Aylin EDGÜ

System Modeling and Control Engineer

aylin.edgu@figes.com.tr

MathWorks Ekosisteminde Model Tabanlı Tasarım

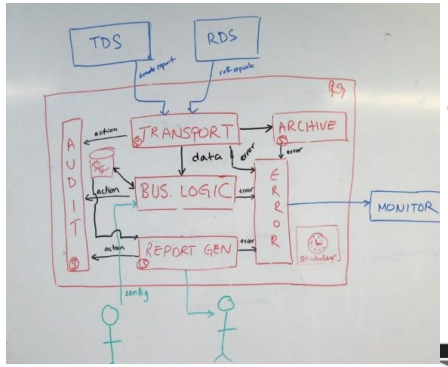
FİGES applied physics & mathematics

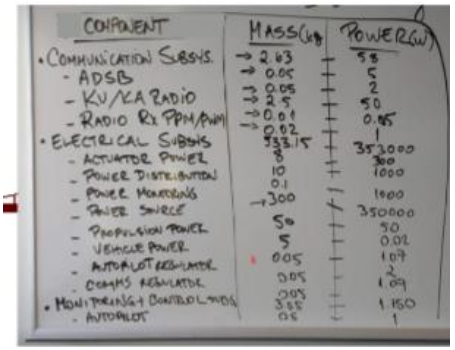
MathWorks® Authorized Reseller

Agenda

1. Traditional vs. Model Based Design
2. Model Based Design Overview
3. Simulink Overview
4. Simscape Overview
5. Stateflow Overview
6. Applications with Model Based Design
7. Q&A

Traditional Development Process





Manual steps introduce **errors** and **slow down** the development process

RESEARCH

REQUIREMENTS

SPECIFICATIONS

DESIGN

EDA	Algorithm Design	MCAD/ MCAE
Electrical Components	Embeddable Algorithms	Mechanical Components

IMPLEMENTATION

C/C++

Embedded Software

INTEGRATION AND TEST

Requirement Documents
- Difficult to analyze
- Difficult to manage as they change

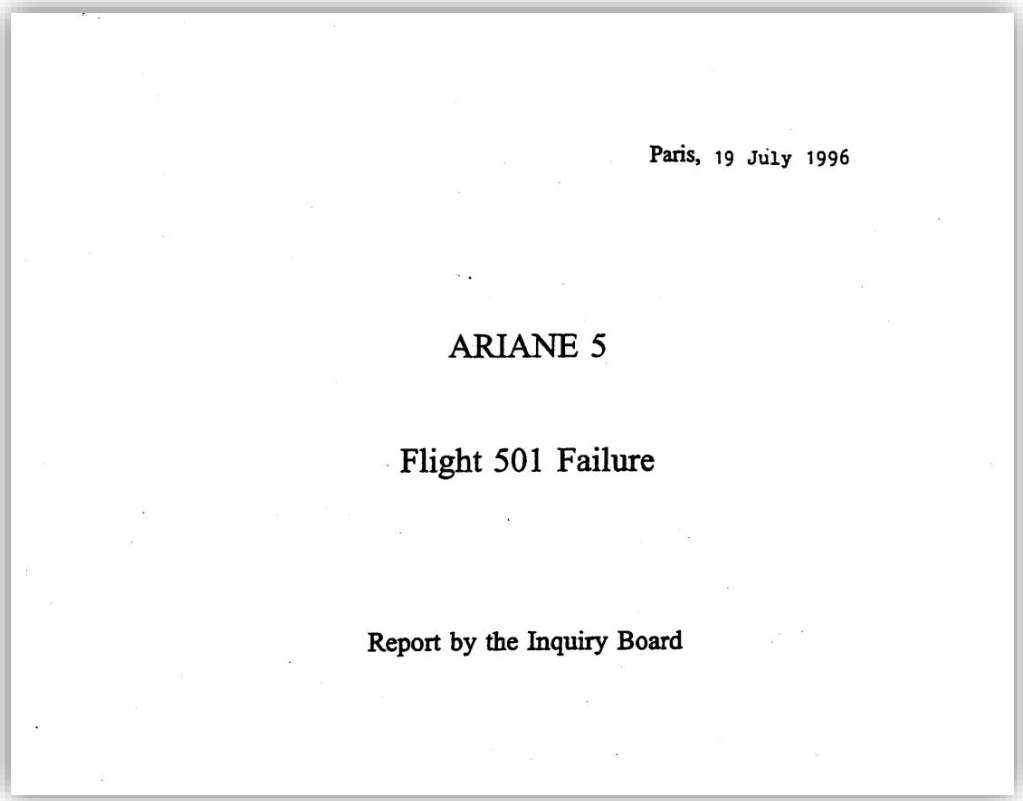
Paper Specifications
- Easy to misinterpret
- Difficult to integrate with design

Physical Prototypes
- Incomplete and expensive
- Prevents rapid iteration
- No system-level testing

Manual Coding
- Time consuming
- Introduces defects and variance
- Difficult to reuse

Traditional Testing
- Design and integration issues found late
- Difficult to feed insights back into design process
- Traceability

Traditional vs. Model-Based Approach

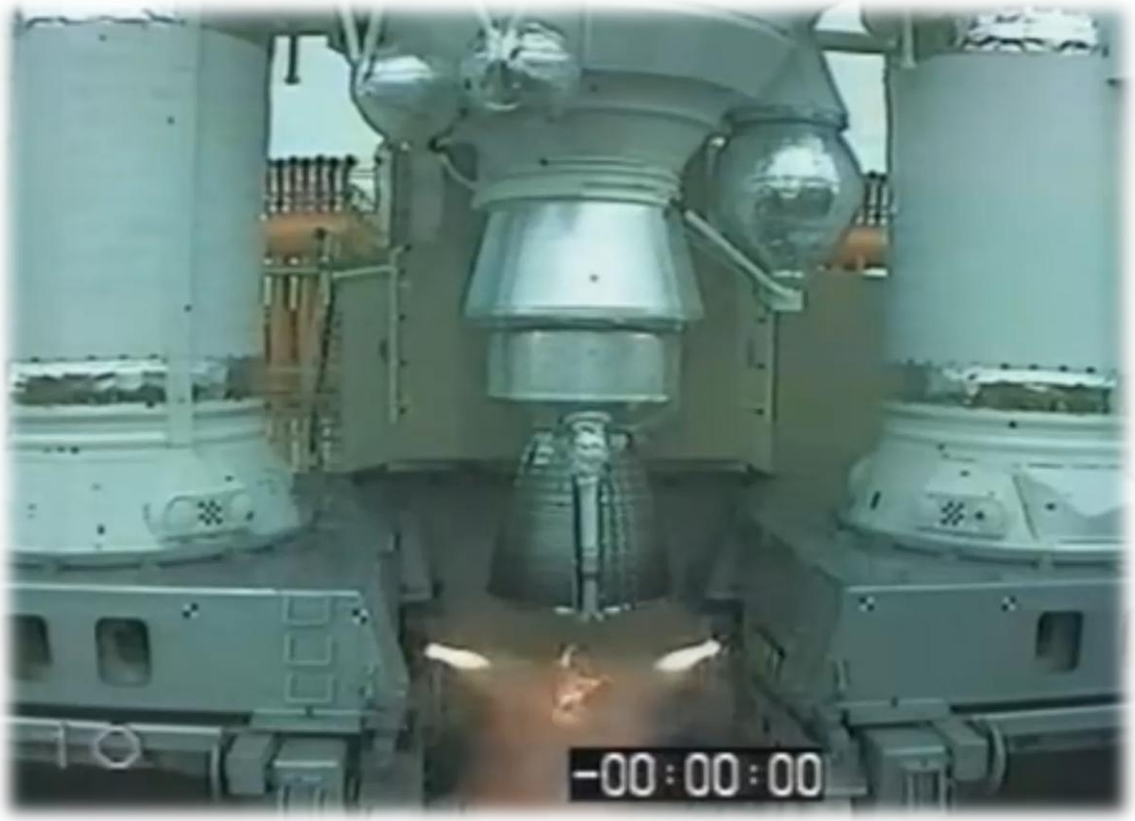


Paris, 19 July 1996

ARIANE 5

Flight 501 Failure

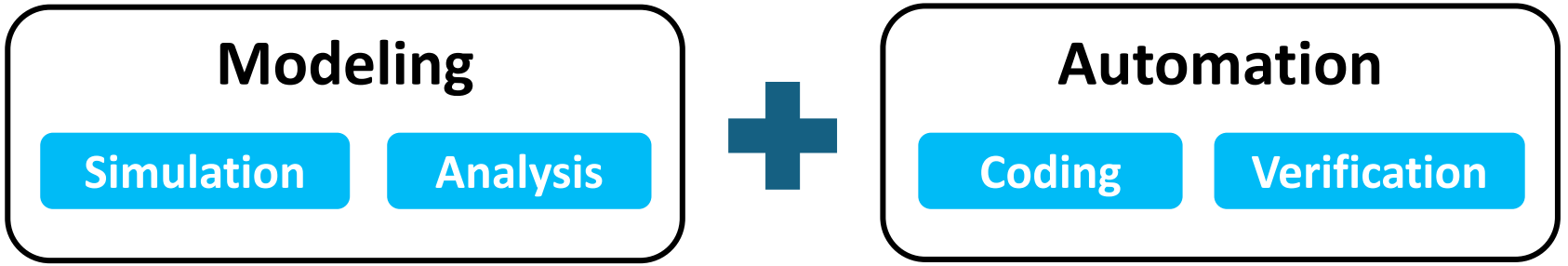
Report by the Inquiry Board



Overflow error

What is Model-Based Design?

Systematic use of **models** throughout the development process

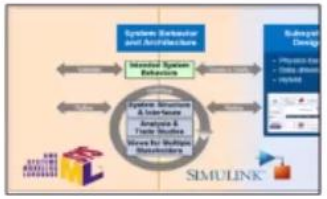


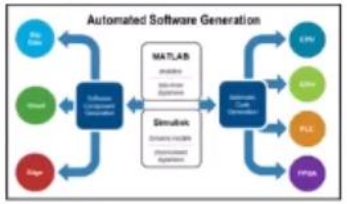
Try out new ideas
Fast repeatable tests

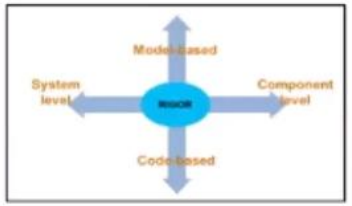
Eliminate manual steps
and **human error**

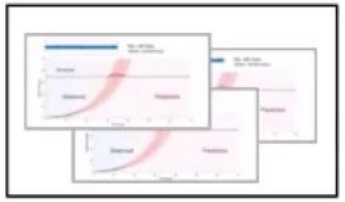
Why Model-Based Design?

System Requirements	System Functionality and Architecture	Subsystem Design	Subsystem Implementation	System Integration and Test	Operations and Sustainment
Use cases Requirement docs and models	Behavior models Architecture models	Physics-based Data-driven	Automatic codegen • C/C++ • VHDL/Verilog • GPU code Hand-code support	Model-based V&V Code-based V&V Certification workflows	Predictive maintenance Digital Twins



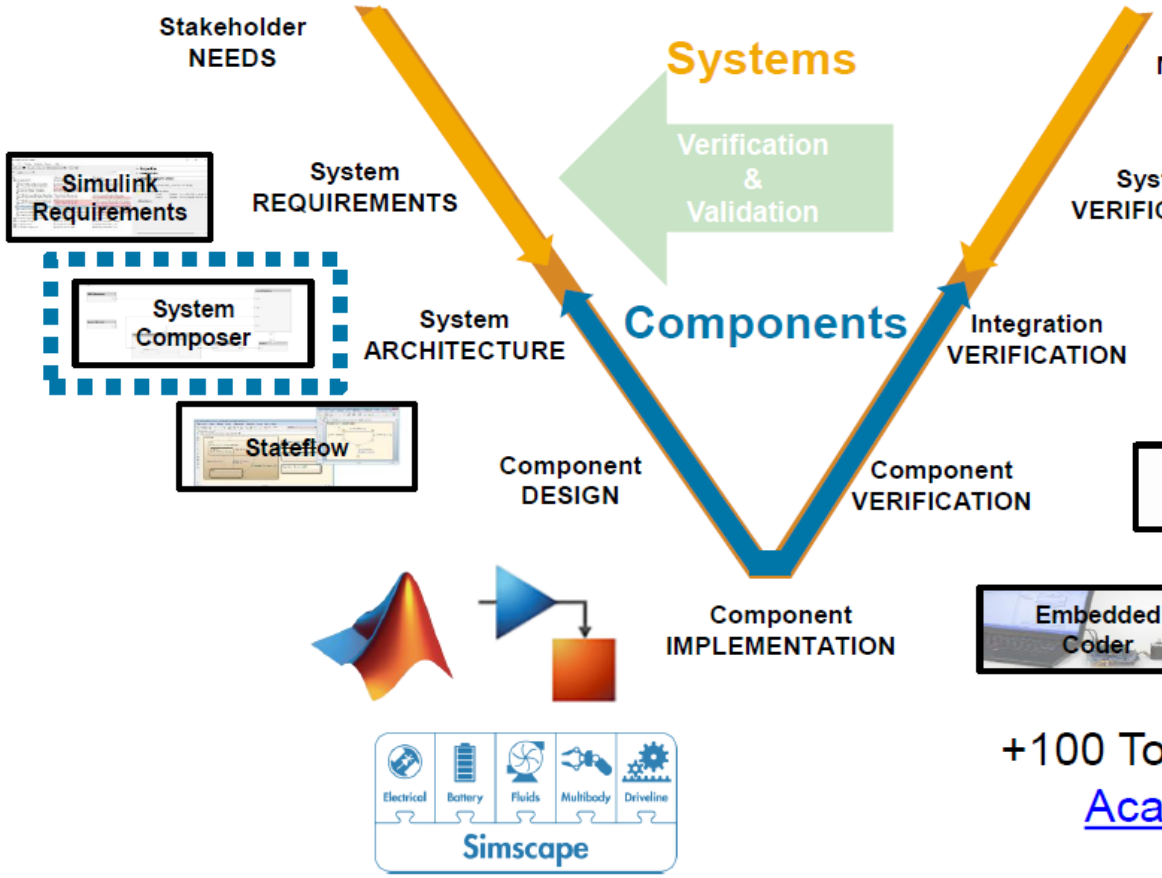






Digital Thread

Model-Based Design with MathWorks Tools



Speedgoat

NVIDIA

STM32

Raspberry Pi

ARDUINO

ARM®

C2000

XILINX®

ZYNQ™

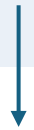
C, C++, HDL, PLC, and CUDA* code

+100 Toolboxes and Blocksets included in [Academics Campus-Wide License]

Data-Driven Versus Physics-Based Modeling

Data-driven

- Data sources
  - Experimental data
  - External tool (complexity-reduction exercise)
- Complex behaviour with many unknowns
- Examples
  - Combustion engine
  - Aerodynamics



- Machine Learning Algorithms
- System Identification
- Reduced Order Model (ROM)

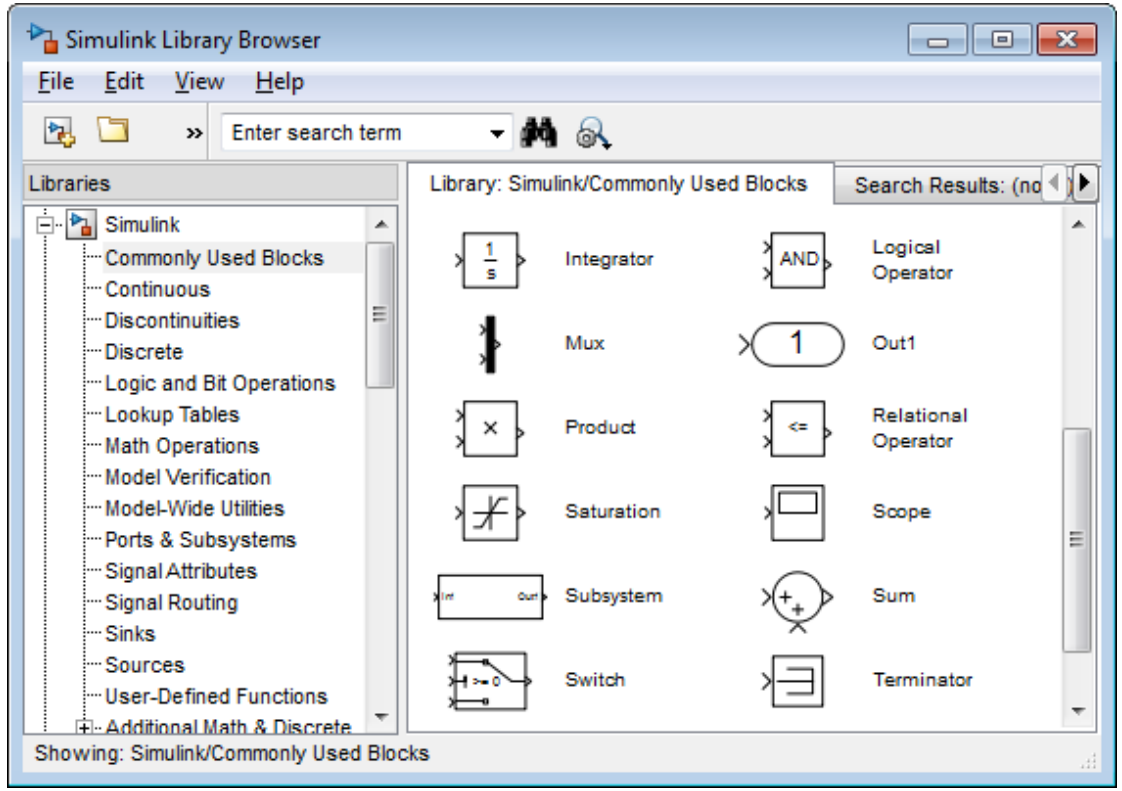
Physics-based

- Mathematical paradigms
  - ODEs & DAEs
  - State charts
  - Physical networks
- Behaviour is adequately approximated by limited set of equations
- Examples
  - Algorithms/control
  - Gears & clutches

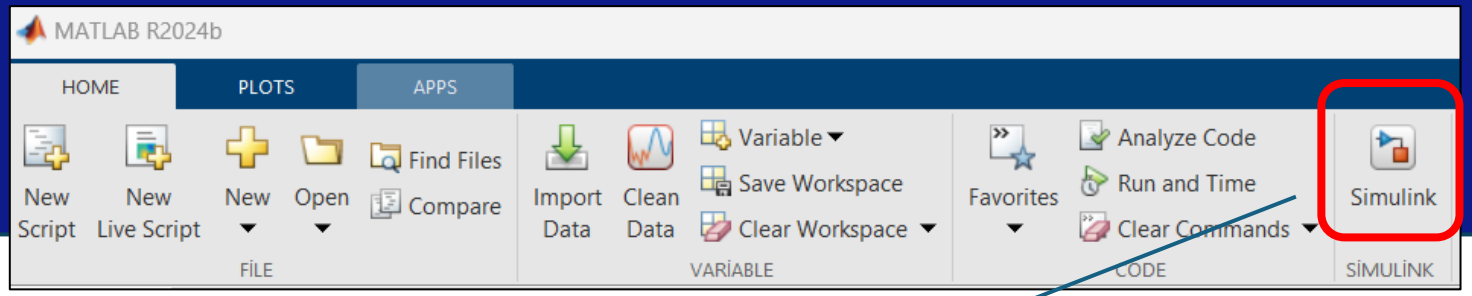
MathWorks Products Overview - **Simulink**

SIMULINK®

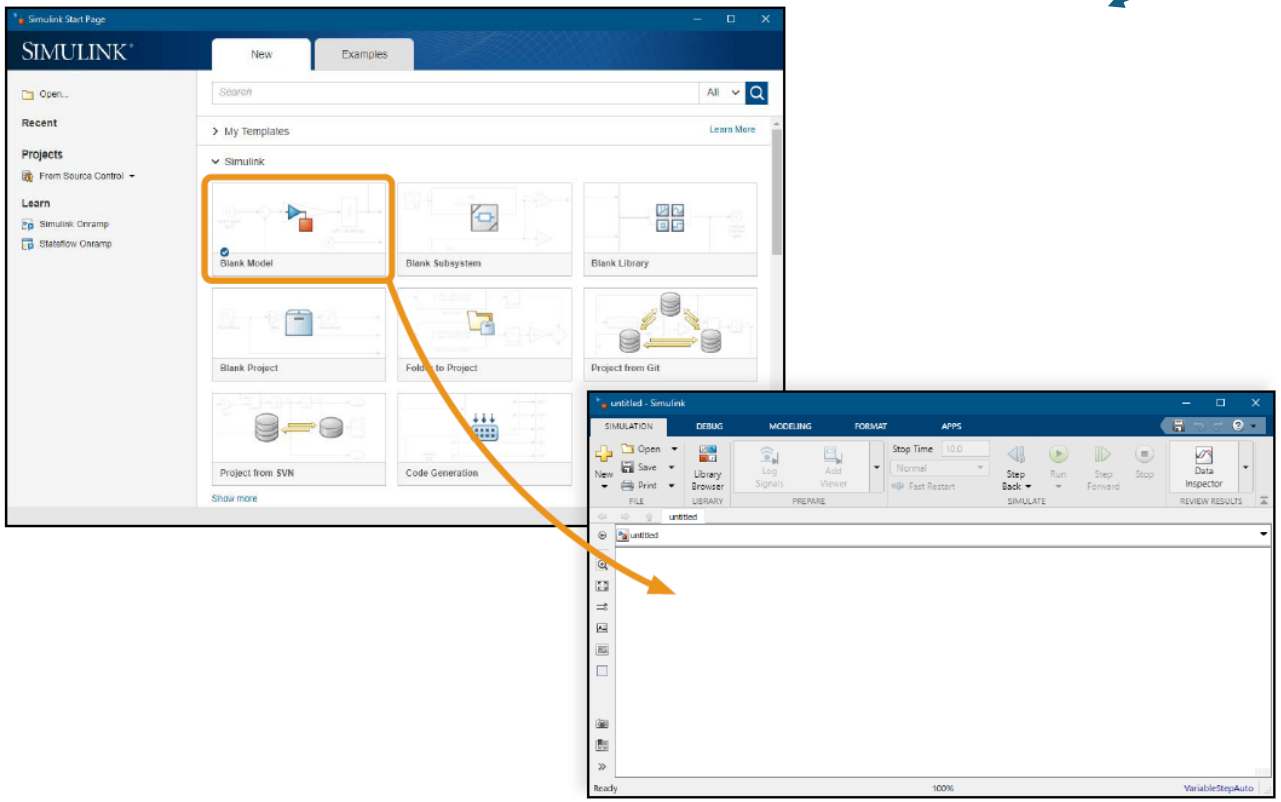
- **Block-diagram** environment(**.slx**)
- Model, simulate, and analyze multi-domain systems
- Design, implement, and test:
  - **Control systems**
  - **Signal processing systems**
  - **Communications systems**
  - **Other dynamic systems**
- Platform for **Model-Based Design**
- Open architecture with links to **third-party tools** and development boards, and instrumentation



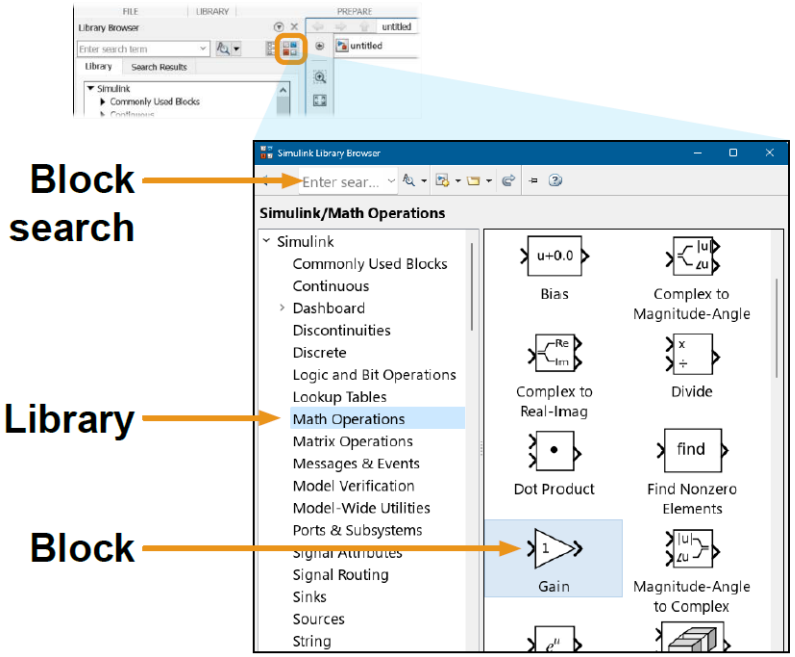
Simulink



Creating a New Simulink Model

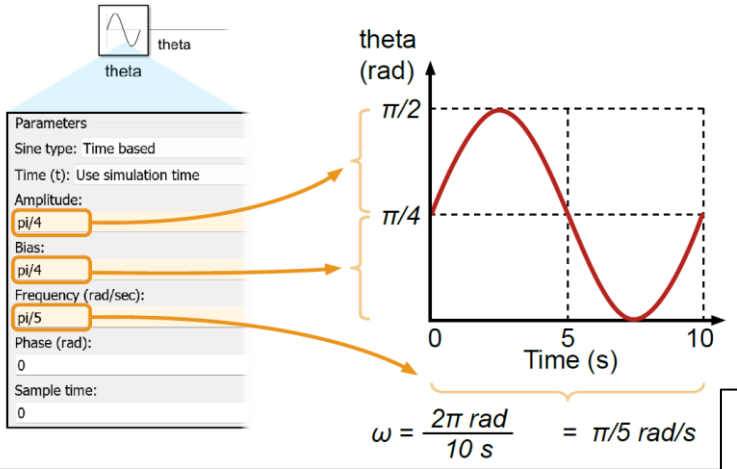


Using the Simulink Library Browser

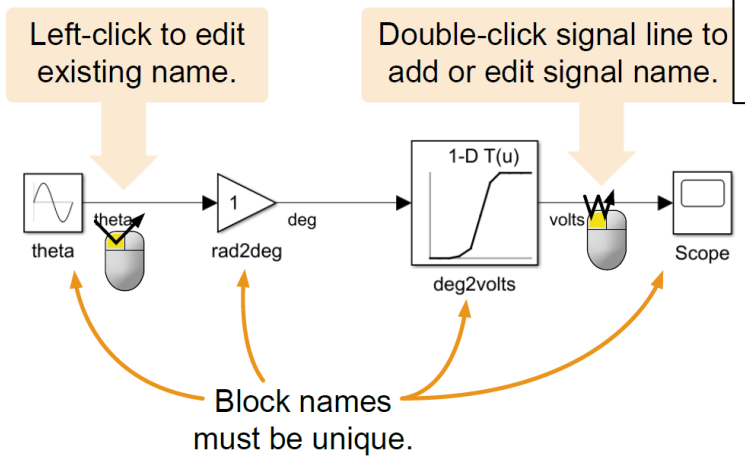


Define Simulink Block Parameters

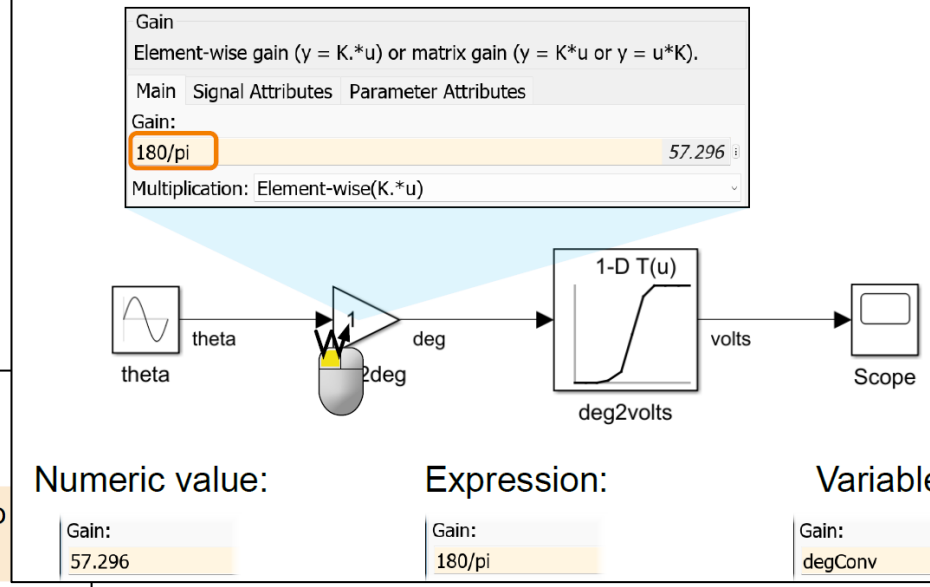
Defining Sine Wave Parameters



Labeling Blocks and Signals



Defining Block Parameters

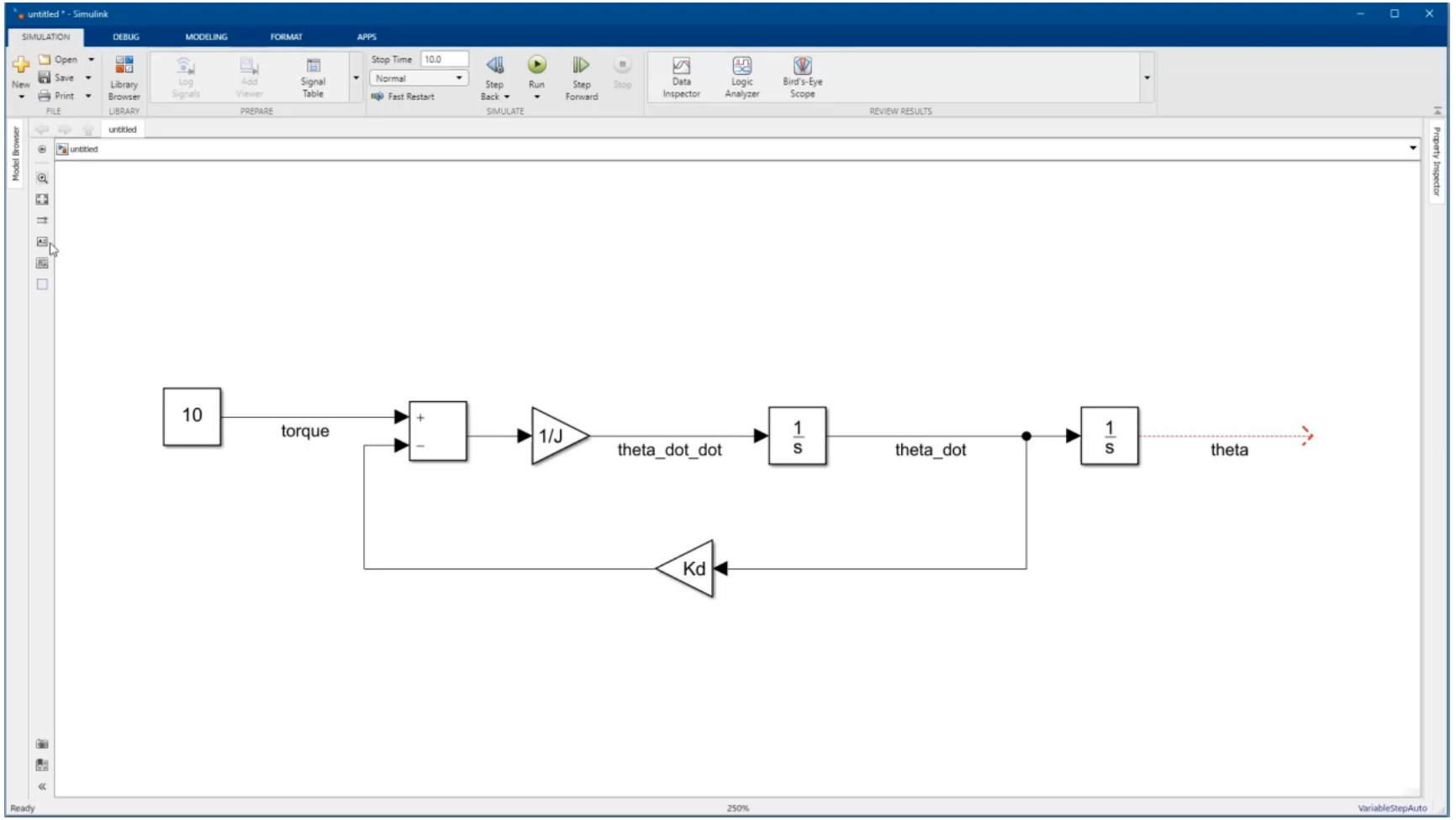


Numeric value:

Expression:

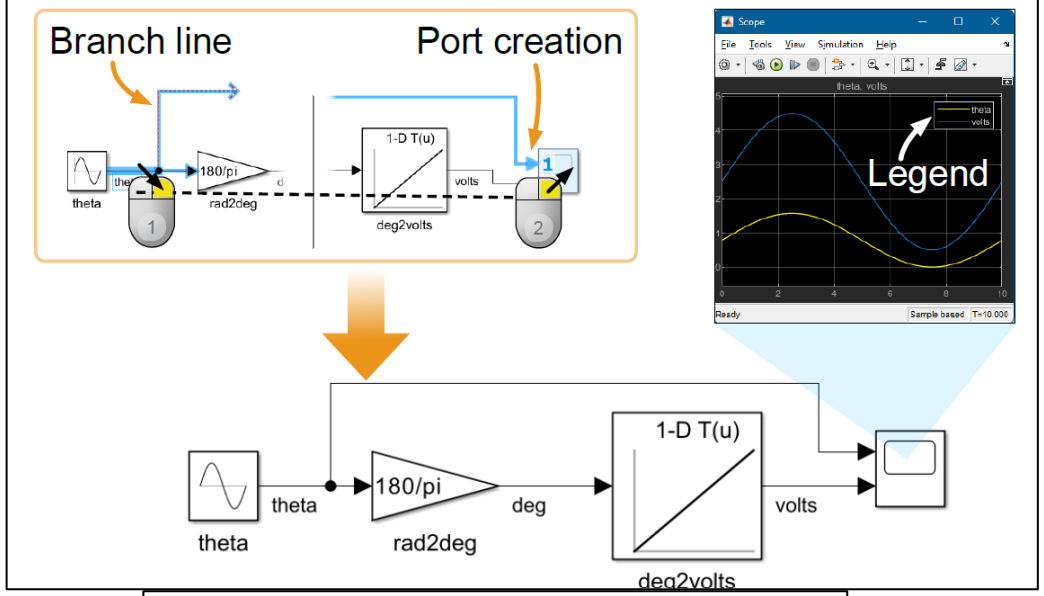
Variable:

Modeling in Simulink

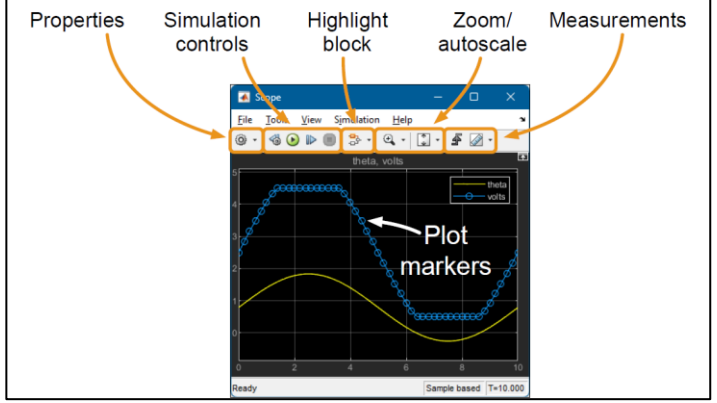


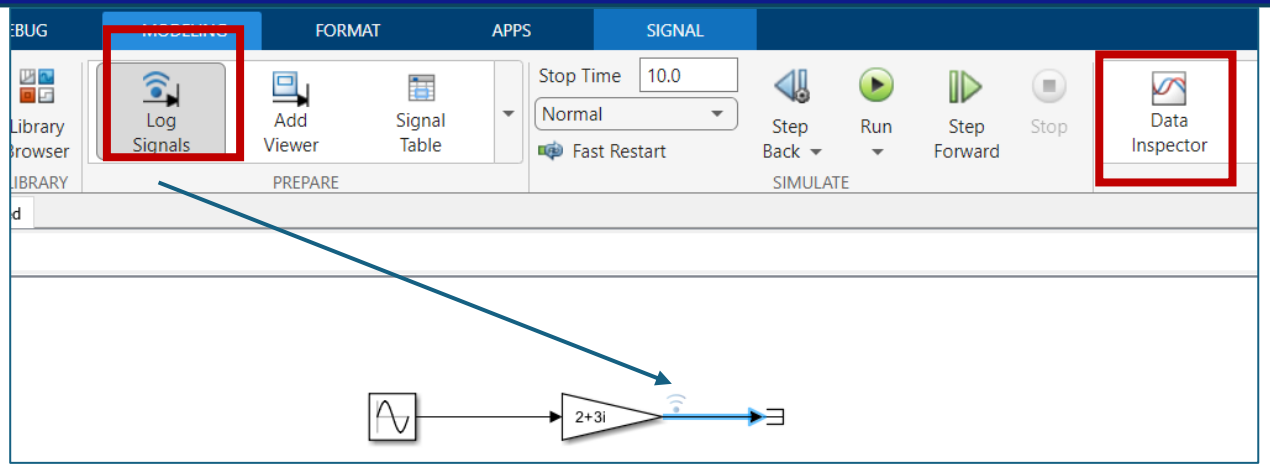
Scope & Data Inspector

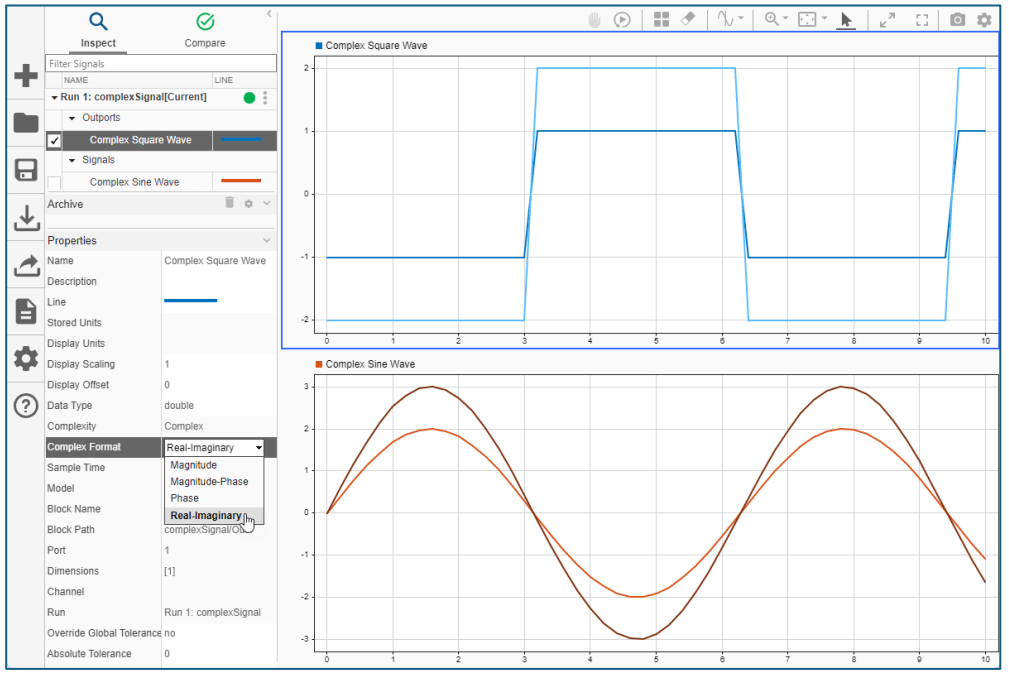
Plotting Multiple Signals



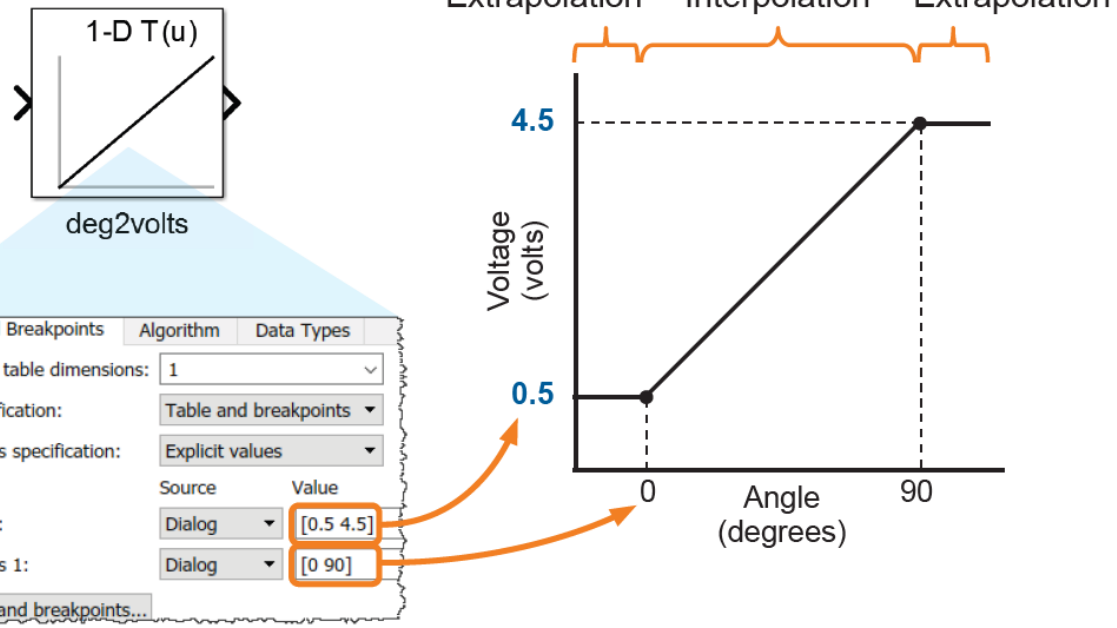
Customizing Scope Appearance

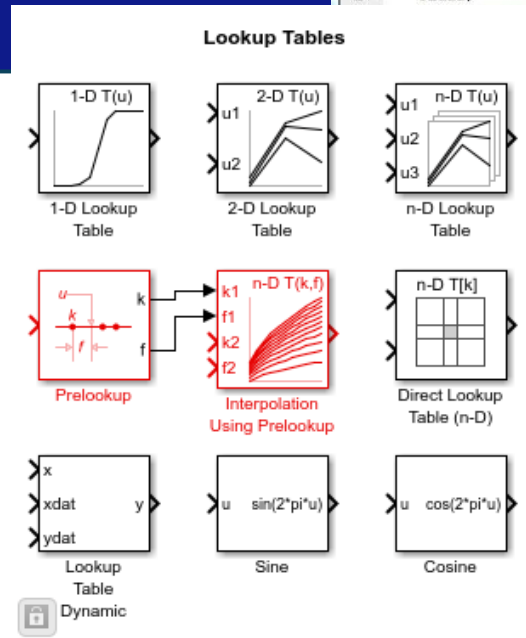


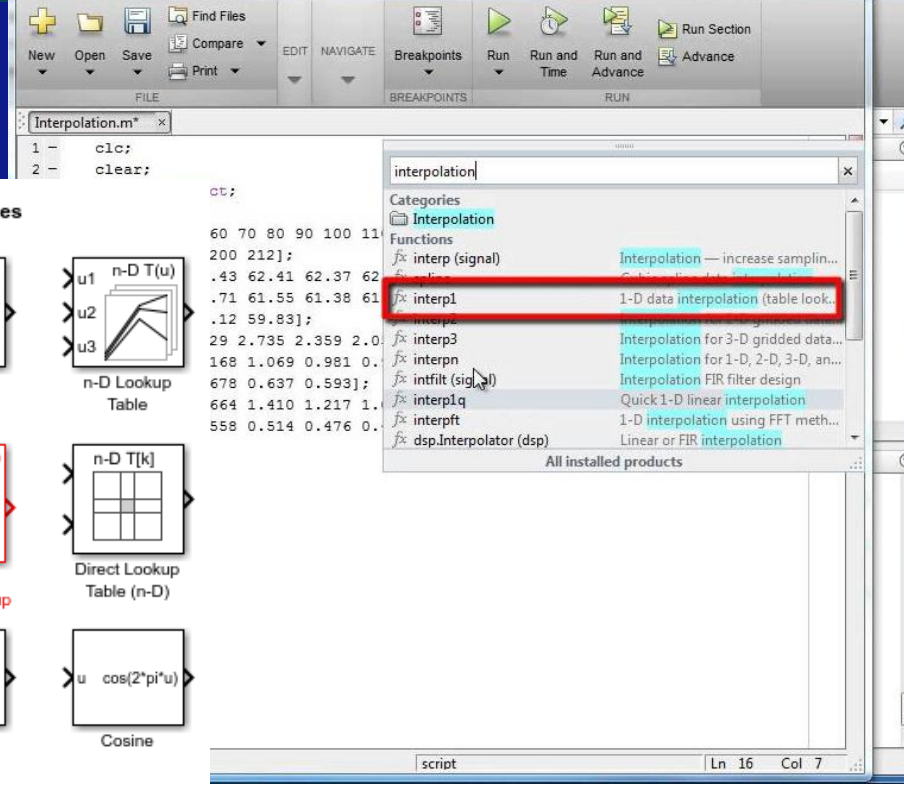


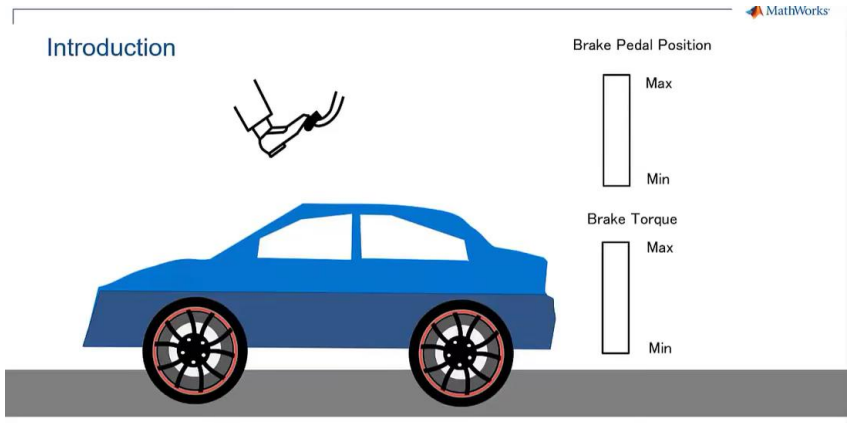


Defining Lookup Table Parameters



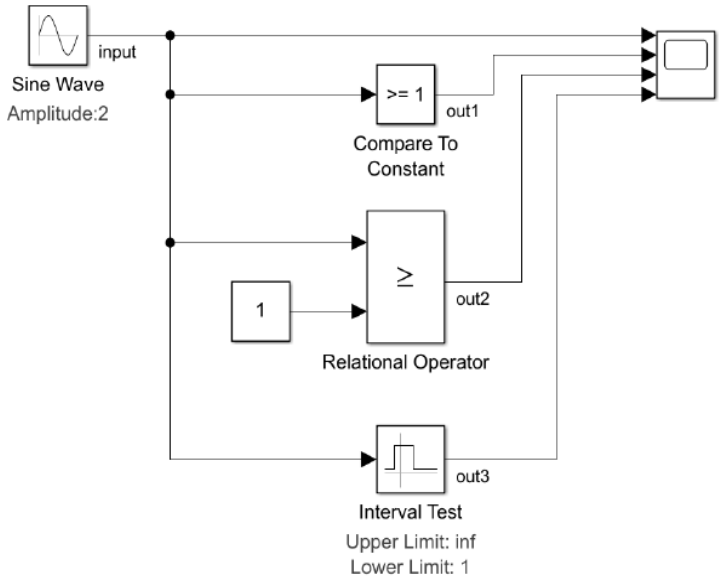






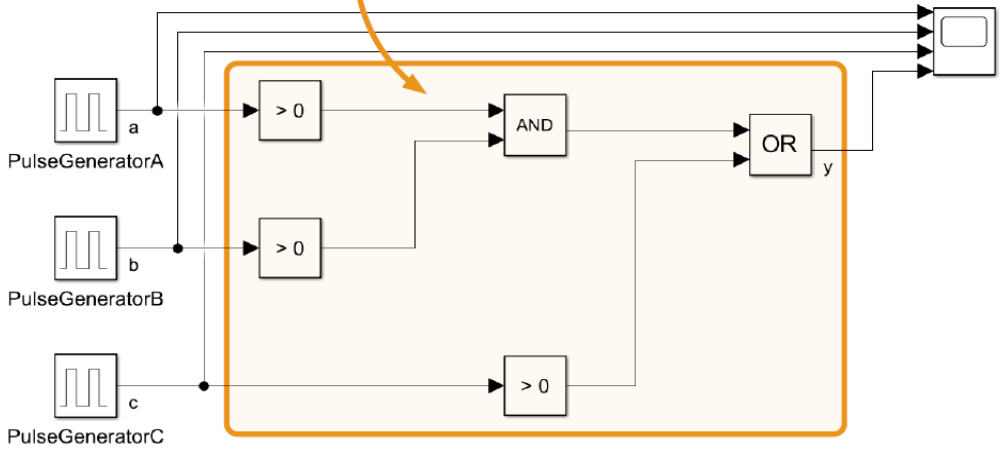
Modeling Programming Constructs

- Relational operators (`<`, `>`, `==`, etc.)
- Logical operators (`AND`, `OR`, `NOT`, etc.)
- Decision statements (`if-else`, `switch-case`, etc.)



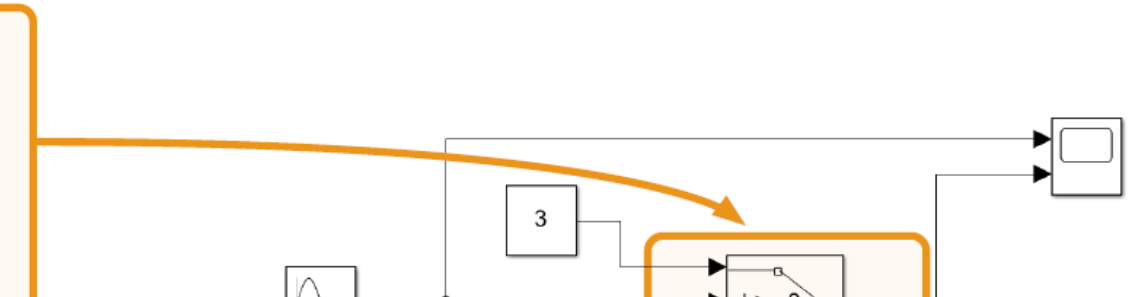
Modeling Comparisons

```
y = ((a > 0) & (b > 0)) | (c > 0)
```

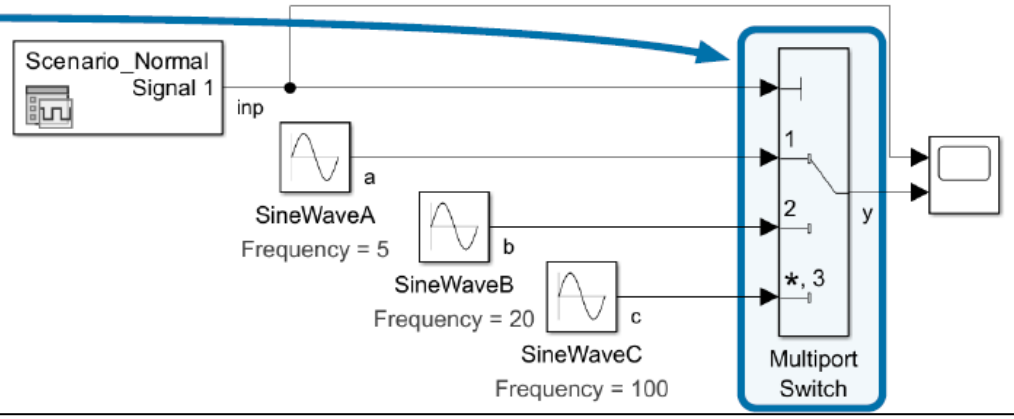


Modeling Decision Statements

```
if input >= 0
  output = 3;
else
  output = -3;
end
```

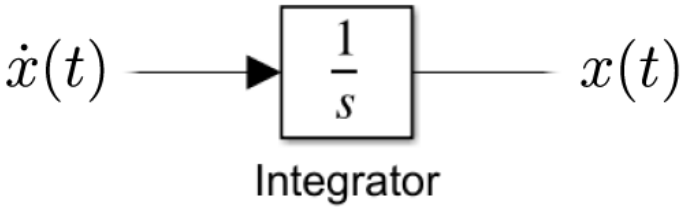



```
switch inp
  case 1
     y = a;
  case 2
     y = b;
  case 3
     y = c;
  otherwise
     y = c;
end
```

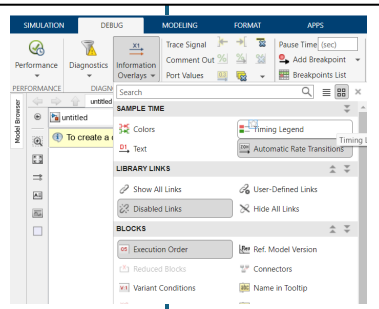


Continuous & Discrete Systems in Simulink

Continuos Systems



- `Ts = 0` – Continuous block execution
- `Ts > 0` – Discrete block execution
- `Ts = -1` – Sample time is inherited



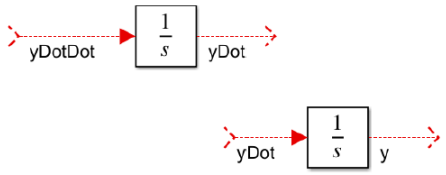
1

$$\ddot{y}(t) = u(t) - y(t) - \dot{y}(t)$$

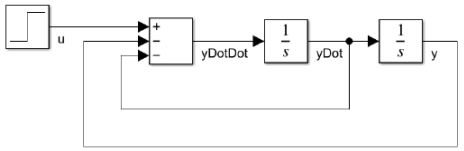
Two Integrator blocks:

$$\ddot{y}(t) \rightarrow \dot{y}(t)$$
$$\dot{y}(t) \rightarrow y(t)$$

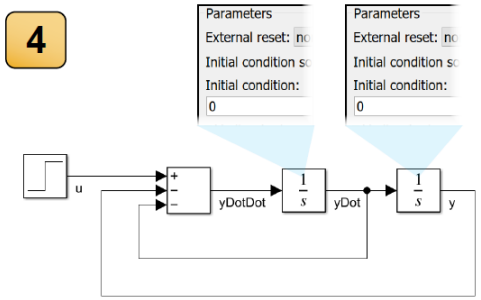
2



3

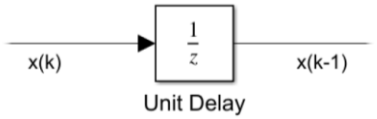


4



Discrete Systems

The Unit Delay block is the generic building block of difference equations.



Sample time `0.5`

Time	x(k)	x(k-1)
0	0.48	0.00
0.5	0.84	0.48
1.0	1.00	0.84
1.5	0.91	1.00

Initial condition `0.0`

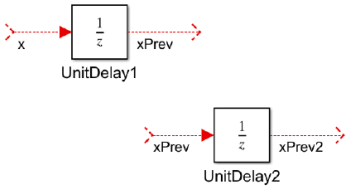
1

$$x(k) = x(k-1) - x(k-2) + u(k)$$

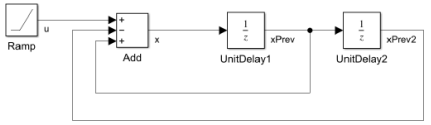
Two Unit Delay blocks:

$$x(k) \rightarrow x(k-1)$$
$$x(k-1) \rightarrow x(k-2)$$

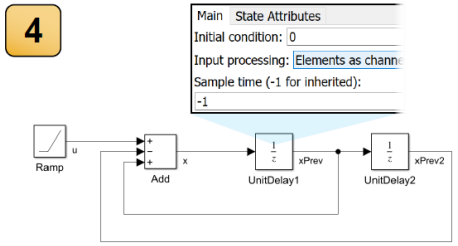
2



3

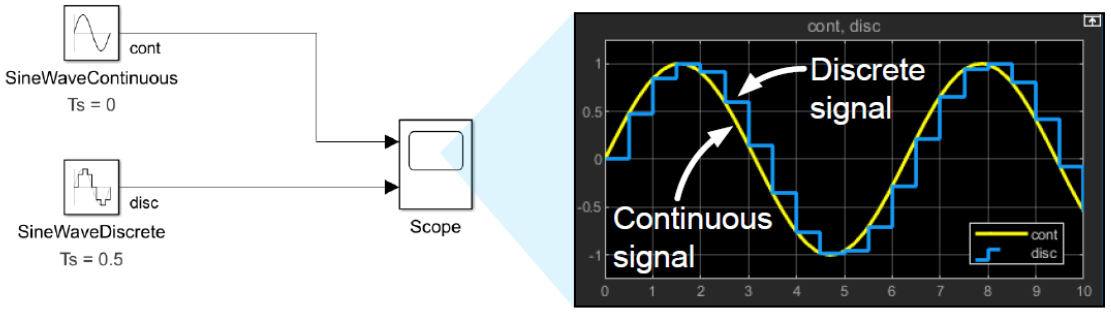


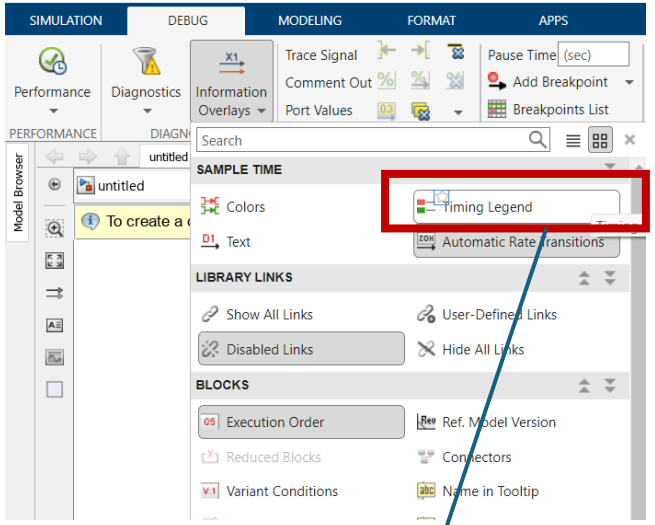
4

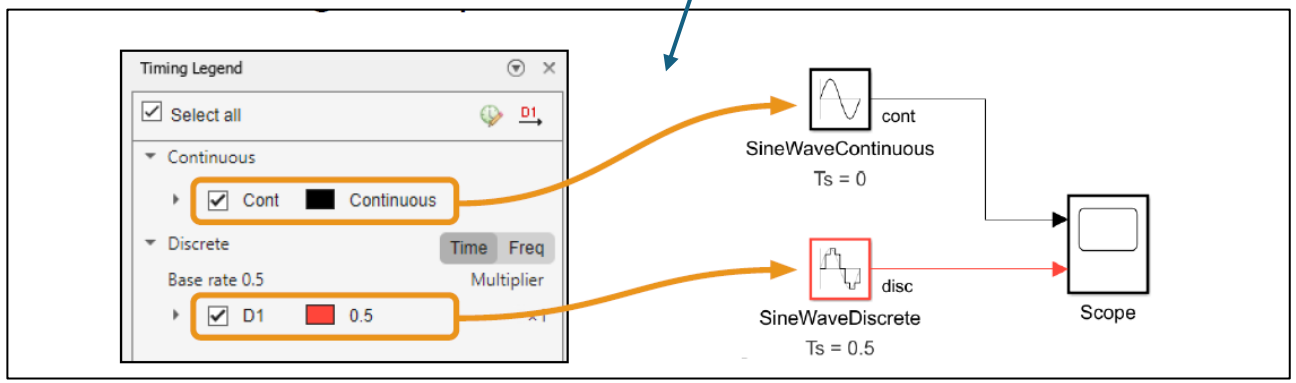


Sample Time

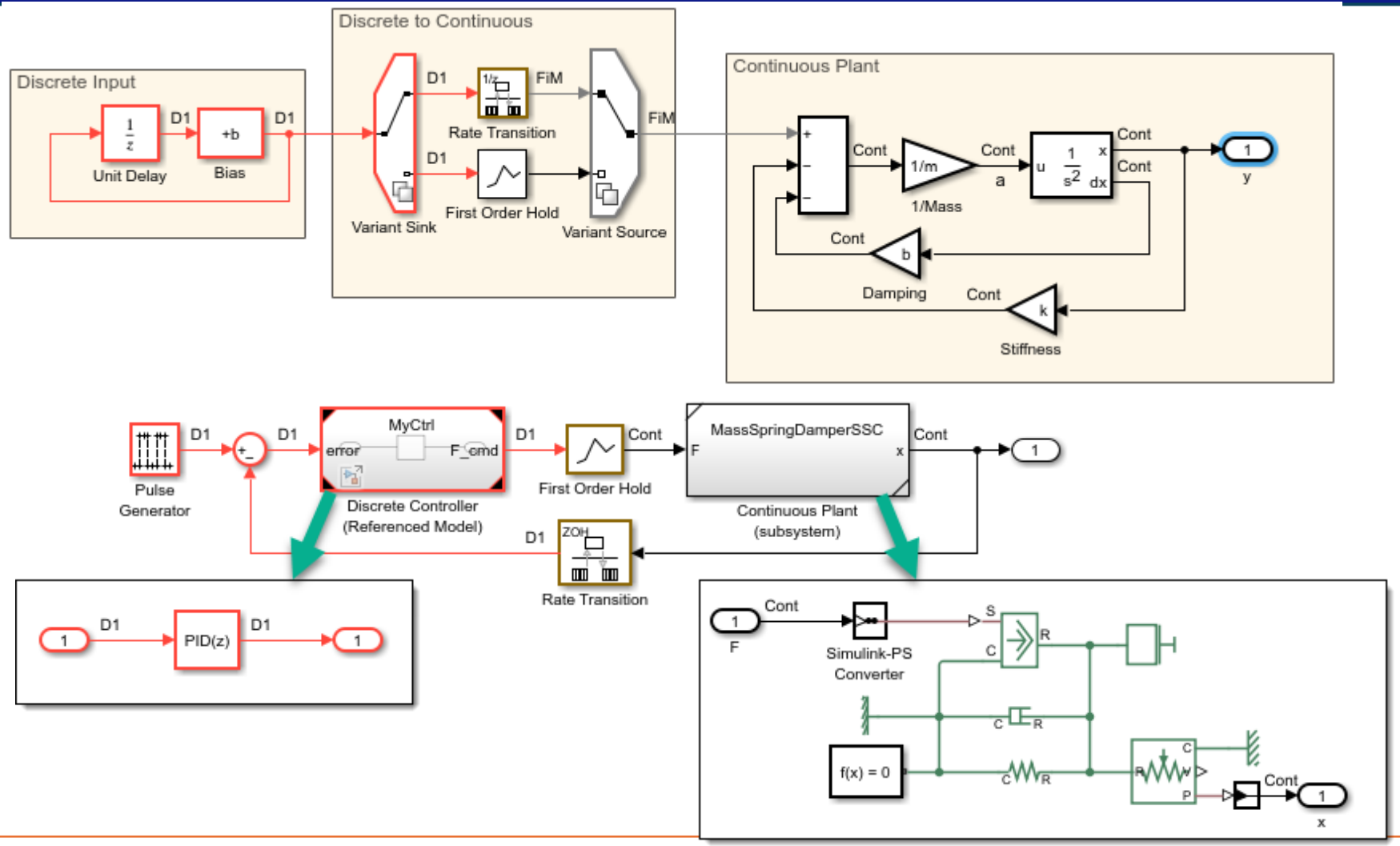
- `Ts = 0` – Continuous block execution
- `Ts > 0` – Discrete block execution
- `Ts = -1` – Sample time is inherited



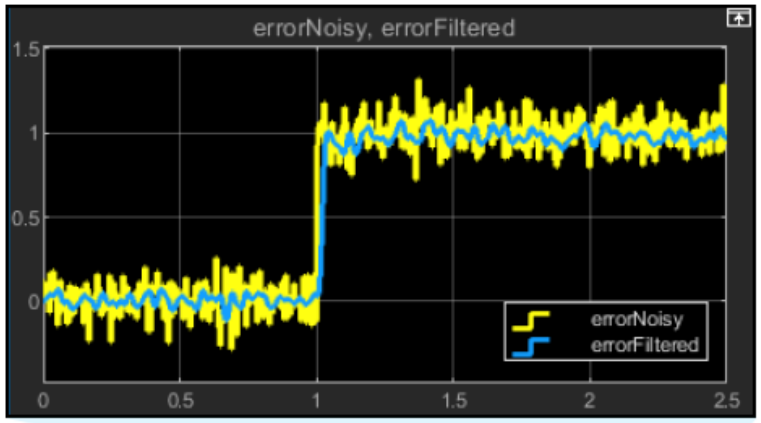


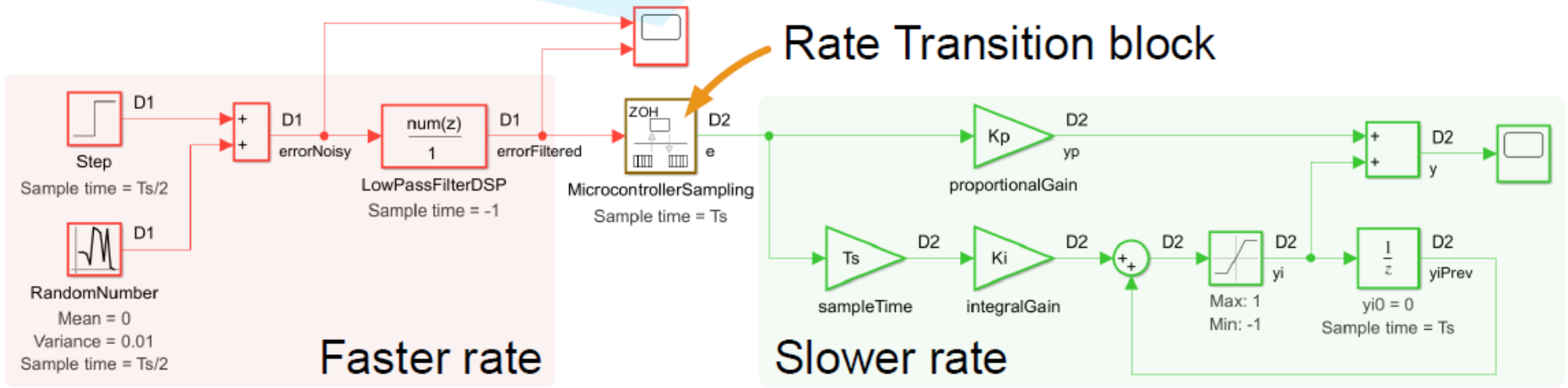


Continuous & Discrete Systems in Simulink

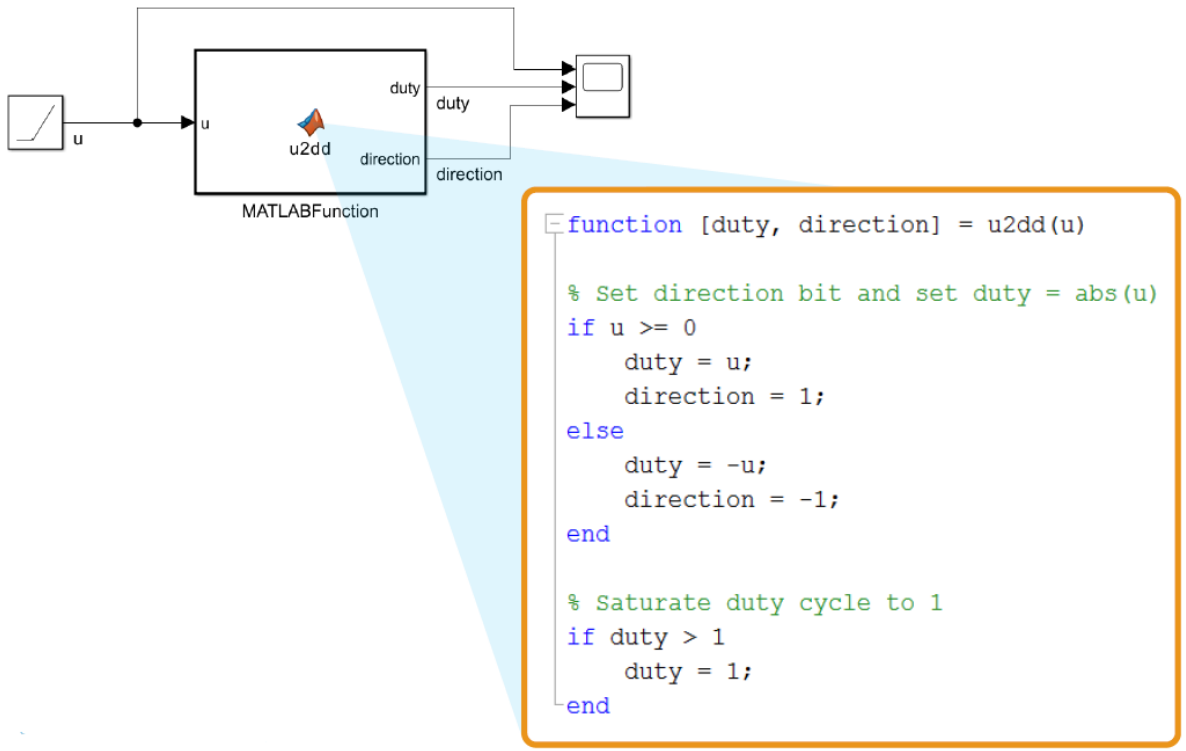


Multirate Systems



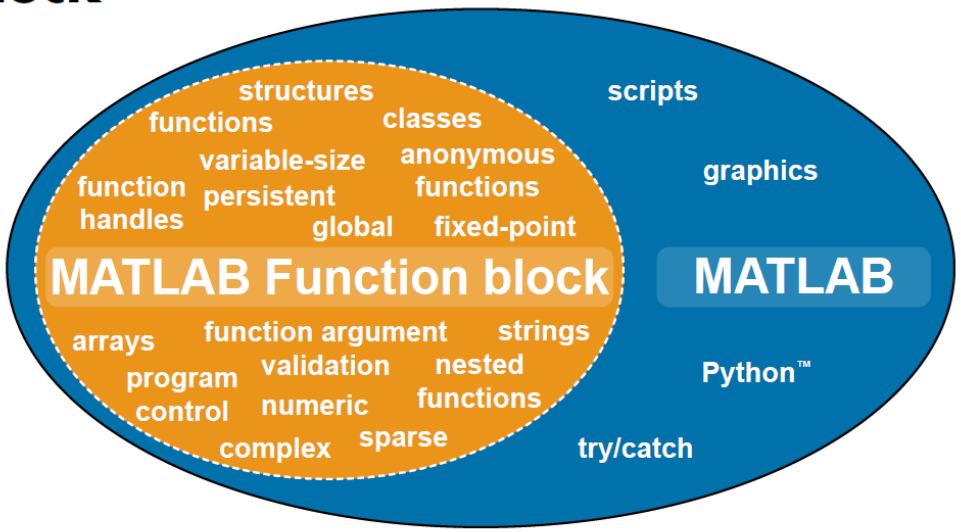


Modeling with MATLAB Function Blocks




```
function [duty, direction] = u2dd(u)

% Set direction bit and set duty = abs(u)
if u >= 0
    duty = u;
    direction = 1;
else
    duty = -u;
    direction = -1;
end

% Saturate duty cycle to 1
if duty > 1
    duty = 1;
end
```

Understanding the MATLAB Function Block



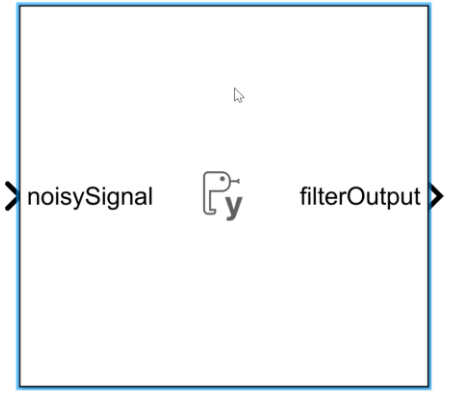
- Subset of supported functions and language features
- Unsupported functions call MATLAB interpreter

Python Code Block

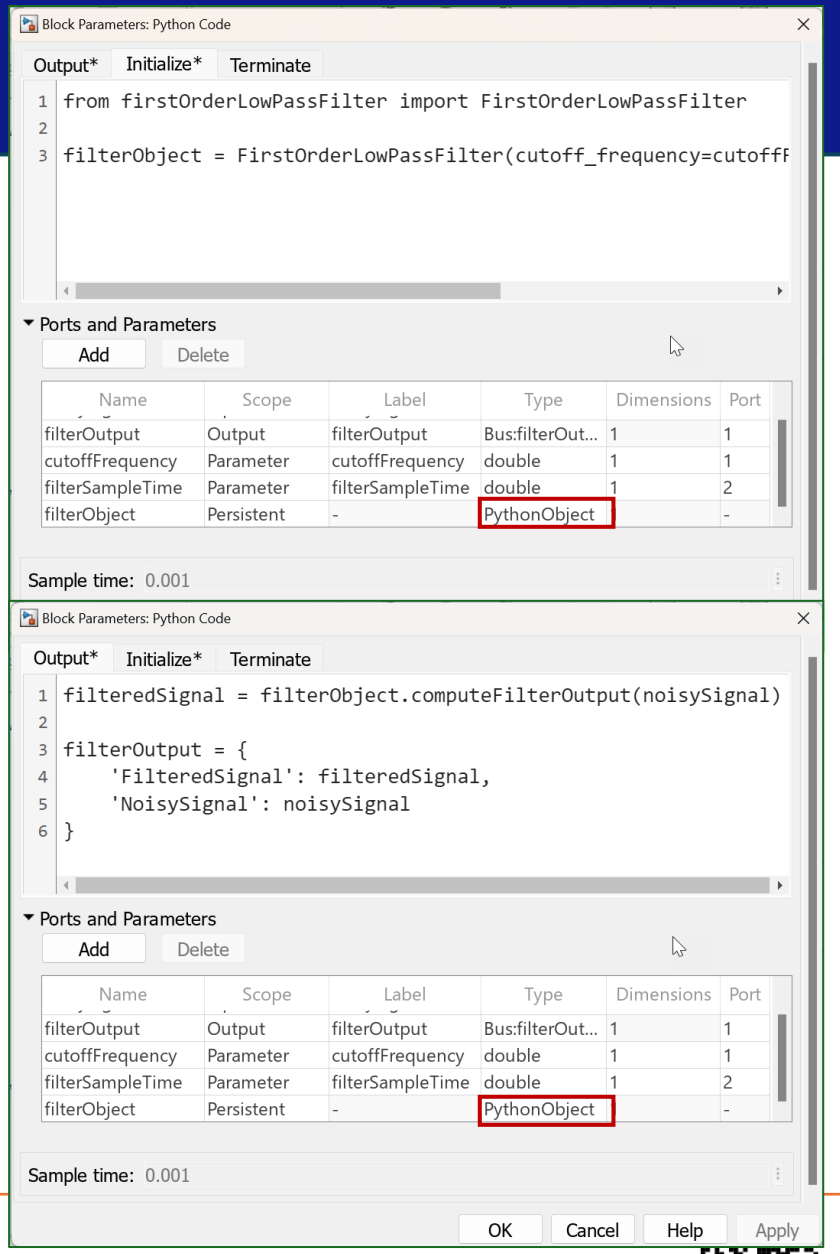
- Easily integrate external Python code into Simulink

R2025a

- Call external Python libraries, functions or methods from Simulink
- Write **native Python code** in the block dialog to support the interface between Simulink and Python
  - No need for “py.” prefix
- Supports all simulation modes
  - normal, accelerator, rapid accelerator, model reference accelerator



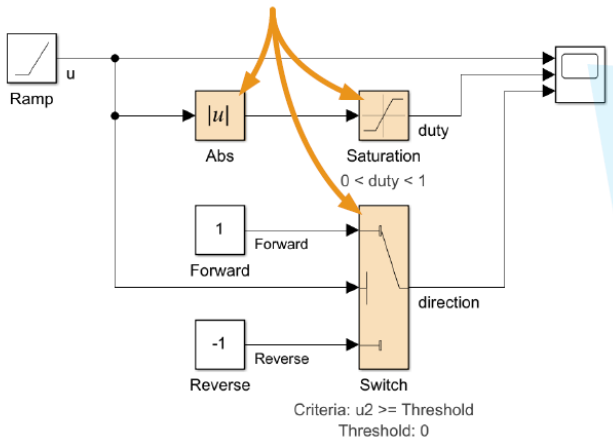
- **Python Code Block** vs **Python Importer**
  - Flexibility vs Reusability



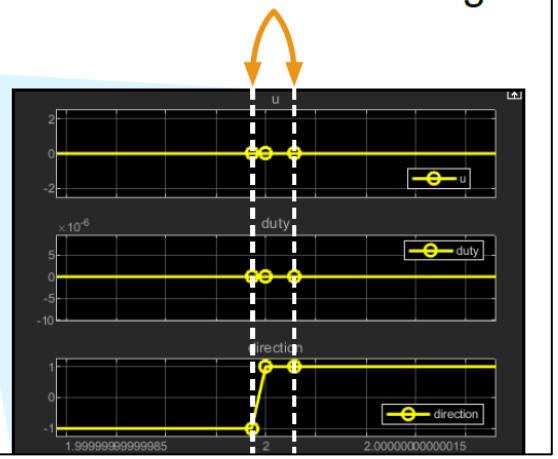
Zero Crossing Detection

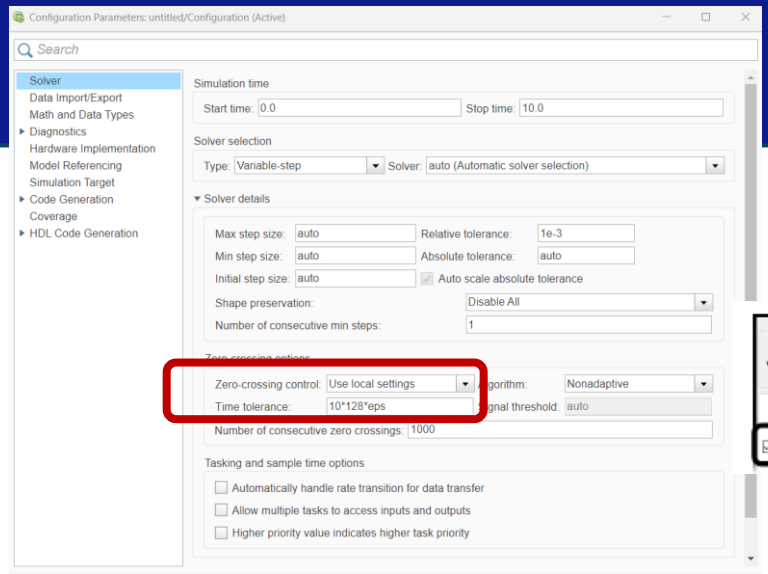
Blocks with discontinuous behavior can cause the solver to take additional time steps before and after a discontinuity.

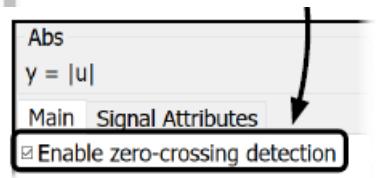
Blocks with zero-crossing detection

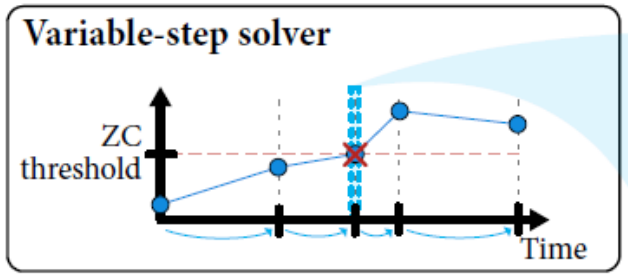


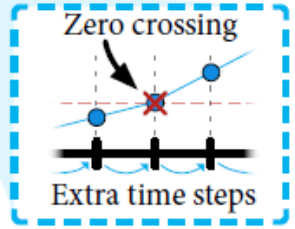
Extra time steps taken around a zero crossing

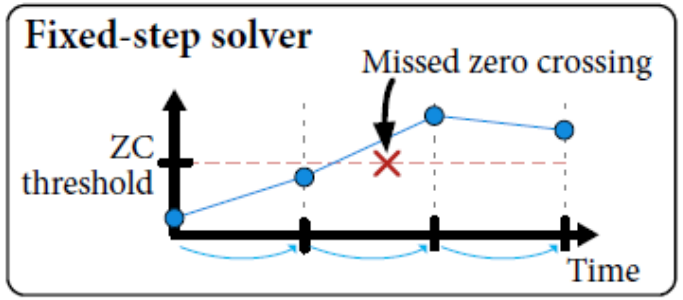










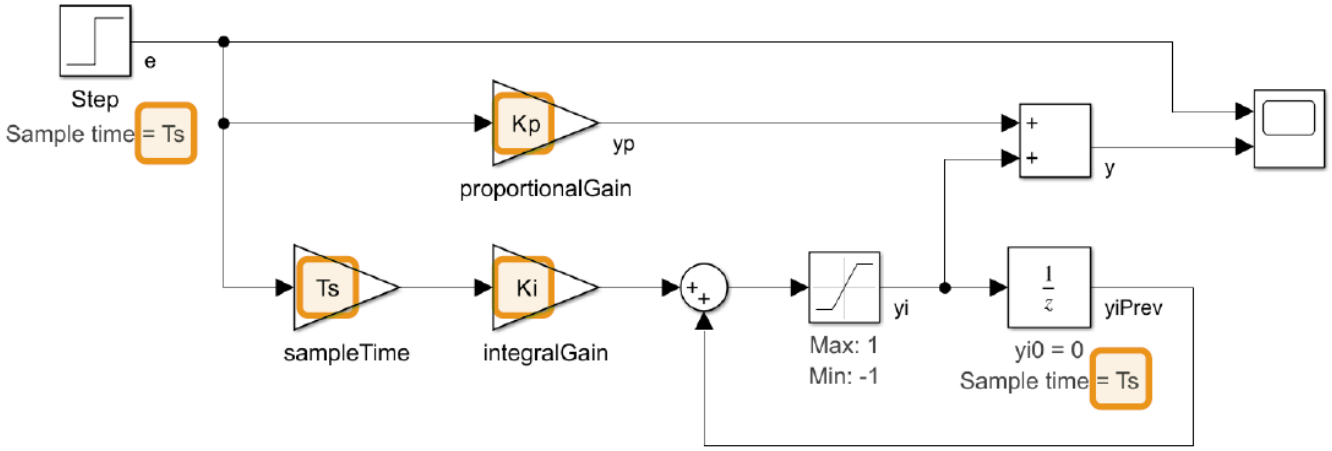


Parameterizing in Simulink

- Make model more readable
- Edit parameters from workspace

Workspace

Name	Value
Ki	2
Kp	0.3
Ts	0.01

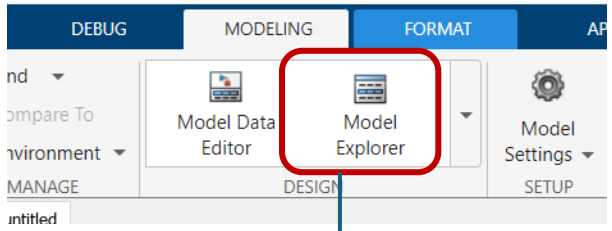


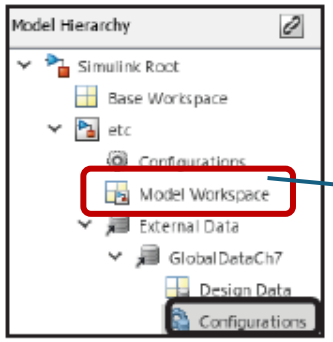
Workspace

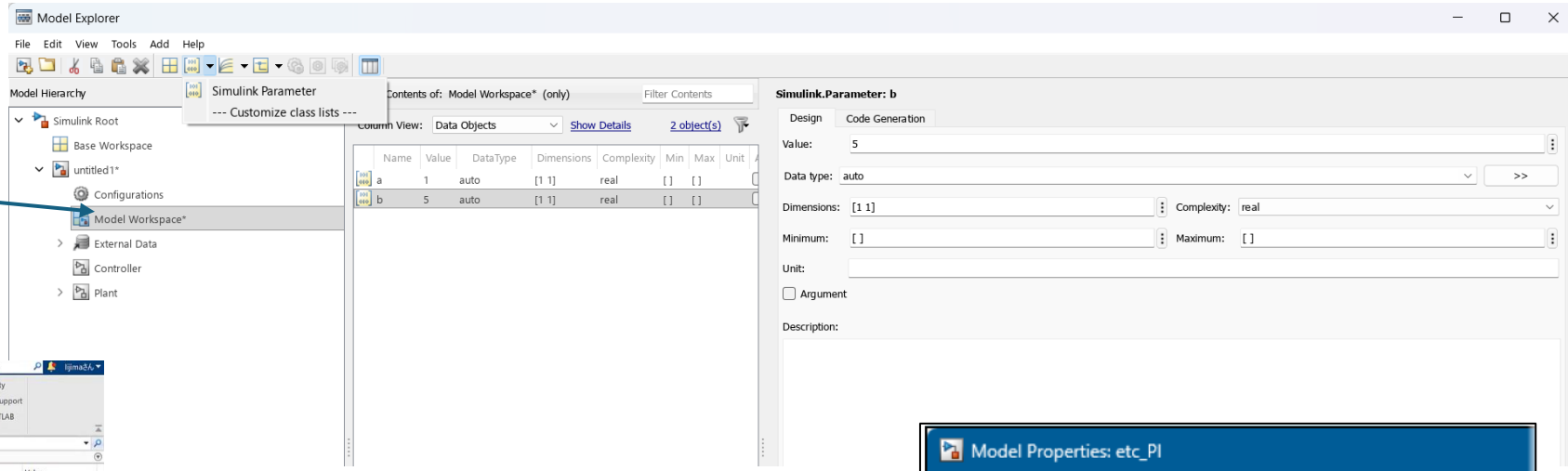
Name	Value	Size	Class	Min	Max
A	[1;2;3]	3×1	double	1	3
b	5	1×1	uint32	5	5
C	*5×5 cell*	5×5	cell		
str	"abc"	1×1	string		
S	*0×0 struct*	0×0	struct		
T	*3×2 table*	3×2	table		

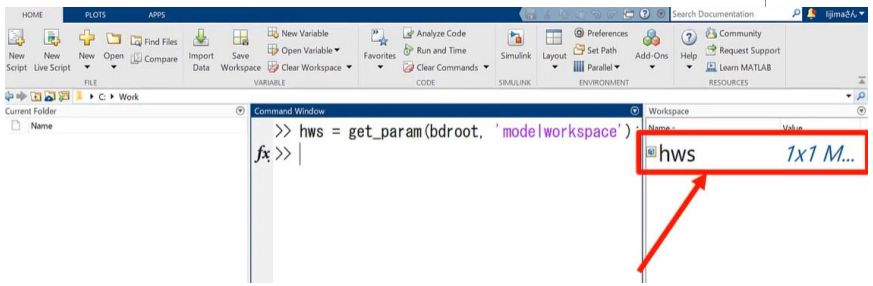
Base Workspace

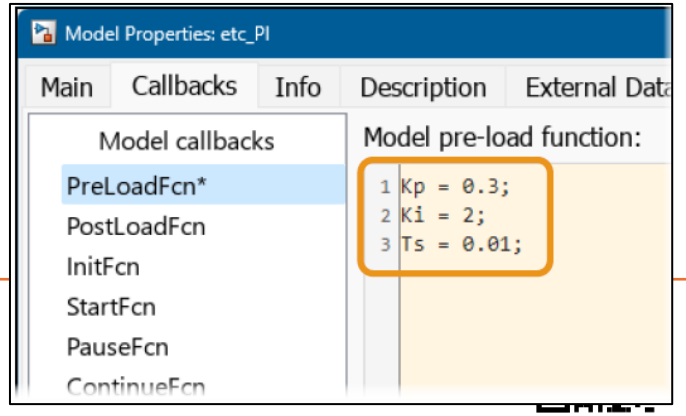
Model Workspace



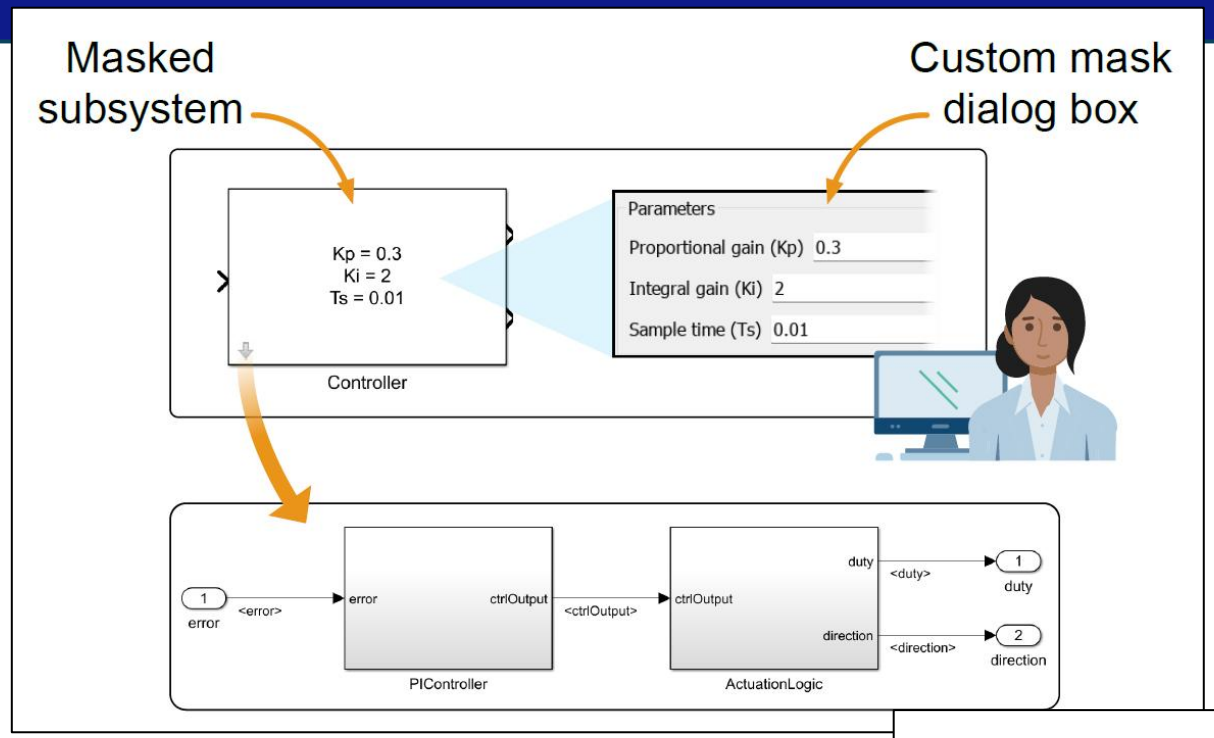


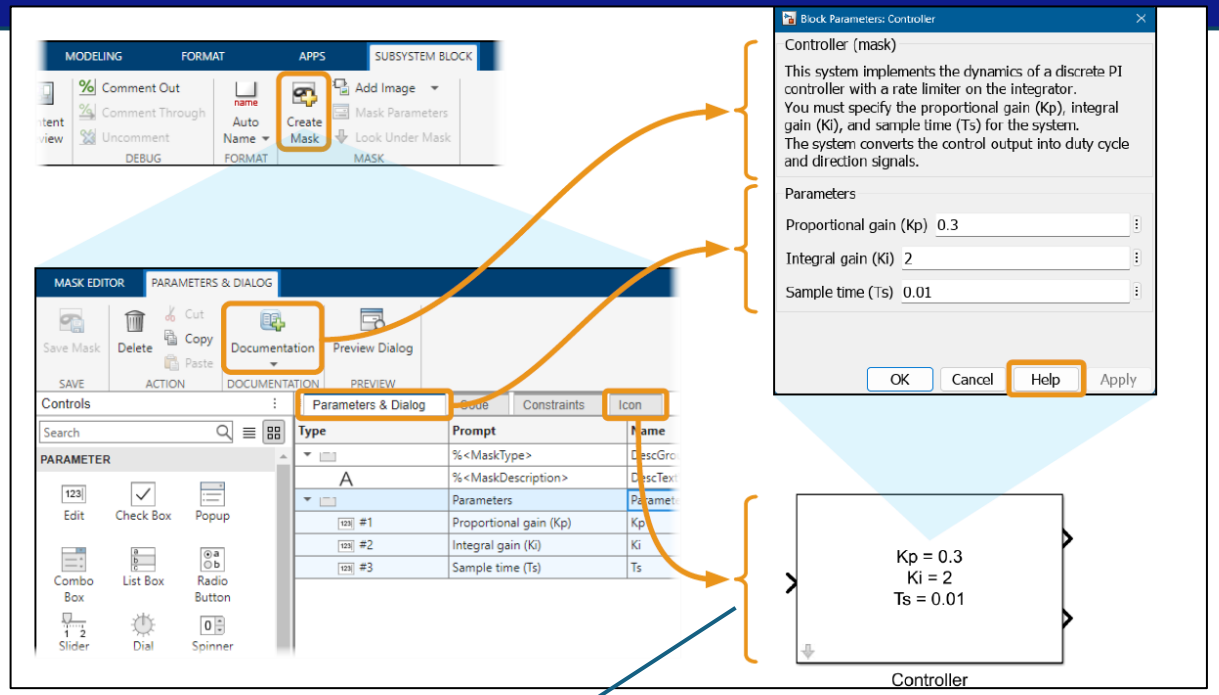


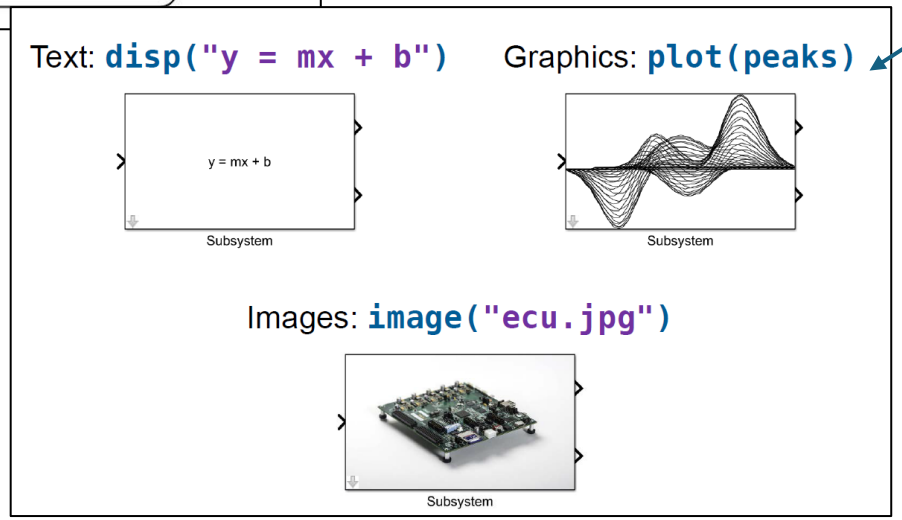




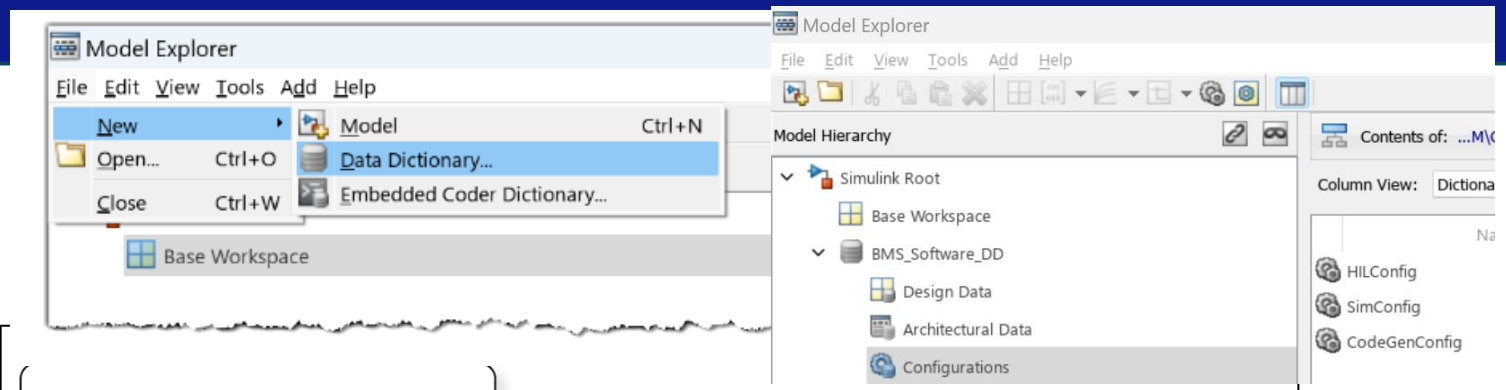
Creating Mask

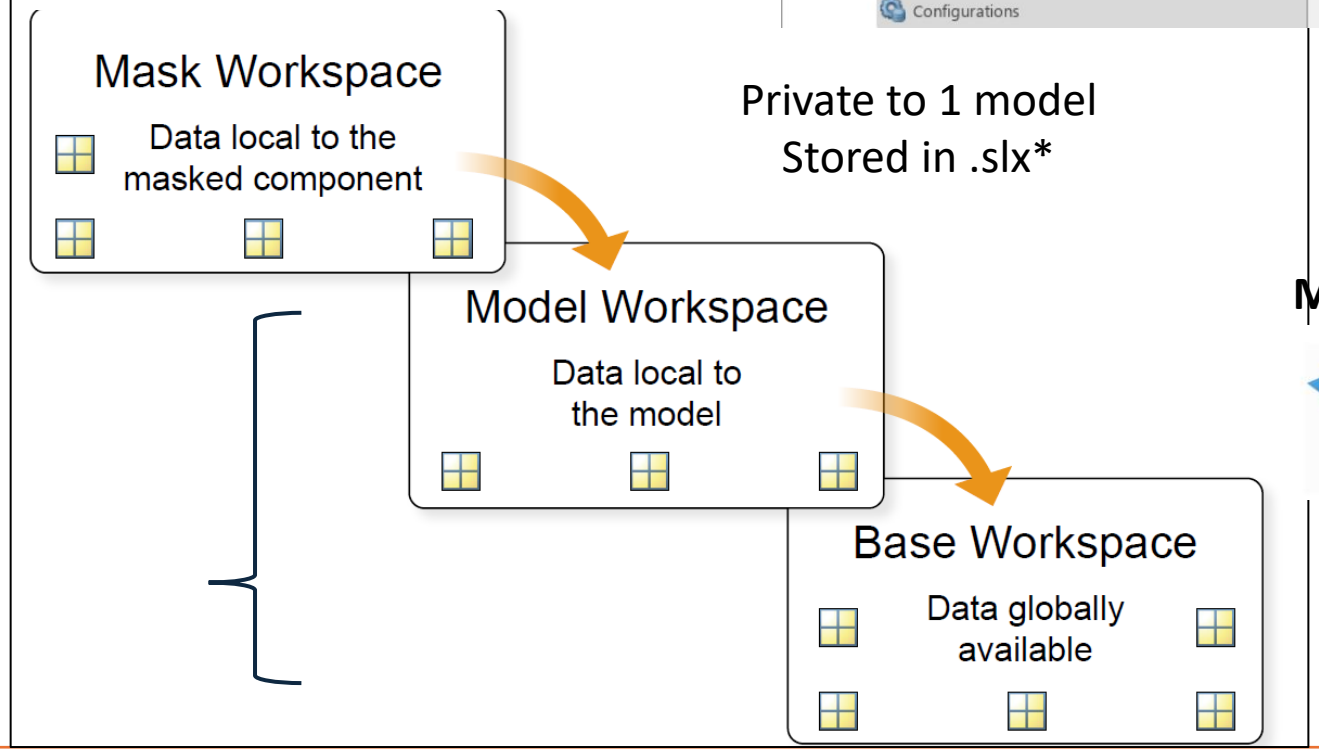






Simulink's options for data storage





Private to 1 model
Stored in .slx*

SLDD

Data Dictionary

Dedicated, persistent file
1 or multiple models

Designed for Simulink.Scalability and partitioning: can reference and be referenced by other SLDDs. Can only be diffed and merged in MATLAB.

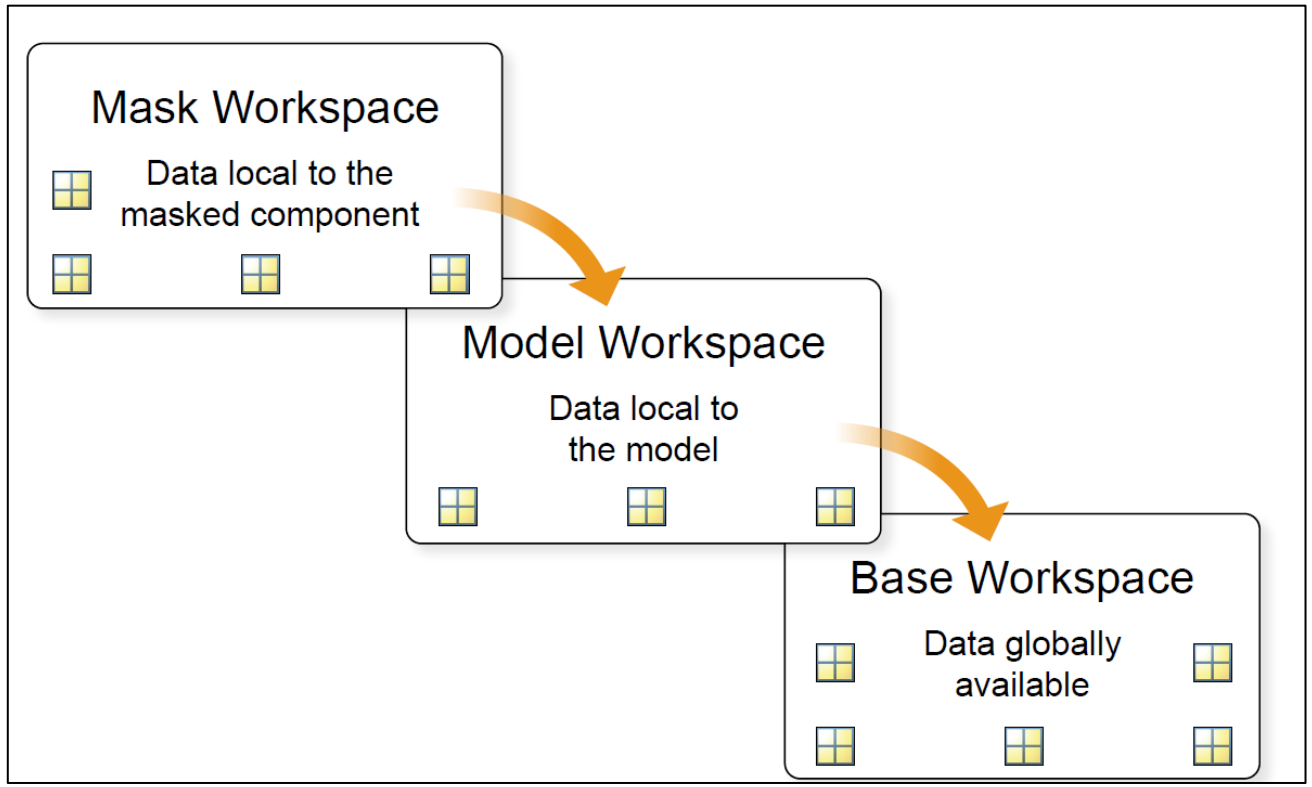
Large Projects!

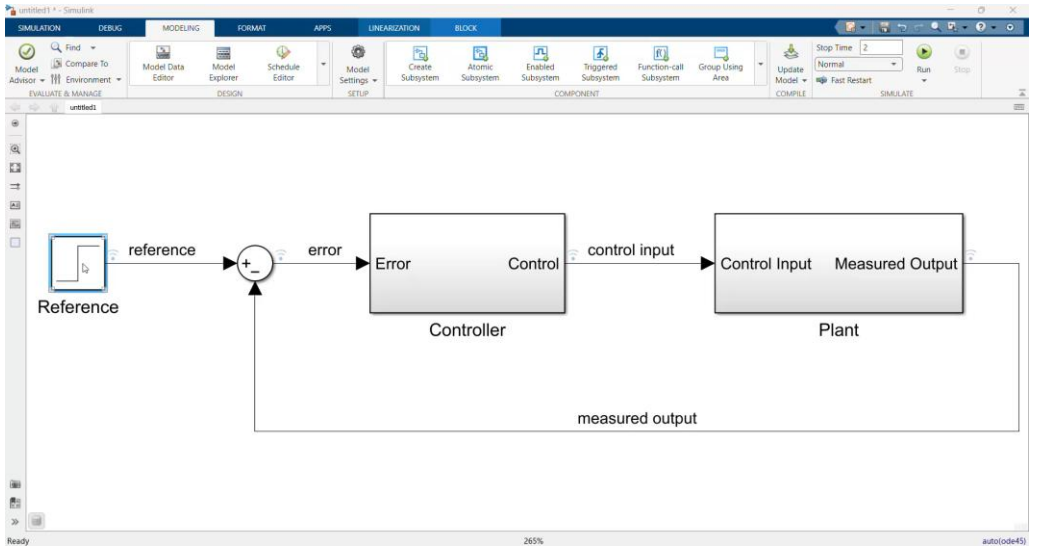
MAT-file **.M script**

Can also be stored as a .mat or .m file

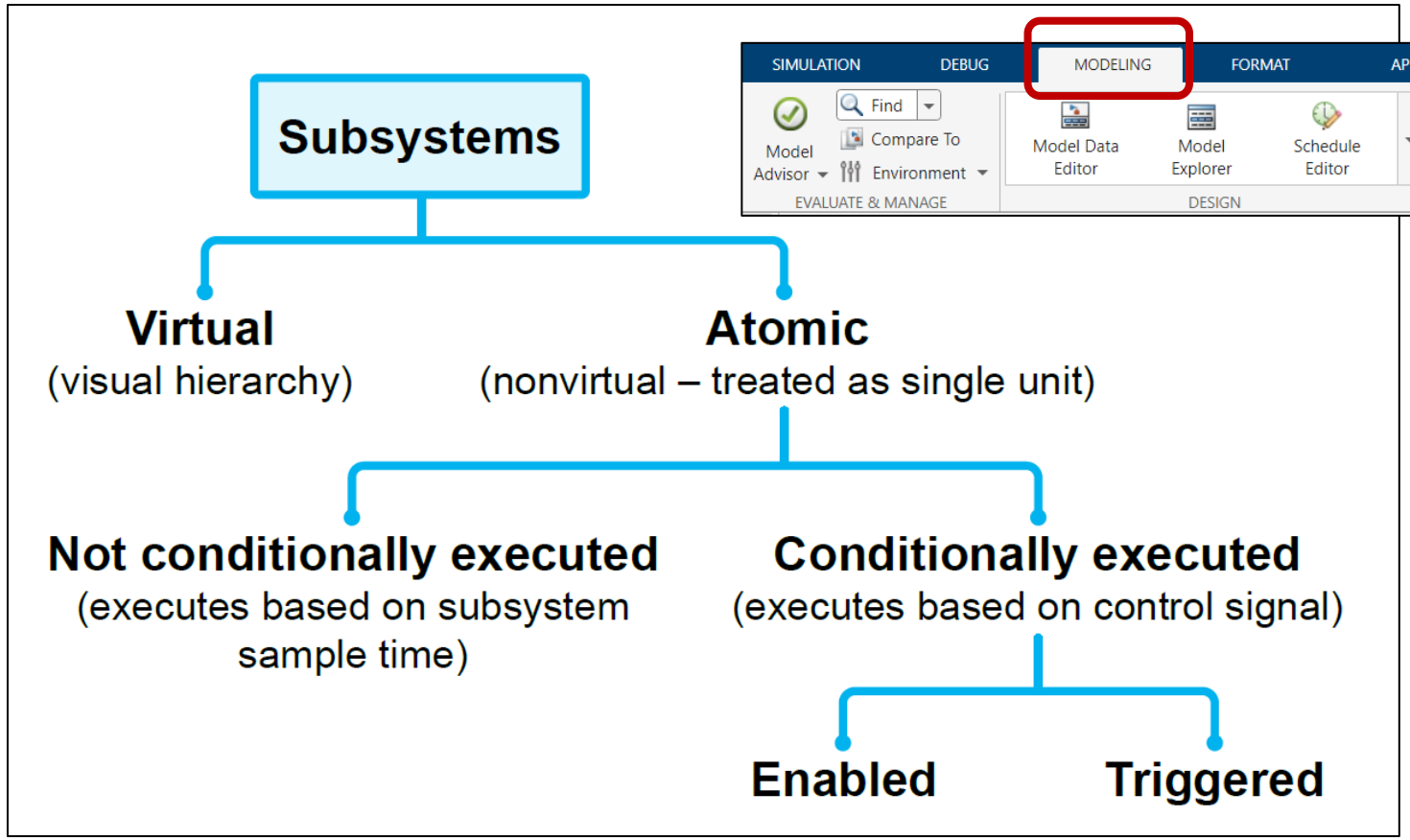
"It's always there"
Visible to all models
Not saved by default

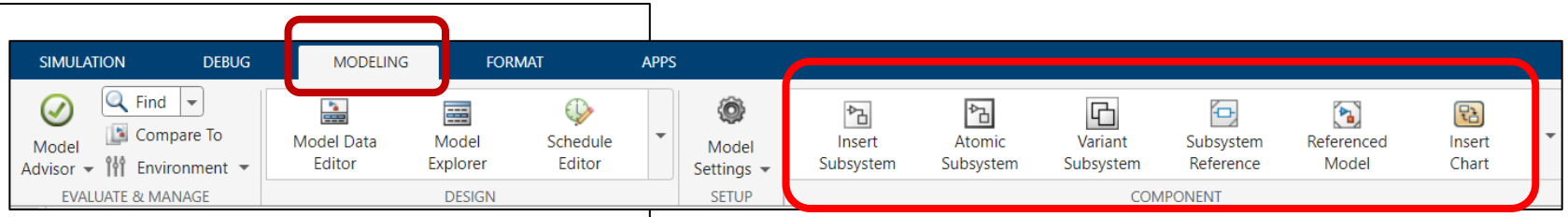
Simulink’s options for data storage



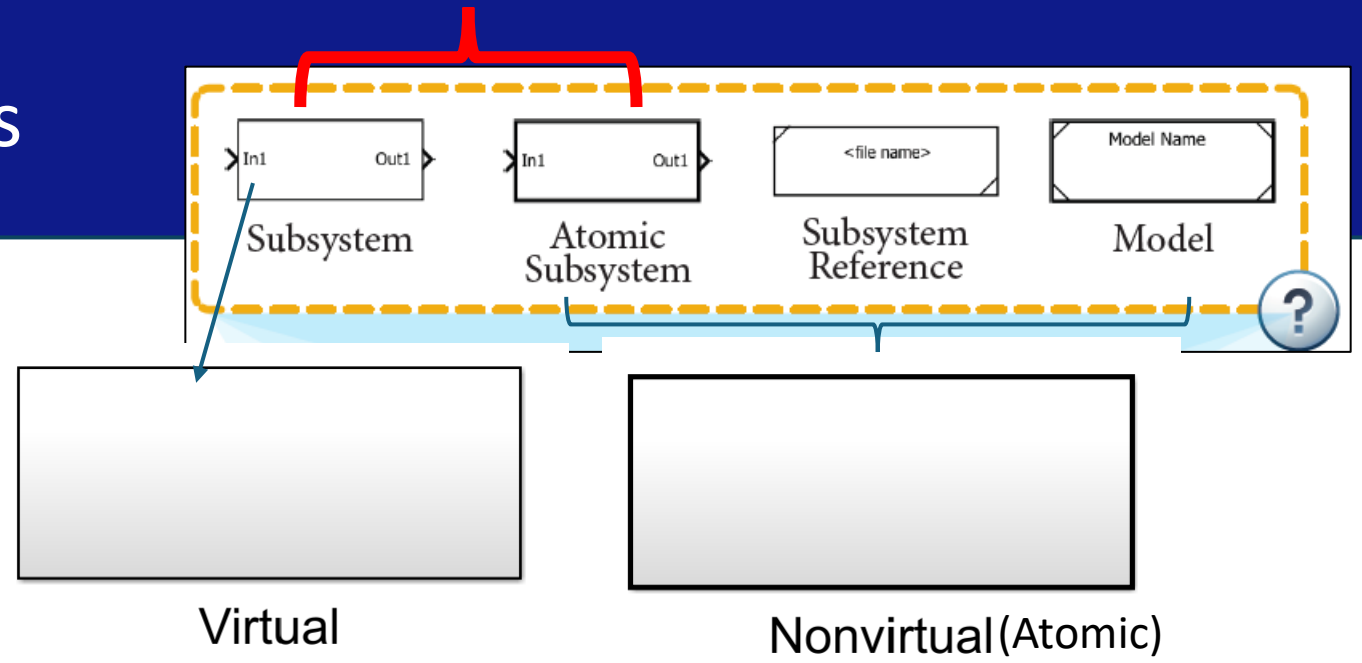


Types of Subsystems



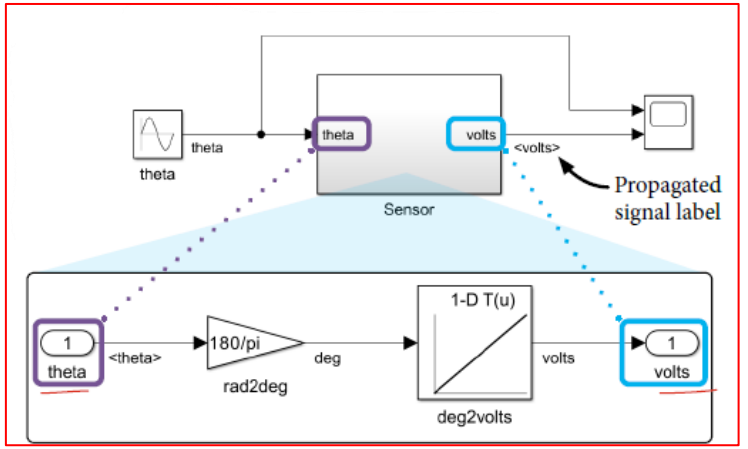


Subsystems



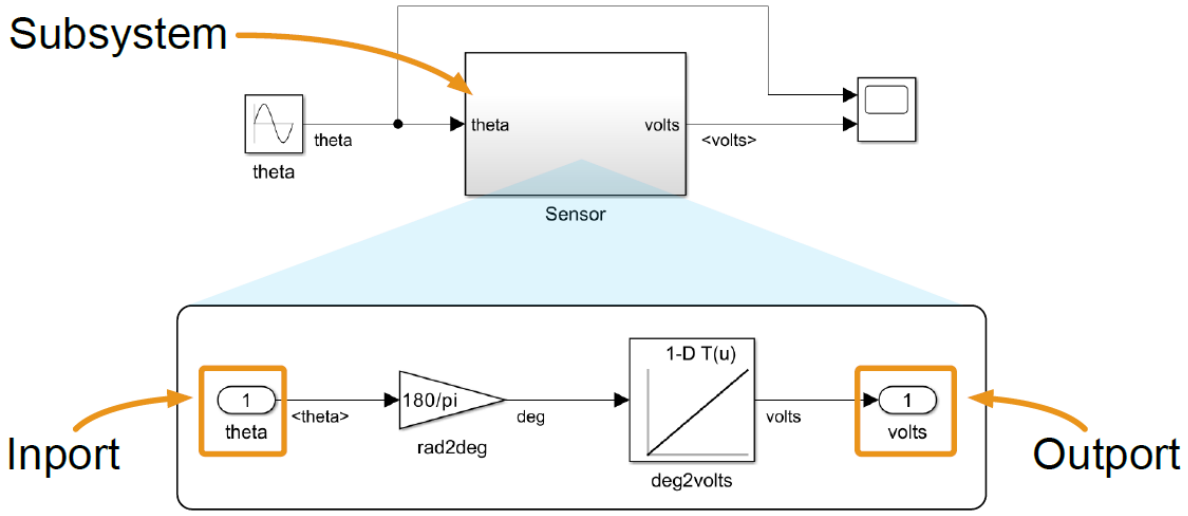
Benefits	Virtual subsystem	Atomic subsystem
Ease of use	+	+
Readability	+	+
Traceability		+
Reusability	-	-
Concurrent development	-	-
Unit testing	-	-
Performance		-

- Atomic (nonvirtual) subsystems **functionally** encapsulate blocks.
  - No outside block will be executed in-between the first and last block of the atomic subsystem.
- Virtual subsystems group blocks v**isually**.
  - The virtual boundary is ignored at compile time.
- Virtual subsystems are useful for:
  - Visually organizing and decluttering models.
  - Workflows involving physical connections (Simscape).
  - Workflows involving HDL code.



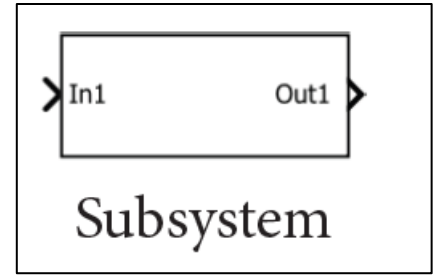
Virtual Subsystem

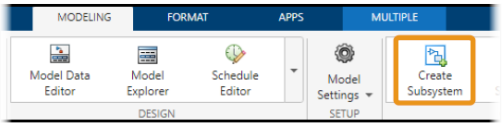
- **Scalability** – Add hierarchy to reduce the number of blocks at a given level
- **Readability** – Group functionally related blocks

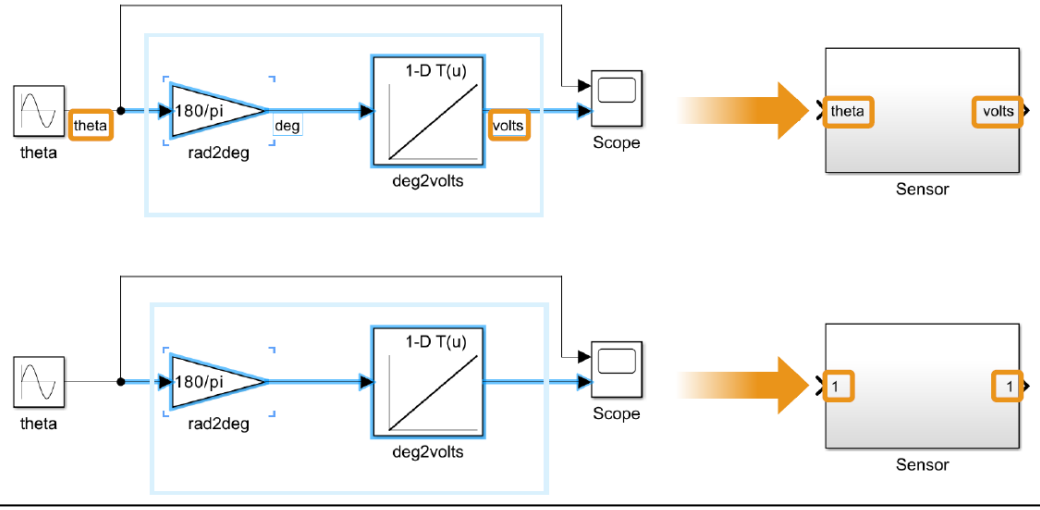


Benefits	Virtual subsystem
Ease of use	✚
Readability	✚
Traceability	–
Reusability	▬
Concurrent development	▬
Unit testing	▬ *
Performance	

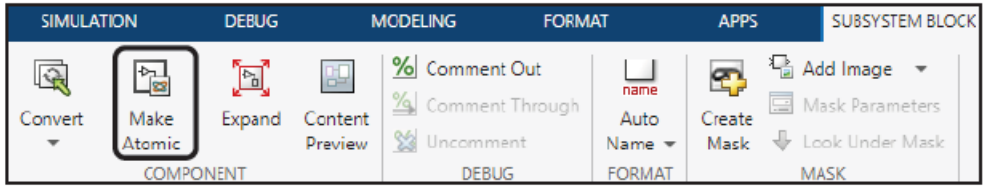
** Simulink Test is needed for unit testing*

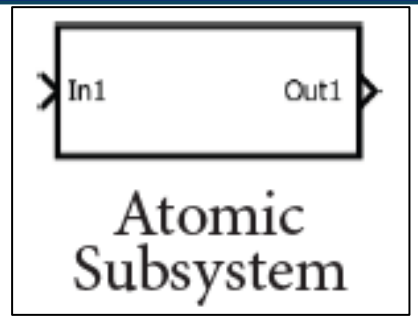




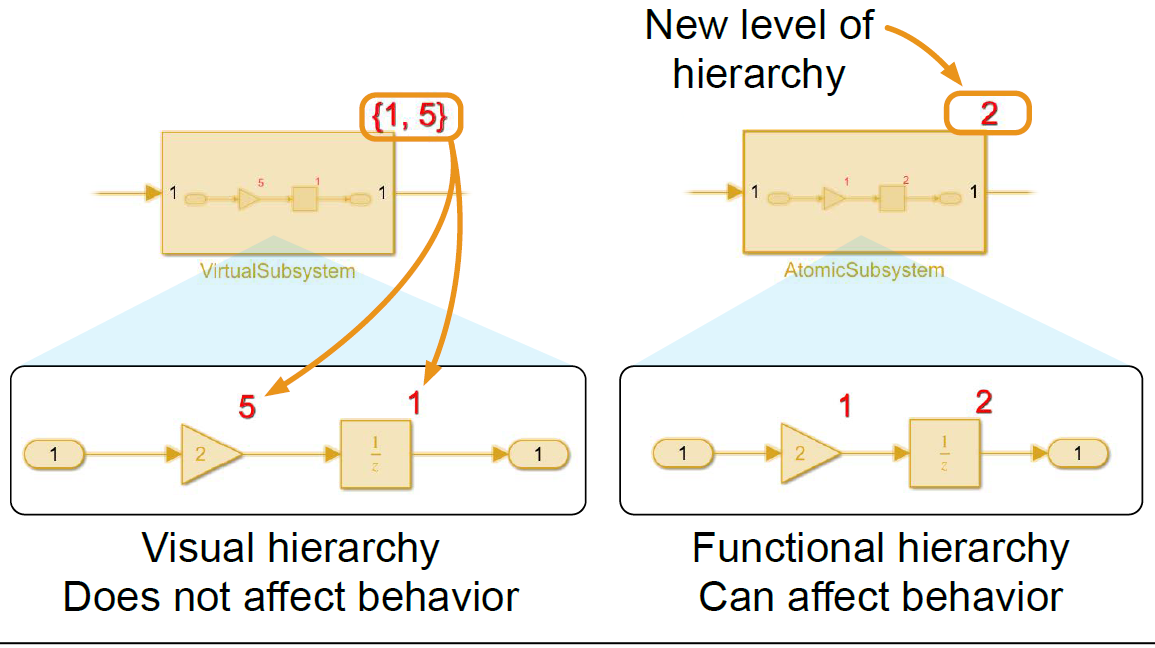


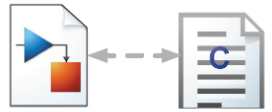
Atomic (Nonvirtual) Subsystems

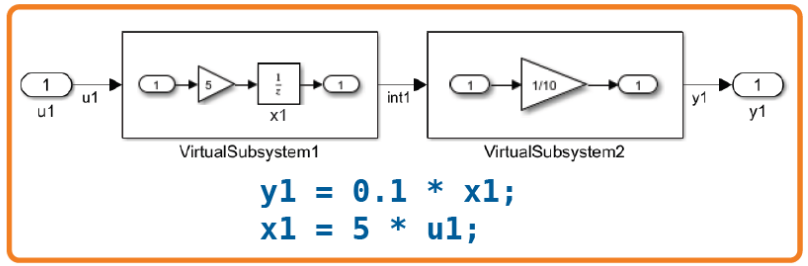


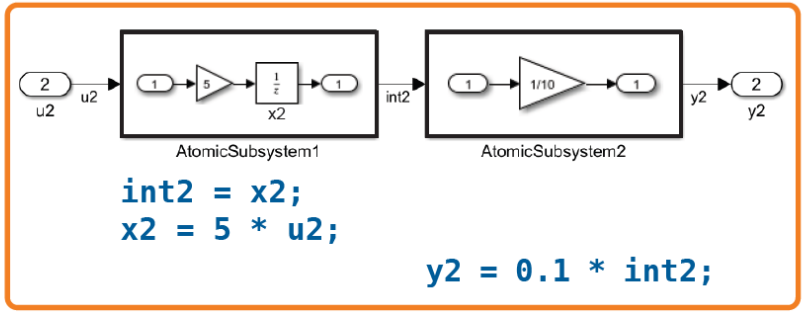


Creating Functional Hierarchy



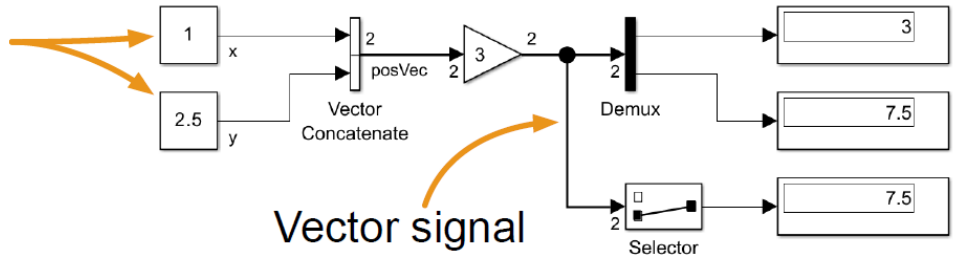






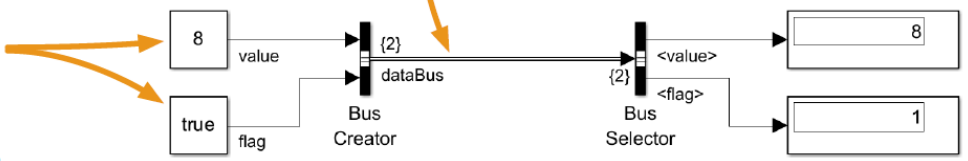
Vector & Bus Signals

Vectors must have **same** data type



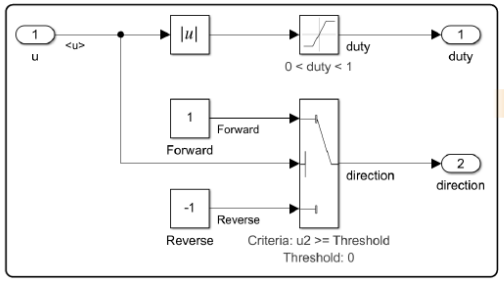
Bus signal

Buses can have **different** data types

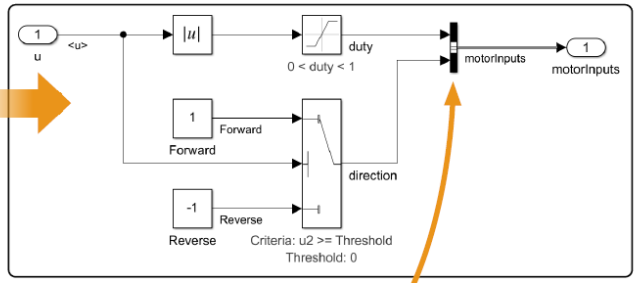


Use the Bus Creator block to group signals into a bus signal.

Multiple output signals:

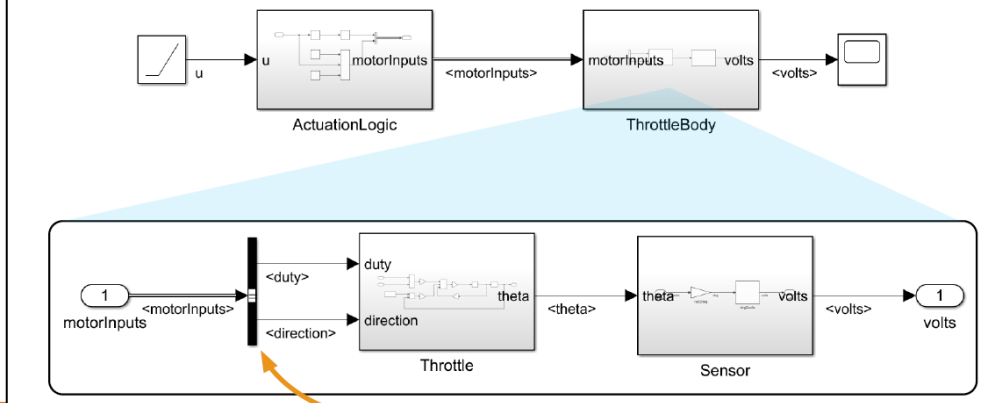


Single bus output signal:



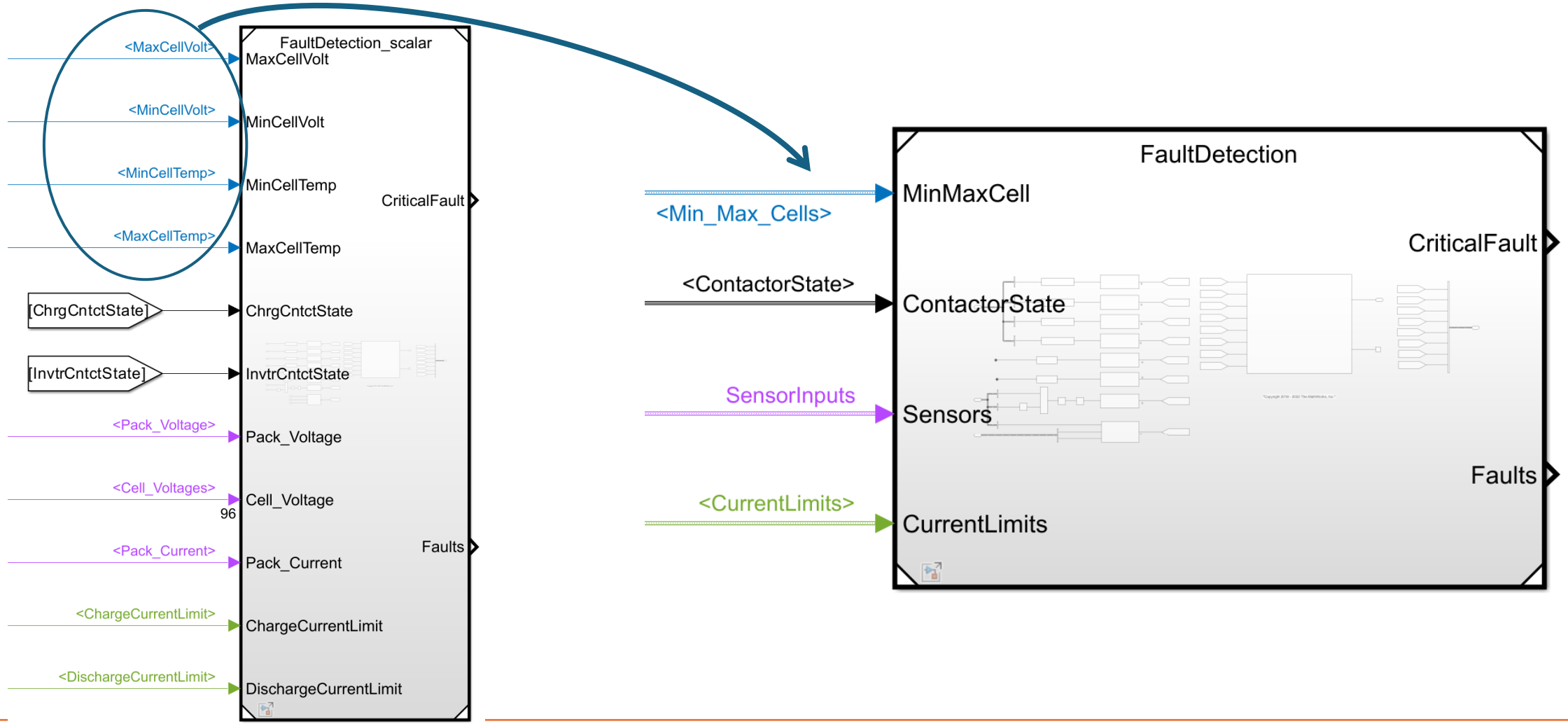
Bus Creator

Use the Bus Selector block to extract signals from a bus.

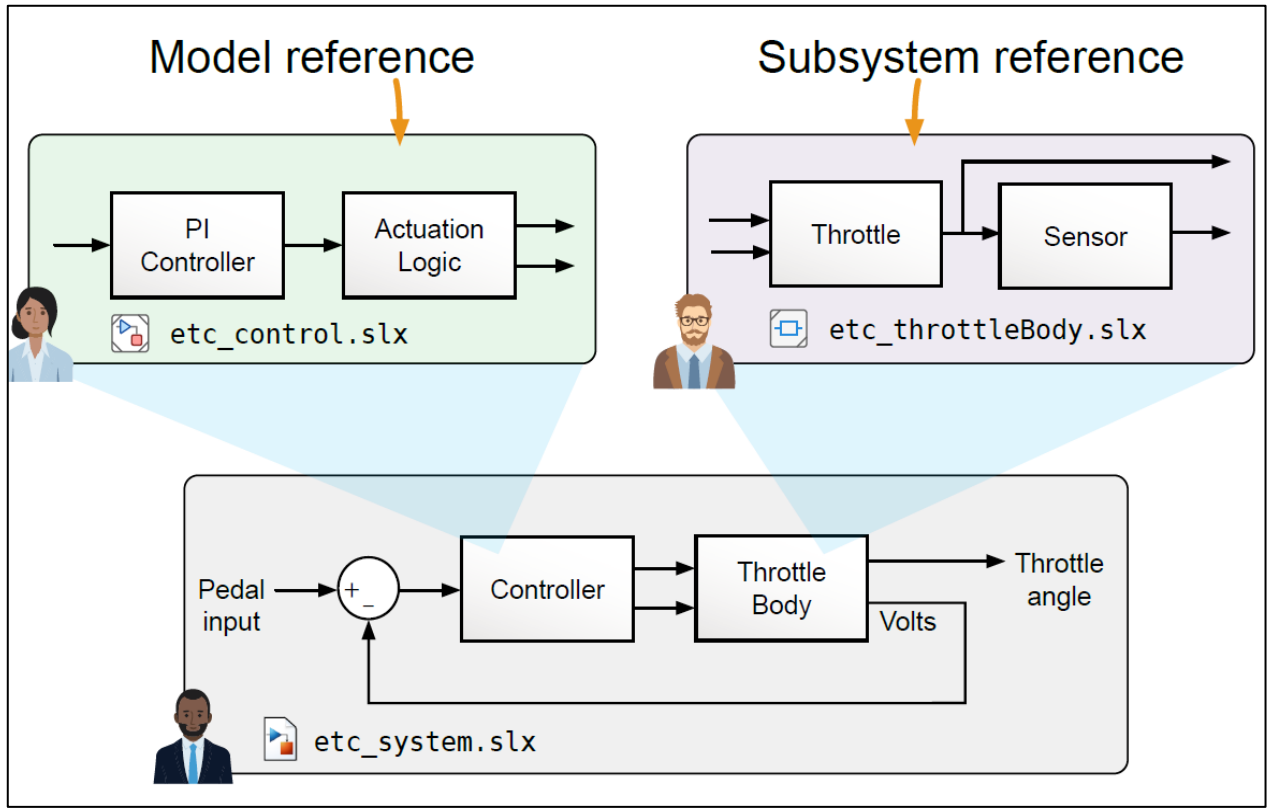


Bus Selector

Simplifying Interfaces with Buses

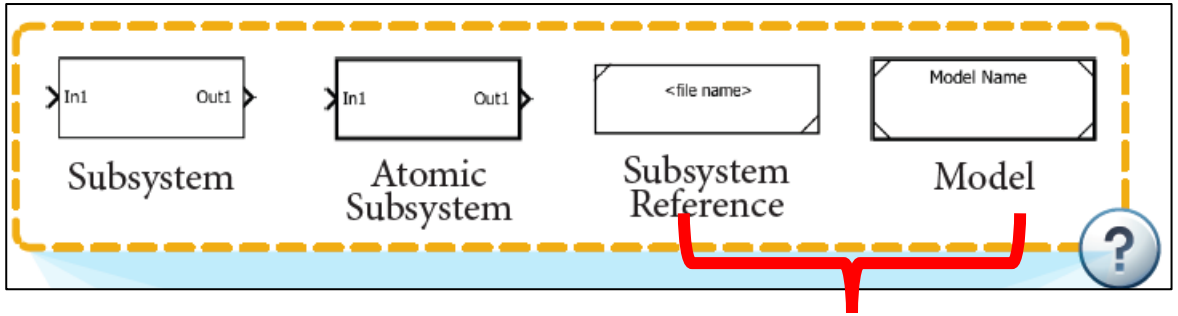


Reference Component Simulink

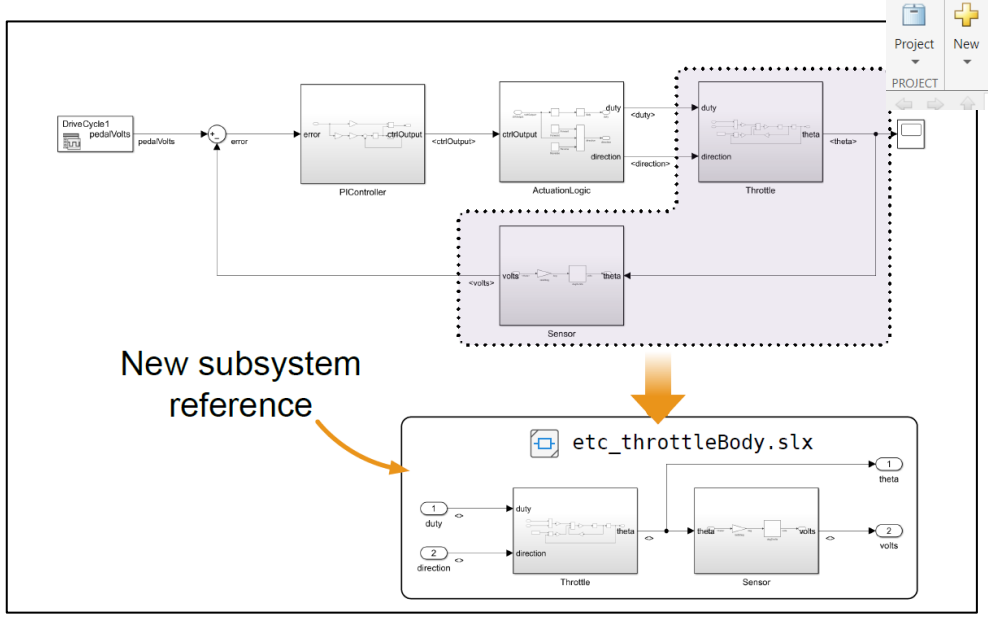


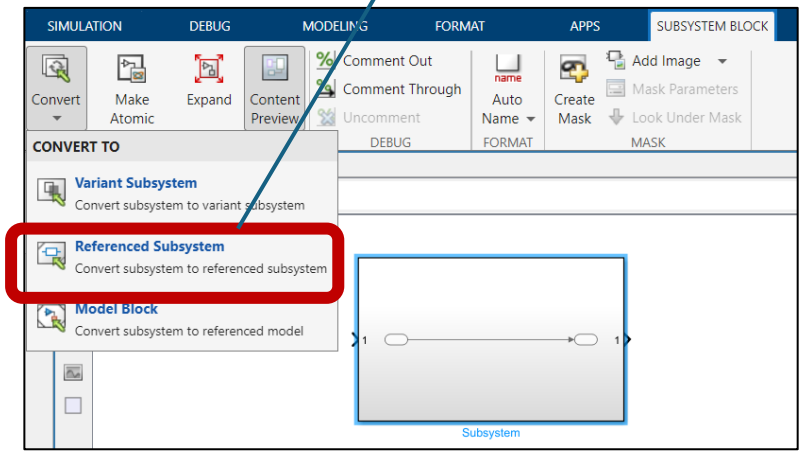
Benefits	Virtual subsystem	Atomic subsystem	Subsystem reference	Model reference
Ease of use	+	+		−
Readability	+	+	+	+
Traceability		+	+ *	+
Reusability	−	−	+	+
Concurrent development	−	−	+	+
Unit testing	−	−	+	+
Performance		−	− *	+ , −

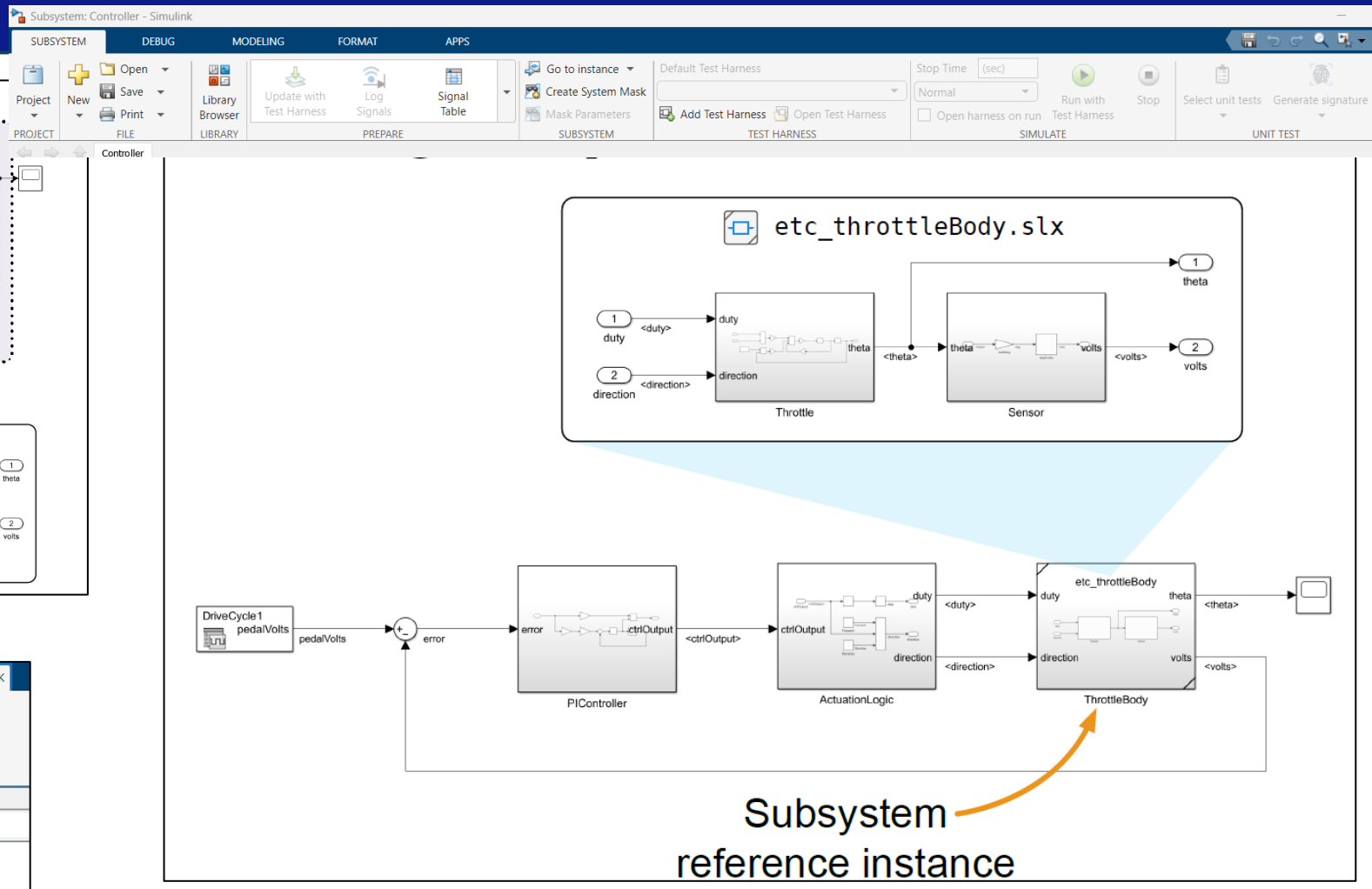
** Depends whether subsystem reference instance is virtual or atomic*



Subsystem References

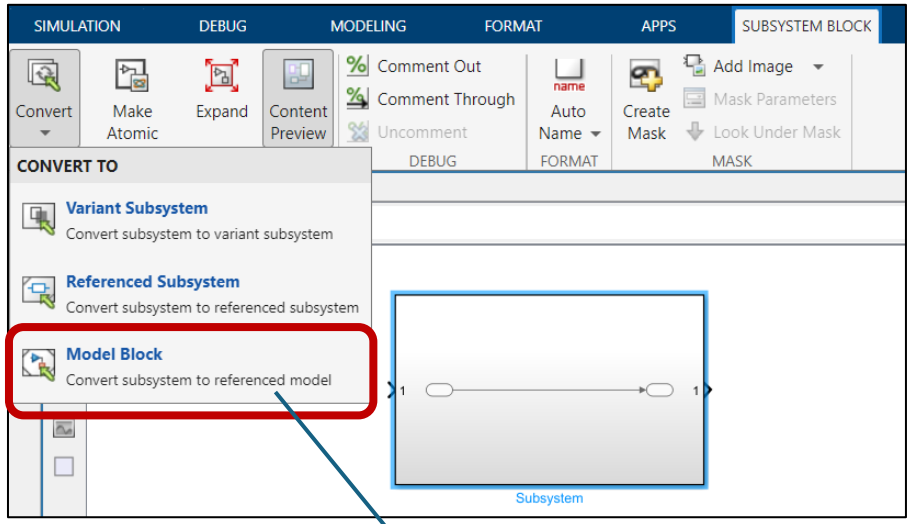


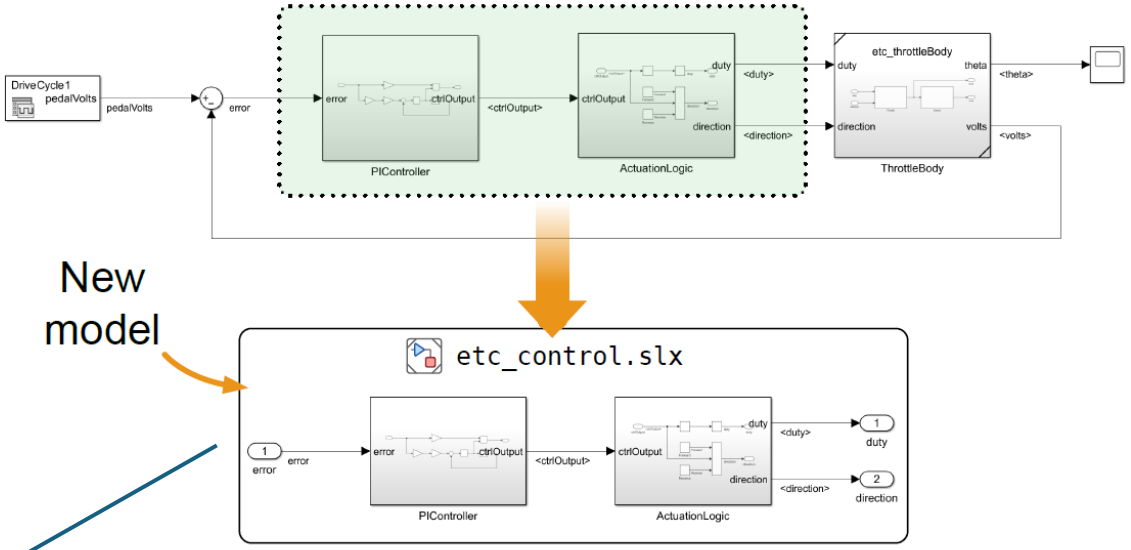


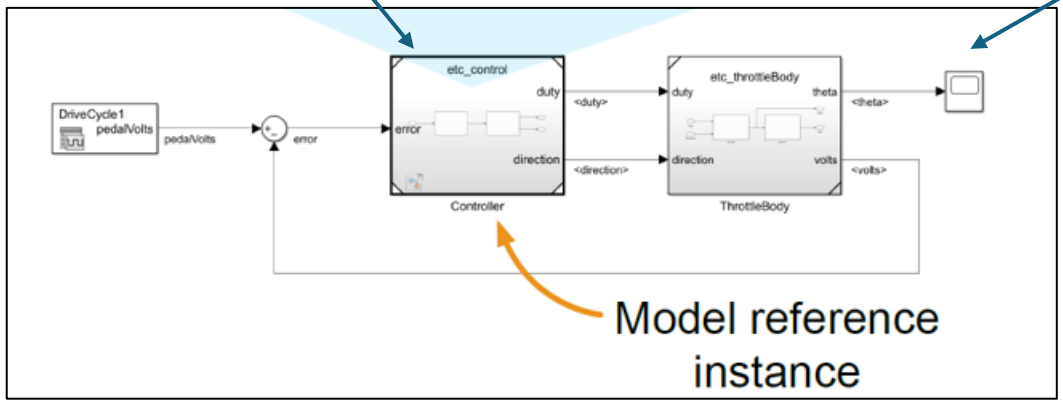


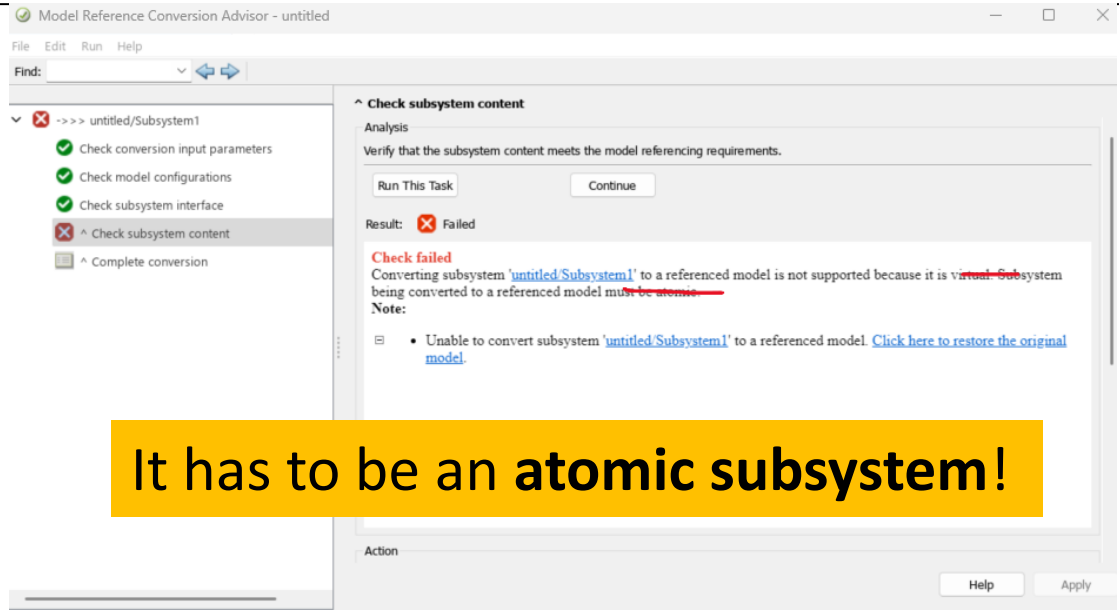
Need not be an atomic subsystem!

Model Reference



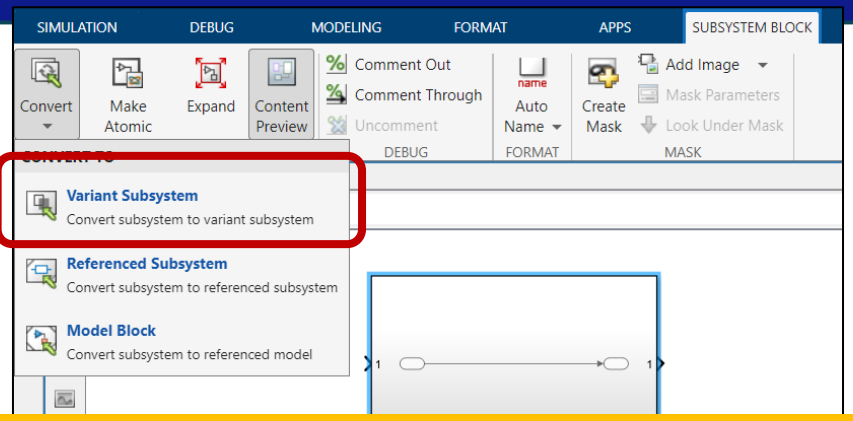




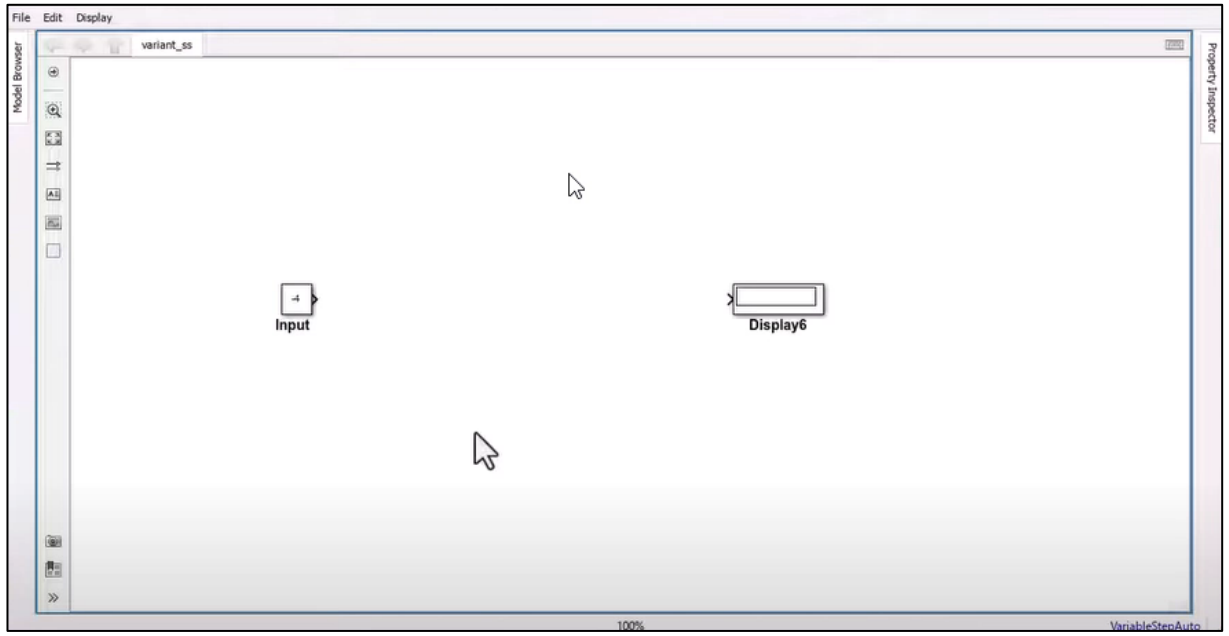


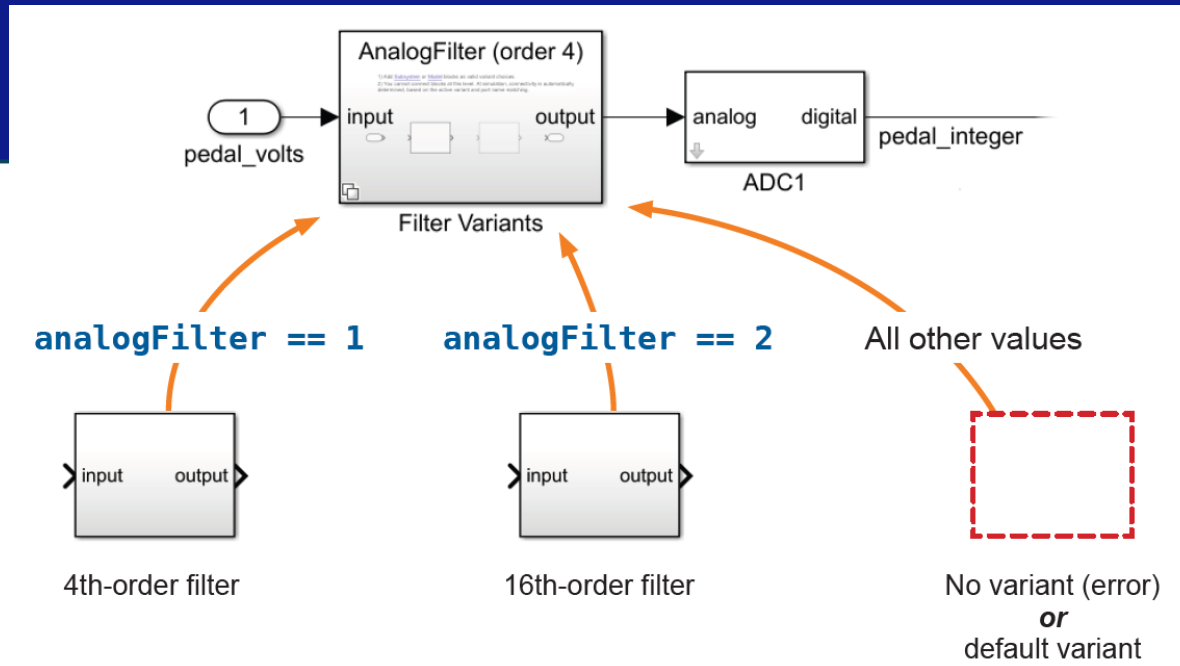
It has to be an **atomic subsystem**!

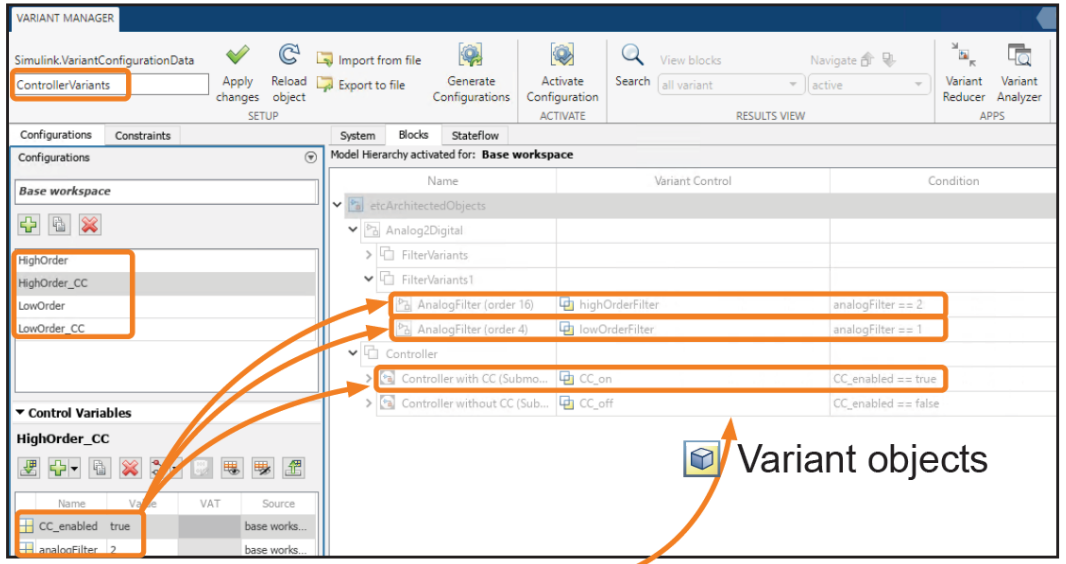
Variant Model



Need not be an **atomic subsystem**!

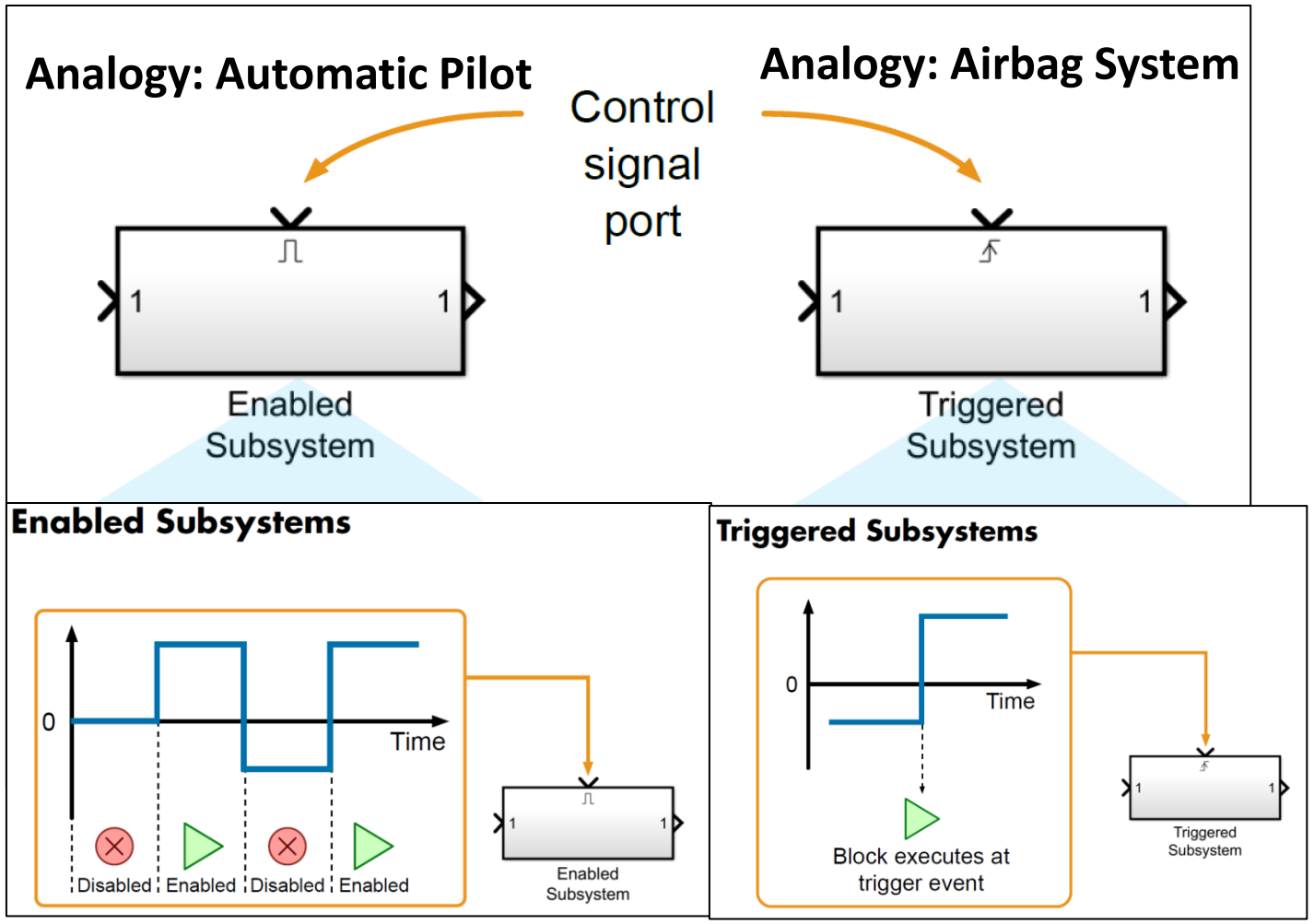


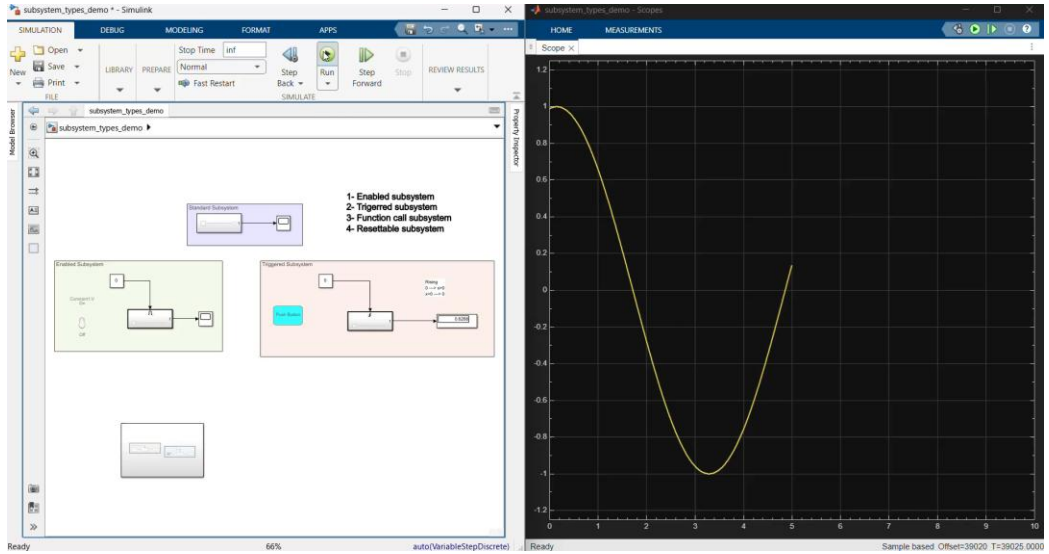




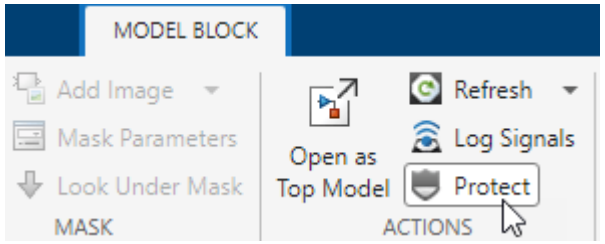

```
>> CC_on = Simulink.Variant;
>> CC_on.Condition = 'CC_enabled == true';
```

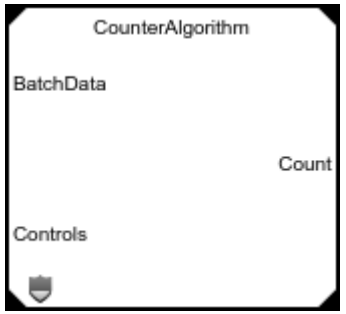
Conditionally Executed Subsystems

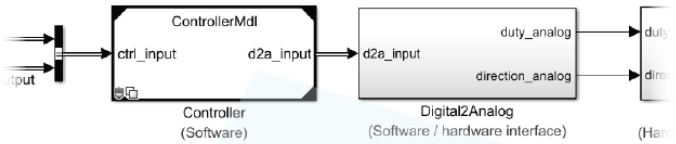




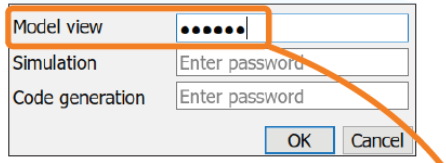
Protected Model

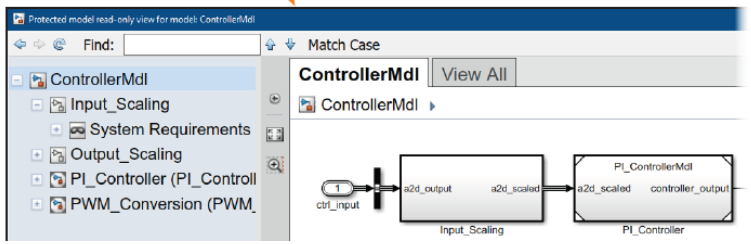


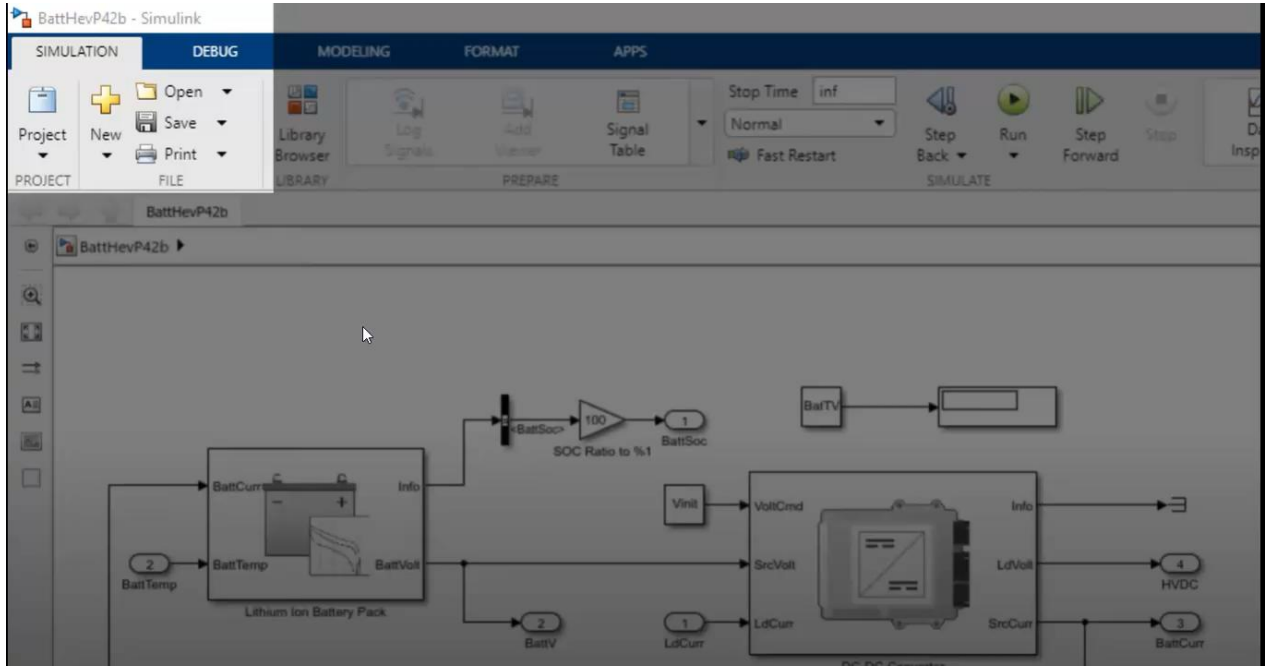




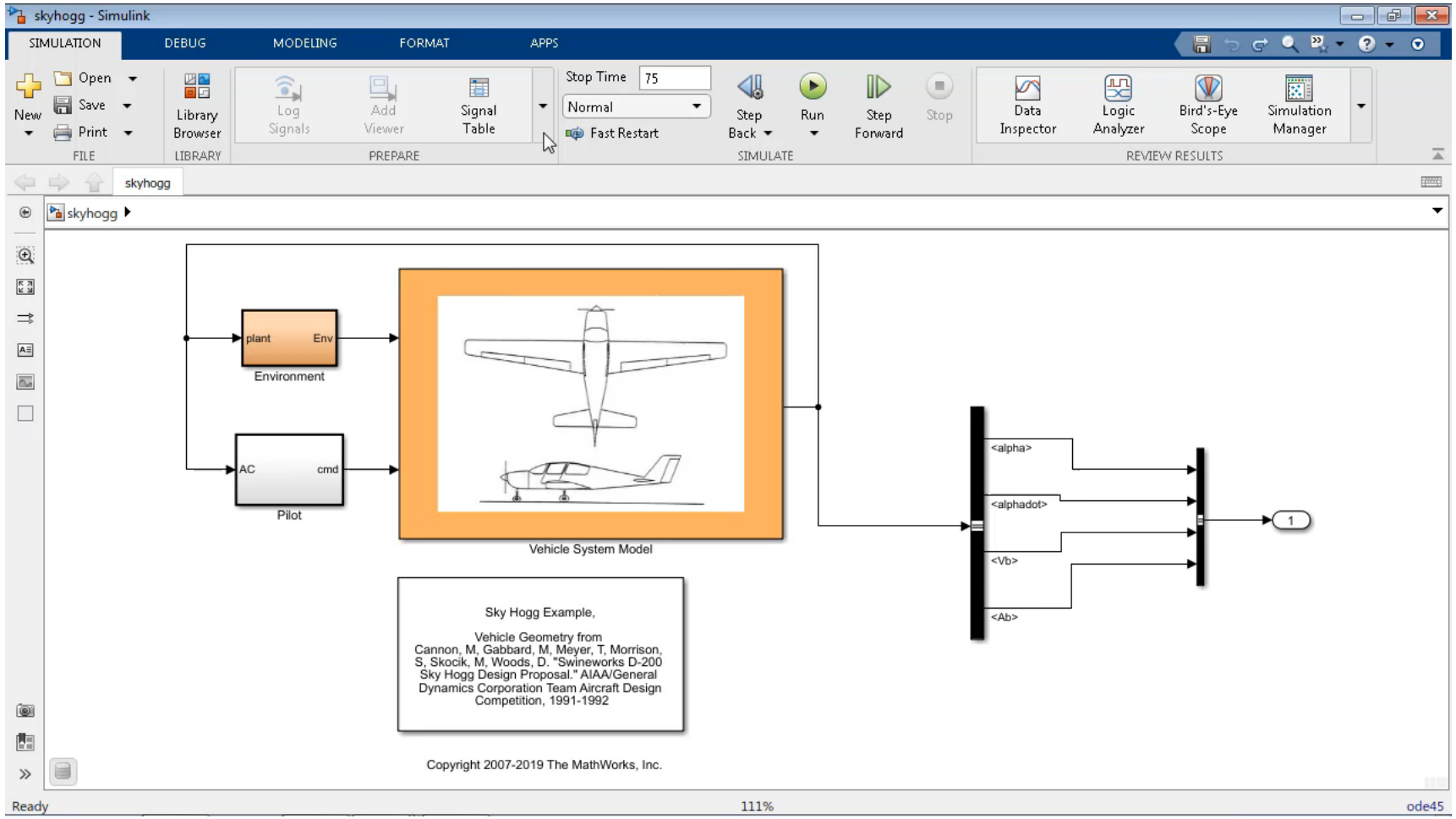
`ControllerMdl.slxp`

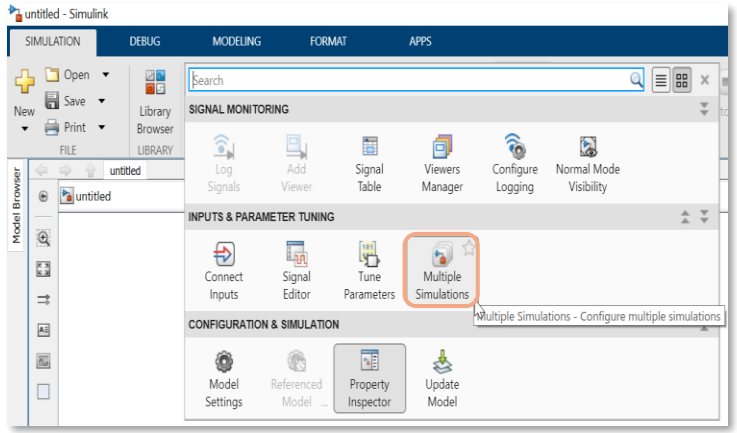




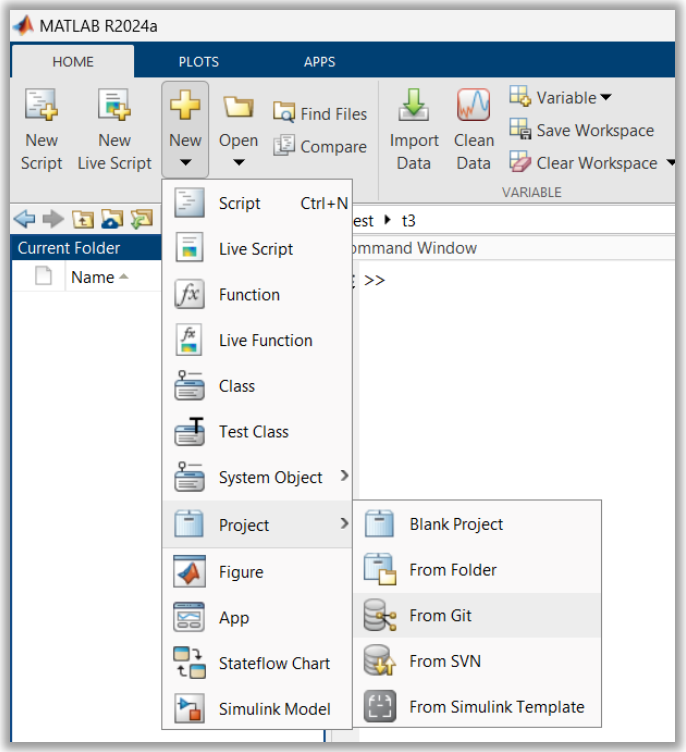


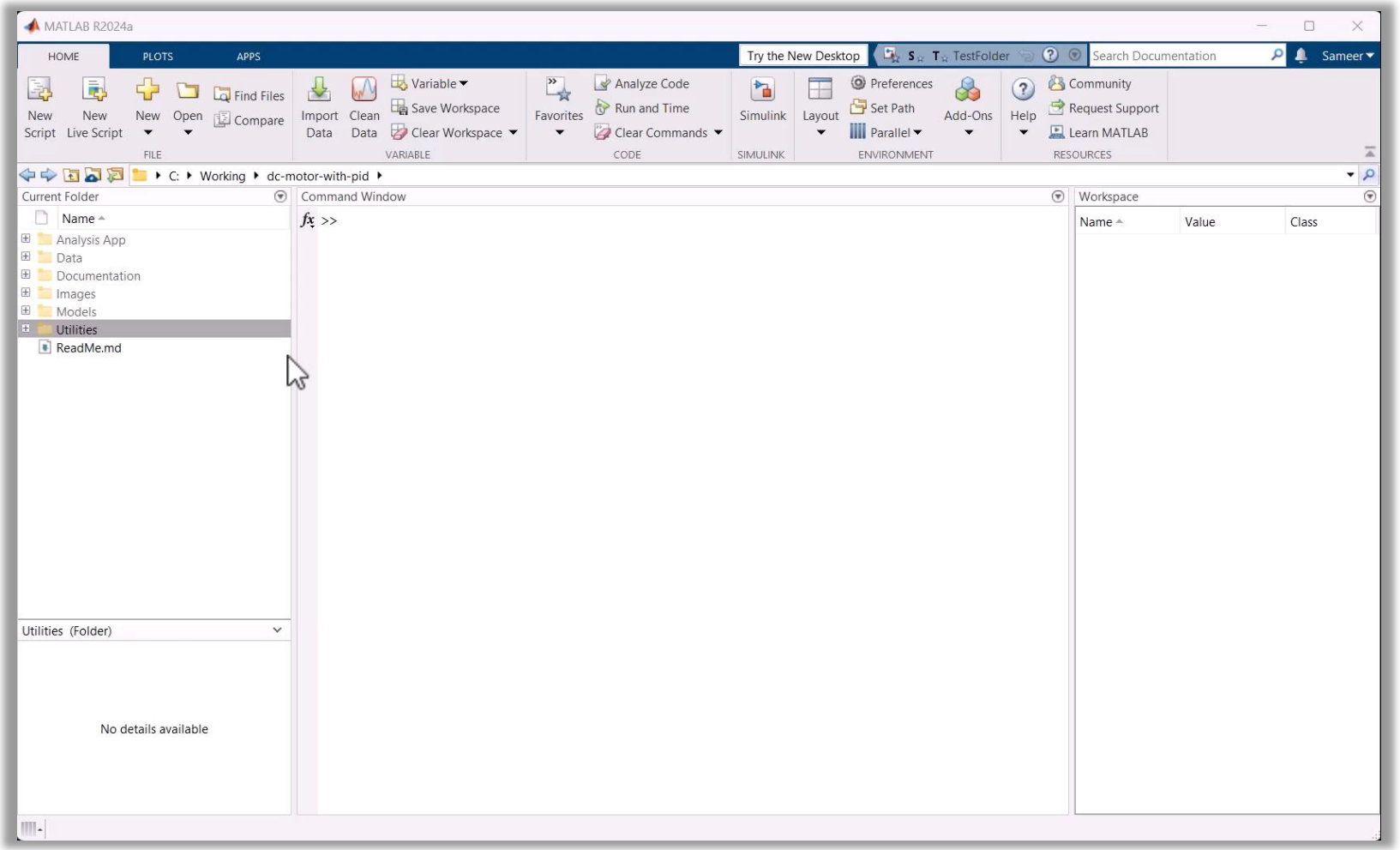
Multiple Simulations



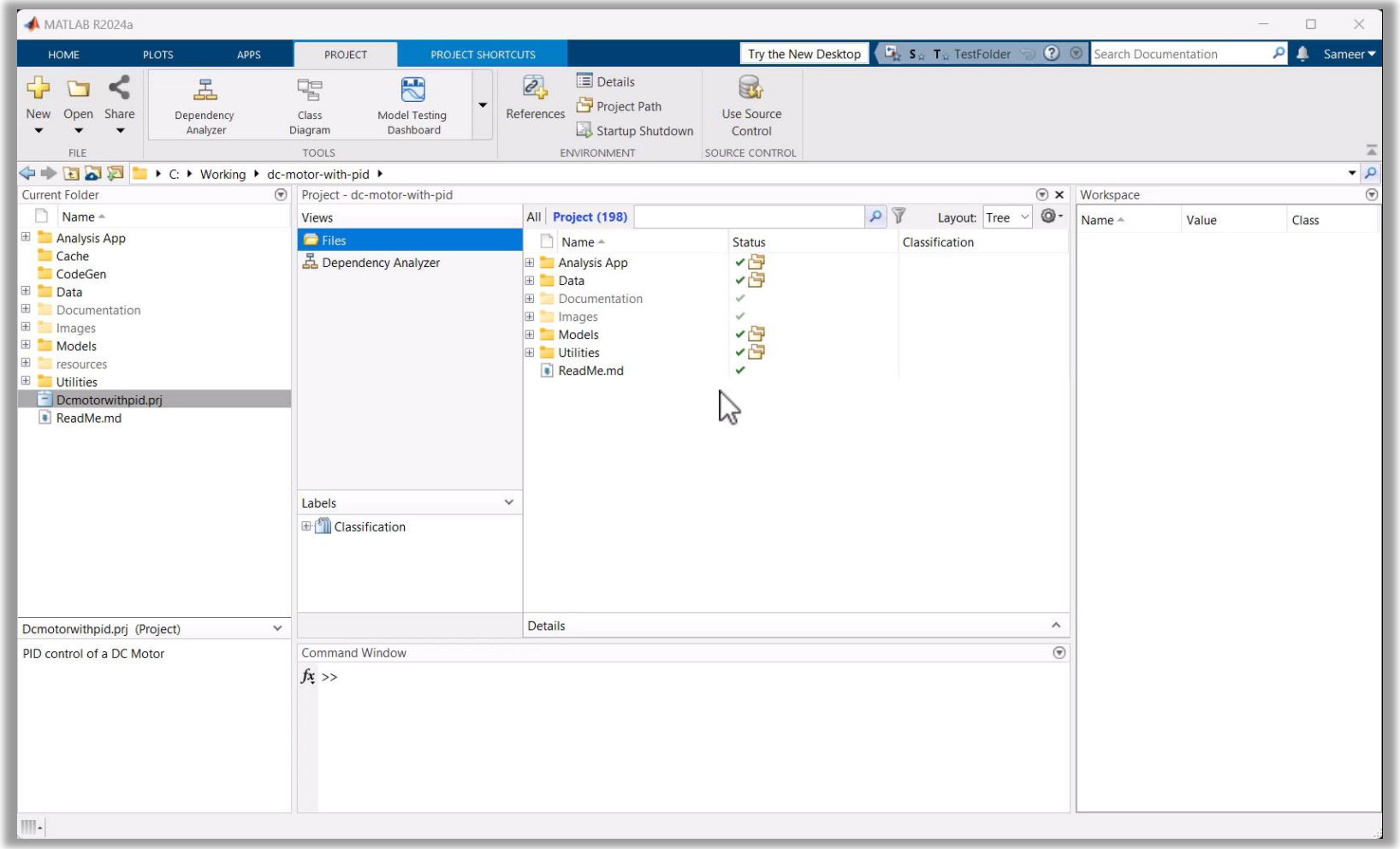


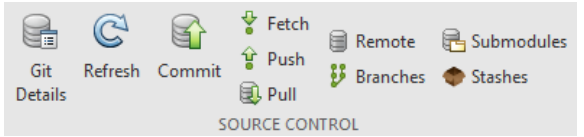
Simulink Projects





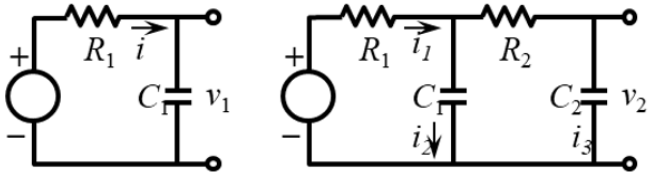
Simulink Projects

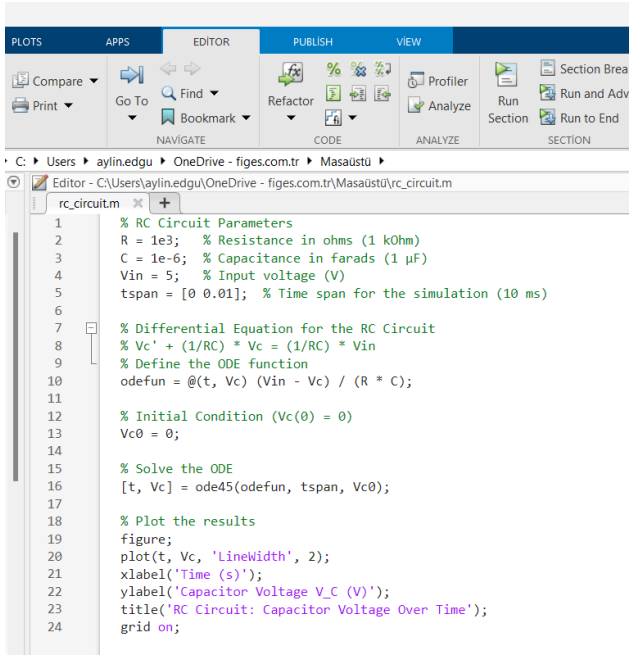




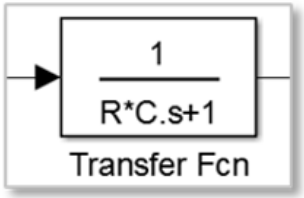
MathWorks Products Overview

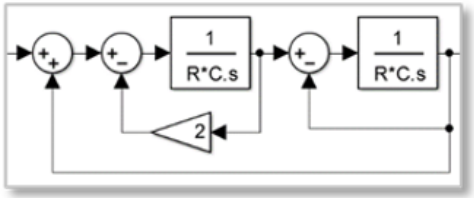
MATLAB®

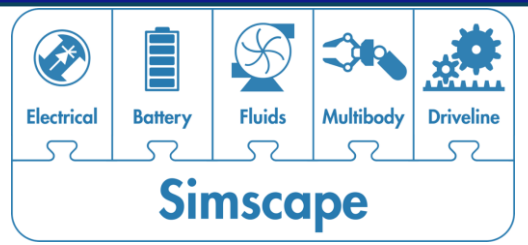


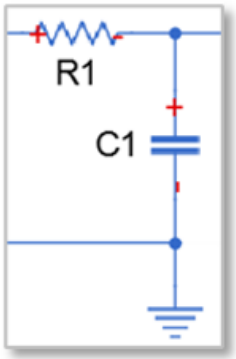


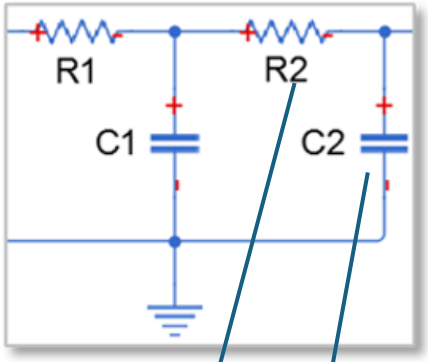
SIMULINK®

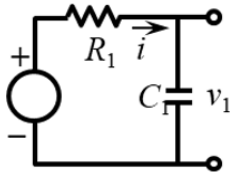


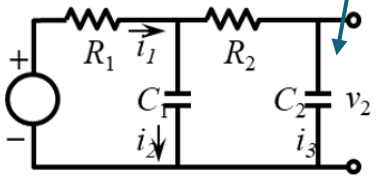












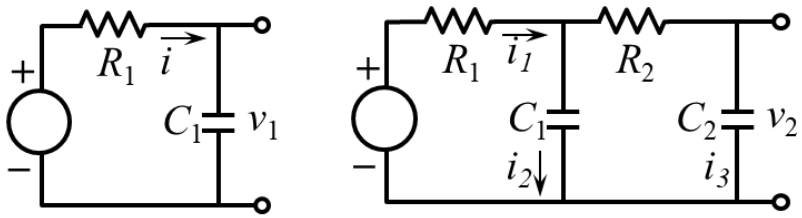
FİGES | MathWorks Authorized Reseller

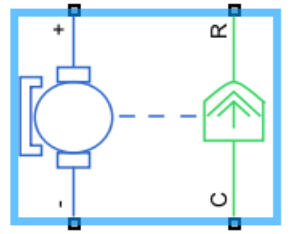
BİZİ TAKİP EDİN!

Modeling Differences Between Simulink and Simscape

Simscape models are easy to build, simple to scale, and straightforward to understand

Systems To Be Modeled

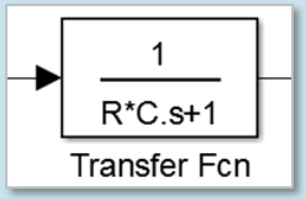


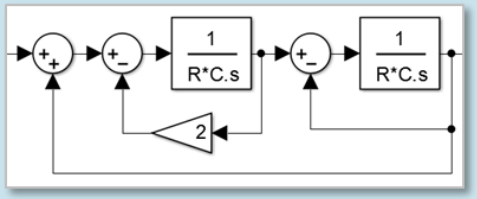


DC Motor

Simulink: Inputs→Outputs

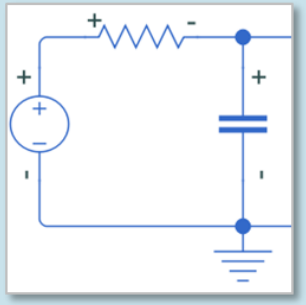
- Model data flow from inputs to outputs
- Model behavior via equations

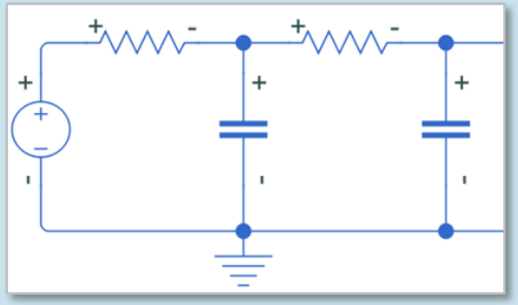


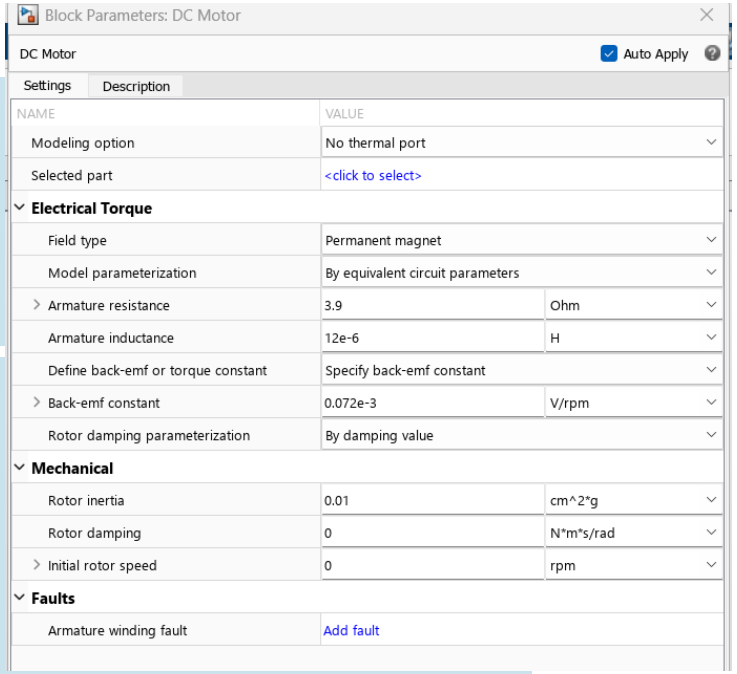


Simscape: Physical Networks

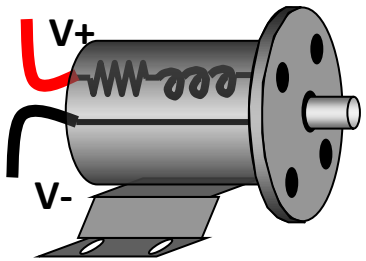
- Assemble schematics that directly represent the physical system
- Underlying equations are automatically derived
- Model bidirectional flow of energy

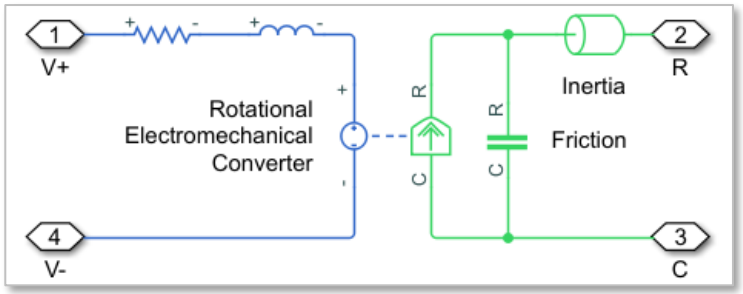


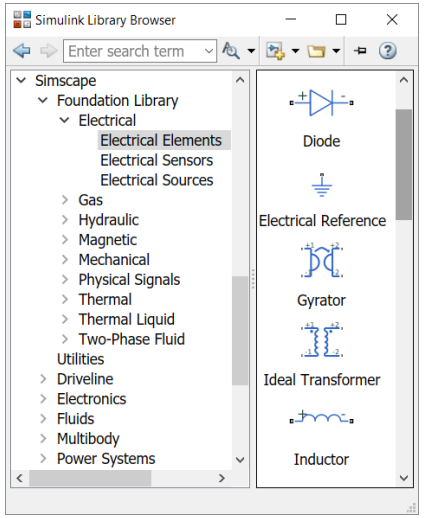




MathWorks Products Overview - **Simscape**

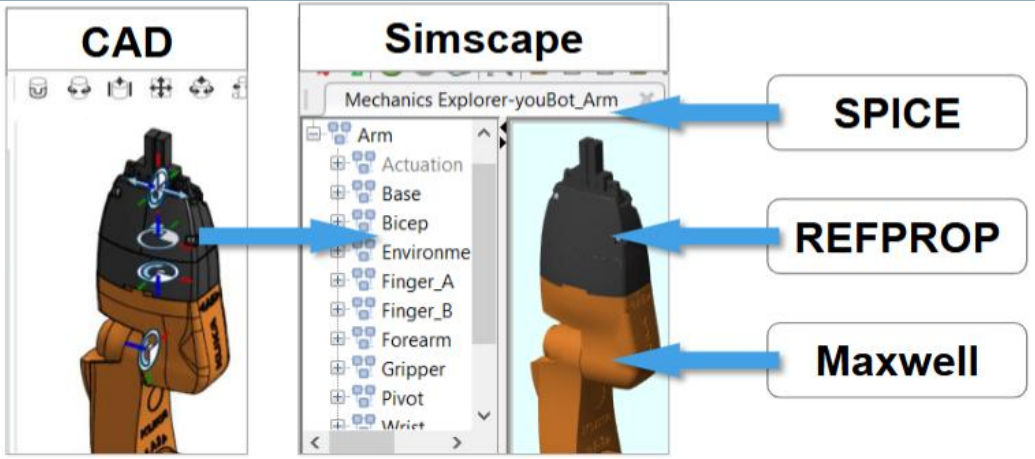




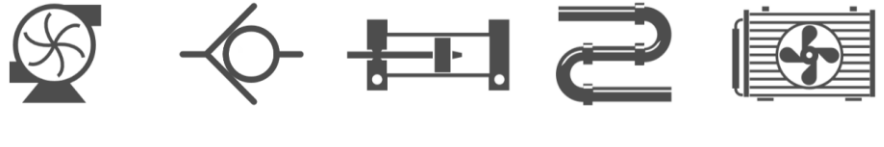


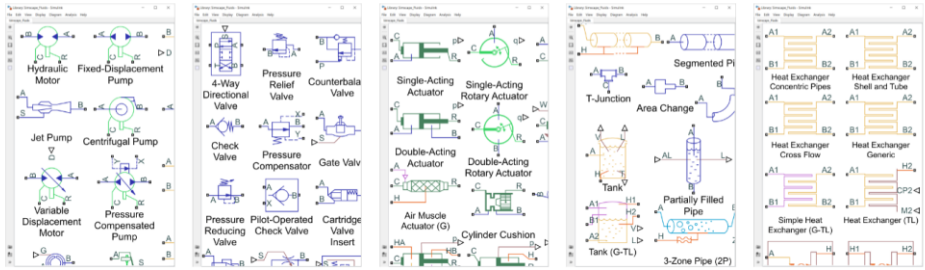
$$i = (C_0 + C_v v)\frac{dv}{dt} + \frac{v}{r_d}$$


```
Editor - C:\+MyComponents\LossyUltraCapacitor.s
40  equations
41    i == (C0 + Cv*vc)*vc.der + vc/Rd;
42    v == vc + i*R;
43  end
```



Simscape Fluid Component Models

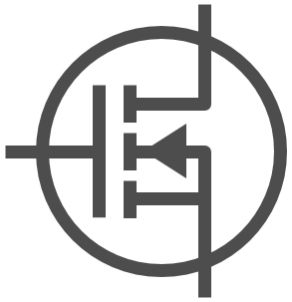




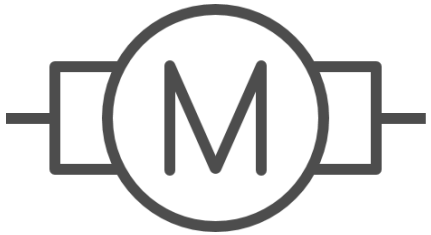
FIGES | MathWorks Authorized Reseller

BİZİ TAKİP EDİN!

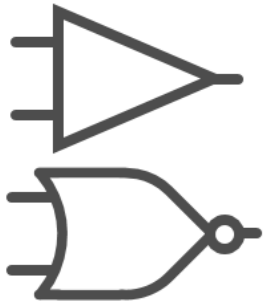
Simscape Electrical Component Models



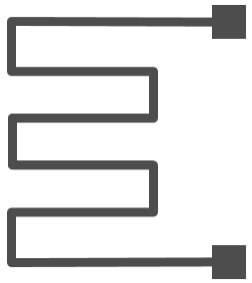
Semi-conductors



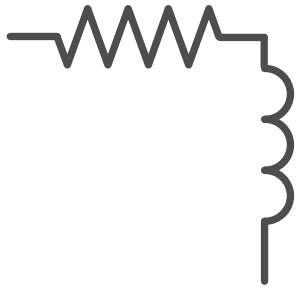
Motors, Actuators



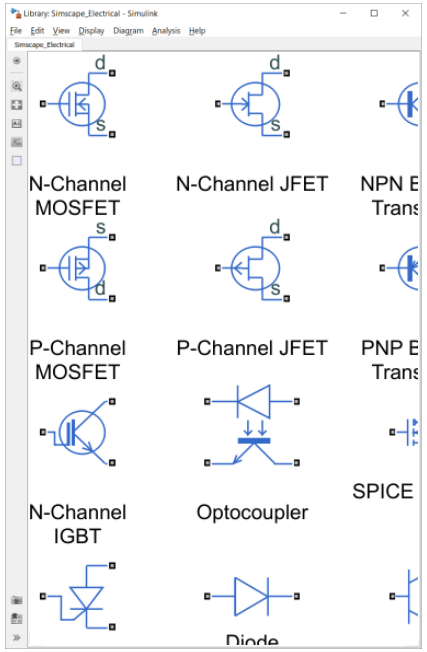
Op-Amps, Logic Gates

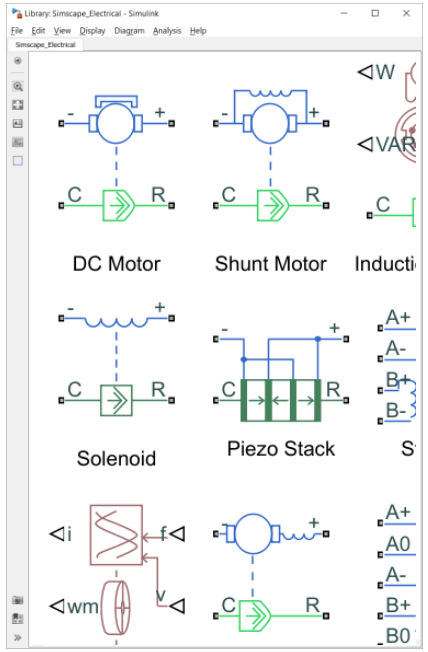


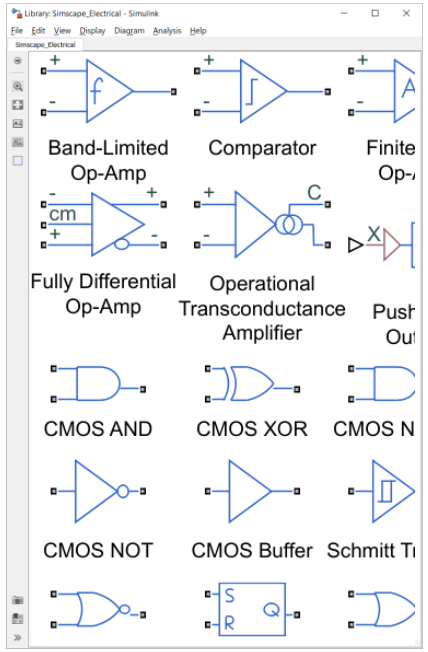
Sensors

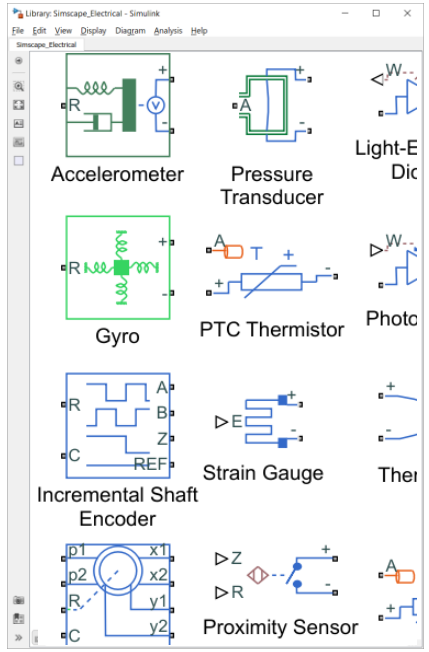


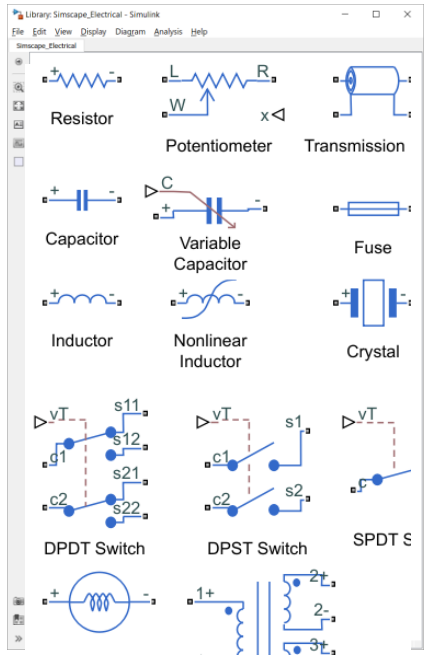
Passive Devices



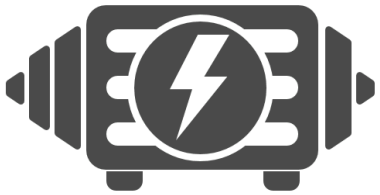




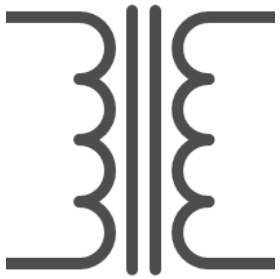




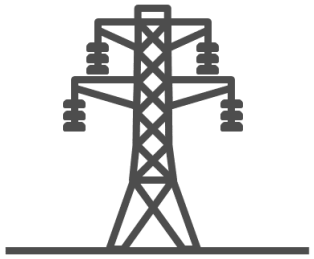
Simscape Electrical Component Models



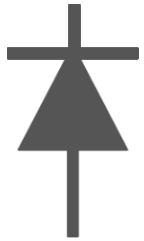
Generators, Motors



Transformers

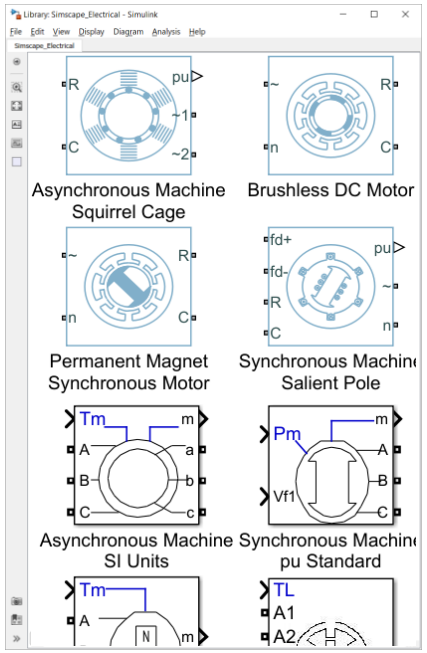


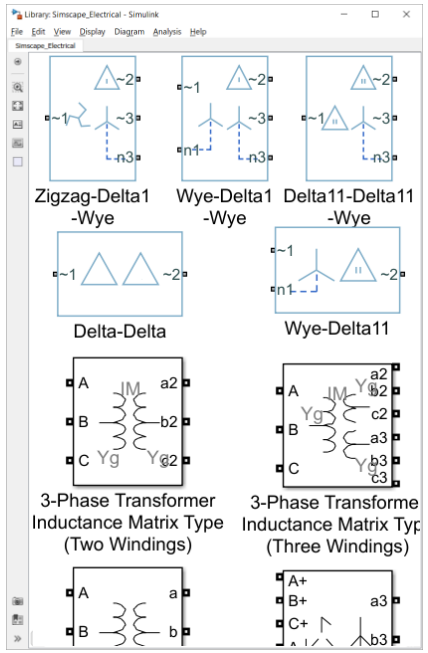
Lines, FACTS

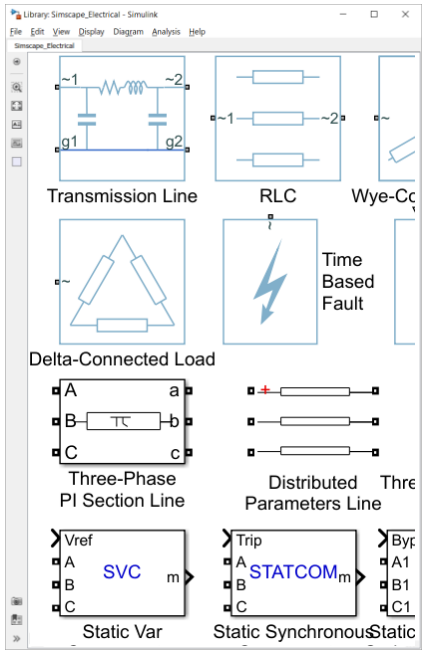


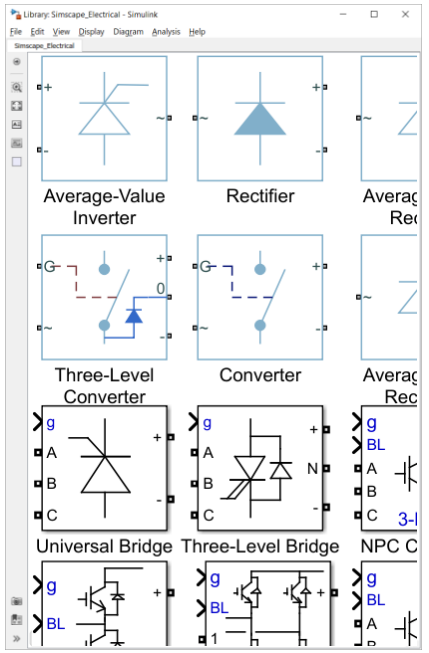
Converters

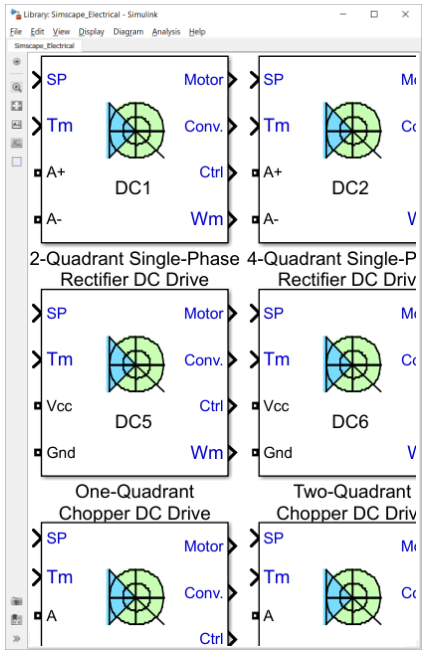
Electric Drives





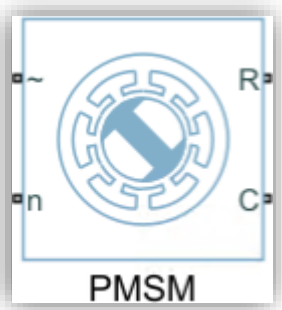






Manufacturer Specific Electric Motors

- Database with parameter values to match **vendor**-specific motors

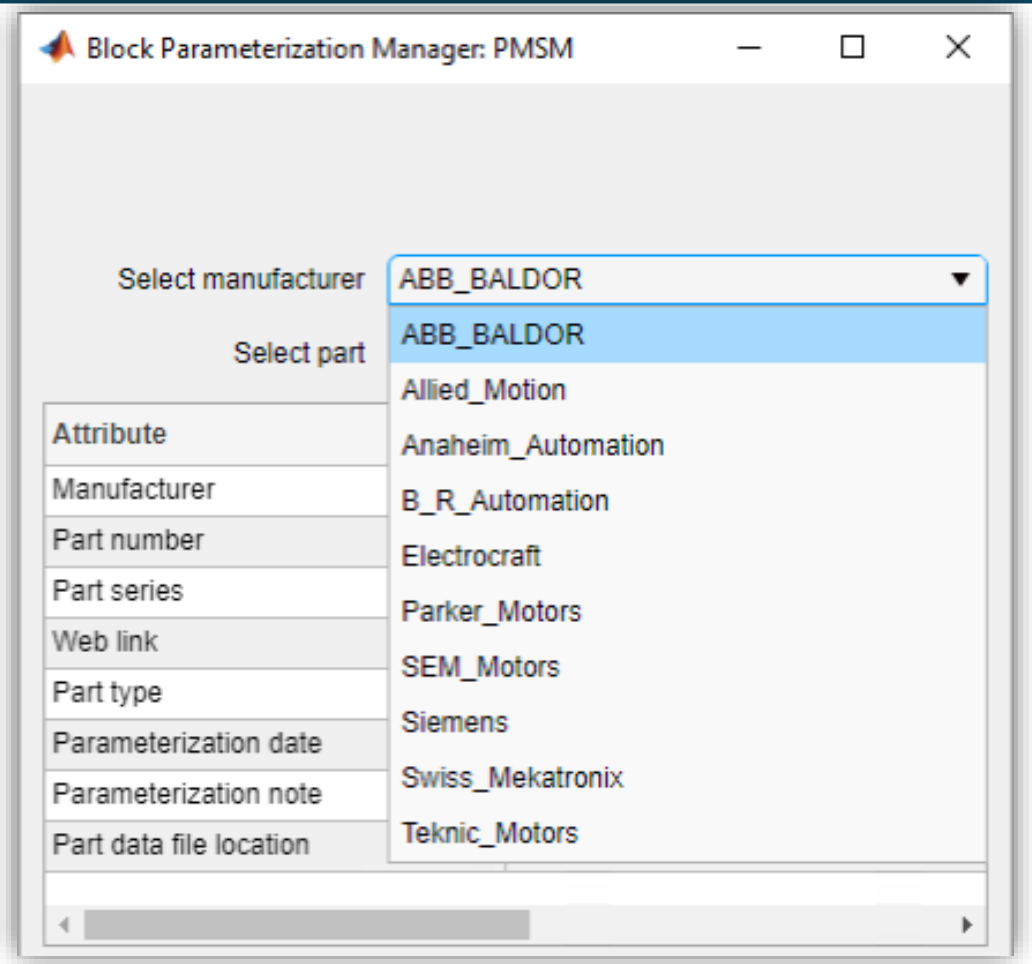


PMSM

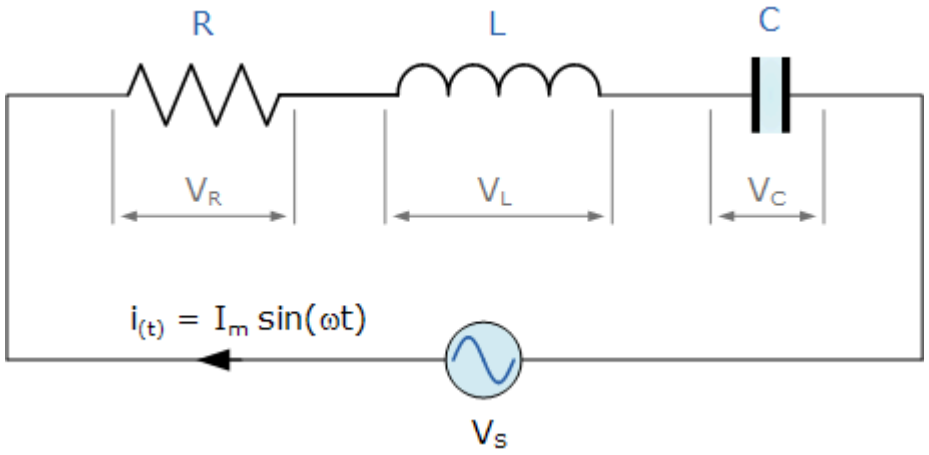
This block represents a permanent magnet synchr

Right-click on the block and select Simscape block

Select a predefined parameterization



RLC Circuit Modeling



Total empedans: $Z(s) = R + sL + \frac{1}{sC}$

Kirchoff voltage law : $v_{in}(t) = v_R(t) + v_L(t) + v_C(t)$

$$v_R = Ri(t)$$

$$v_L = L\frac{di(t)}{dt}$$

$$v_C = \frac{1}{C}\int i(t)dt$$

Transfer functions form:

$$I(s) = \frac{1}{R + sL + \frac{1}{sC}} V_{in}(s)$$

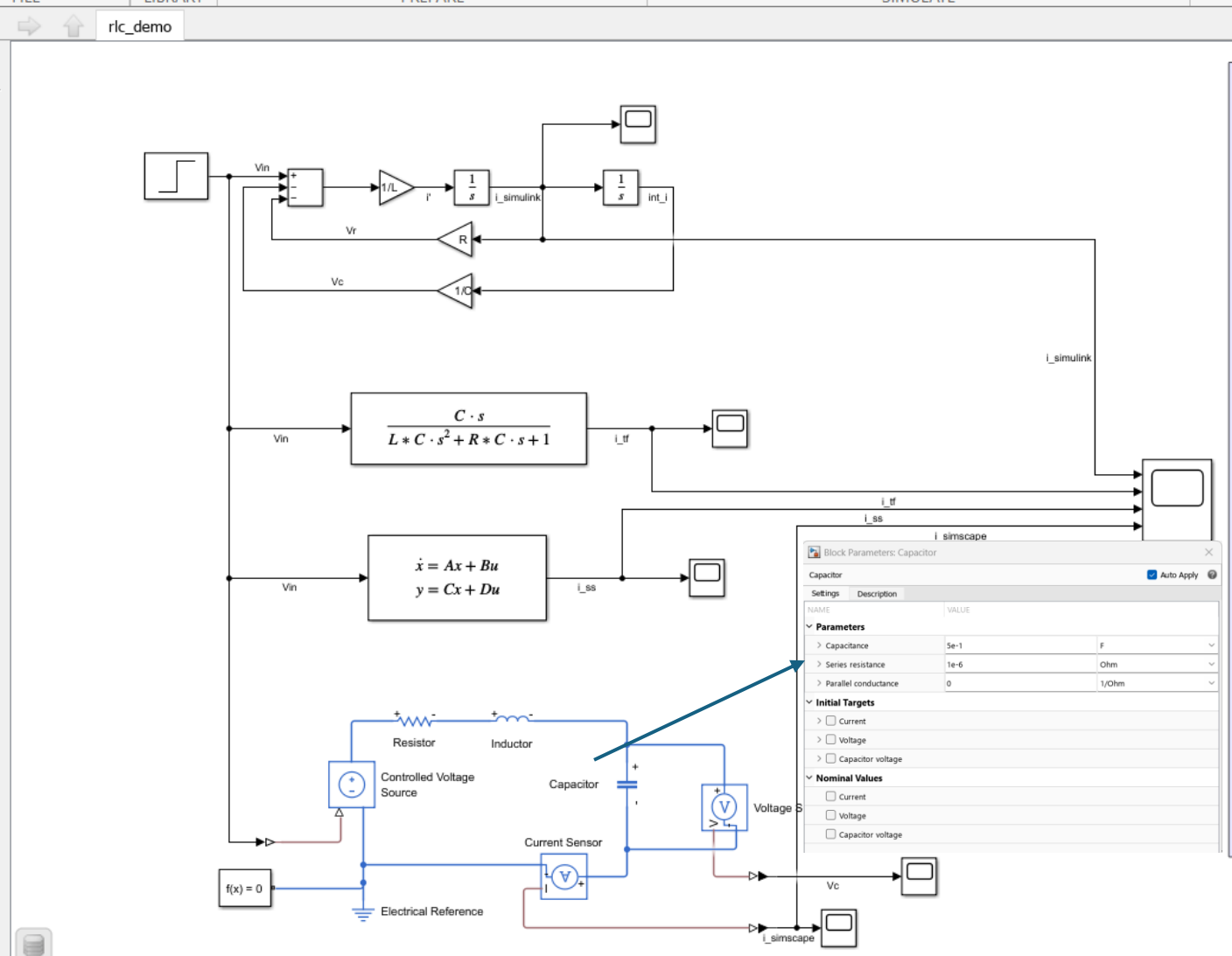
$$H(s) = \frac{I(s)}{V_{in}(s)} = \frac{sC}{LCs^2 + RCs + 1}$$

State-space form: $x_1 = i(t) \quad x_2 = v_C(t)$

$$\dot{x}_1 = \frac{1}{L}(v_{in}(t) - Rx_1 - x_2) \qquad \text{y(t)}= i(t)$$

$$\dot{x}_2 = \frac{x_1}{C}$$

RLC Circuit Modeling



$$v_{in}(t) = Ri(t) + L\frac{di(t)}{dt} + \frac{1}{C}\int i(t)dt$$

$$H(s) = \frac{I(s)}{V_{in}(s)} = \frac{sC}{LCs^2 + RCs + 1}$$

Durum değişkenleri

$$x_1 = i(t), \qquad x_2 = v_C(t)$$

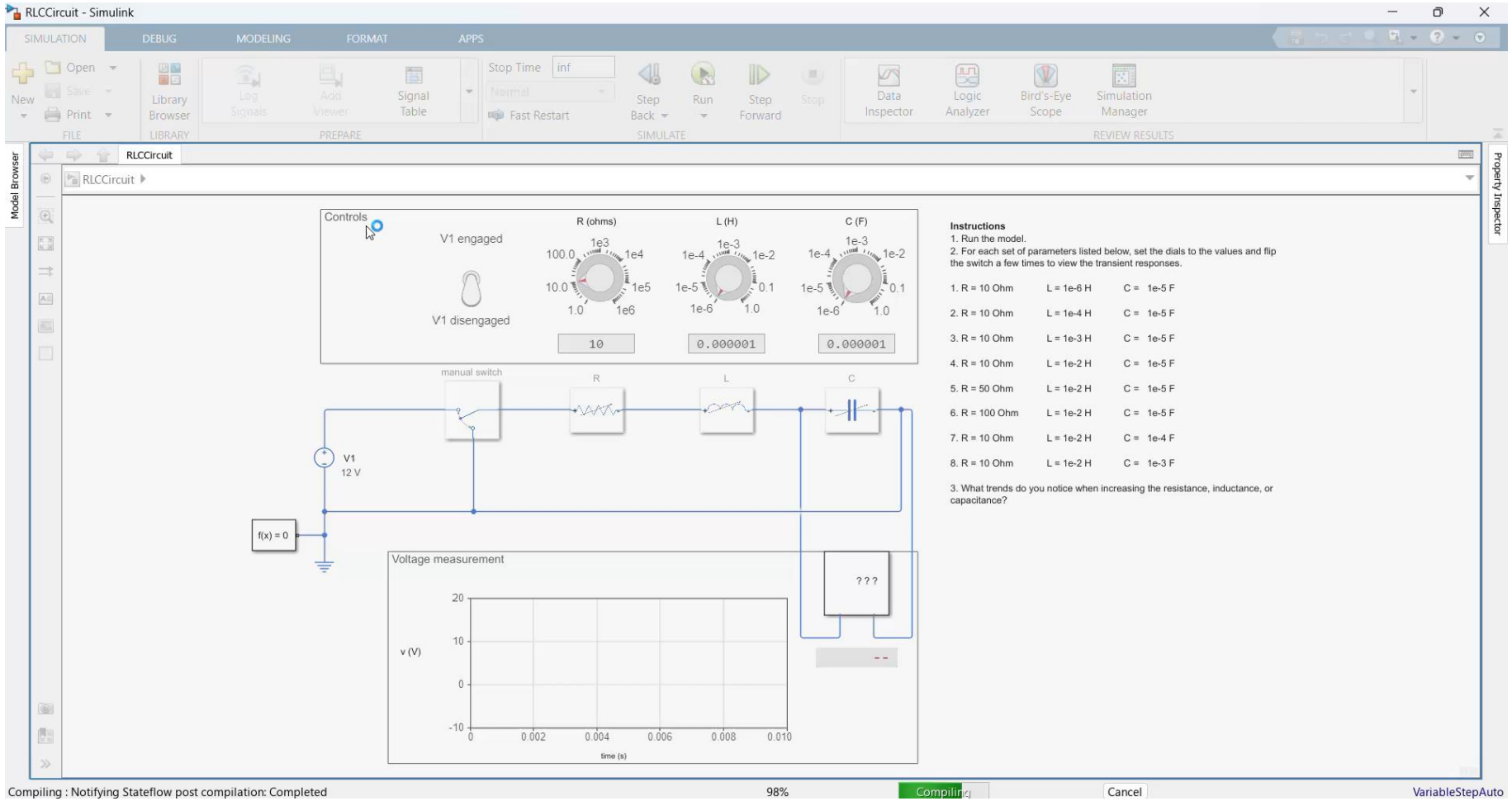
State-space modeli

$$\dot{x}_1 = \frac{1}{L}\left(v_{in}(t) - Rx_1 - x_2\right)$$
$$\dot{x}_2 = \frac{x_1}{C}$$

Çıkış

$$y(t) = i(t) = x_1$$

RLC Circuit Modeling



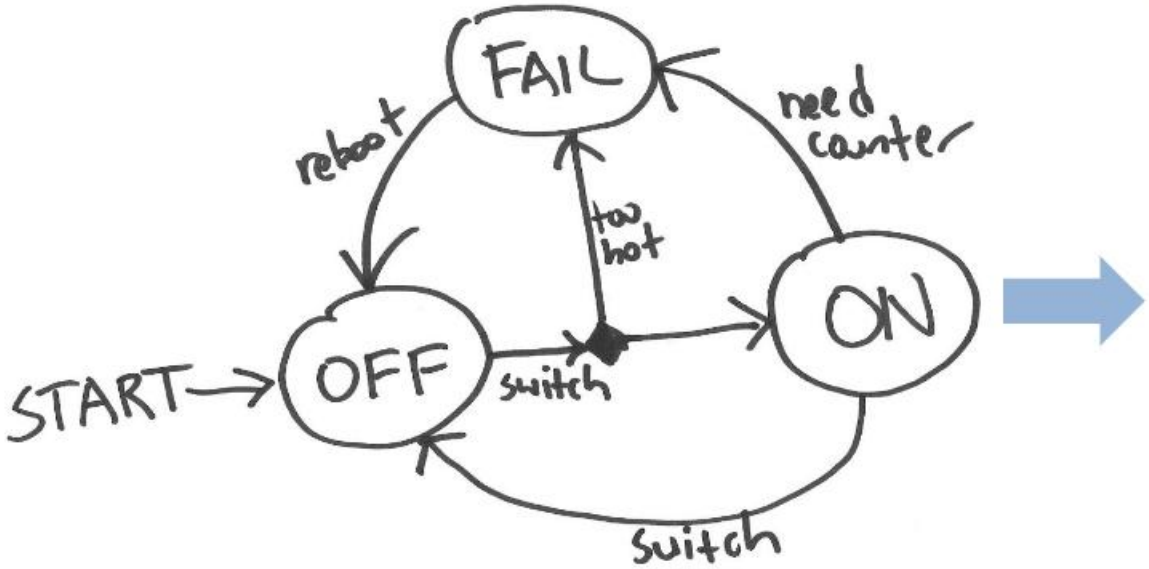
Can you Find the Mistakes?

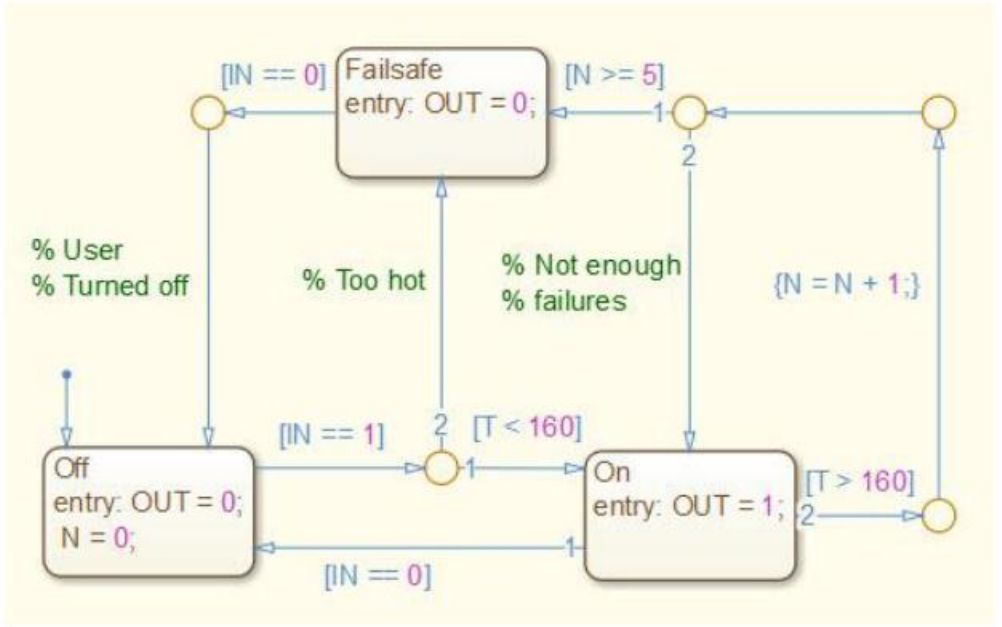
```
function Logic = fcn(GoodComms, GCSCmds, IsBall, AngularState, ...
                     TranslationalState, IMUStatus, BATT_SENS)
persistent LogicOut count;

if isempty(LogicOut)
    LogicOut.CalibrateSensors = false;
    LogicOut.Mode = uint8(1);
    count = 0;
end

TORampTime = 2;
ts_comms = 0.2;
TM_SettleTime = 2;

switch LogicOut.Mode
    case 1 %WaitForComms
        if GoodComms && IMUStatus.GoodIMUData && BATT_SENS > 3.3
            LogicOut.Mode = uint8(2);
        end
    case 2 %Init
        if GCSCmds.GCS_CalibrateCmd
            LogicOut.Mode = uint8(3);
            LogicOut.CalibrateSensors = true;
        end
    case 3 %Calibration
        if IMUStatus.CalibrationDone
            LogicOut.Mode = uint8(4);
        elseif GCSCmds.GCS_EMERGENCY_OFF || ~GoodComms
            LogicOut.Mode = uint8(10);
            count = 0;
        end
        LogicOut.CalibrateSensors = false;
    case 4 %Standby
        if CrashCheck(GoodComms, GCSCmds, AngularState)
            LogicOut.Mode = uint8(10);
            count = 0;
        elseif GCSCmds.GCS_MissionMode == 1
            LogicOut.Mode = uint8(6);
            count = 0;
        elseif GCSCmds.GCS_CalibrateCmd
            LogicOut.Mode = uint8(3);
            LogicOut.CalibrateSensors = true;
        elseif BATT_VOL > 2 && GCSCmds.GCS_TAKEOFF
            LogicOut.Mode = uint8(6);
        end
    case 5 %Takeoff
        if CrashCheck(GoodComms, GCSCmds, AngularState)
            LogicOut.Mode = uint8(10);
            count = 0;
        elseif ~GoodComms
            LogicOut.Mode = uint8(9);
        elseif count >= uint8((TORampTime+2)/ts_comms) && IsBall
            LogicOut.Mode = uint8(7);
        elseif GCSCmds.GCS_TestCmd
            LogicOut.Mode = uint8(6);
            count = 0;
        elseif BATT_VOL <= 2
            LogicOut.Mode = uint8(4);
        end
        count = count + 1;
    case 6 %TestManeuvers
        if CrashCheck(GoodComms, GCSCmds, AngularState)
            LogicOut.Mode = uint8(10);
            count = 0;
        elseif count >= uint8(TM_SettleTime/ts_comms - 1)
            if IsBall
                LogicOut.Mode = uint8(7);
            else
                LogicOut.Mode = uint8(8);
                count = 0;
            end
        end
        count = count + 1;
    case 7 %Track3D
        if GCSCmds.GCS_MissionMode == 2 || ~GoodComms || ...
                BATT_SENS <= 2
            LogicOut.Mode = uint8(9);
        elseif CrashCheck(GoodComms, GCSCmds, AngularState)
            LogicOut.Mode = uint8(10);
            count = 0;
        elseif ~IsBall
            LogicOut.Mode = uint8(8);
            count = 0;
        elseif GCSCmds.GCS_TestCmd
            LogicOut.Mode = uint8(6);
            count = 0;
        end
    case 8 %LostBall
        if CrashCheck(GoodComms, GCSCmds, AngularState)
            LogicOut.Mode = uint8(10);
            count = 0;
        elseif count >= (3/ts_comms - 1) || ~GoodComms
            LogicOut.Mode = uint8(9);
        elseif IsBall
            LogicOut.Mode = uint8(7);
        elseif GCSCmds.GCS_TestCmd
            LogicOut.Mode = uint8(6);
            count = 0;
        end
        count = count + 1;
    case 9 %Land
        if abs(AngularState.Angles(1)) < 0.08727 && ...
                abs(AngularState.Angles(2)) < 0.08727 && ...
                    TranslationalState.Acc_scaled(3) < -15
            LogicOut.Mode = uint8(1);
            count = 0;
        elseif CrashCheck(GoodComms, GCSCmds, AngularState)
            LogicOut.Mode = uint8(10);
            count = 0;
        end
    case 10 %EmergencyOff
        count = count + 1;
end %End Switch

Logic = LogicOut;
end %End Function
```

```
function crash = CrashCheck(GoodComms, GCSCmds, AngularState)
    crash = false;
    if GCSCmds.GCS_EMERGENCY_OFF || ~GoodComms || ...
            AngularState.Angles(1) > 0.5235 || ...
                AngularState.Angles(2) > 0.5235
        crash = true;
    end
end
```

When should I switch from Simulink to Stateflow?

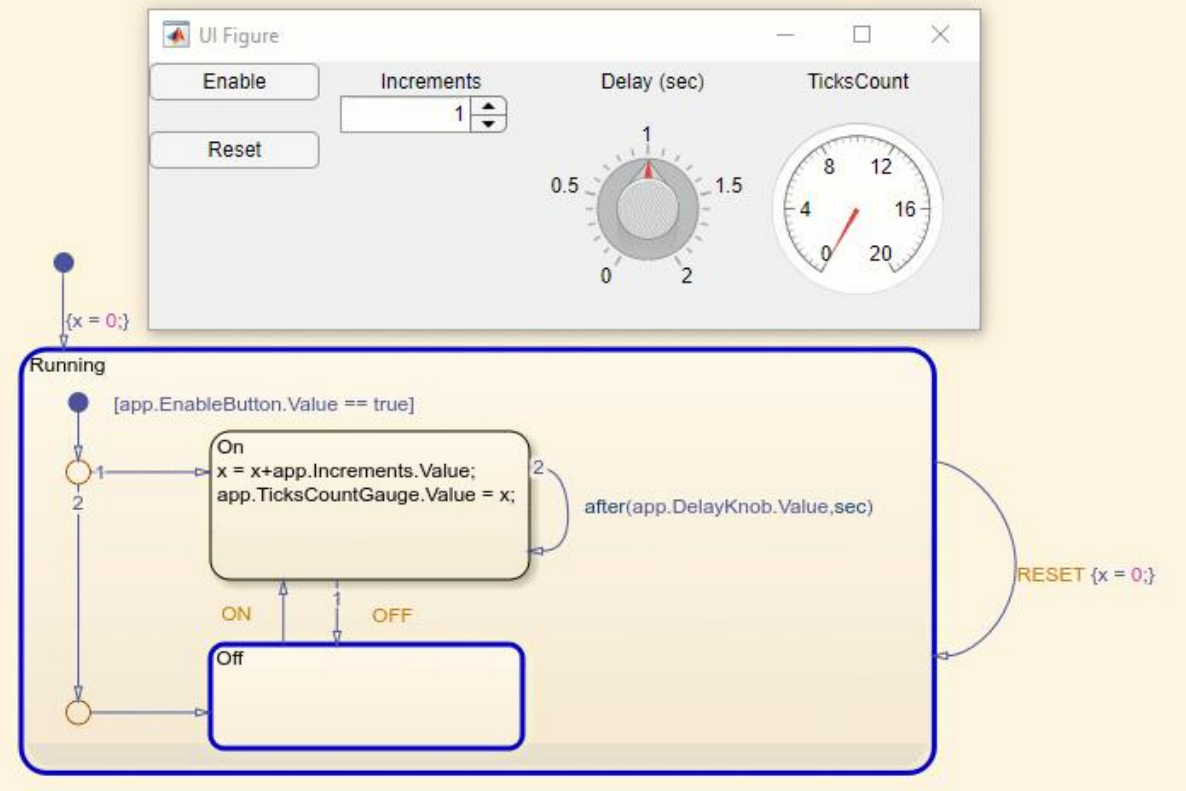
- A graphical state machine design environment that sits on top of Simulink
- What is a state machine?

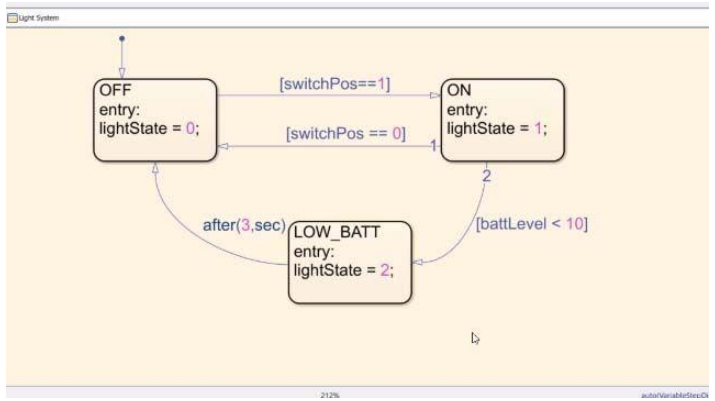


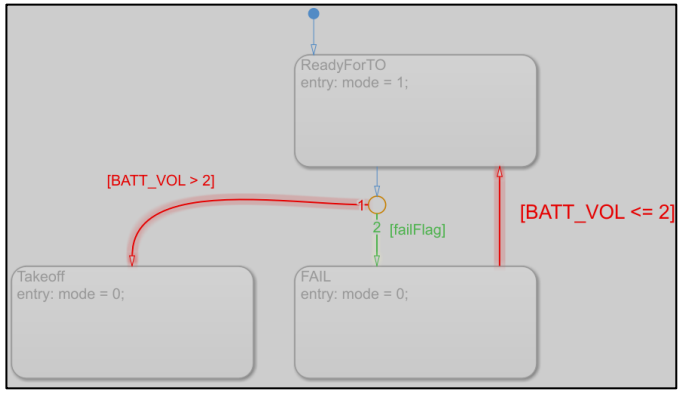


MathWorks Products Overview - **Stateflow**

- Easily design and maintain **complex logic**
- Animations help **visually debug systems**

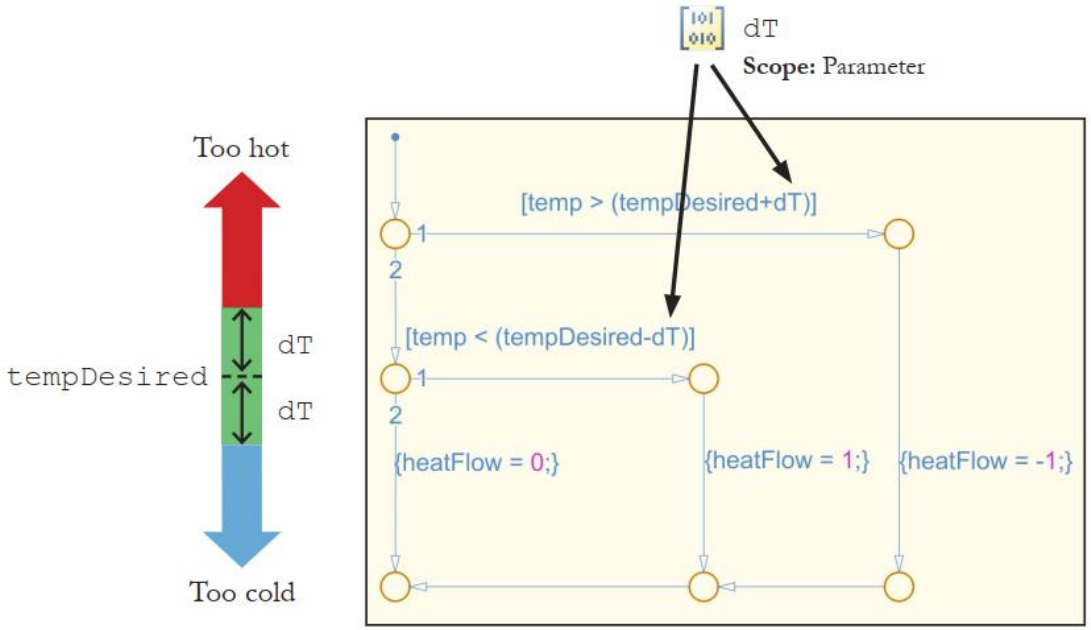


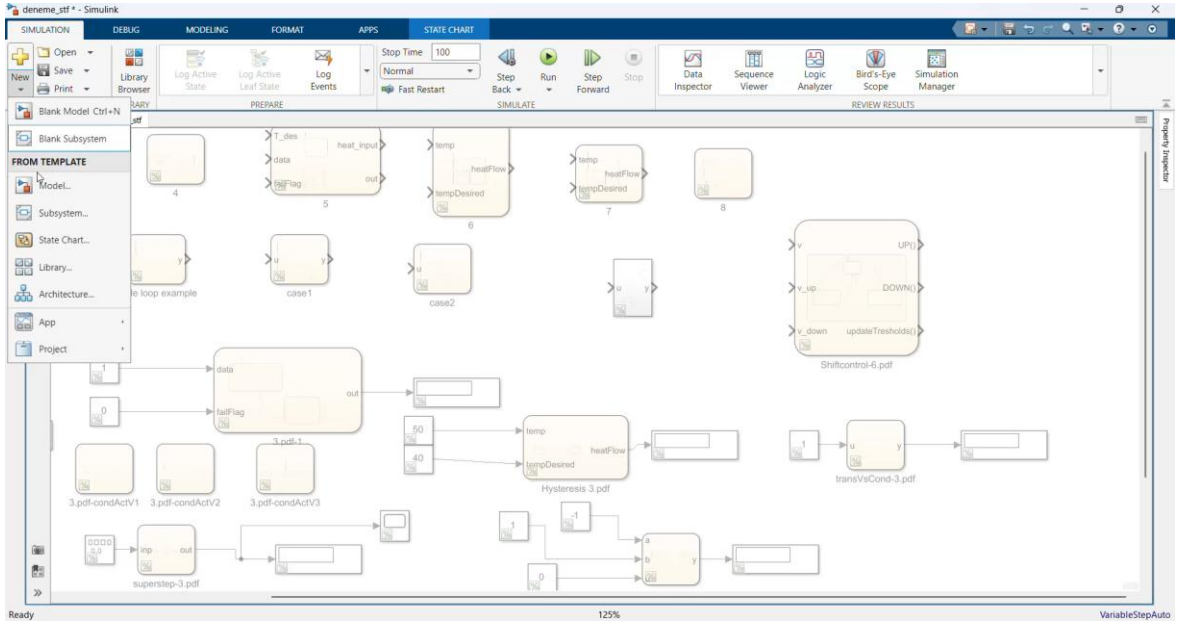





```
if (untitled_DW.is_active_c3_untitled == 0U) {
  /* Entry: Chart */
  untitled_DW.is_active_c3_untitled = 1U;

  /* Entry Internal: Chart */
  /* Transition: '<S1>:5' */
  untitled_DW.is_c3_untitled = untitled_IN_ReadyForTO;

  /* Entry 'ReadyForTO': '<S1>:3' */
} else {
  switch (untitled_DW.is_c3_untitled) {
   case untitled_IN_FAIL:
    /* During 'FAIL': '<S1>:1' */
    /* Transition: '<S1>:2' */
    untitled_DW.is_c3_untitled = untitled_IN_ReadyForTO;

    /* Entry 'ReadyForTO': '<S1>:3' */
    break;

   case untitled_IN_ReadyForTO:
    /* During 'ReadyForTO': '<S1>:3' */
    /* Transition: '<S1>:4' */
    break;

   default:
    /* During 'Takeoff': '<S1>:8' */
    break;
  }
}
```

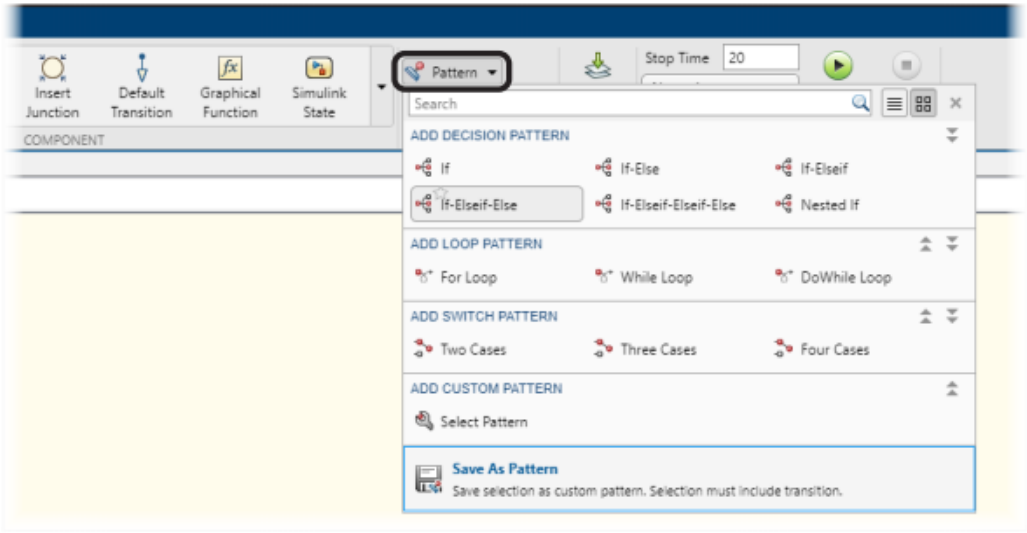
Modeling Flow Charts

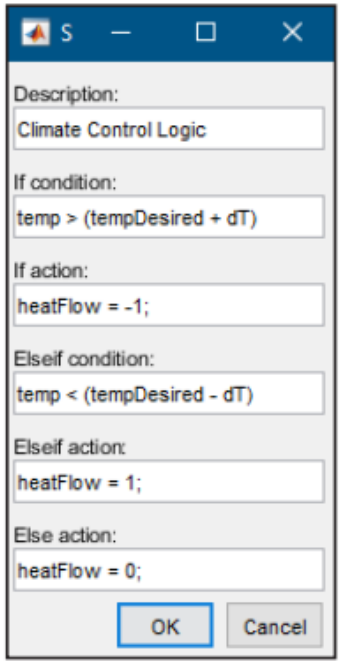


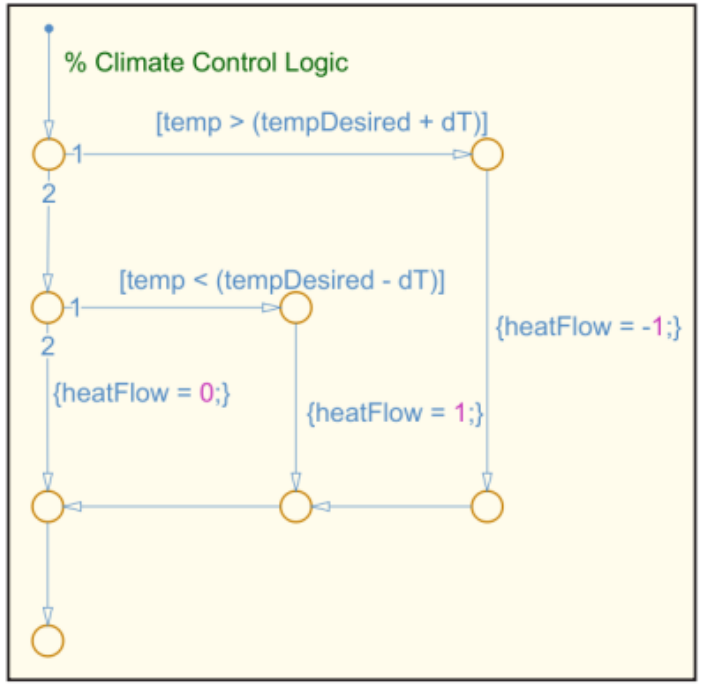


Pattern Wizard

- There are standard patterns that exist for modeling common programming constructs (such as if-elseif and for loops).
- You can automatically generate these patterns using the Stateflow Pattern Wizard



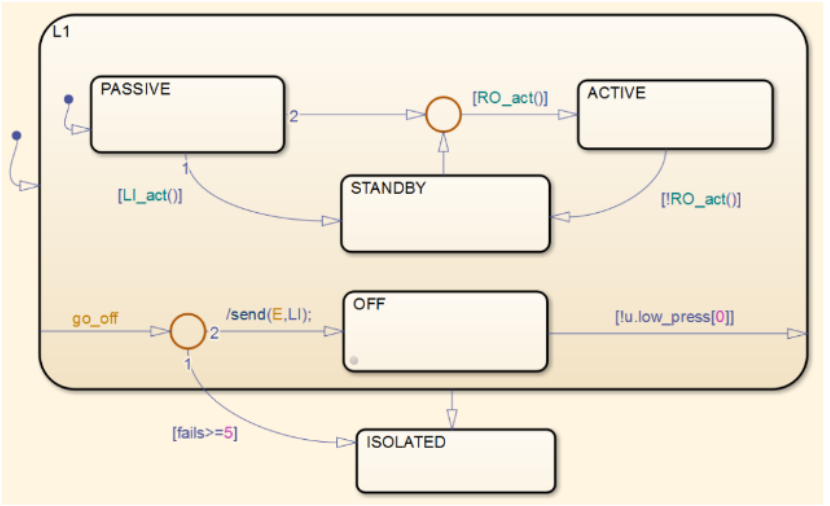




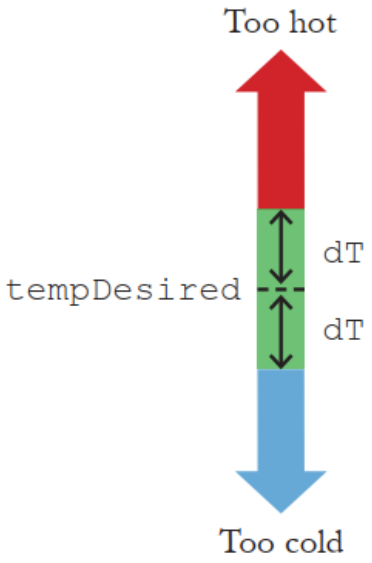
Modeling State Machines

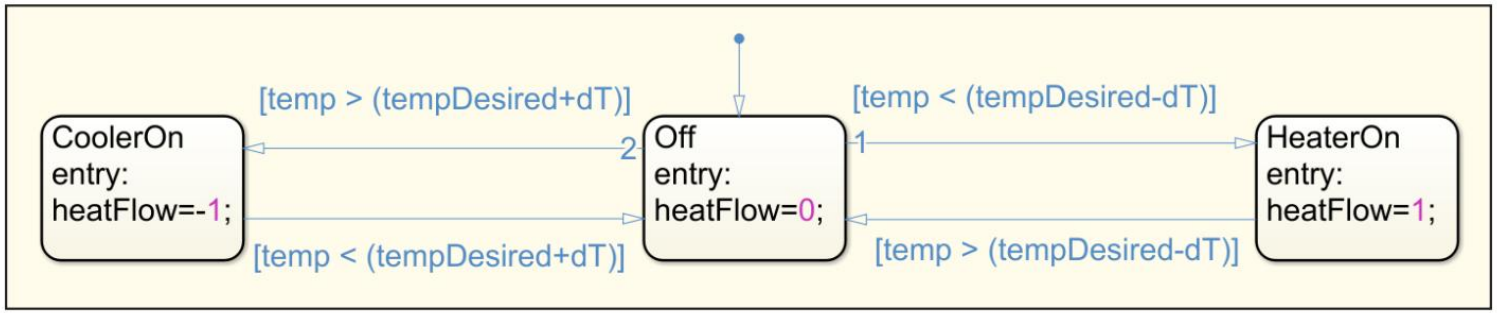
A state transition diagram contains:

- The possible states within the state machine
- The logic required to transition from one state to another
- The actions taken by the system for each particular state



E.g. Fault Management



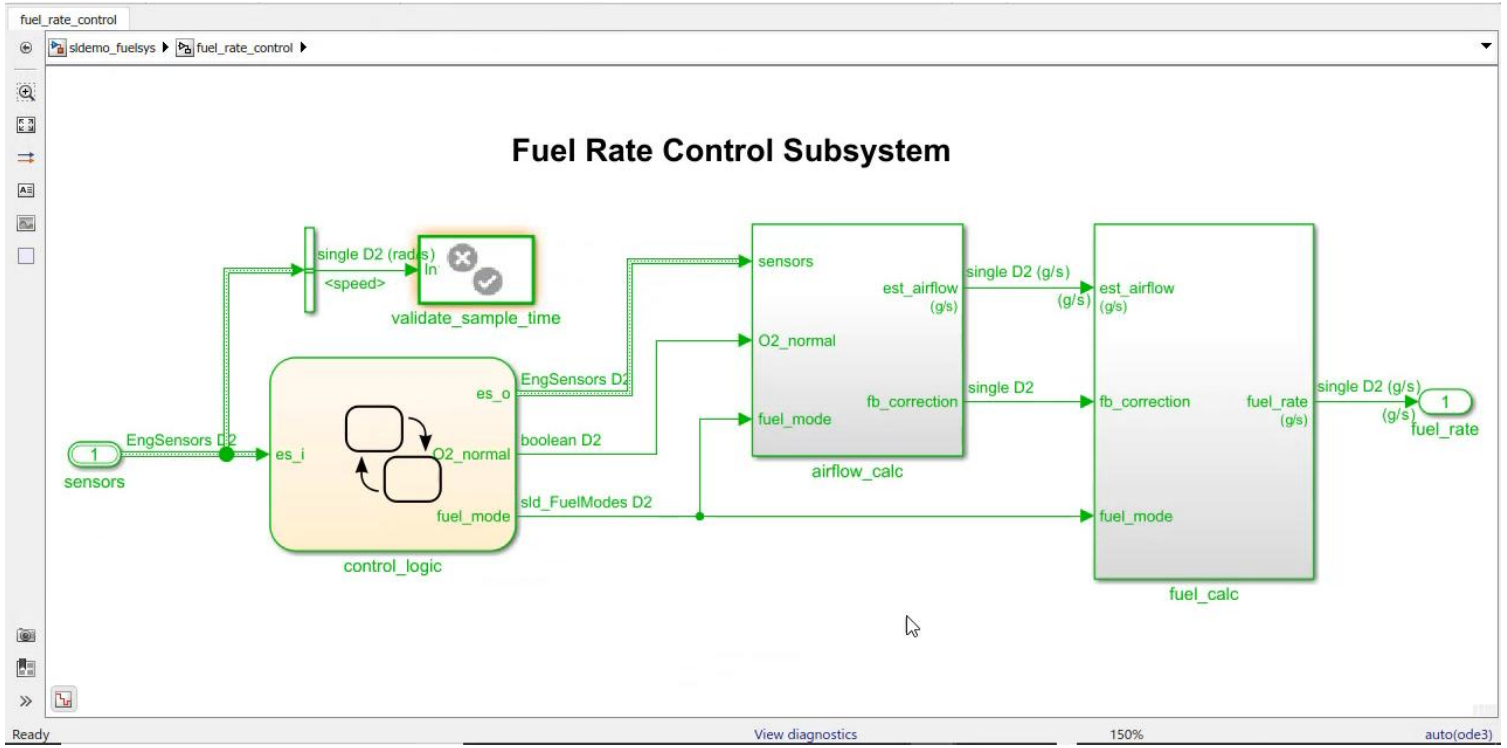


Automatically Generated Code Behaves The Same As The System Model

Integrate generated code

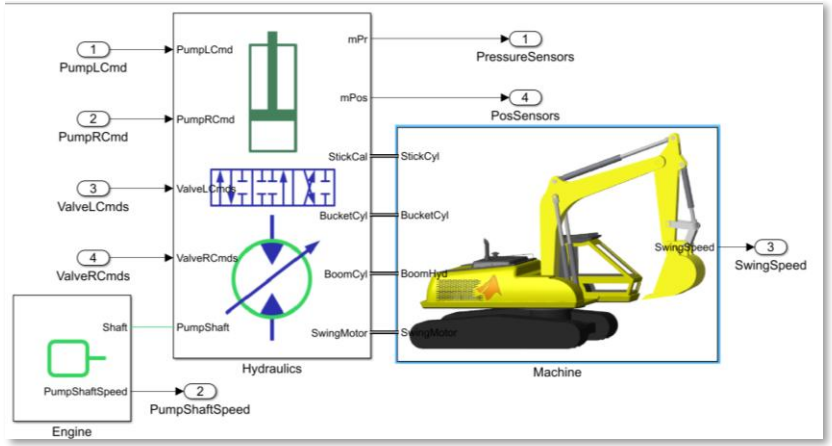
Reduce development time

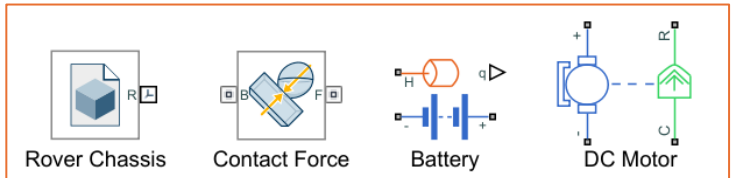
Eliminate hand-coding errors



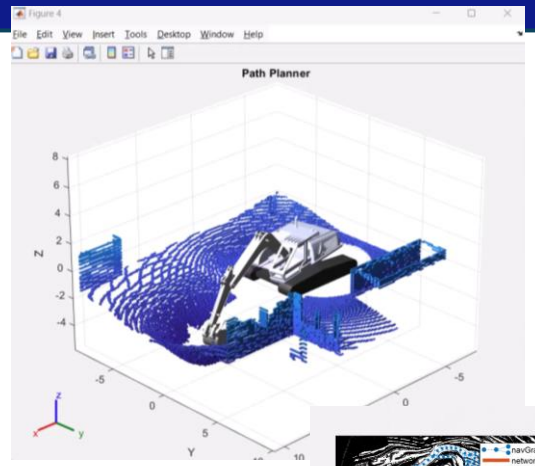
Physical Modeling of Offroad Vehicles

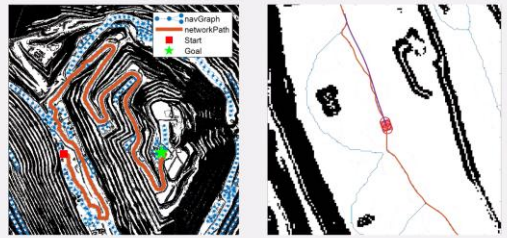
ISO 6015:2006
Earth-moving machinery — Hydraulic excavators and backhoe loaders — Methods of determining tool forces

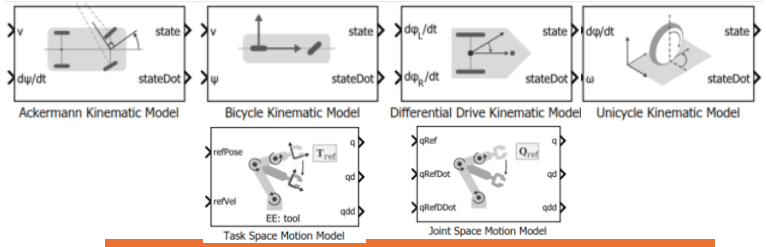




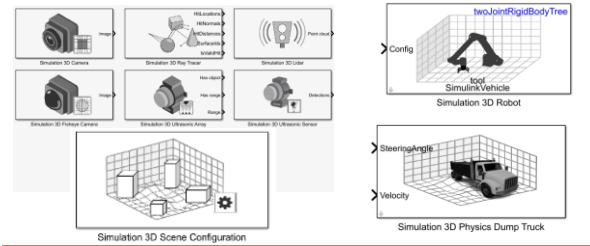
Dynamic Models







Kinematic Models

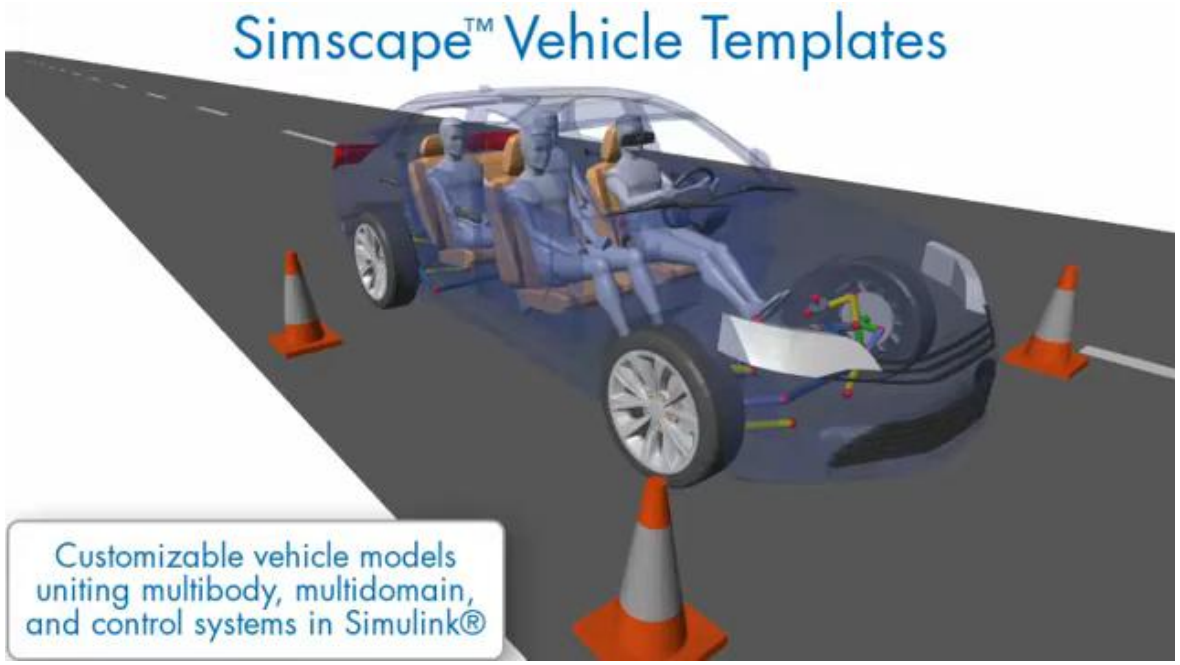


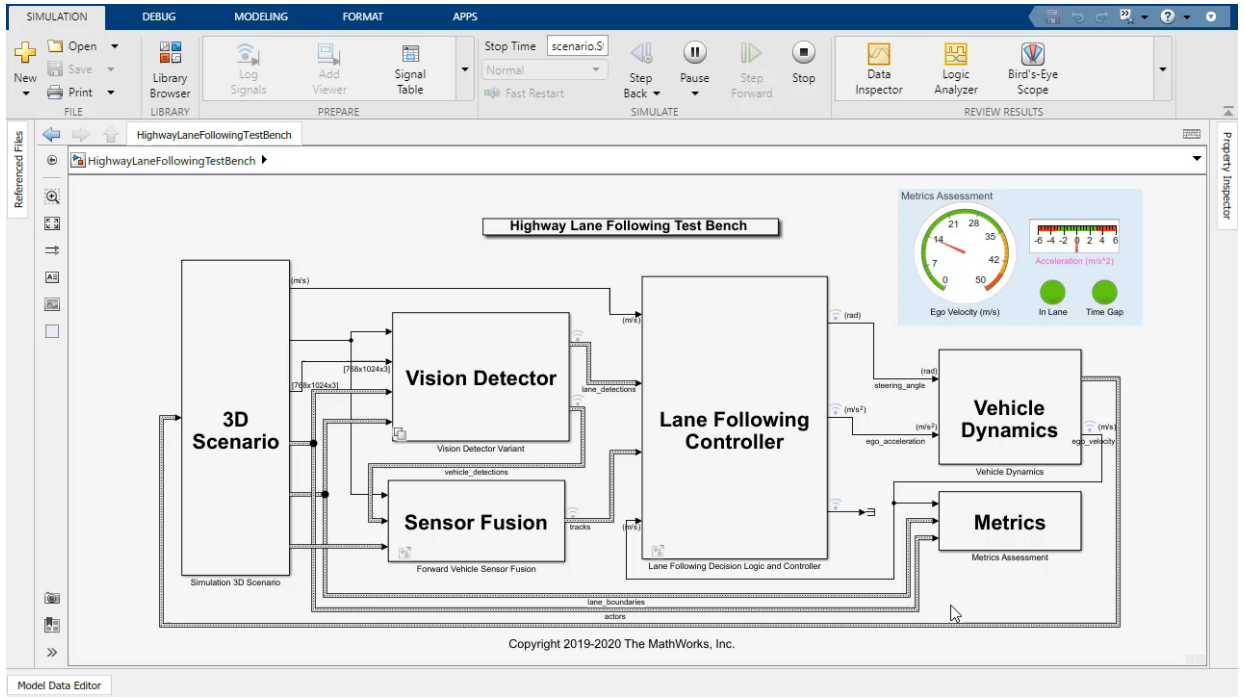
Scene, Scenario, and Agent Models

FIGES applied physics & mathematics | MathWorks Authorized Reseller

BİZİ TAKİP EDİN!

Simscape Applications





https://www.mathworks.com/matlabcentral/fileexchange/79484-simscape-vehicle-templates

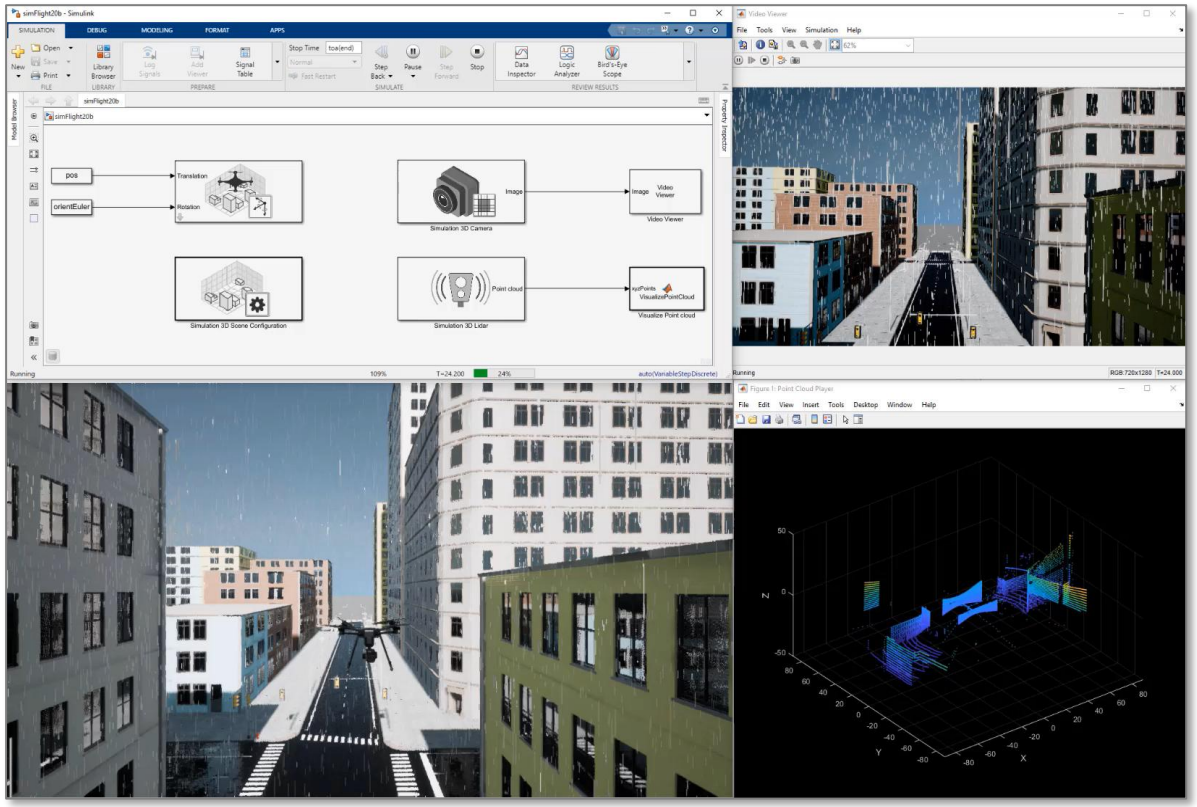
Aerospace Applications

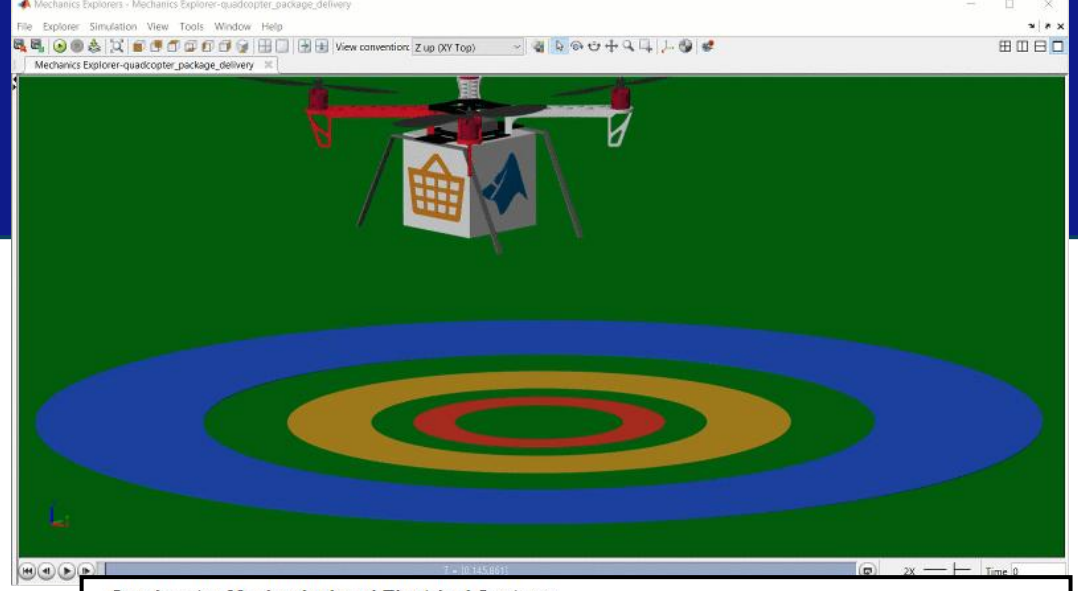
Fly the De Havilland Beaver with Unreal Engine® Visualization

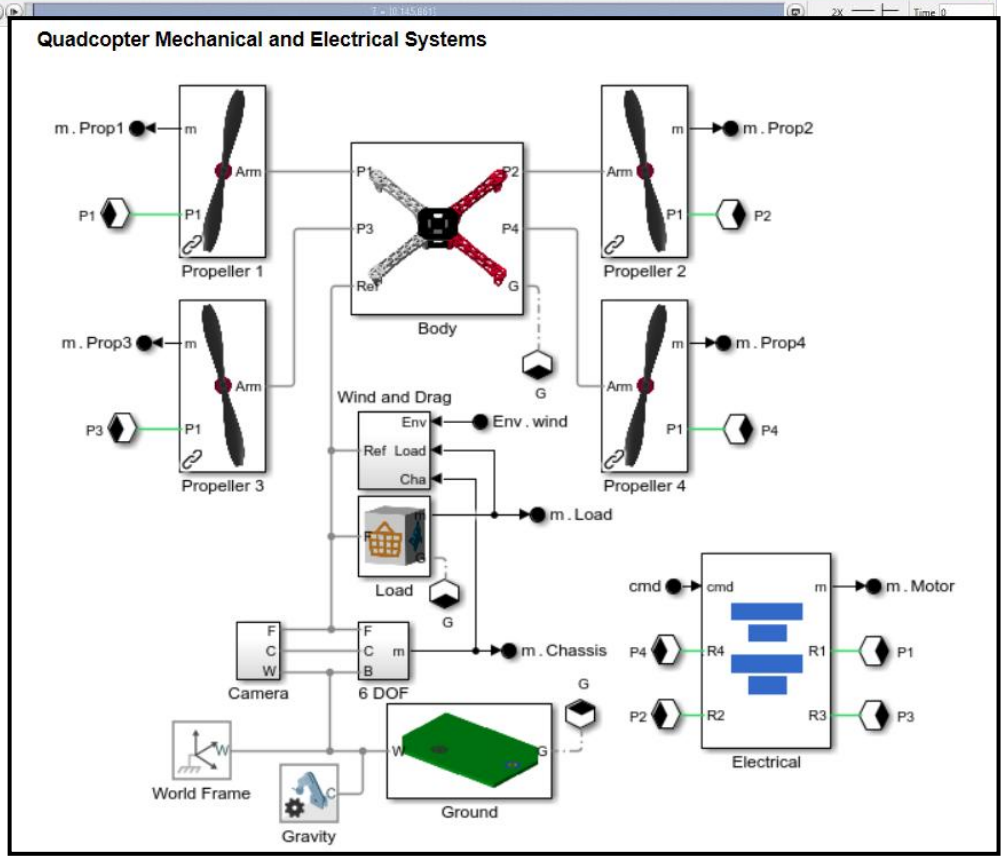
Cesium Ion®

Rotorcraft
Unreal Engine Visualization Requires Simulink® 3D Animation™

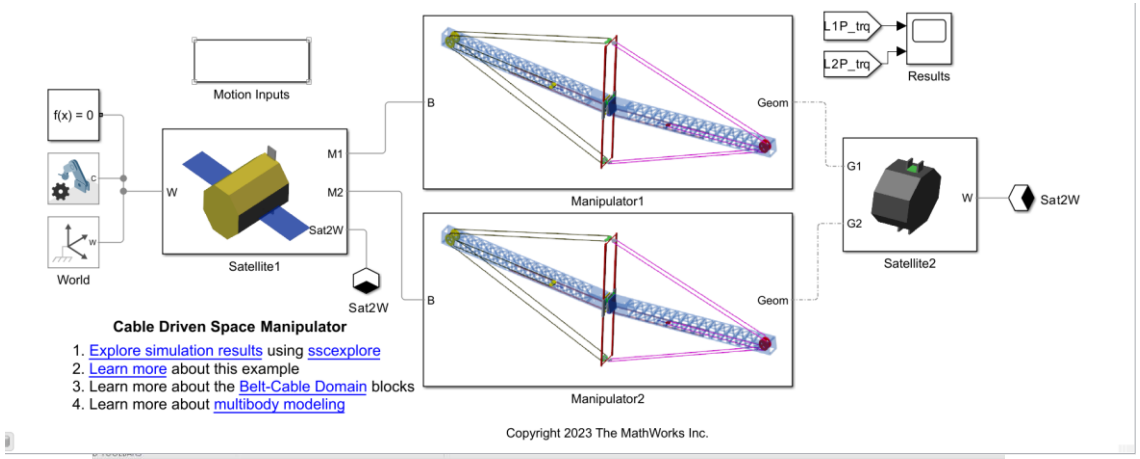
Aerospace Applications

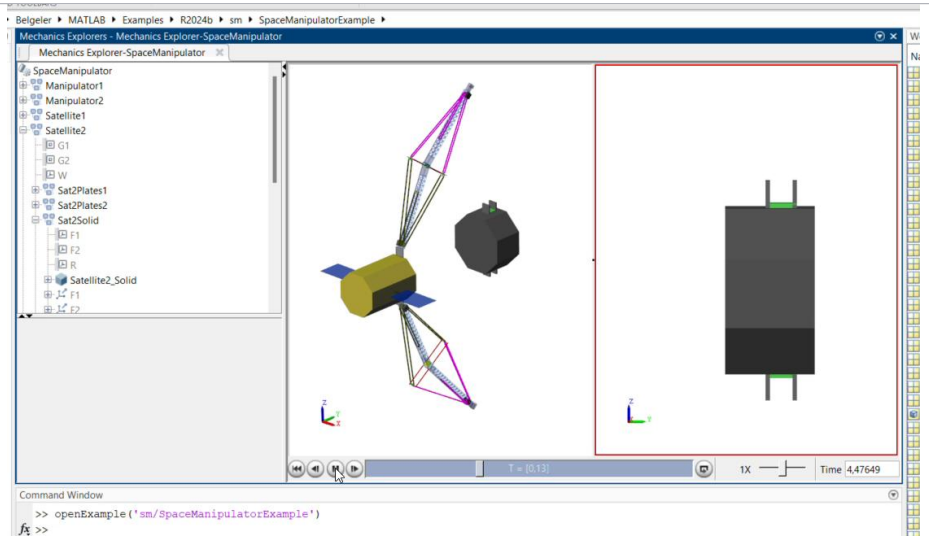






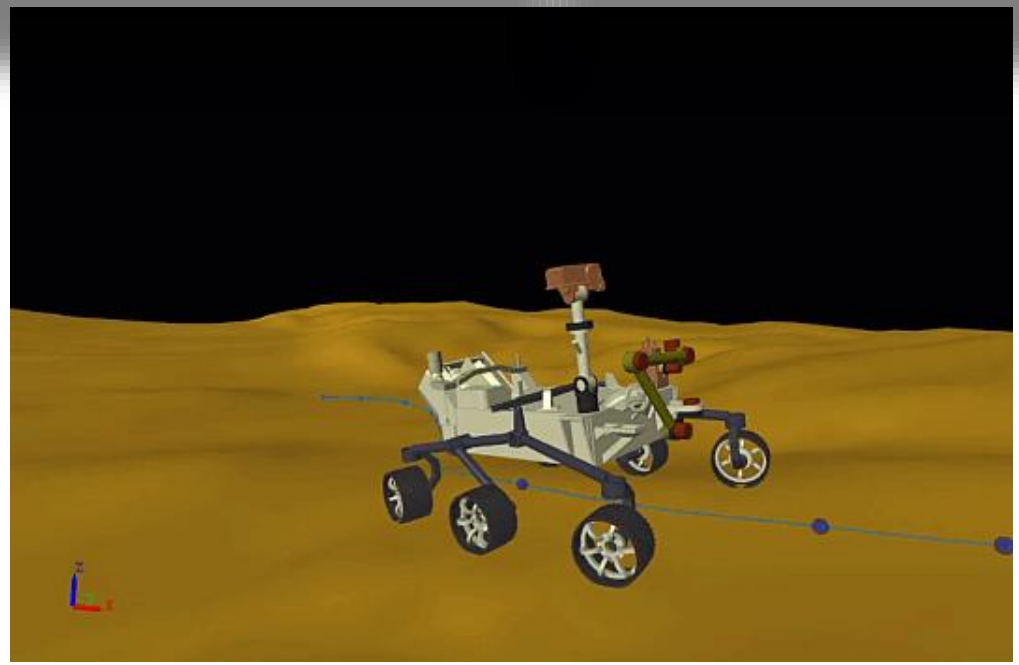
Simscape Multibody Applications





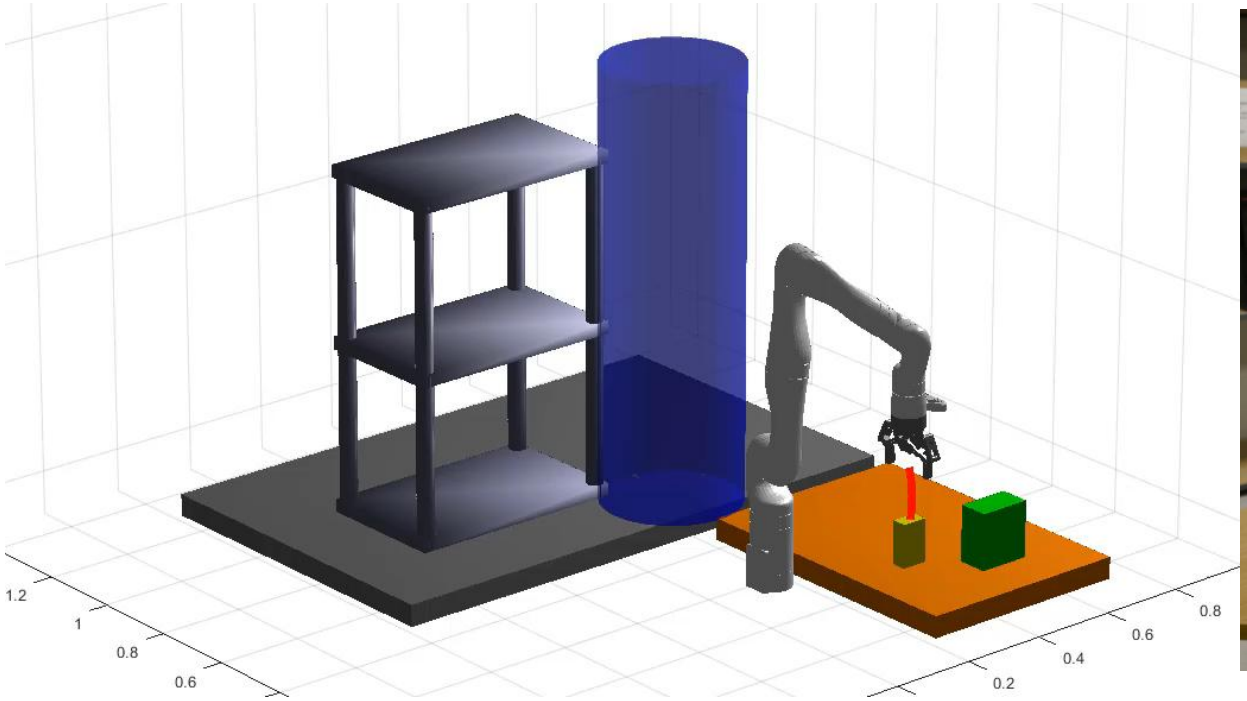
Cable Driven Space Manipulator

https://www.mathworks.com/help/sm/ug/cable-driven-space-manipulator.html



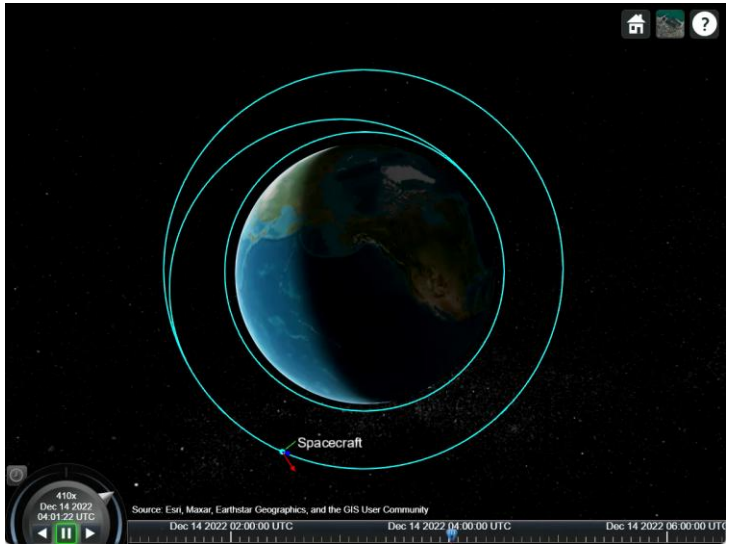
Modeling and Control of a Mars Rover in Simscape Multibody

Robotic Applications

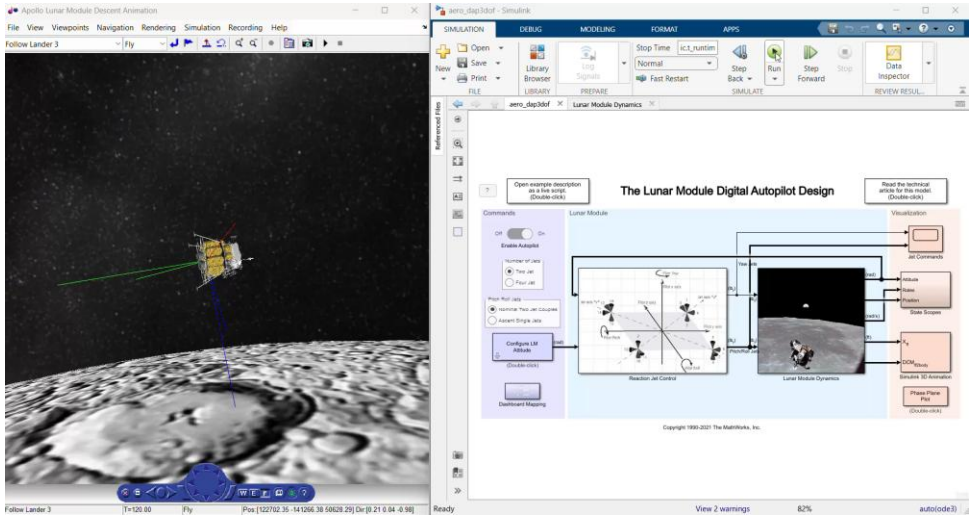


Robotic System Toolbox
Stateflow

Space Applications



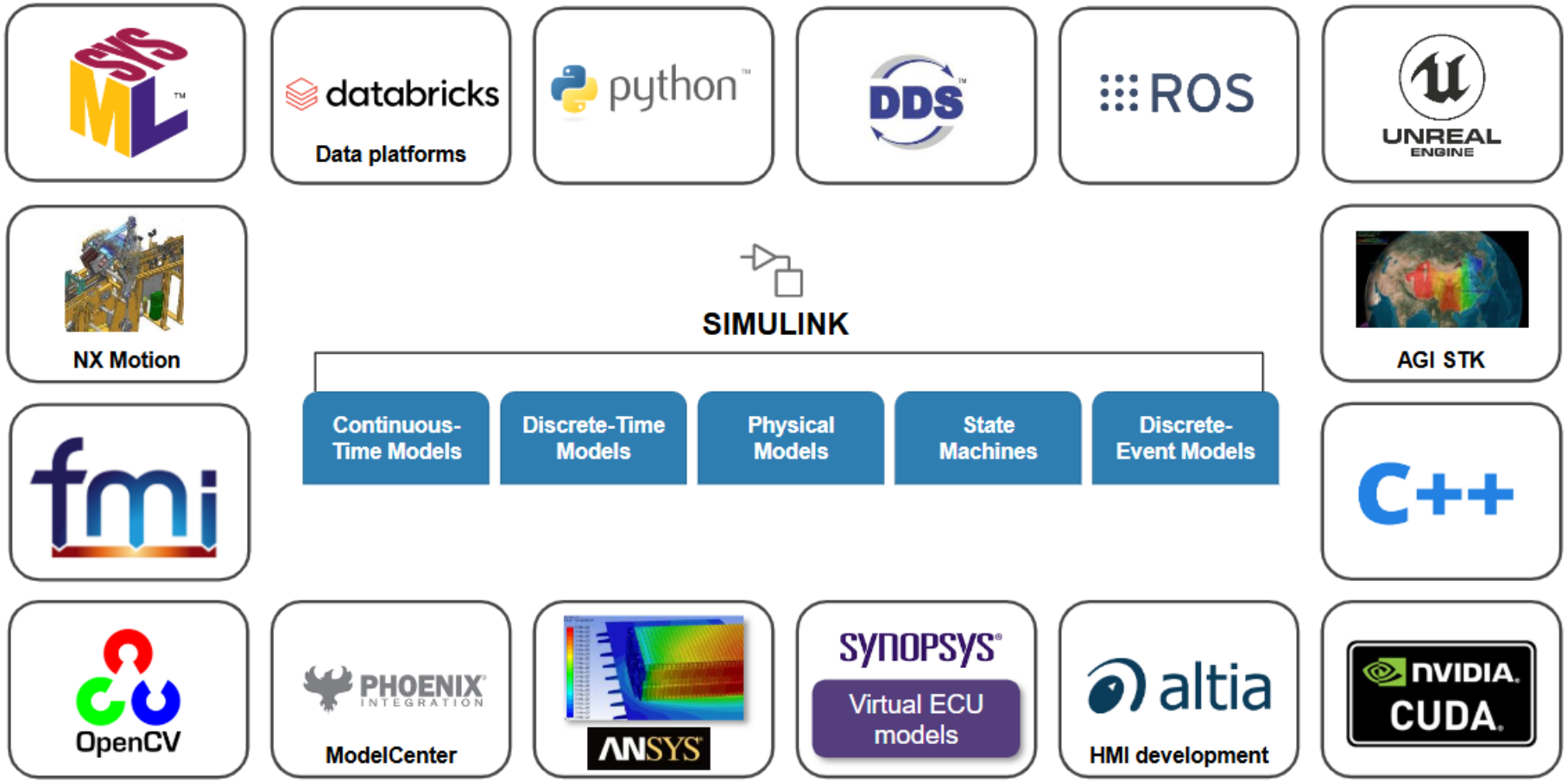
Hohman Transfer Model



Apollo Lunar Module Digital Autopilot

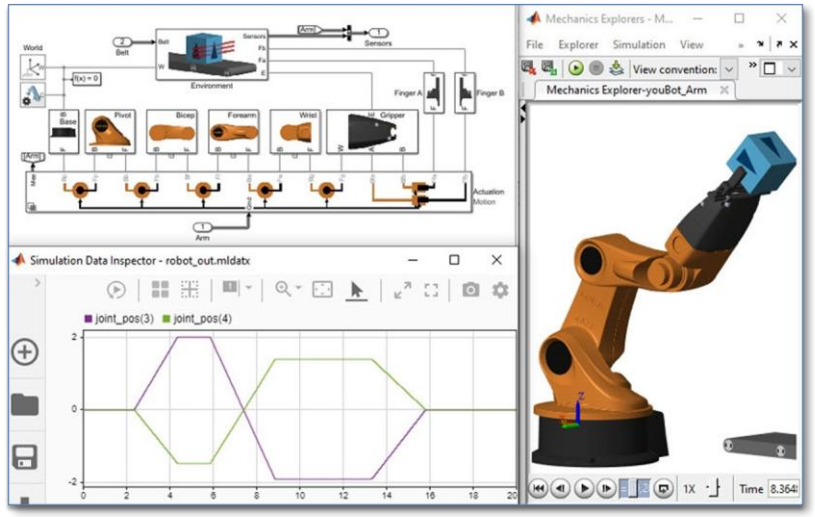
Satellite Conjuction Finder Analysis Model

Ecosystem with 100+ Third-Party Tools and Languages



There are three key pieces to Model-Based Design

Modeling & Simulation



Code Generation

```
604       /* End of Saturate: '<S210>/Saturation' */
605
606       /* RelationalOperator: '<S196>/NotEqual' */
607       NotEqual_n = (0.0F != Switch_f);
608
609       /* Signum: '<S196>/SignPreSat' */
610       if (Switch_f < 0.0F) {
611         Switch_f = -1.0F;
612       } else {
613         if (Switch_f > 0.0F) {
614           Switch_f = 1.0F;
615         }
616       }
```

Test & Verification

Conditions analyzed

Description	True	False
Condition 1, "alt>10000"	4 U1.1	185 U1.1
Condition 2, "anomaly"	0	4 U1.1

MC/DC analysis (combinations in parentheses did not occur)

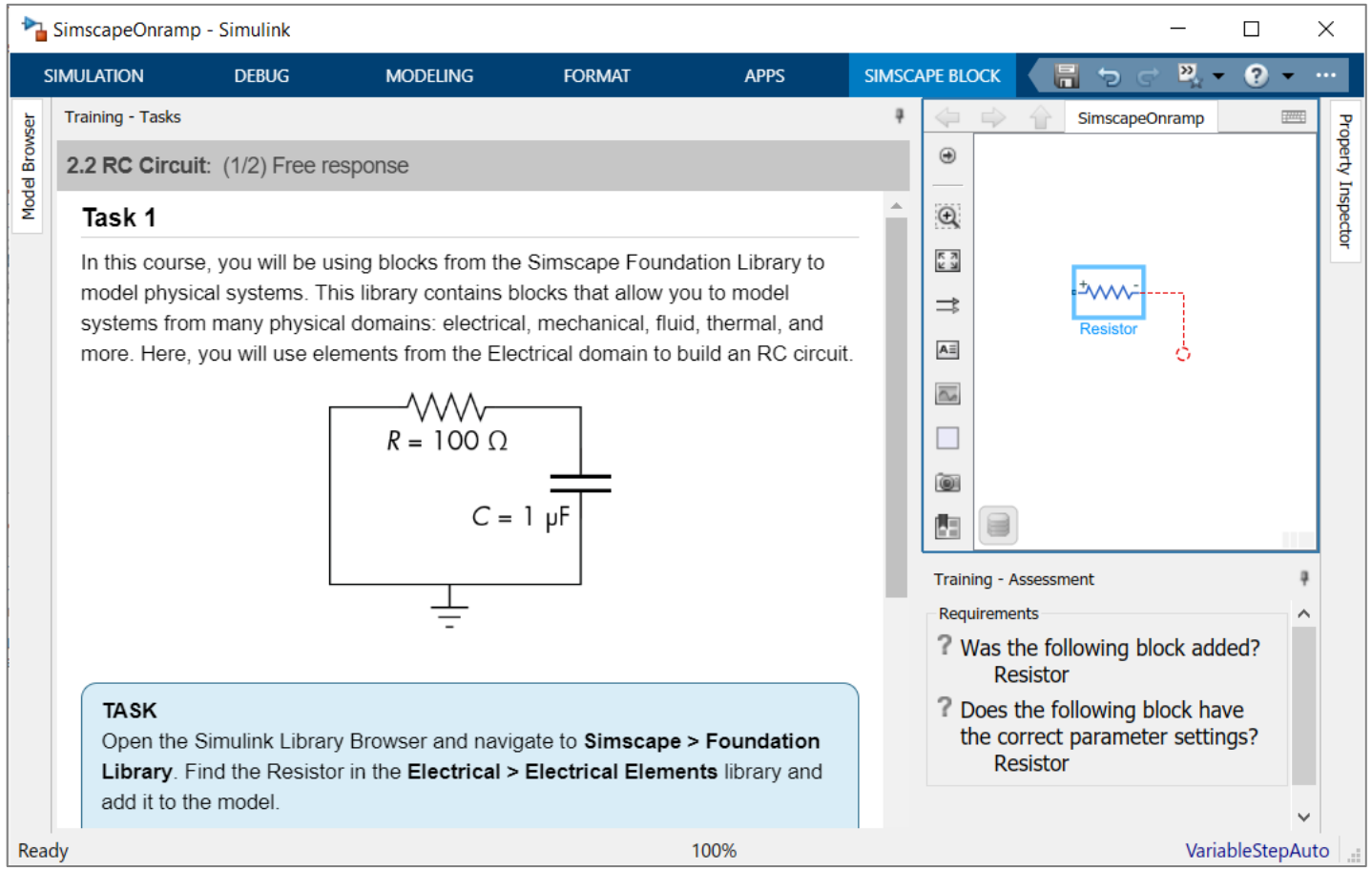
Decision/Condition	True Out	False Out
Transition trigger expression		
Condition 1, "alt>10000"	TF U1.1	Fx U1.1

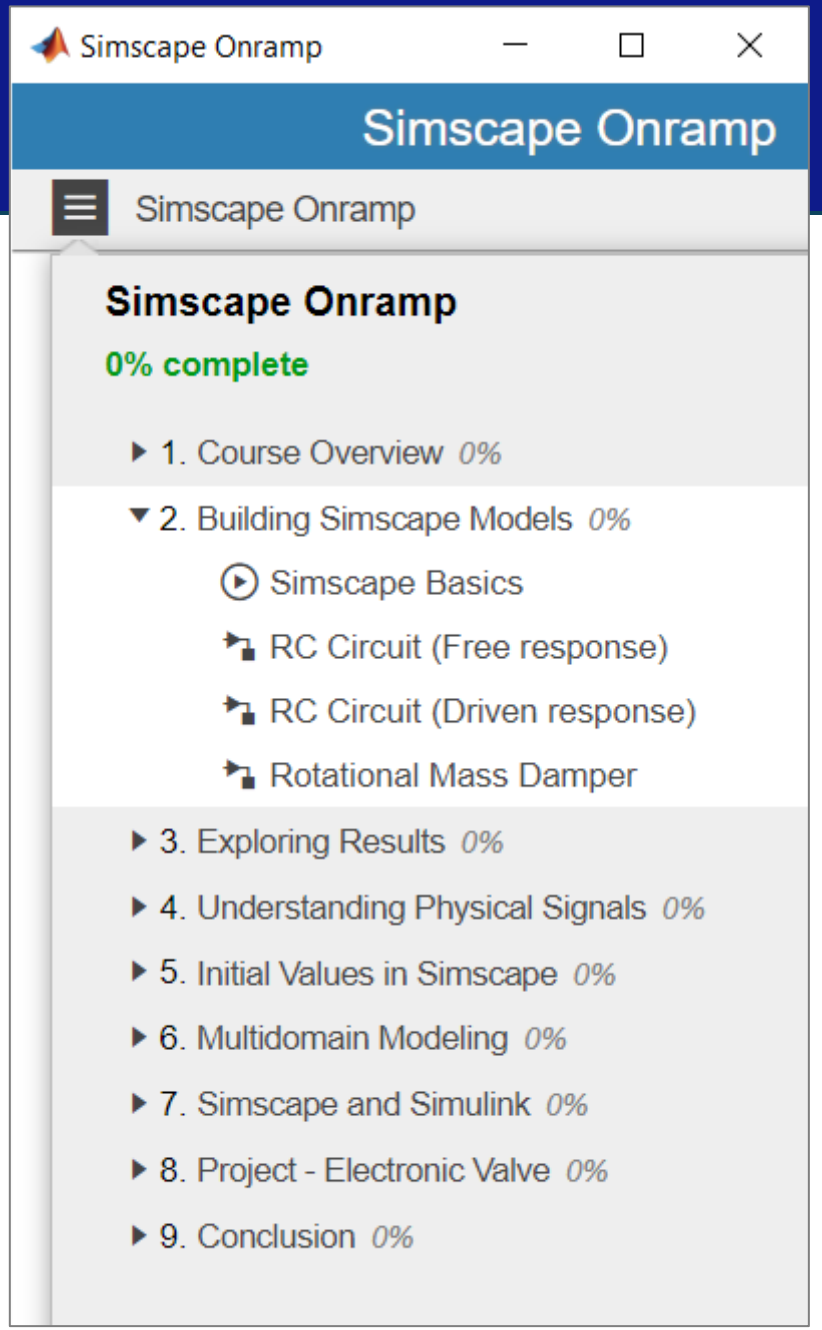
SIMULINK®

Simulation and Model-Based Design

Simscape
Simscape Onramp

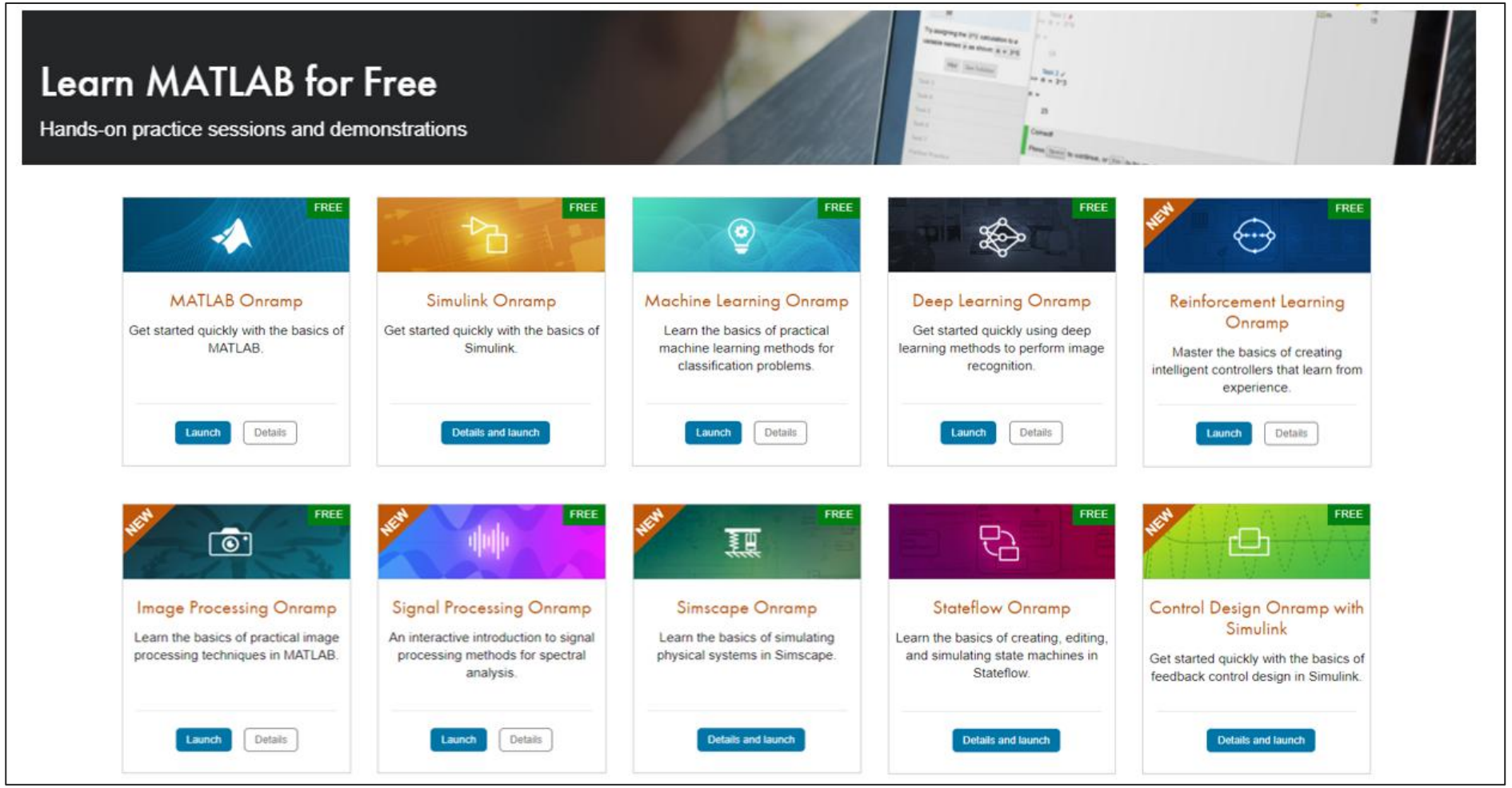
- Self-paced, interactive tutorial for getting started with Simscape





https://www.mathworks.com/services/training.html

Resources and Training



Q & A

aylin.edgu@figes.com.tr

FİGES applied physics & mathematics | MathWorks® Authorized Reseller | TEŞEKKÜRLER