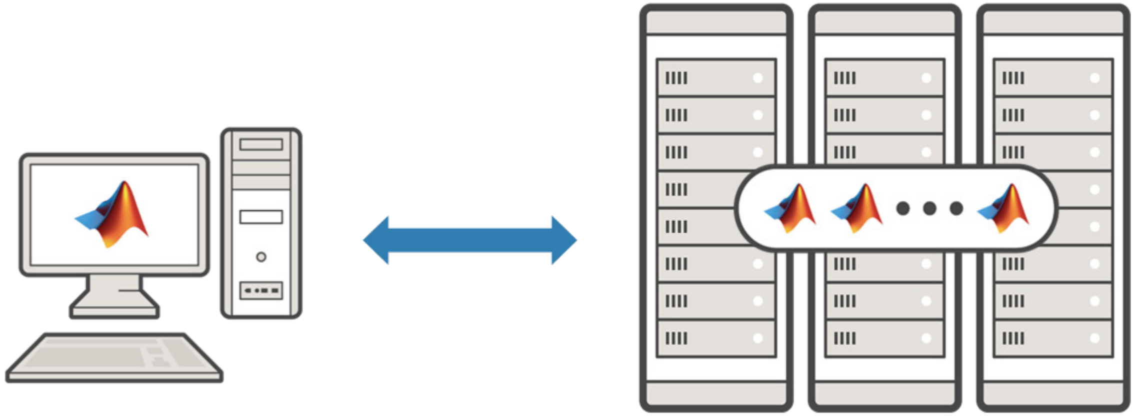
- Versatile ways to integrate your custom code
  - Simply calling your function
  - Reuse your code as a Simulink library
  - Scripting algorithm with discrete states
  - Dynamic system creation

Demo - Virtual Vehicle ADAS Applications

- Integrating custom C code for lane marker detection



Simulation Scalability



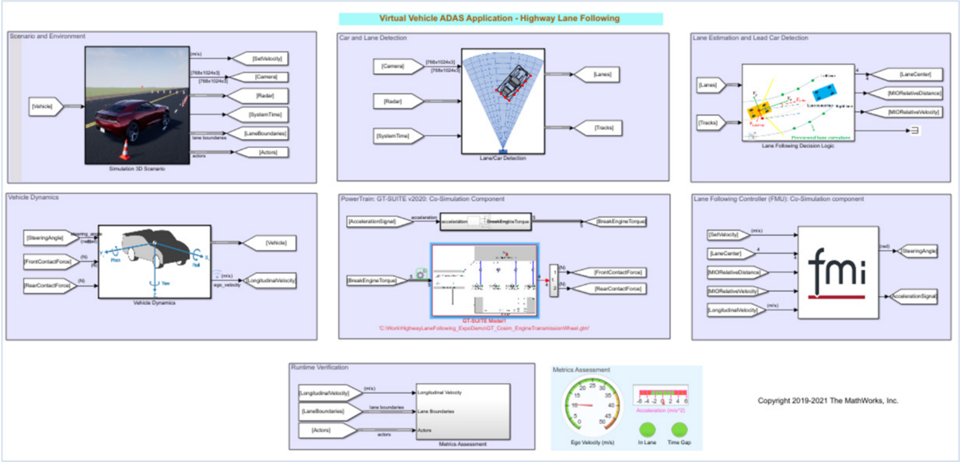
Scale Up System Simulation

- System-level simulation problems may involve a large number of simulation iterations due to the complexity of design combinations
- Complex system simulation takes time to execute
- The capability to scale up is a must-have of an integration platform to deliver quick simulation insights

Full vehicle simulation

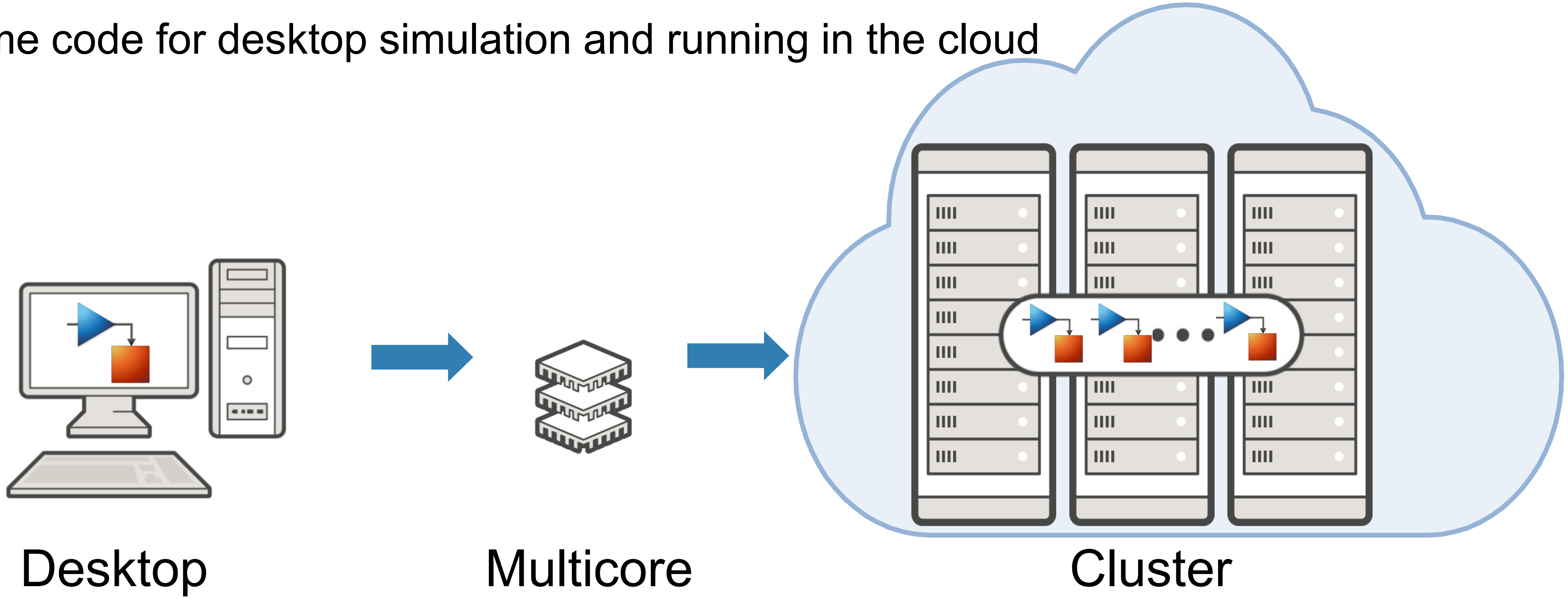
10 drive cycles
10 weather conditions
10 vehicle loadings
10 gear ratios
10 tire sizes

-> 100,000 simulations



Scale Up System Simulations

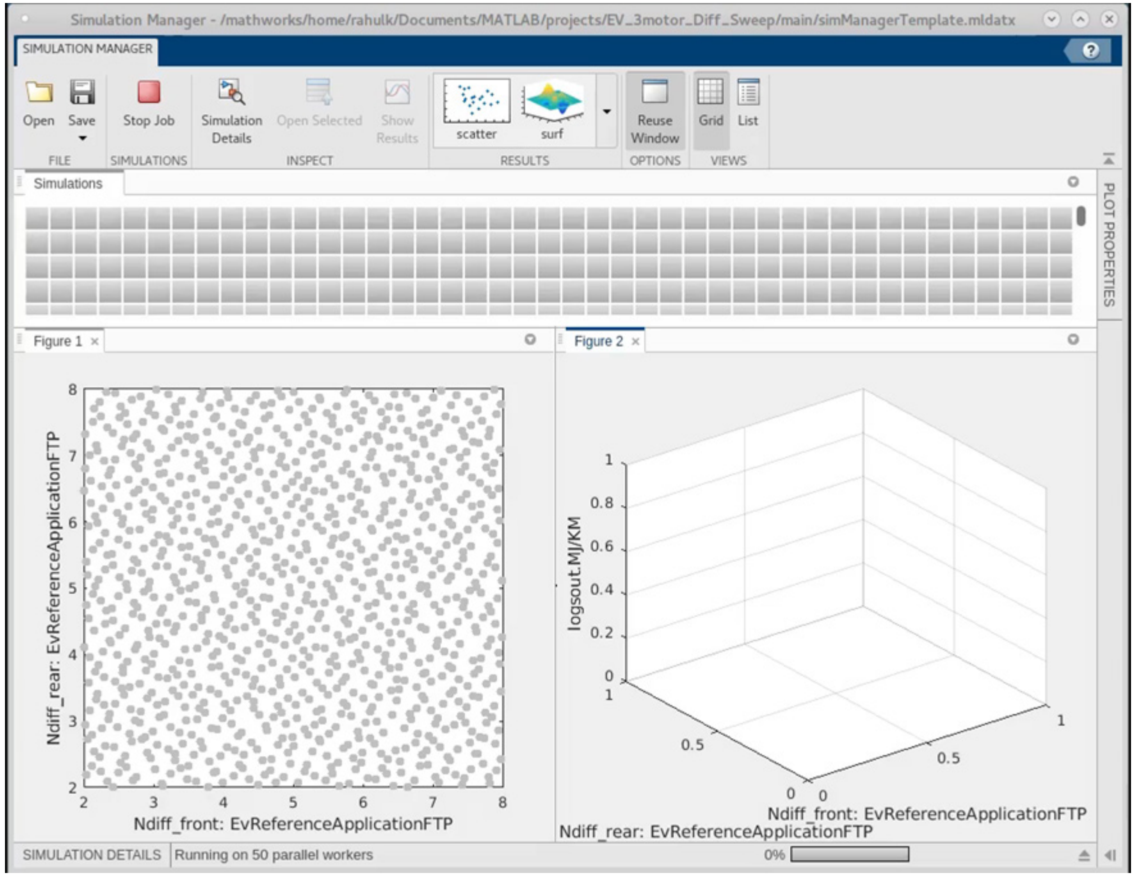
- The same code for desktop simulation and running in the cloud




```
for i = 10000:-1:1
    in(i) = Simulink.SimulationInput('my_model');
    in(i) = in(i).setVariable('my_var', i);
end
out = parsim(in);
```

Scale Up System Simulations

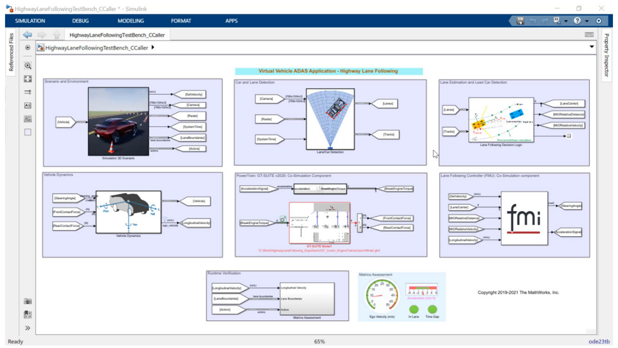
- Manage and visualize the simulations as the simulations are progressing

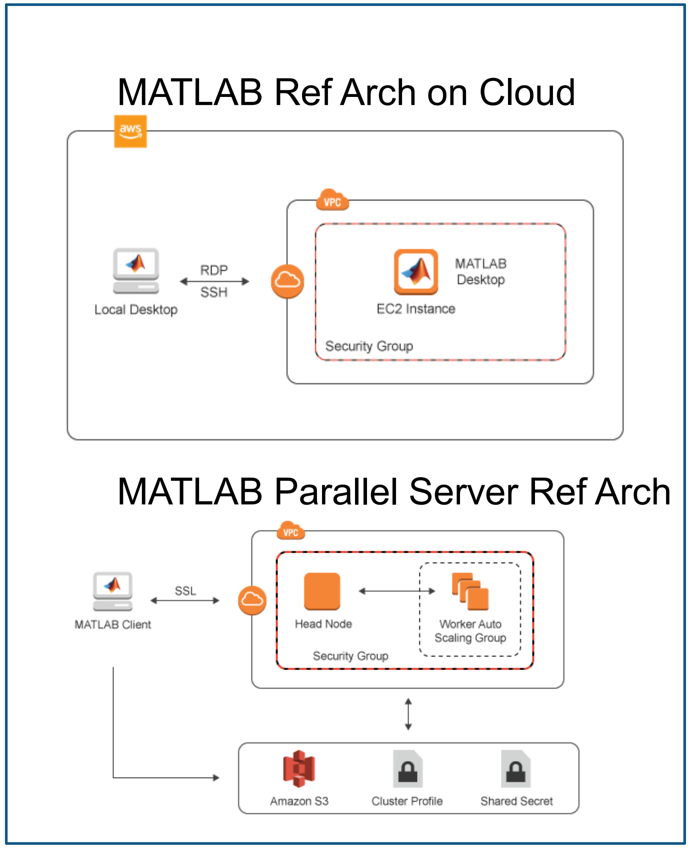


Scale Up System Simulations

- Move the simulation to the Cloud by leveraging a Prebuilt Cloud Configuration via Reference Architecture

System Simulation





[Use MATLAB/Simulink in the Cloud]

Summary

As an integration platform Simulink provides key capabilities to scale up your complex, system-level simulations:

- Standard-based interfaces to integrate 3rd party simulation models
- Co-simulation numeric robustness with automatic signal compensation
- Bringing in custom C/C++ code made easy
- Utilizing parallel simulation capabilities to speed up system level simulations

MATLAB EXPO
2021

Thank you

MathWorks®