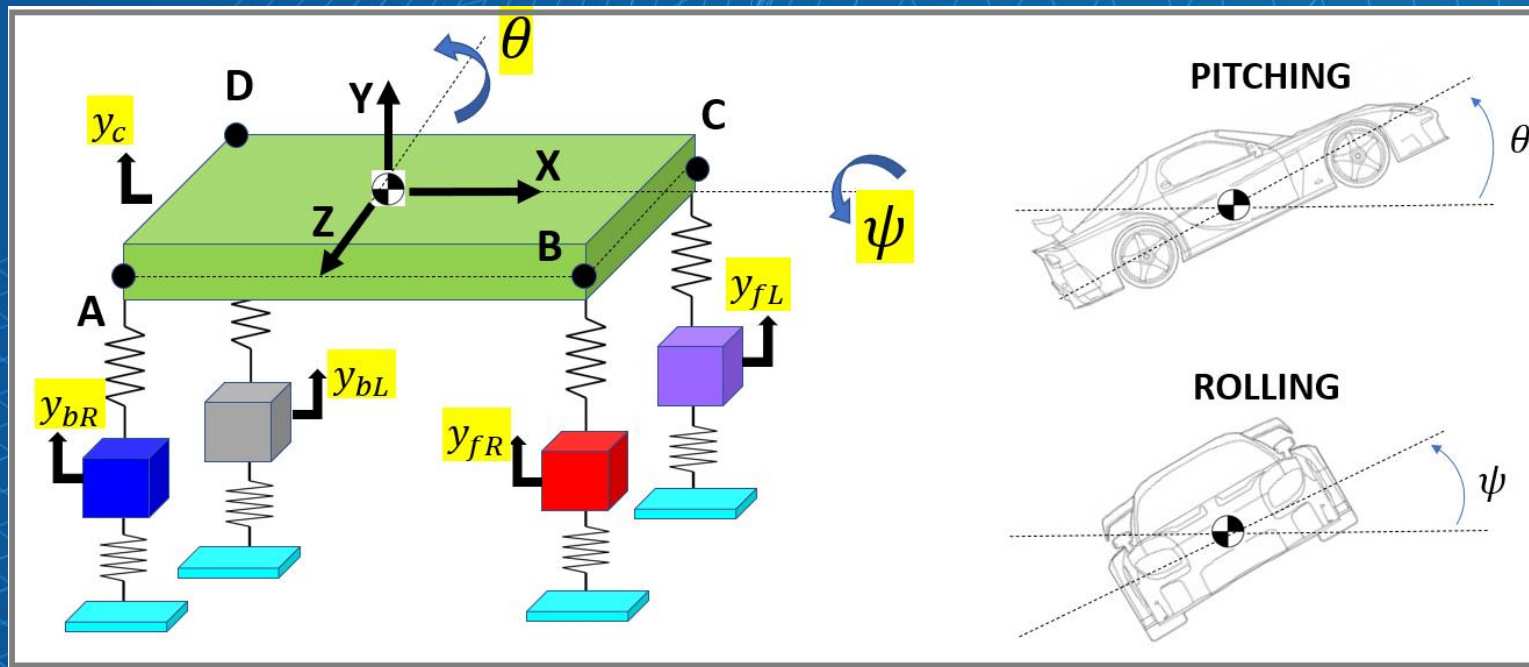
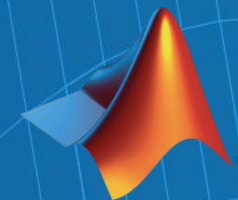


7-dof Transverse Car Dynamics

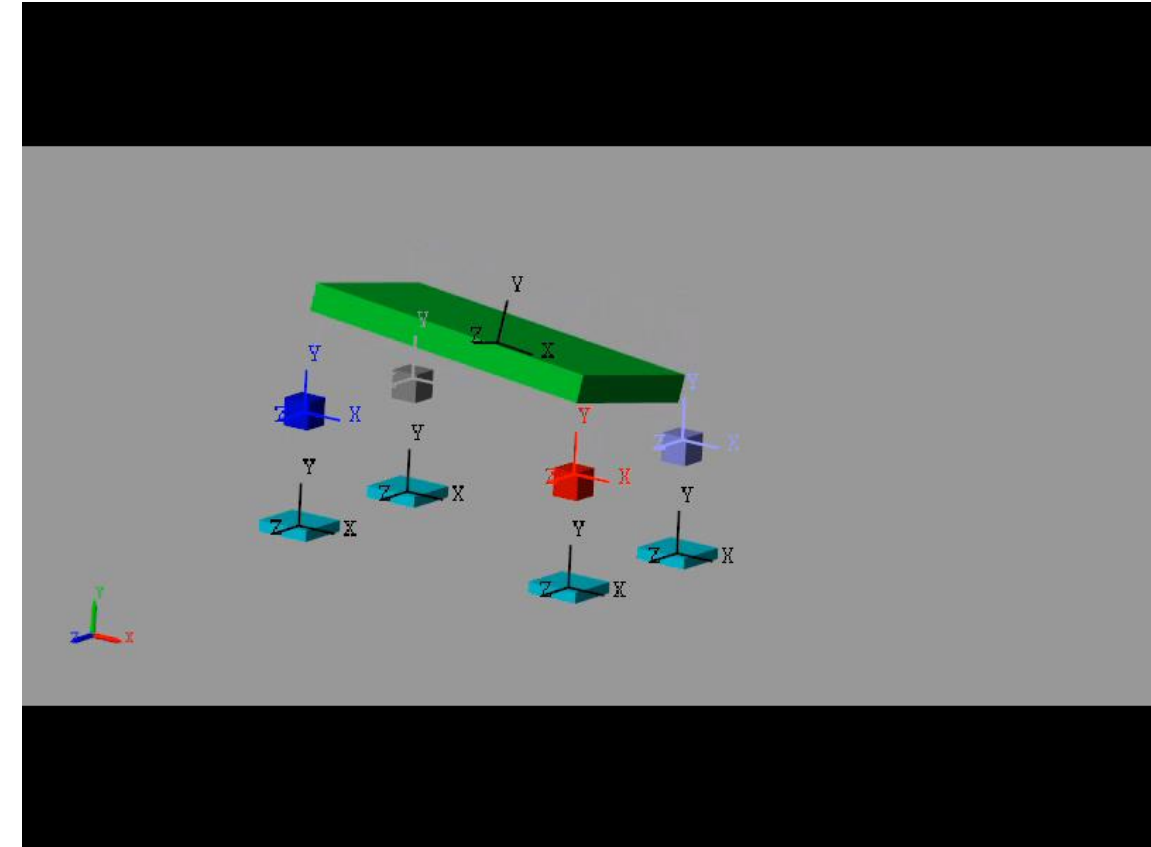


Modelling and Simulation in
MATLAB



Agenda

- A review of the required resources for this class
- A refresher on modelling with **Simulink**
 - Integrating a time varying signal
 - 2 ways to model a spring mass damper
- The **Road Profile model**
 - Define it !
 - Implement and simulate
- The **7-dof Car Dynamics model**
 - LECTURE mode
 - **Deriving** the dynamic equations of motion in MATLAB
 - Doing a **MODAL analysis**
 - Implementing the dynamic equations of motion in **Simulink**
 - **Simulating** the model in Simulink
 - An alternate approach to modelling – **Simscape**
- Next steps (for YOU)
 - Do the Spring-Mass-Damper Hands on course Module

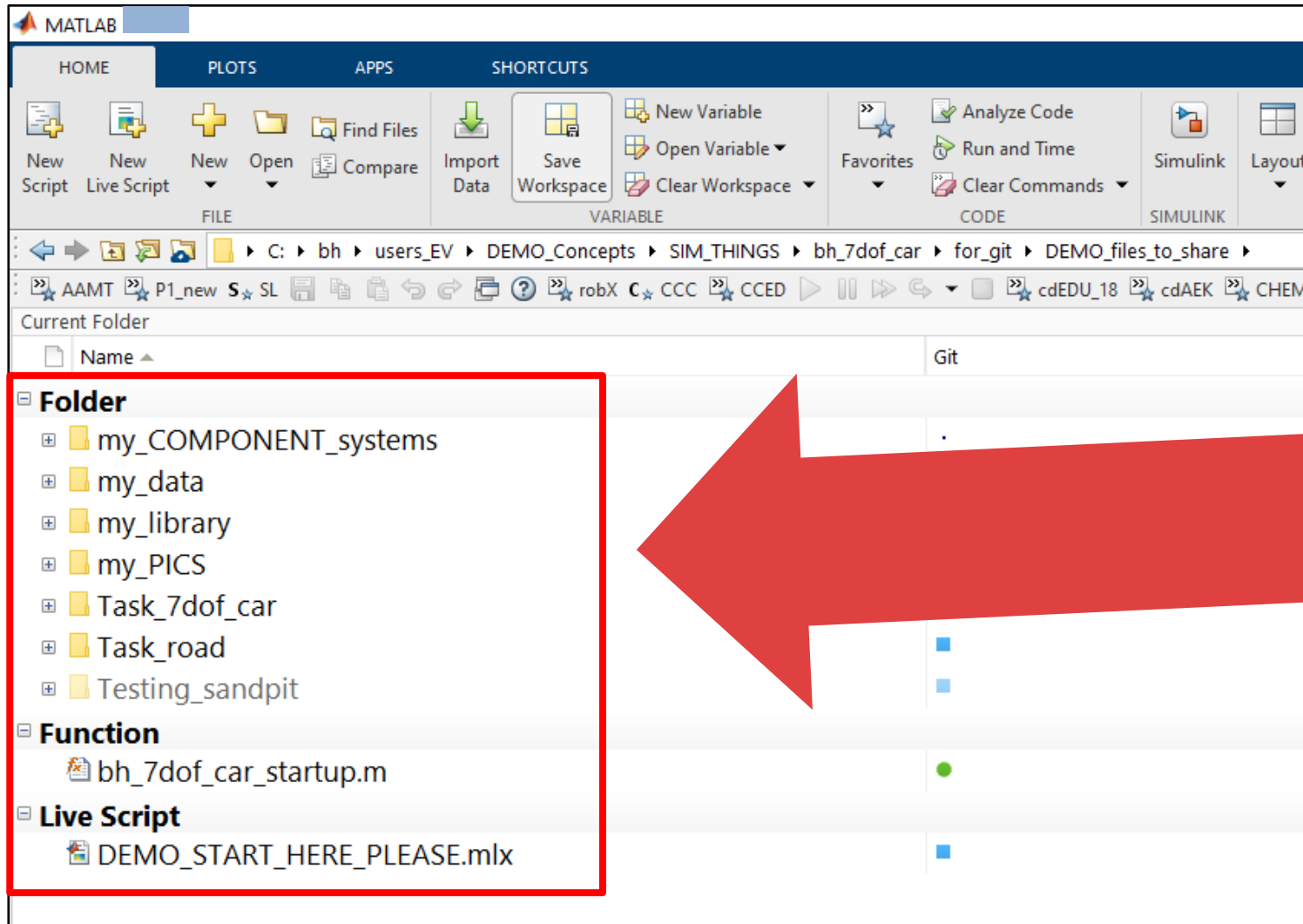


Required Resources for this class



Required Resources – part 1 of 3

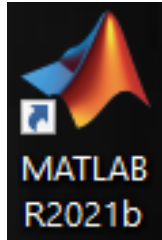
- Do you have the DEMO files ?



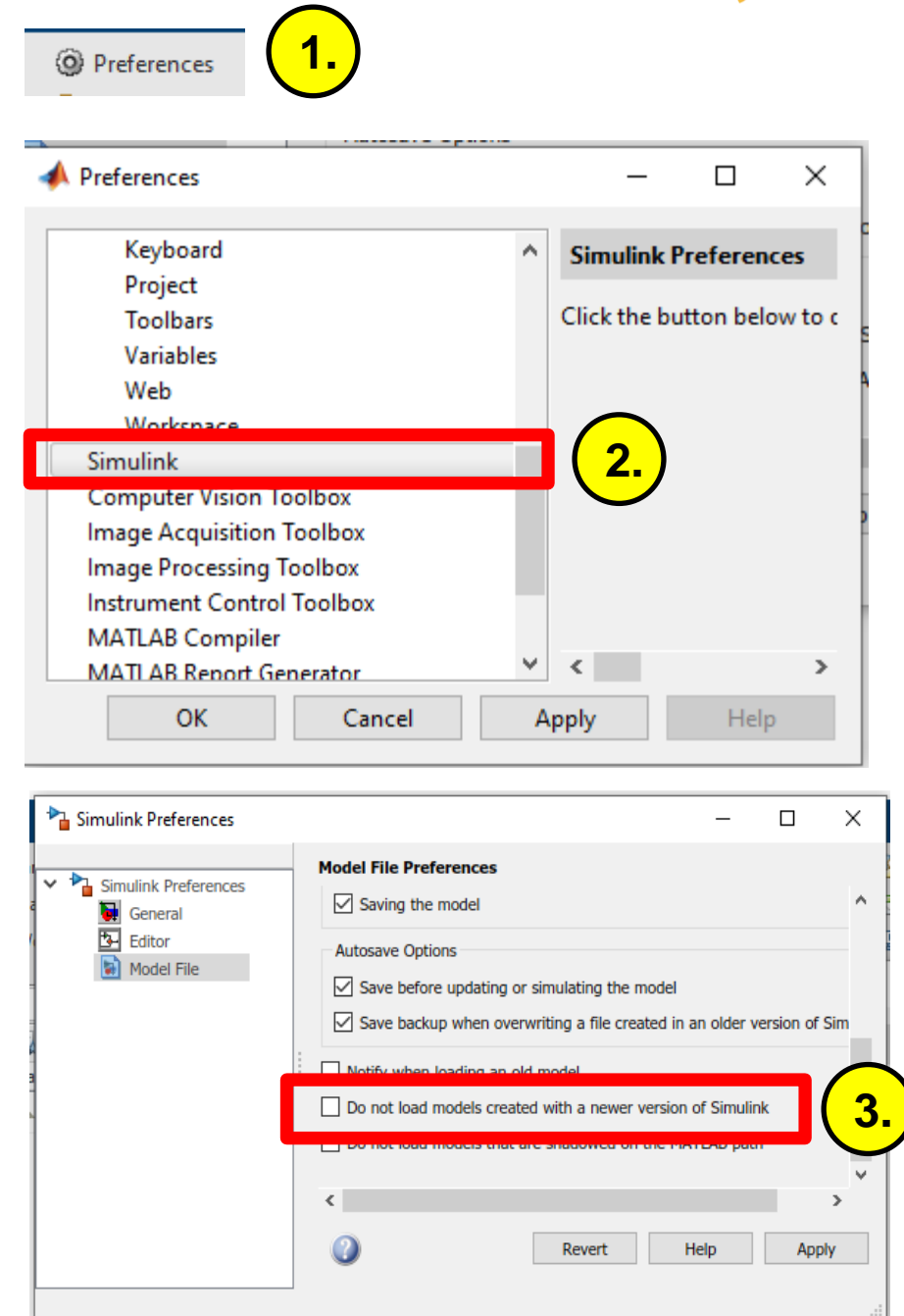
Have
you got
this ?

Required Resources – part 2 of 3

- Do you have the MATLAB License installed ?
 - You need the **R2021b** release installed



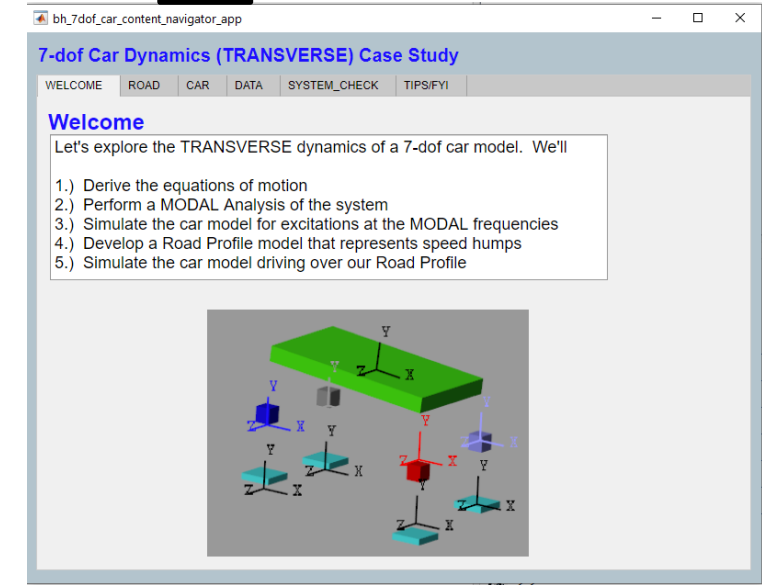
- As a minimum you will need:
 - MATLAB*
 - Symbolic Math Toolbox*
 - Simulink*
 - Simscape*
 - Simscape Multibody*
- But you are recommended to install the FULL campus license
- IFFF You are using an OLDER release, you must do the following



Required Resources – part 3 of 3

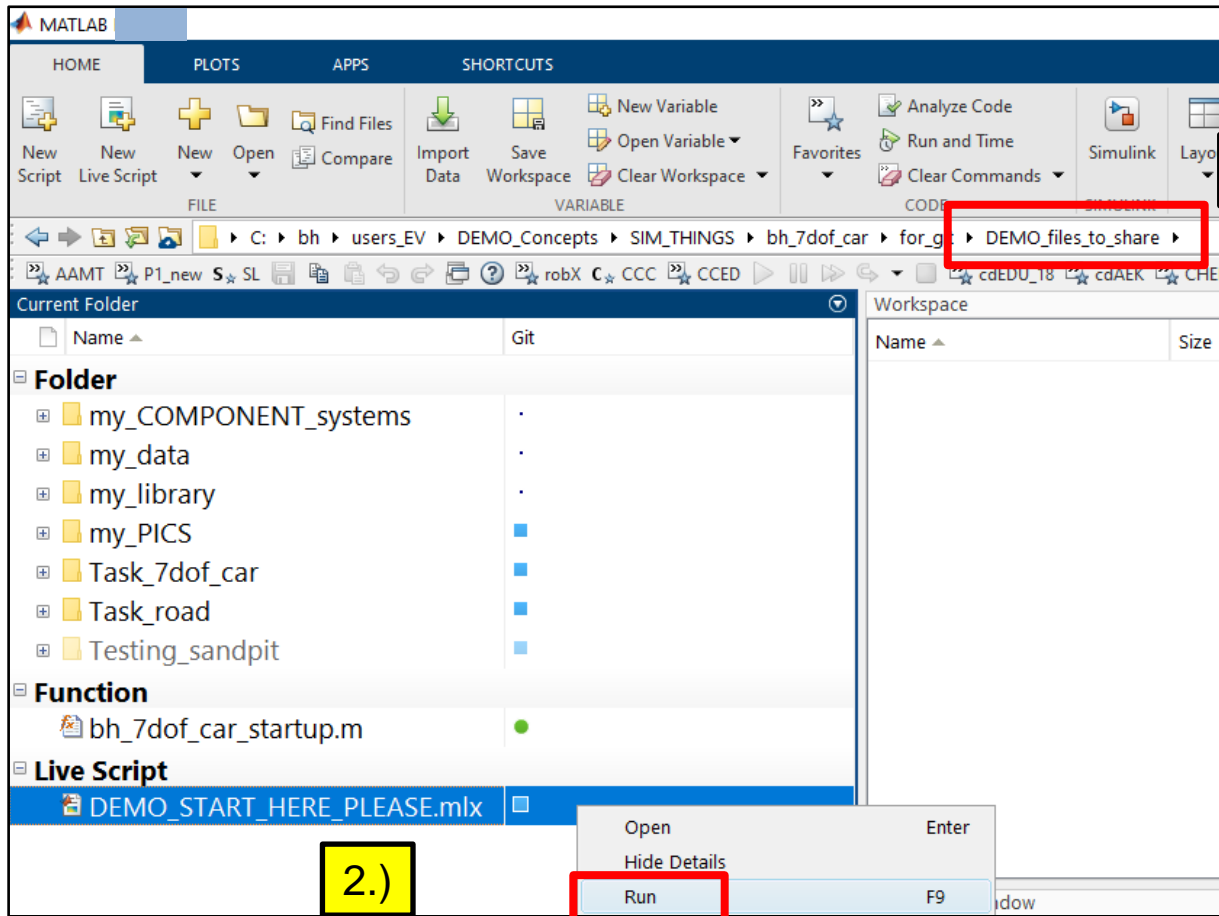
3.)

THIS APP should appear



1.)

You are in THIS folder

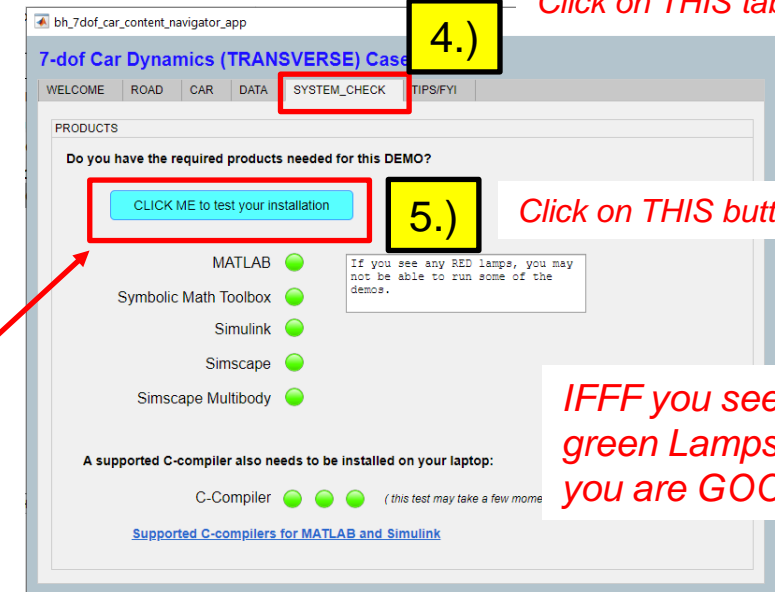


2.)

Right click THIS file and select Run

4.)

Click on THIS tab



5.)

Click on THIS button

NOTE: This test may take 1-2 minutes !

IFFF you see ALL green Lamps .. Then you are GOOD to go !



A refresher on modelling dynamic systems in Simulink

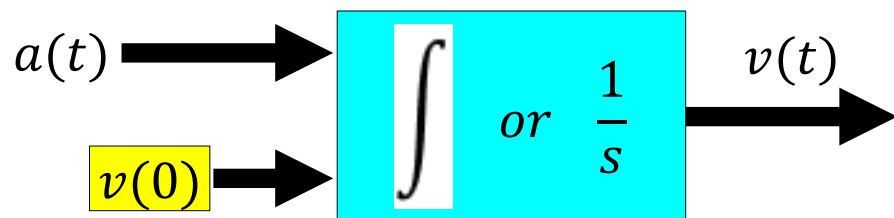
The concept of integrating a dynamic signal

Integrate **acceleration** to get **velocity**

$$\int_0^t a(\tau) \cdot d\tau = \int_0^t \frac{dv(\tau)}{d\tau} \cdot d\tau = v(t) - v(0)$$



$$v(t) = v(0) + \int_0^t a(\tau) \cdot d\tau$$

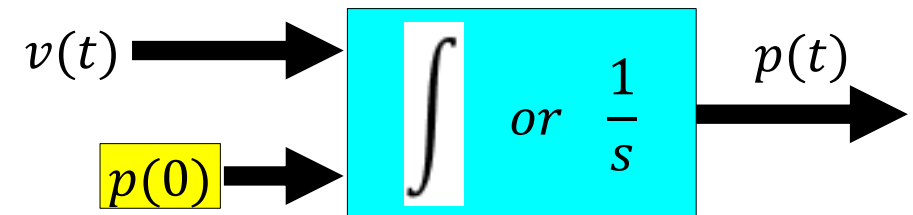


Integrate **velocity** to get **position**

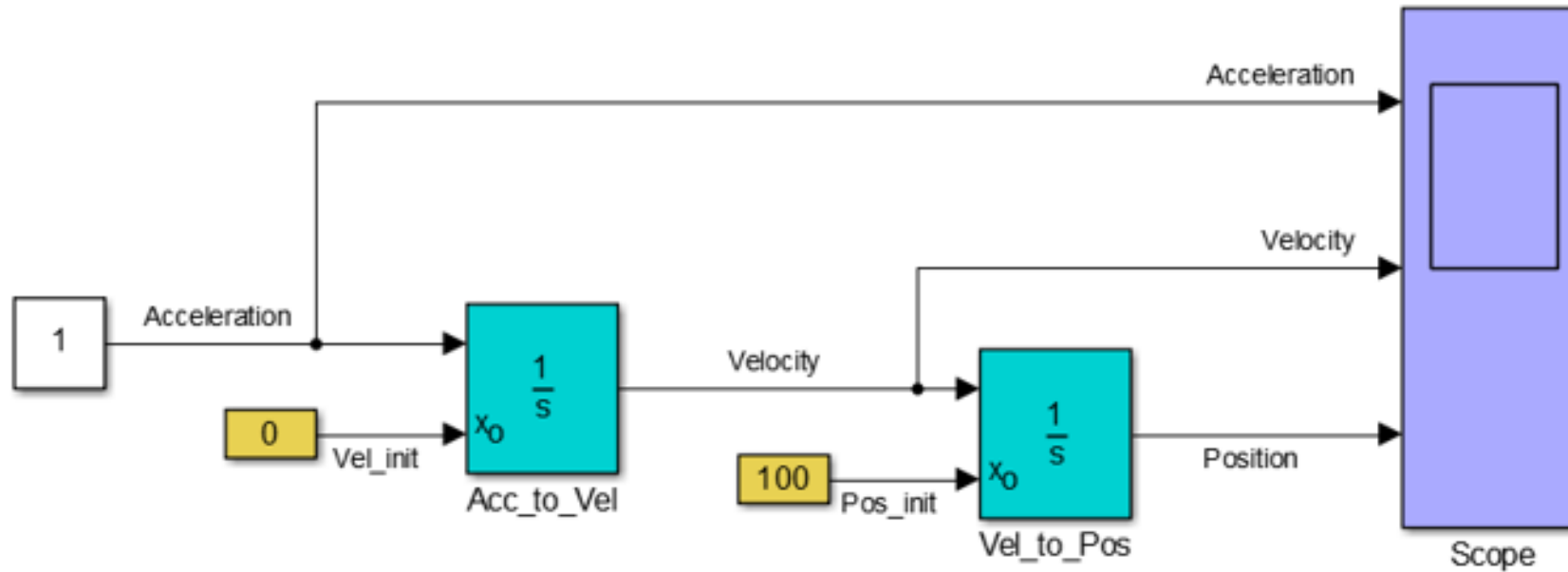
$$\int_0^t v(\tau) \cdot d\tau = \int_0^t \frac{dp(\tau)}{d\tau} \cdot d\tau = p(t) - p(0)$$



$$p(t) = p(0) + \int_0^t v(\tau) \cdot d\tau$$

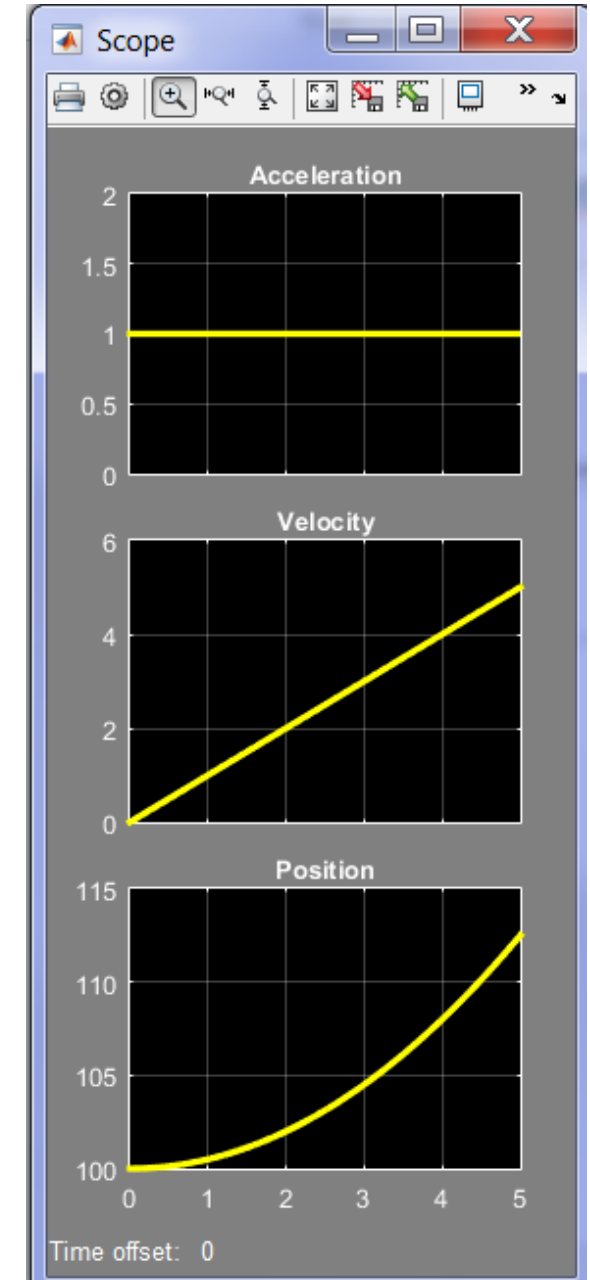


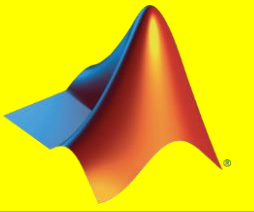
The concept of integrating a dynamic signal



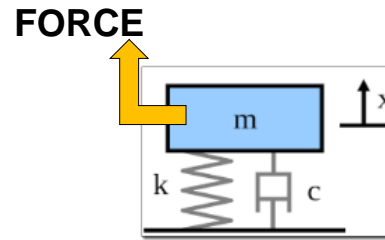
$$v(t) = v(0) + \int_0^t a(\tau) \cdot d\tau$$

$$p(t) = p(0) + \int_0^t v(\tau) \cdot d\tau$$

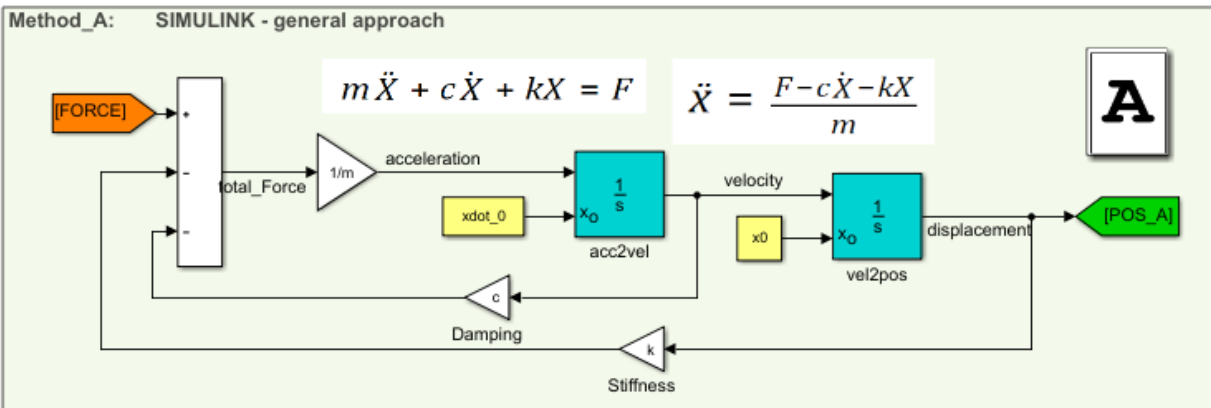




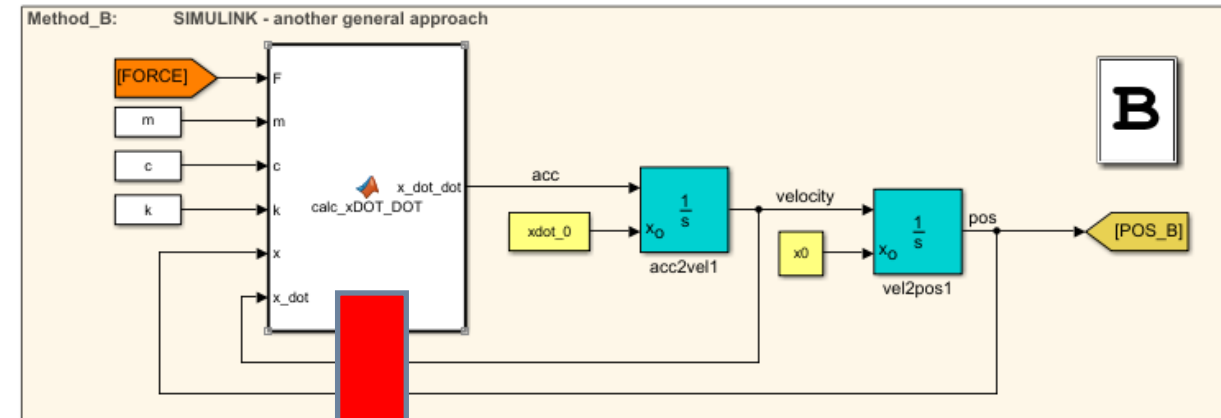
A Hello World model in Simulink



Sometimes you use blocks for everything



Sometimes you write a MATLAB function for some of it



$$\ddot{X} = \frac{F - c.\dot{X} - k.X}{m}$$

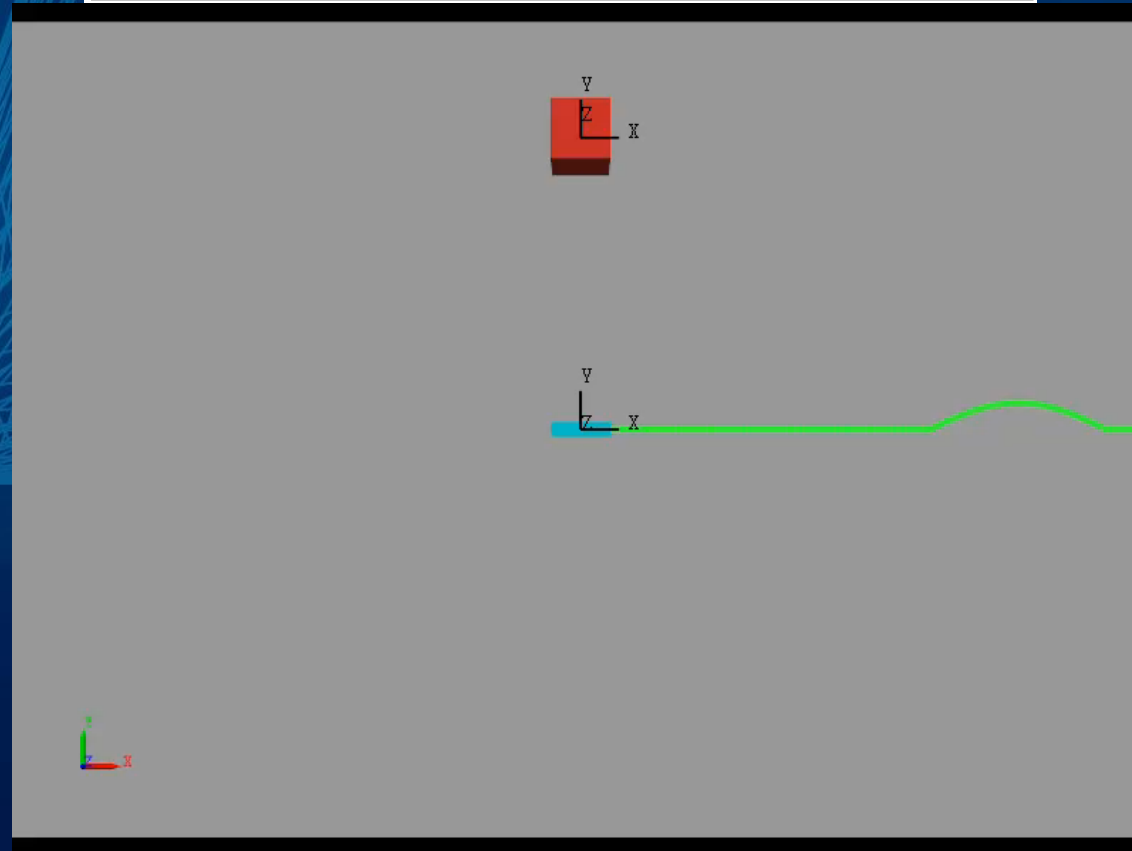
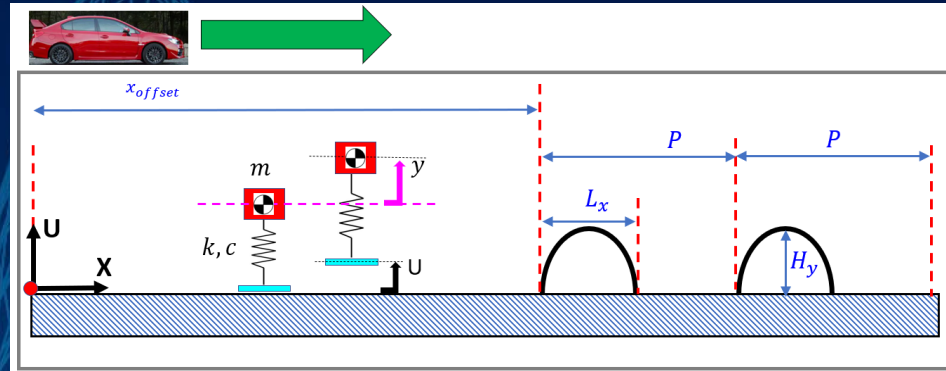


```
function x_dot_dot = calc_xDOT_DOT(F,m,c,k,x,x_dot)

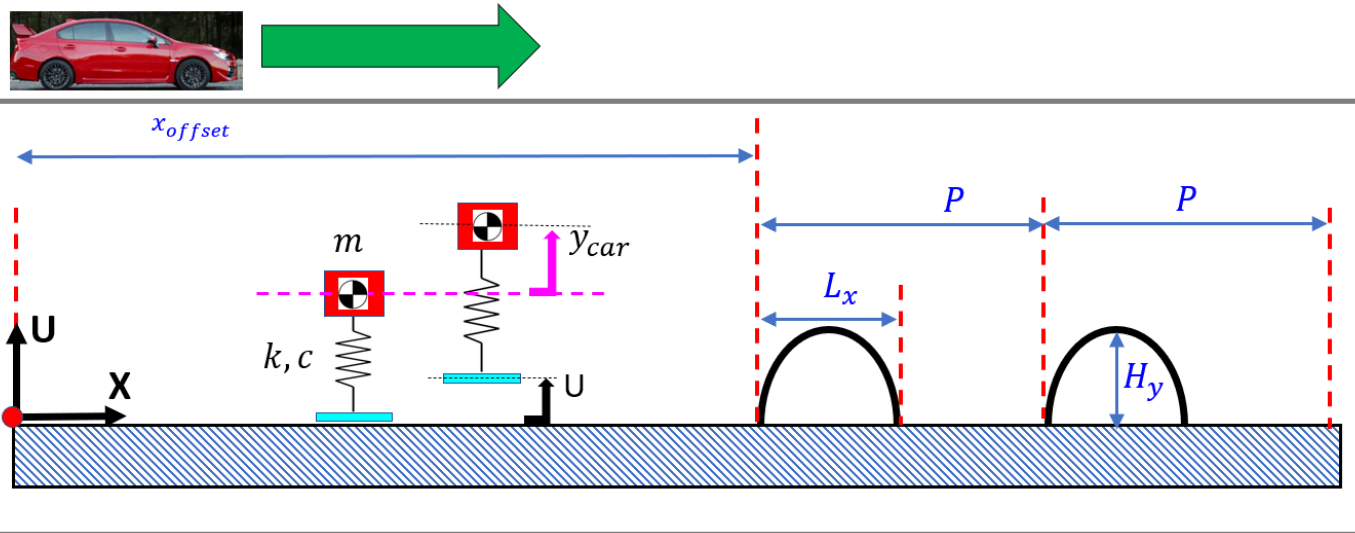
    x_dot_dot = (F - c*x_dot - k*x)/m;

end
```

A Road Profile model



A speed hump Road Profile model – an introduction, Part 1 of 5



The "Hump" portion is just a simple Sine wave of the form:

$$U(x) = H_y \cdot \sin\left(\frac{2 \cdot \pi \cdot x}{\lambda}\right) = H_y \cdot \sin\left(\frac{2 \cdot \pi \cdot x}{2 \cdot L_x}\right) = H_y \cdot \sin\left(\frac{\pi \cdot x}{L_x}\right)$$

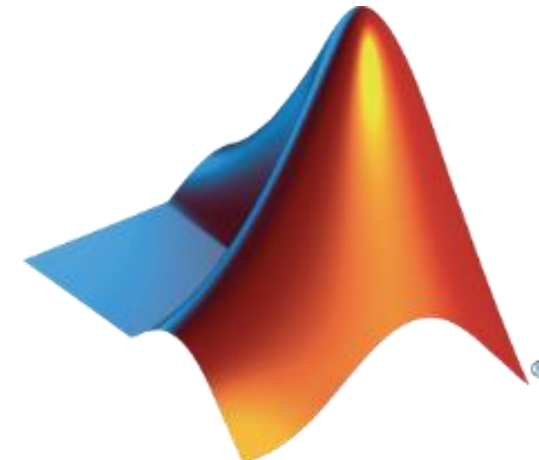
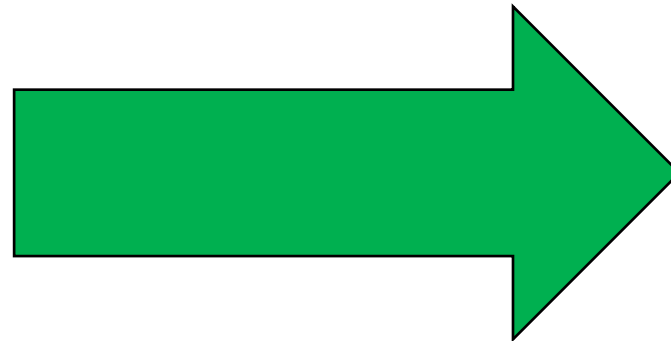
Function INPUTS:

1. **x** : a list of **X-values** at which to compute the height of the road
2. **x_offset** : the horizontal distance at which the FIRST hump begins
3. **Lx** : the **width** of the hump
4. **Hy** : the **Height** of the Hump
5. **P** : the spatial Period of the repeating block

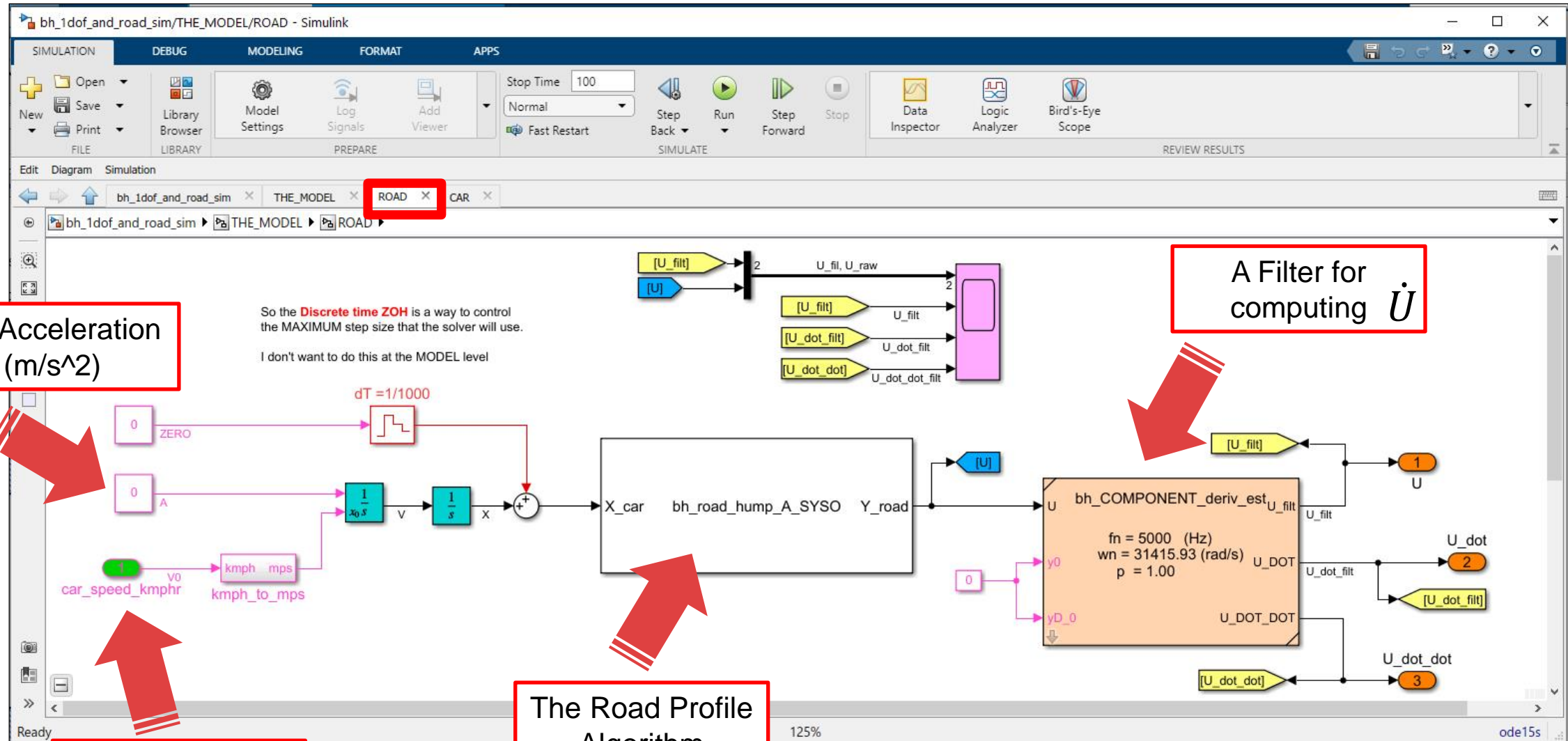
Function OUTPUTS:

1. **U** : the HEIGHT of the road at the specified **X-values**

Implemented in MATLAB
Tested in Simulink

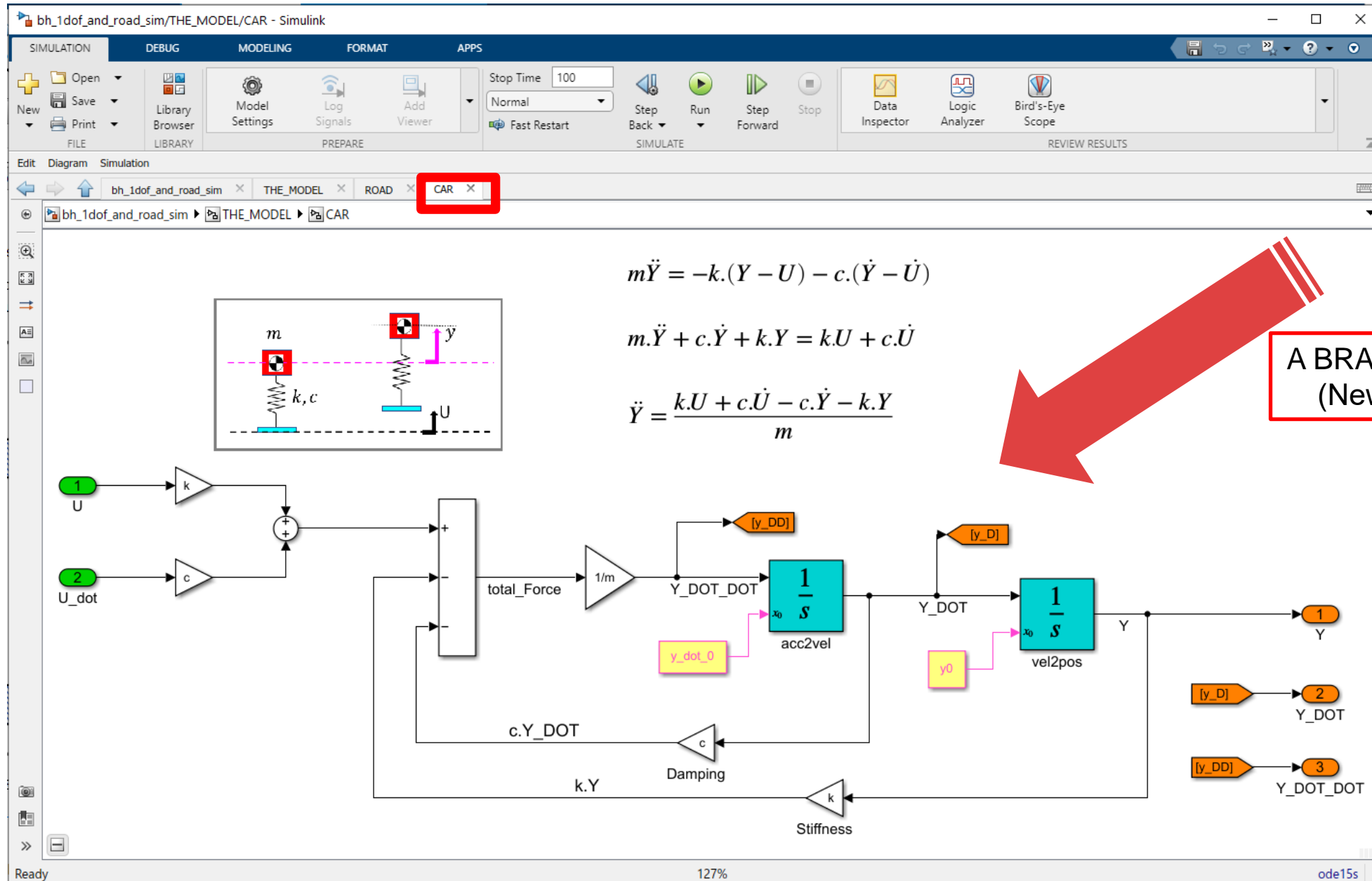


A speed hump Road Profile model – an introduction, Part 2 of 5



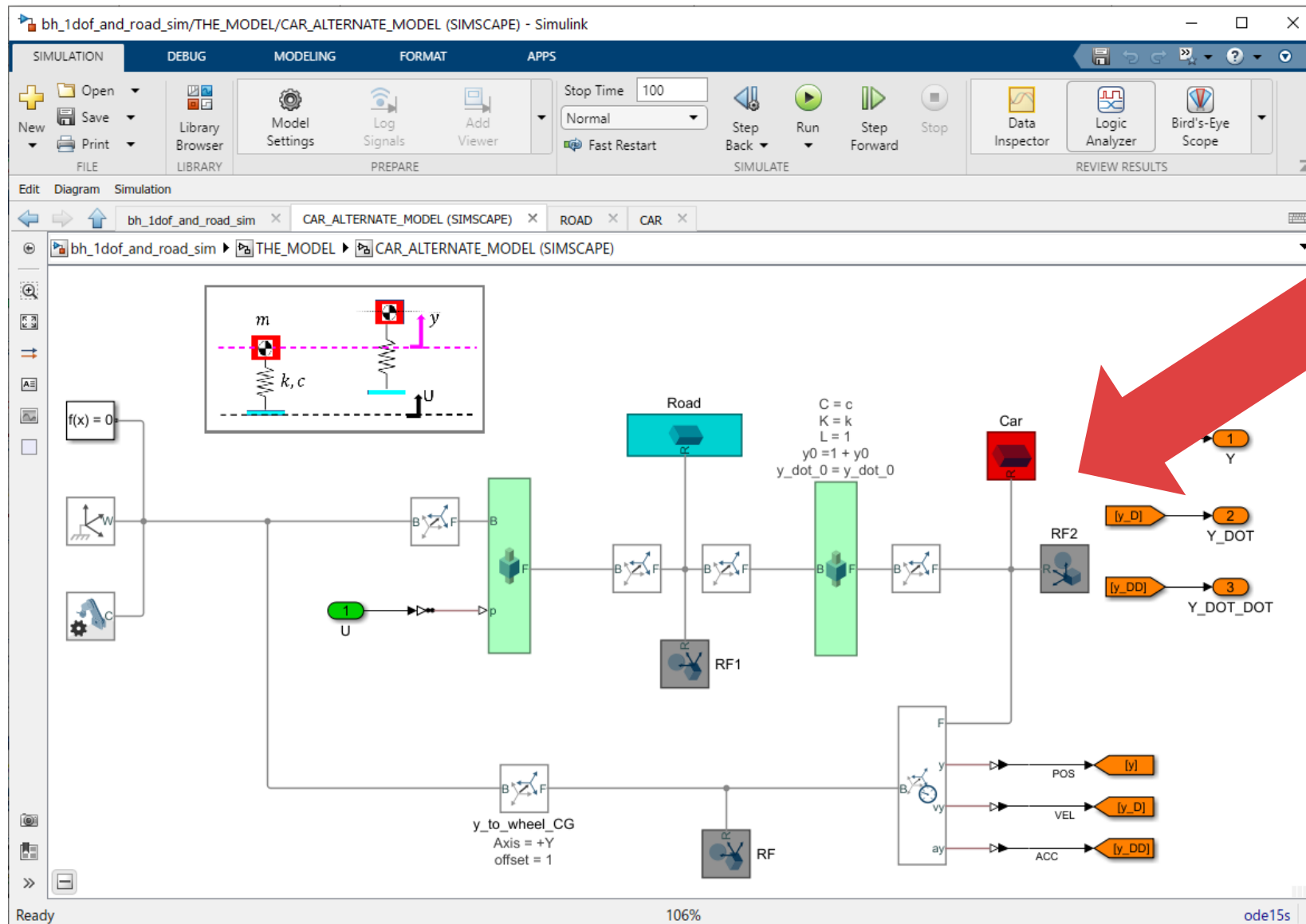
A speed hump Road Profile model – an introduction, Part 3 of 5

Our 1-dof Car
dynamics
model



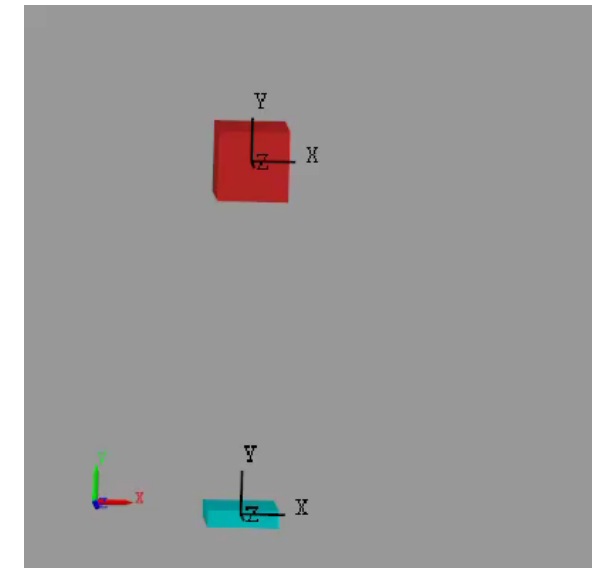
A BRAIN derived model
(Newton's 2nd Law)

A speed hump Road Profile model – an introduction, Part 4a of 5



An **Alternate** way of modelling the car dynamics
(**Simscape MultiBody**)

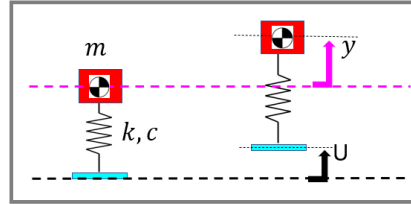
The 3D CAD-like viewer appears automatically !



A speed hump Road Profile model – an introduction, Part 4b of 5

So to be clear: There are 2 different approaches to modelling the Car Dynamics

Approach #1



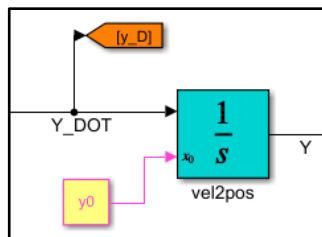
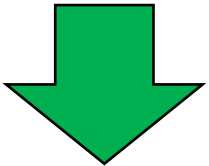
Approach #2



*I will show you
BOTH
techniques*

Brain – First Principles

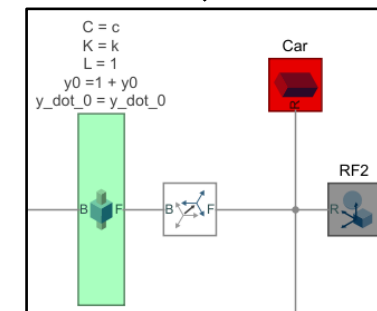
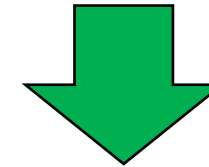
- Apply Newton's 2nd law
- Derive the system ODEs
- Implement ODEs in **Simulink**



*Use
integrator
blocks*

Physical component Modelling

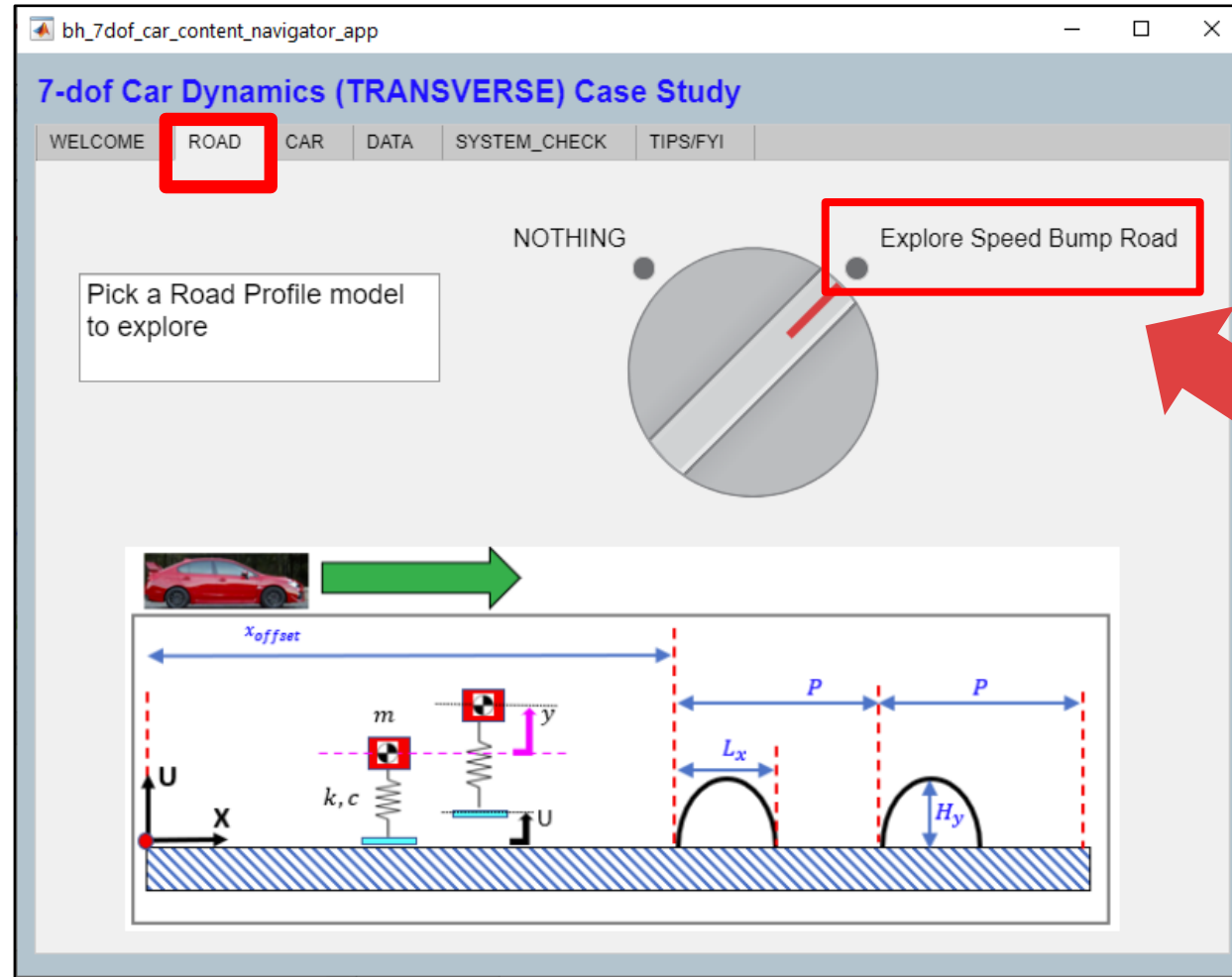
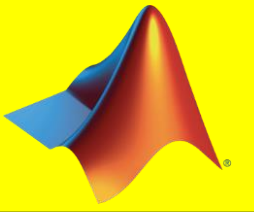
- Use blocks of physical components
 - Mass, joint(with springs) blocks
- **Simscape Multibody**
(inside Simulink)



*... and a 3D
CAD viewer
appears
automatically !*

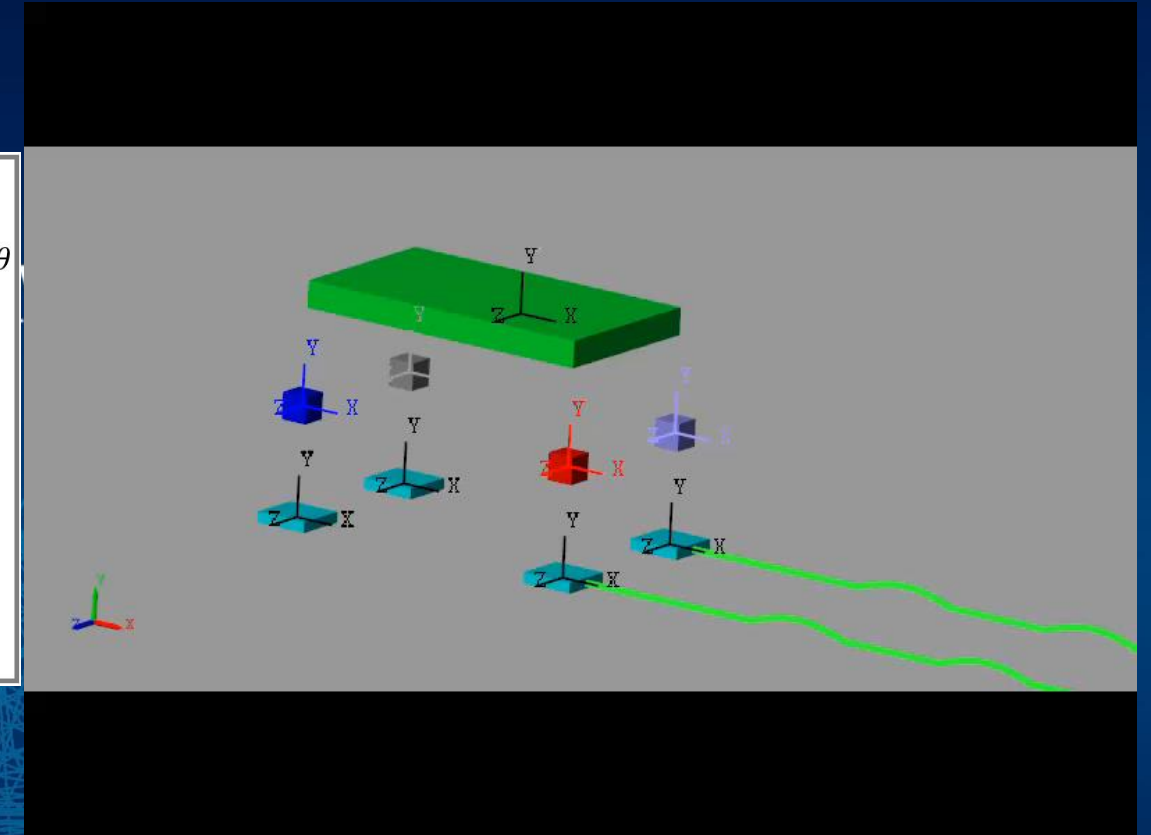
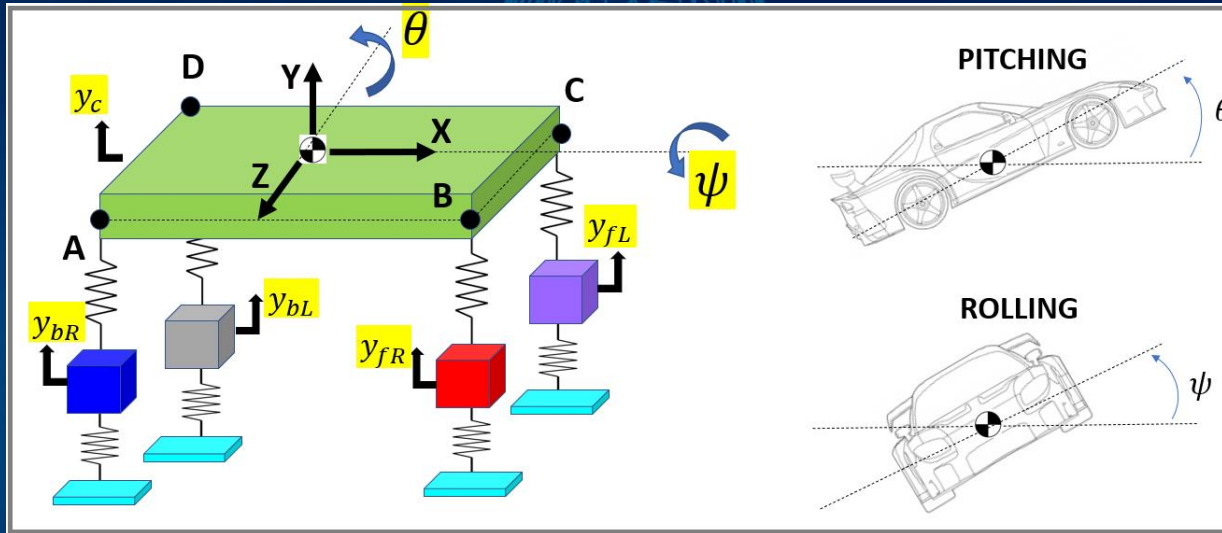
A speed hump Road Profile model – an introduction, Part 5 of 5

DEMO



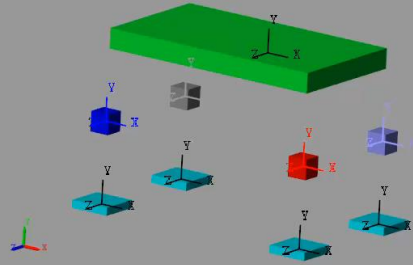
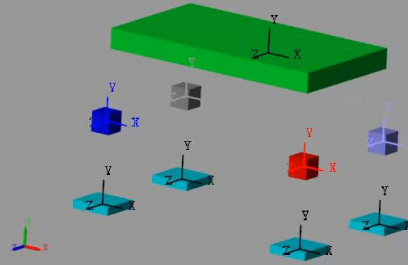
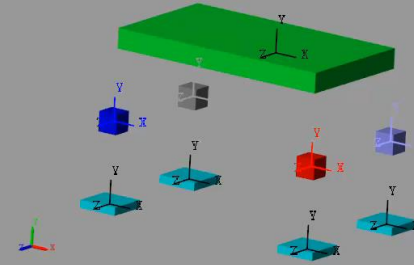
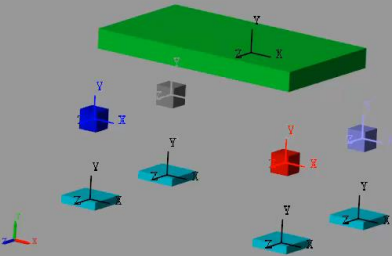
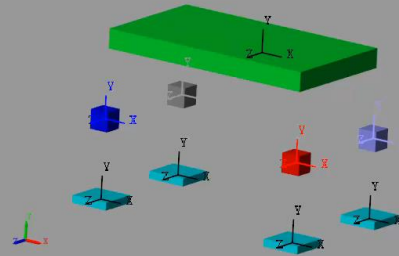
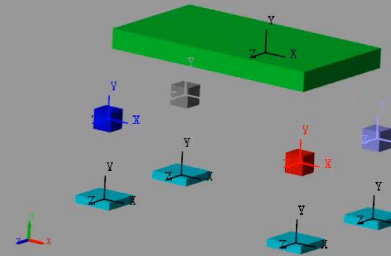
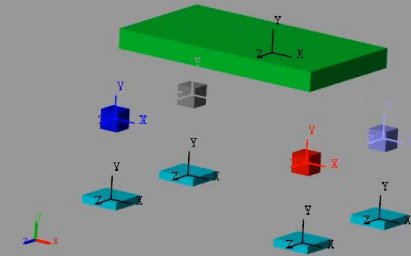
Try it !

Today's Case Study



The 7-dof Car Dynamics (Transverse)

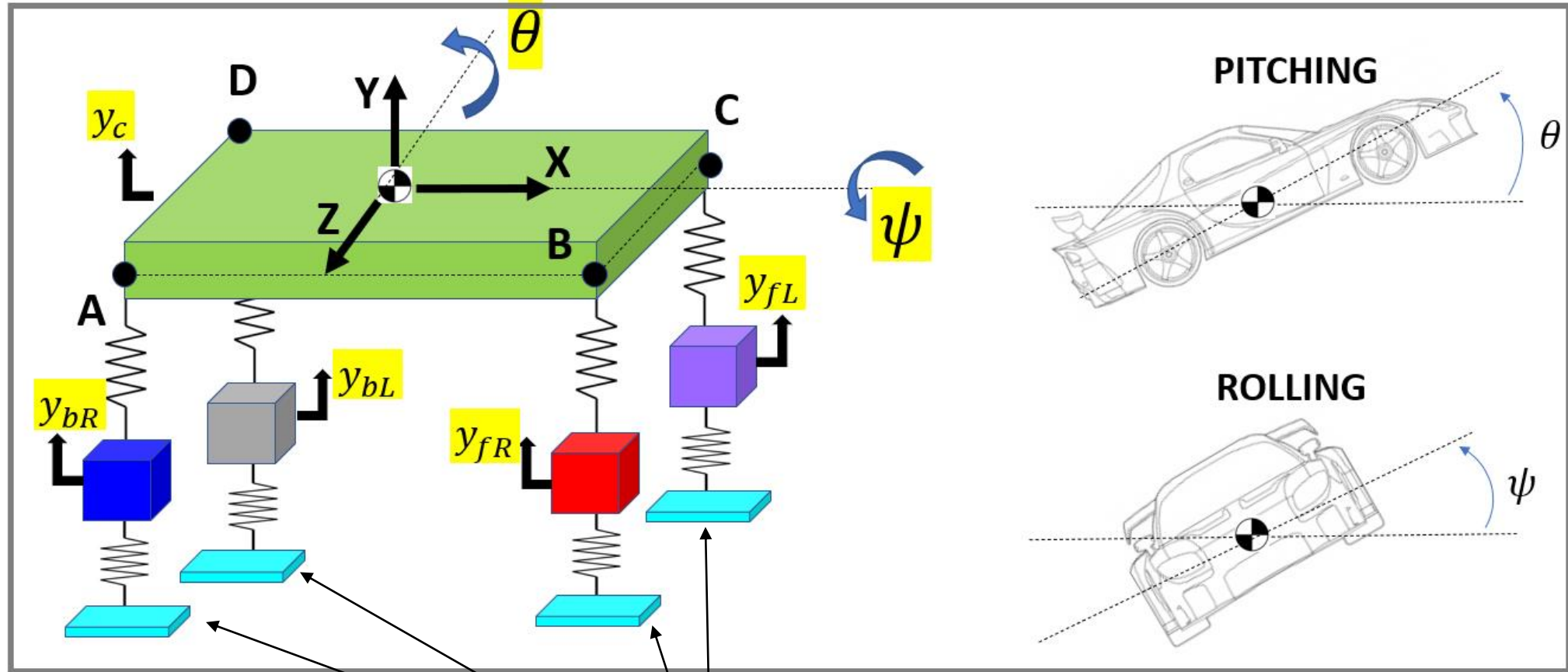
The 7 Transverse Undamped MODES

Mode #1**Mode #2****Mode #3****Mode #4****Mode #5****Mode #6****Mode #7**

The 7-dof Car – an introduction, Part 1 of 3

The 7 degrees of freedom

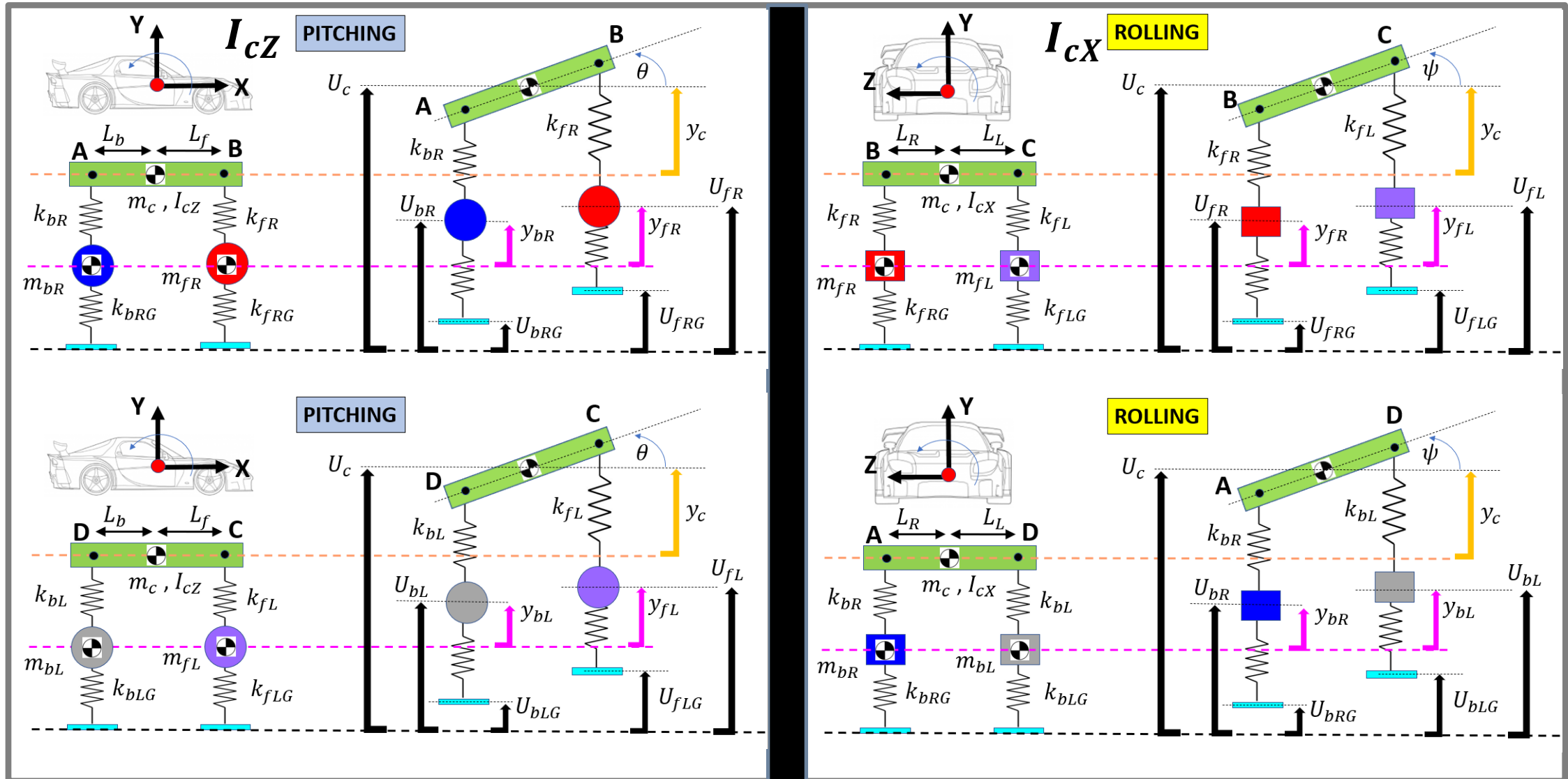
$$\begin{bmatrix} y_c \\ \theta \\ \psi \\ y_{bR} \\ y_{fR} \\ y_{bL} \\ y_{fL} \end{bmatrix}_{7 \times 1}$$



Road surface displacements

The 7-dof Car – an introduction, Part 2 of 3

Springs, Dampers, Lengths, Masses, Inertias and Road displacements

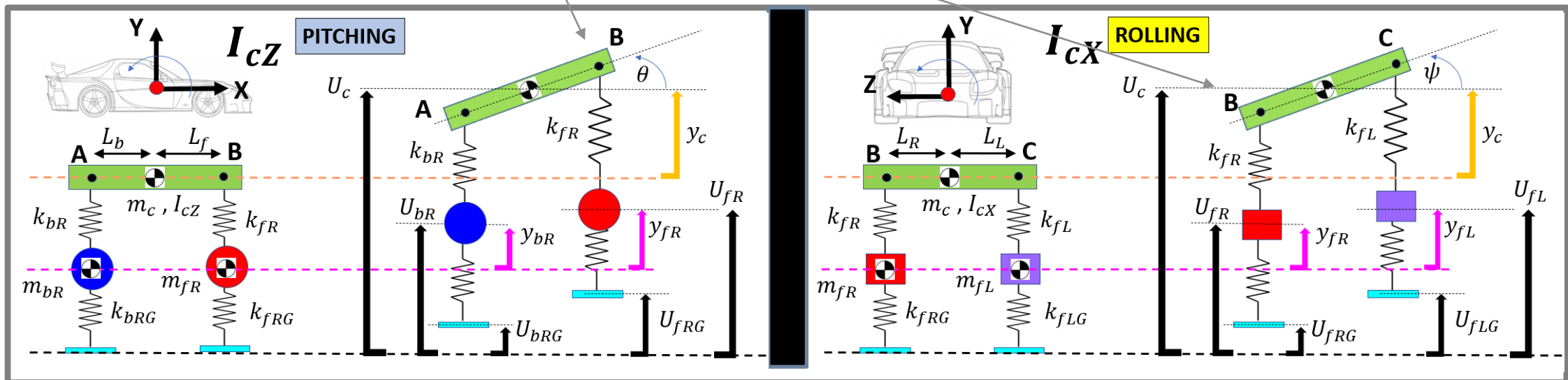
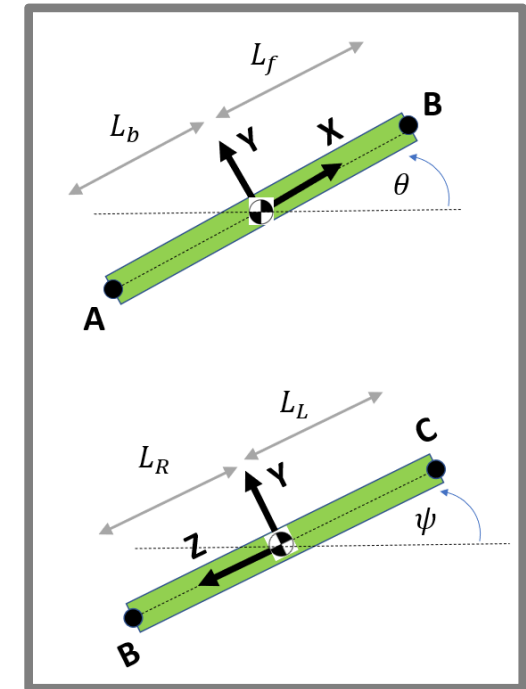


The 7-dof Car – an introduction, Part 3 of 3

Small angle assumption $\sin(\theta) \approx \theta$
 $\sin(\psi) \approx \psi$

E.g.: total displacement of point B, is:

$$\Delta y_B = y_{car} + L_f \cdot \theta - L_R \cdot \psi$$



What we'll do: in MATLAB/Simulink



Try it !



- Derive the equations of motion (Newton's 2nd Law)

- Your Engineering BRAIN + MATLAB**

$$M.\ddot{X} + C.\dot{X} + K.X = F$$

- Perform a MODAL Analysis

$$I = V^T.M.V$$

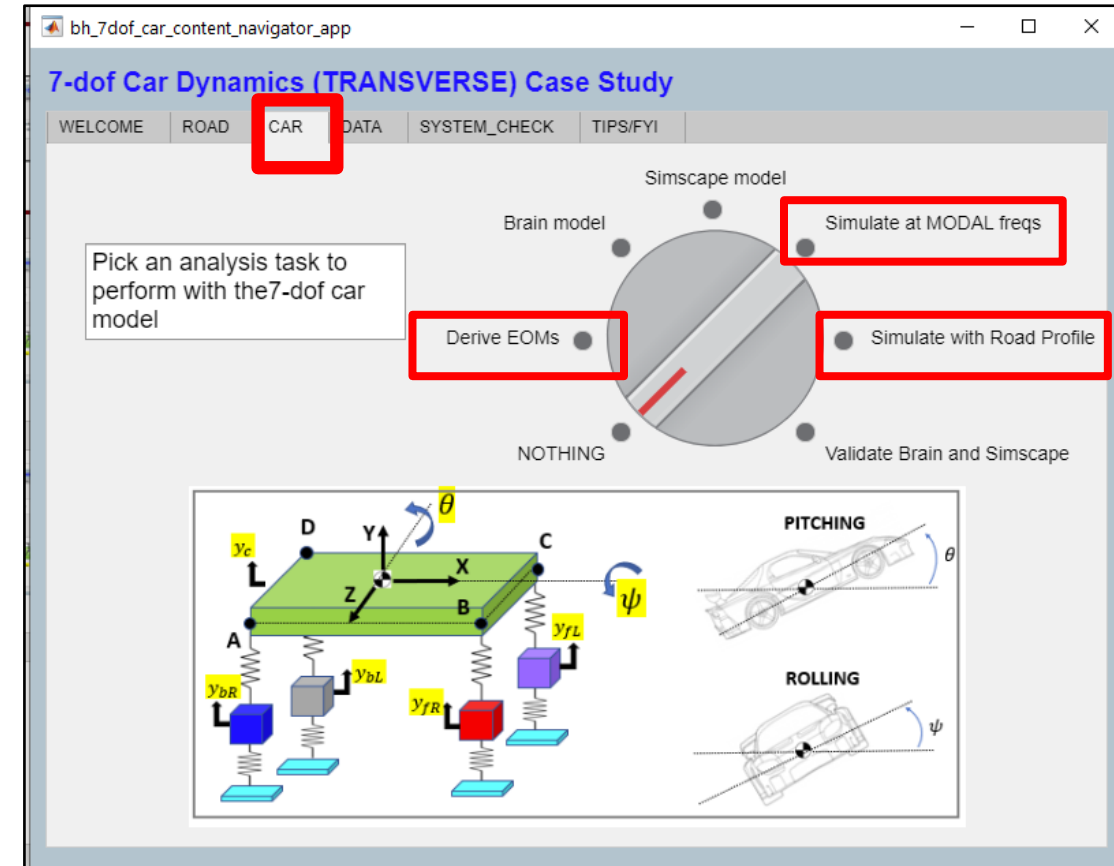
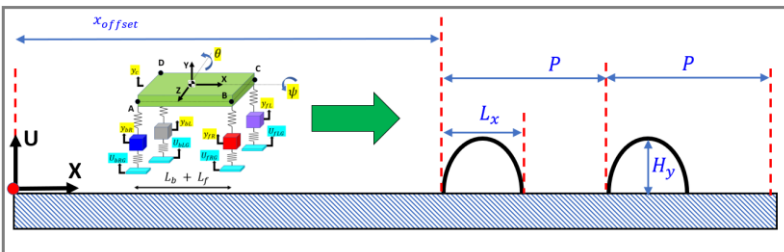
$$\Omega^2 = V^T.K.V$$

- Simulate in **Simulink**

- VERY lightly damped system, excited at MODAL frequencies
 - Validate BRAIN derived model against **Simscape** model

- Simulate in Simulink

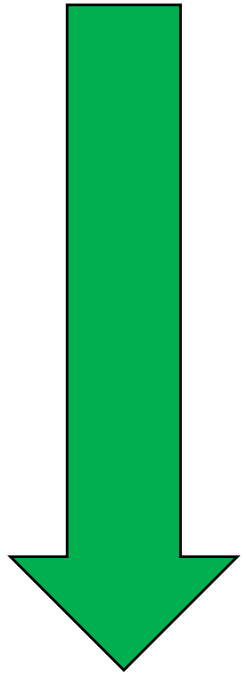
- A more "authentic" car configuration
 - Excite the car using our "speed hump" Road Profile






Next Steps

Next Steps:



Download our Hands
on **(interactive)**
Courseware module

<https://www.mathworks.com/academia/courseware/mass-spring-damper-systems.html>

 Products Solutions Academia Support Community Events

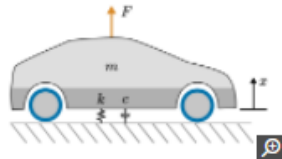
Get MATLAB

Educators

Search MathWorks.com

Teach with MATLAB and Simulink Curriculum Resources Online Teaching Campus-Wide License

"Mass-Spring-Damper Systems" Courseware



Course Materials Include:

- Courseware description
- 3 MATLAB live scripts
- Simulink models
- 3 MATLAB live script solutions

Download the free mechanical engineering courseware module using MATLAB and Simulink.

[Download from GitHub \(.zip\)](#)

[View license on GitHub](#)

[How to Use Live Scripts in MATLAB \(3:41\)](#)

Mass-Spring-Damper Systems

This curriculum module contains interactive MATLAB [live scripts](#) and [Simulink](#) models that explore mass-spring-damper systems. Throughout the module, students apply Simulink models to study the dynamics of the physical systems. Students learn to create and work with mass-spring-damper models in guided activities. These techniques are motivated by two applications: tuning the damping of a vehicle's suspension and analyzing a building's response to an earthquake. These lessons can be used as part of a lecture, as activities in an instructional setting, or as interactive assignments to be completed outside of class.

The live scripts include:

- massSpringDamper.mlx: interactive lesson that teaches how to model a single mass-spring-damper in Simulink; students apply their knowledge to tune the damping of a vehicle's suspension system
- doubleMassSpringDamper.mlx: interactive lesson that teaches how to model a double mass-spring-damper in Simulink; students apply their knowledge to identify the resonant frequencies present in a two-story building model
- multipleMassSpringDamper.mlx: lesson that enables the practice of more advanced skills; students identify the resonant frequencies of a mass-spring-damper building model by computing the power spectrum of a displacement signal

Learning goals:

images

models

doubleMassSpringDamper.mlx

doubleMassSpringDamperSoln.mlx

massSpringDamper.mlx

massSpringDamperSoln.mlx

multipleMassSpringDamper.mlx

multipleMassSpringDamperSoln.mlx

2.)

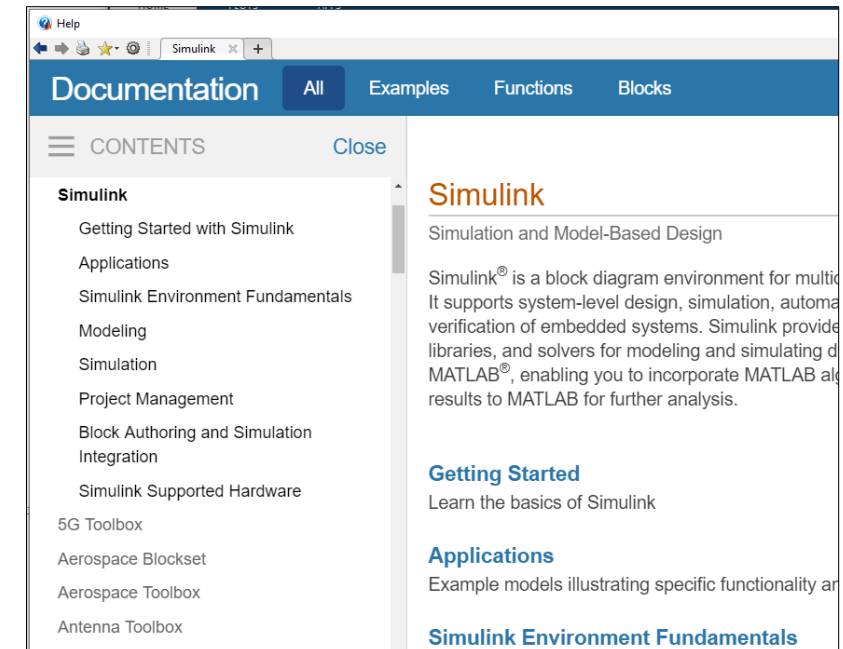
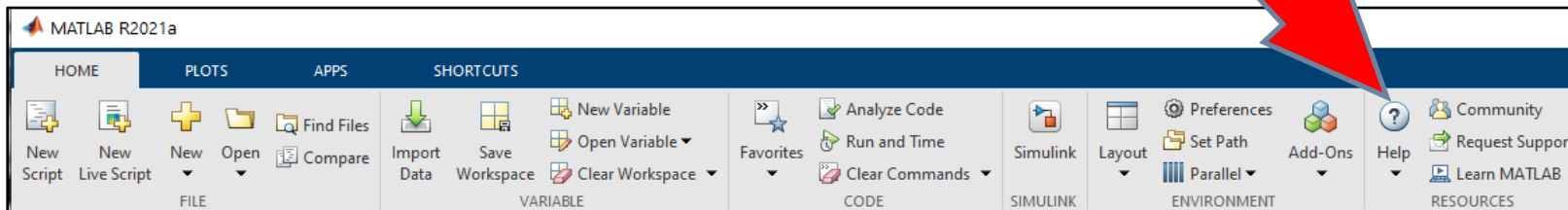
1.)

3.)

Thank you.

$$\left[\begin{array}{c} \text{Your BRAIN} \\ \text{Your ENGINEERING} \end{array} \right] + \text{MATLAB} = \text{Very COOL Things happen}$$

- **PS:**
 - Please use the **MATLAB Help Browser** **OFTEN** !





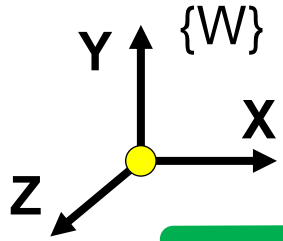
Appendix A:



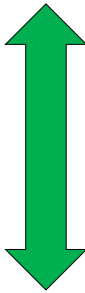
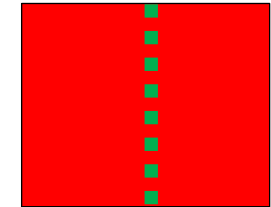
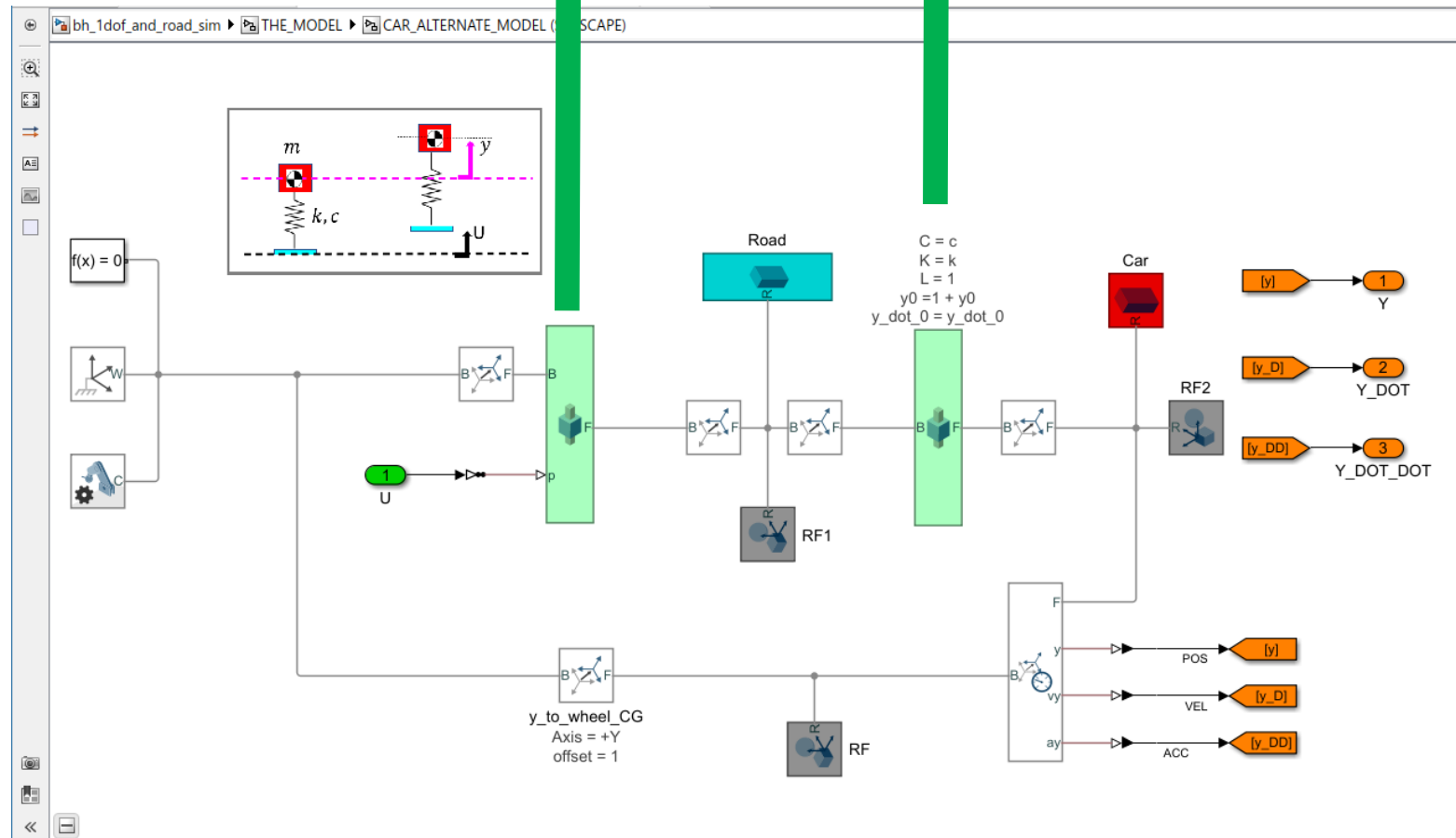
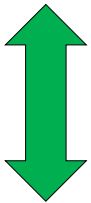
Appendix B:

A teardown of the Simscape 1-dof model

Setting up motion constraints – part 1

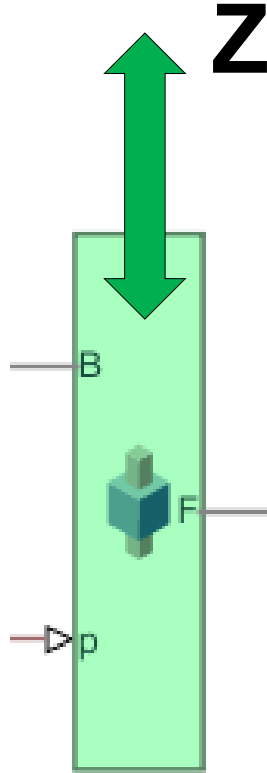
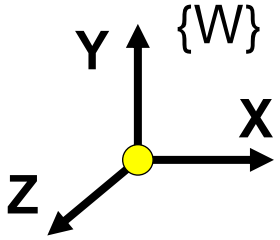


BLUE road body can translate relative to **GROUND**.



Red car body can translate relative to **BLUE** road body

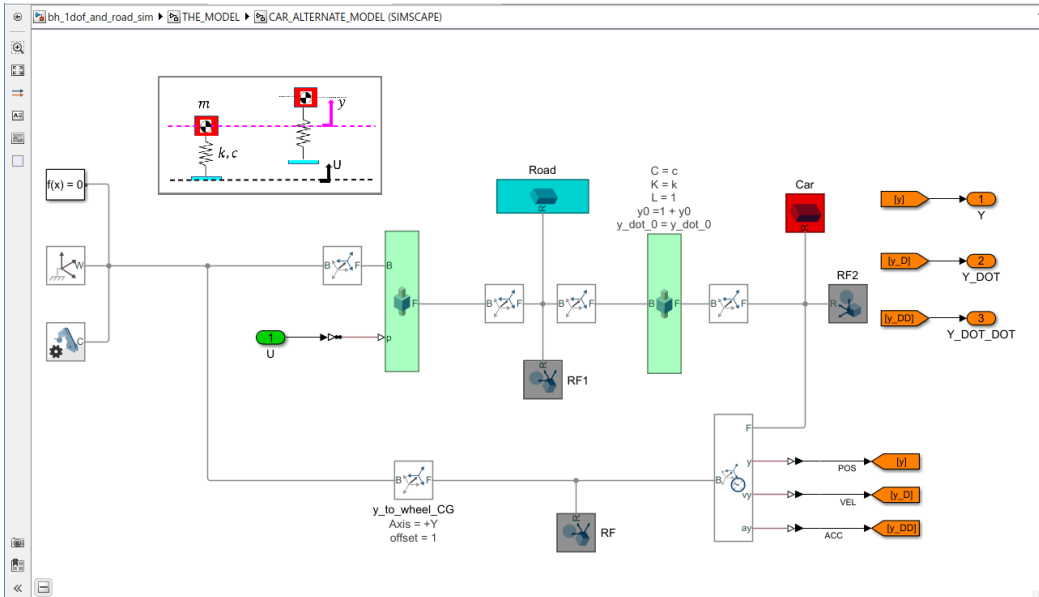
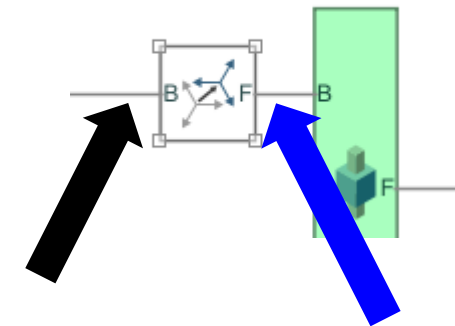
Setting up motion constraints – part 2



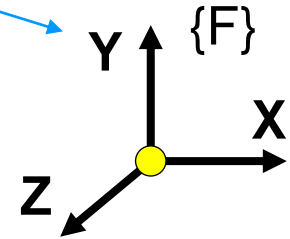
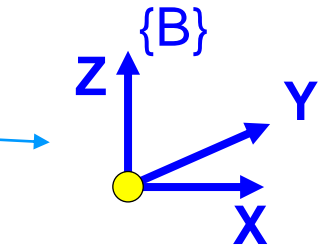
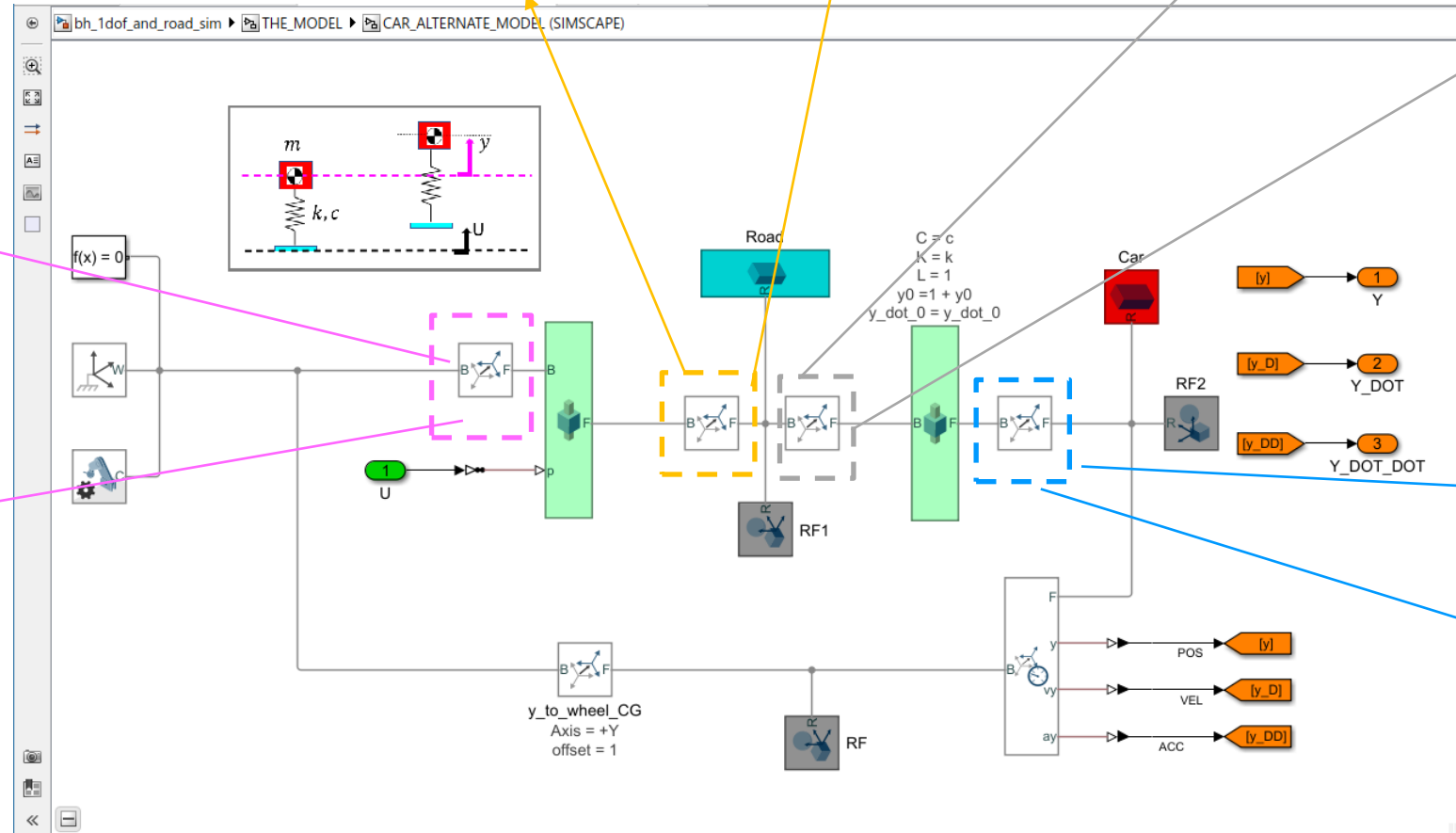
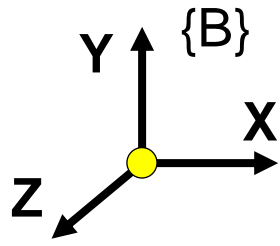
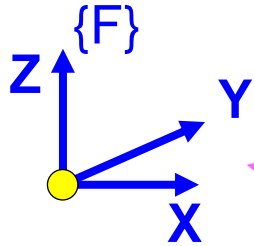
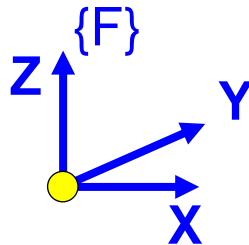
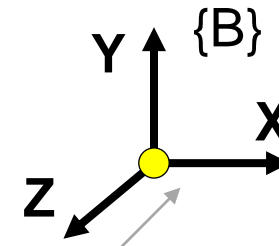
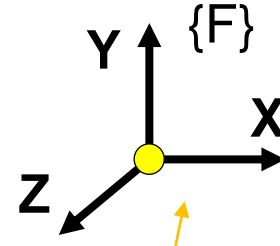
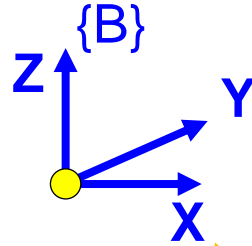
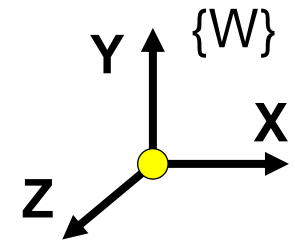
... BUT: the **Prismatic joint** offers Relative motion along “a” Z-axis.

We want the TRANSLATIONAL Motion to be along the Y-axis !!

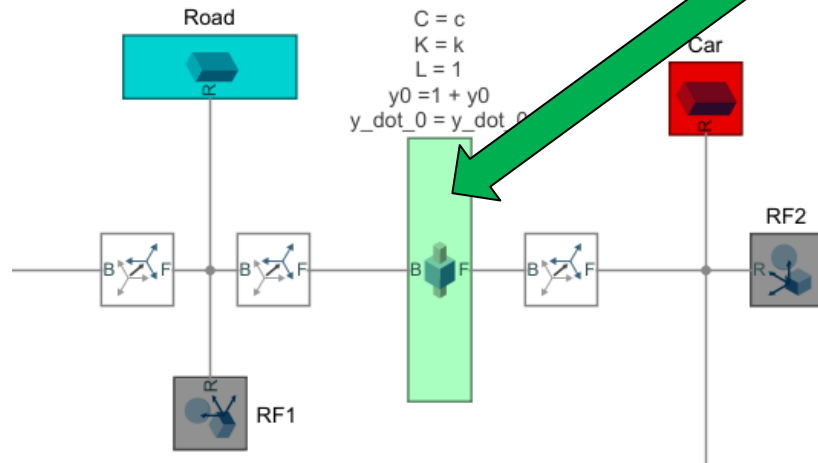
So ? – So we use the **Rigid Transform** Block to create NEW frames that are Orientated according to our needs



Setting up motion constraints – part 3



Setting up the spring



Prismatic Joint : Prismatic Joint

Description

Represents a prismatic joint between two frames. This joint has one translational degree of freedom represented by one prismatic primitive. The joint constrains the follower origin to translate along the base z-axis, while the base and follower axes remain aligned.

In the expandable nodes under Properties, specify the state, actuation method, sensing capabilities, and internal mechanics of the primitives of this joint. After you apply these settings, the block displays the corresponding physical signal ports.

Ports B and F are frame ports that represent the base and follower frames, respectively. The joint direction is defined by motion of the follower frame relative to the base frame.

Properties

Z Prismatic Primitive (Pz)

State Targets

Specify Position Target		
Priority	High (desired)	
Value	$1 + y_0$	m

Specify Velocity Target

Specify Velocity Target		
Priority	High (desired)	
Value	\dot{y}_{dot_0}	m/s

Internal Mechanics

Internal Mechanics		
Equilibrium Position	1	m
Spring Stiffness	k	N/m
Damping Coefficient	c	N/(m/s)

Limits

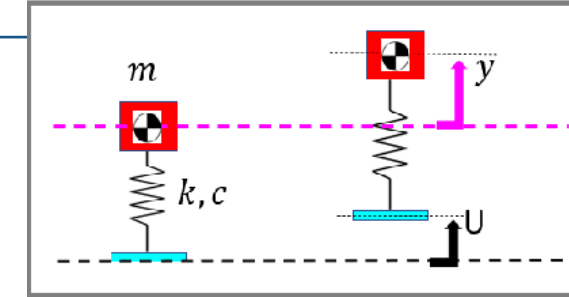
Actuation

Sensing

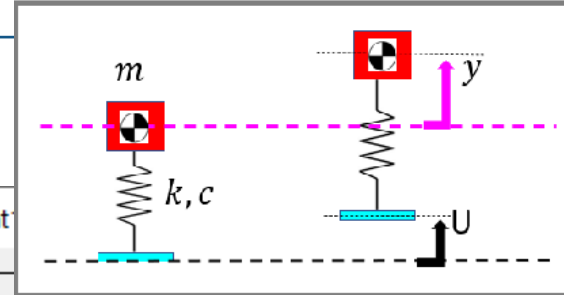
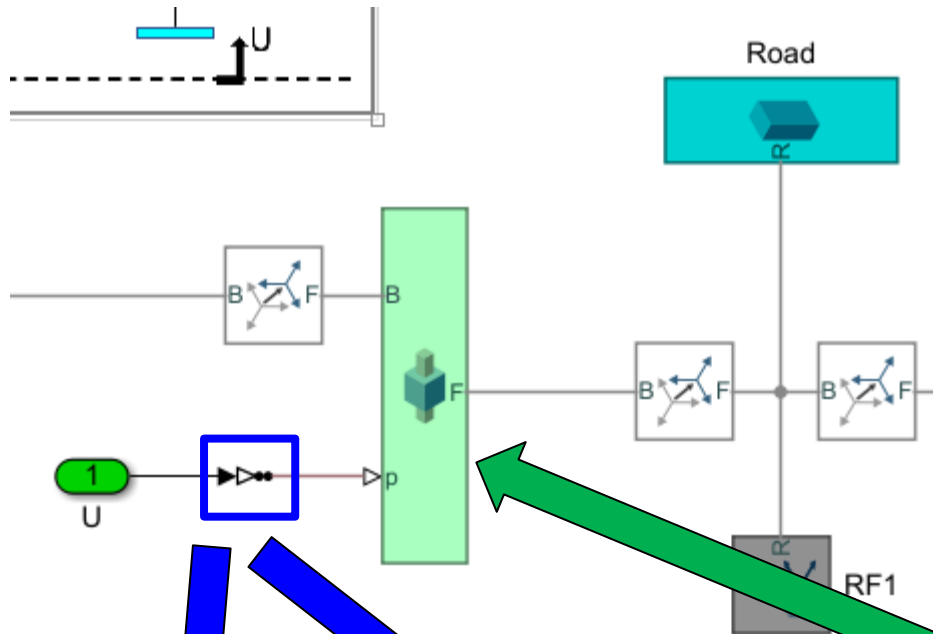
Mode Configuration

Composite Force/Torque Sensing

OK Cancel Help Apply



Setting up the road displacement



Prismatic Joint : Prismatic Joint

Description

Represents a prismatic joint between two frames. This joint has one translational degree of freedom represented by one prismatic primitive. The joint constrains the follower origin to translate along the base z-axis, while the base and follower axes remain aligned.

In the expandable nodes under Properties, specify the state, actuation method, sensing capabilities, and internal mechanics of the primitives of this joint. After you apply these settings, the block displays the corresponding physical signal ports.

Ports B and F are frame ports that represent the base and follower frames, respectively. The joint direction is defined by motion of the follower frame relative to the base frame.

Properties

- ☒ Z Prismatic Primitive (Pz)
- ☒ State Targets
- ☒ Internal Mechanics
- ☒ Limits
- ☒ Actuation
 - Force: Automatically Computed
 - Motion: Provided by Input
- ☒ Sensing
 - ☒ Mode Configuration
 - ☒ Composite Force/Torque Sensing

OK Cancel Help Apply

Parameters

Units Input Handling

Input signal unit:

☐ Apply affine conversion

Parameters

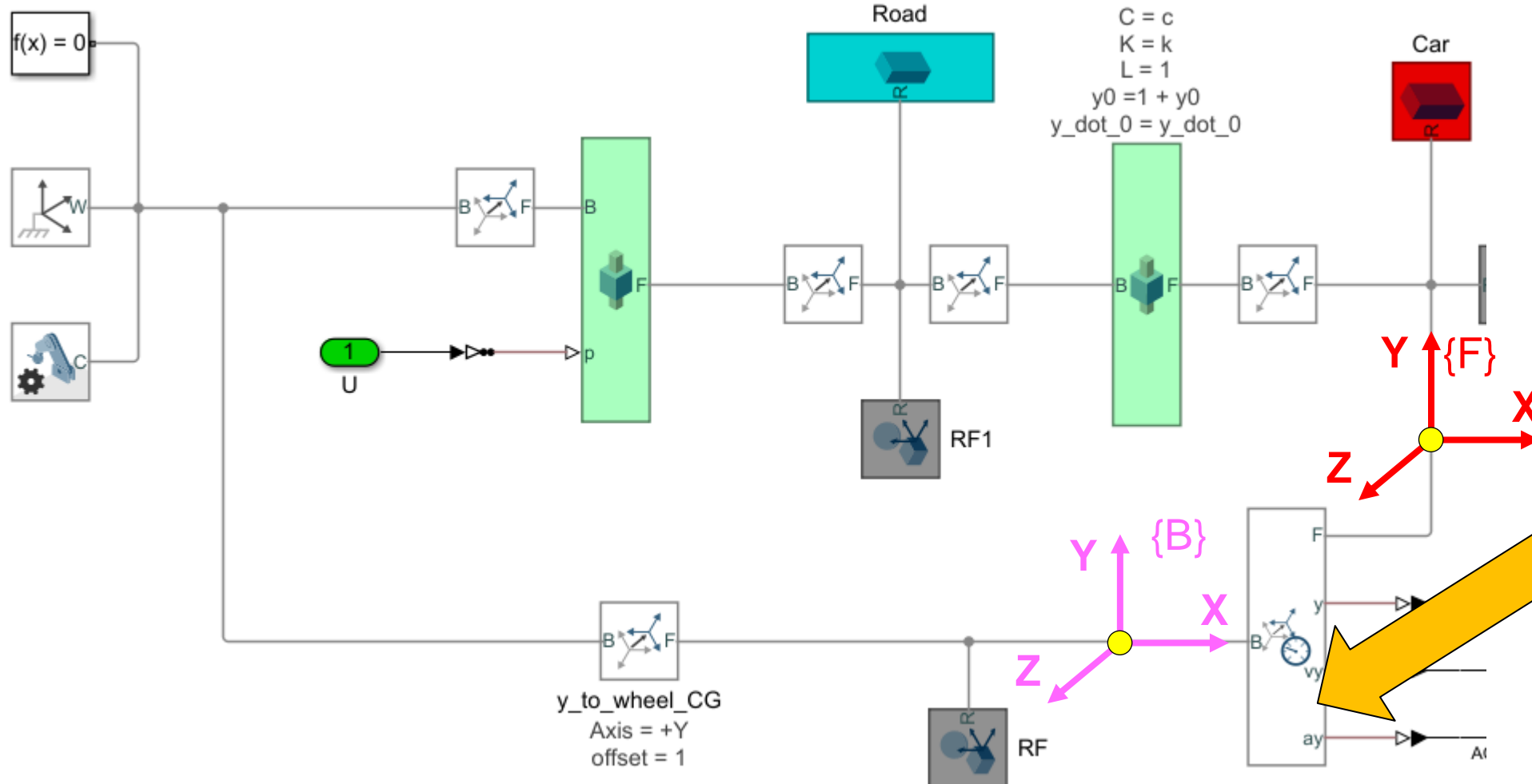
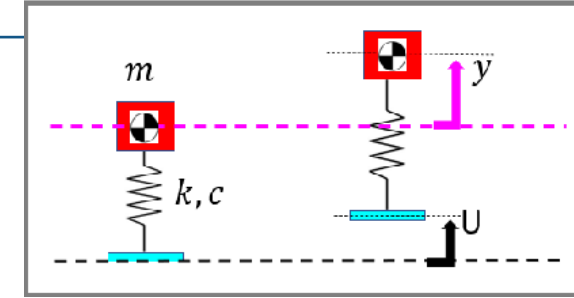
Units Input Handling

Filtering and derivatives:

Input filtering order:

Input filtering time constant (in seconds):

Taking measurements



the [Transform Sensor](#) block Measures the relationship between the FOLLOWER frame relative to the BASE frame