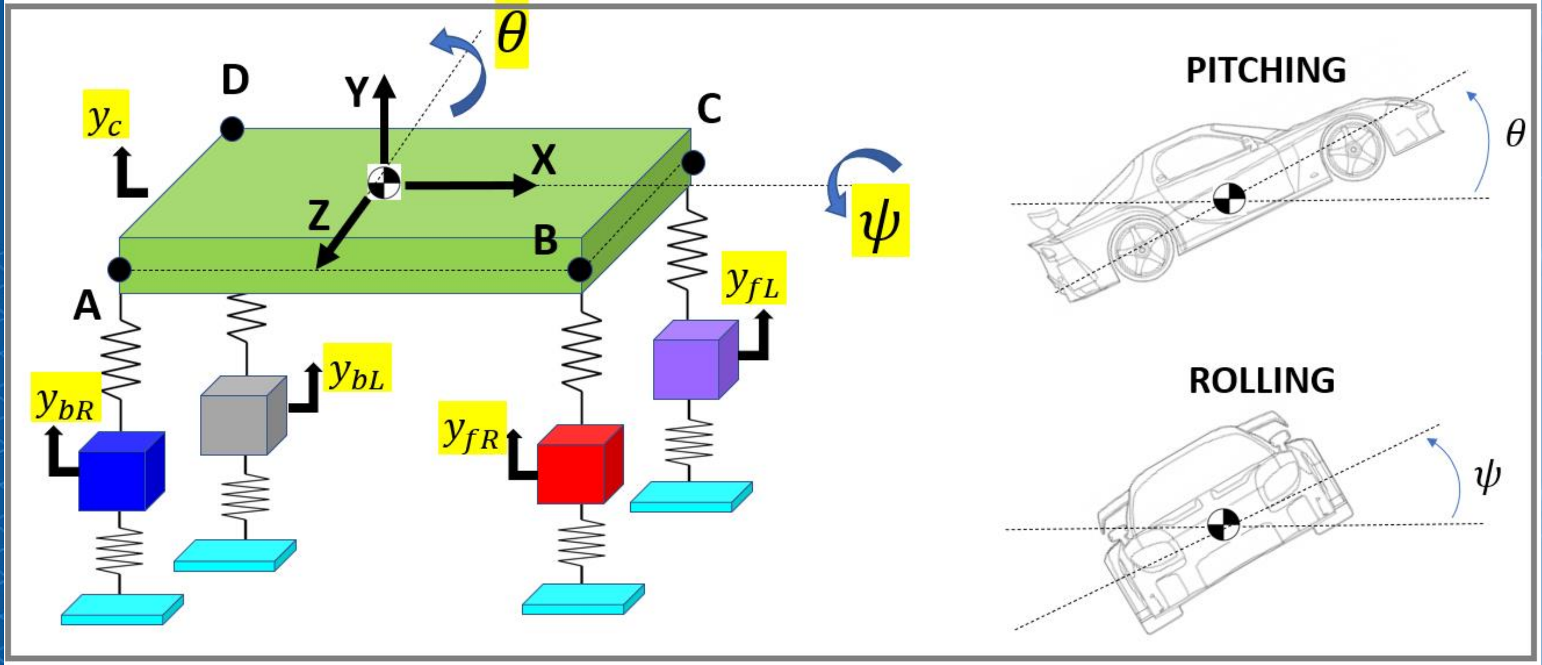
7-dof Transverse Car Dynamics



Modelling and Simulation in MATLAB

Brad Horton
Engineer
MathWorks

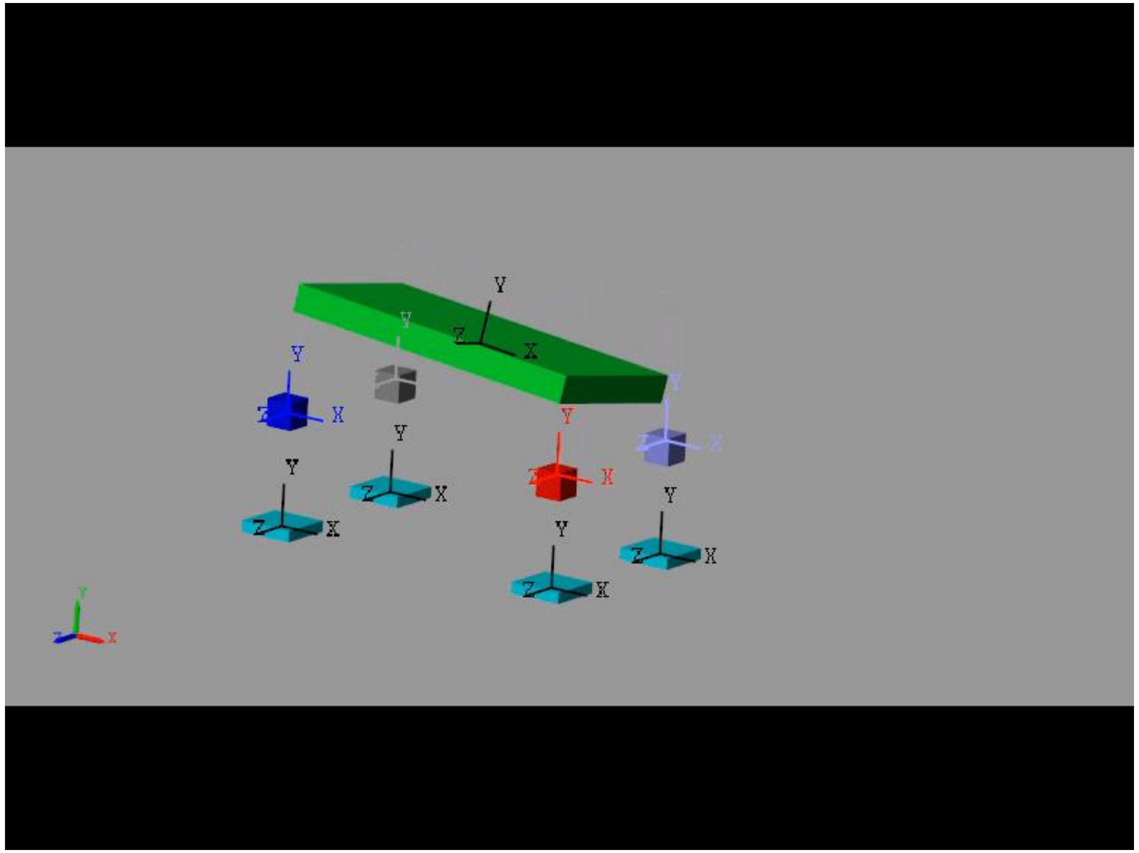
Agenda

- 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**

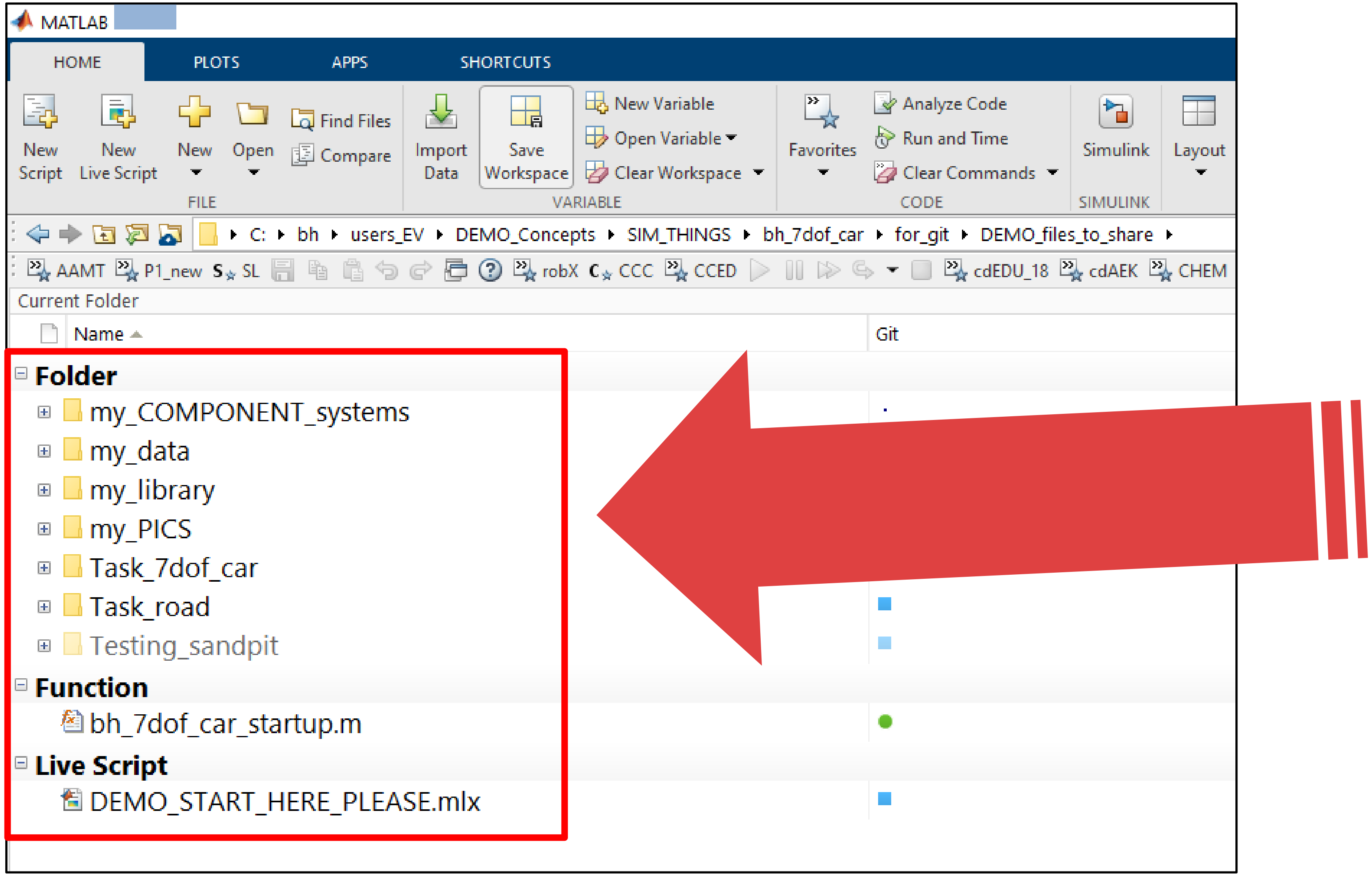


- Next steps (for YOU)
  - Do the Spring-Mass-Damper Hands on course Module

Required Resources for this class

Required Resources – part 1 of 3

- Do you have the DEMO files ?



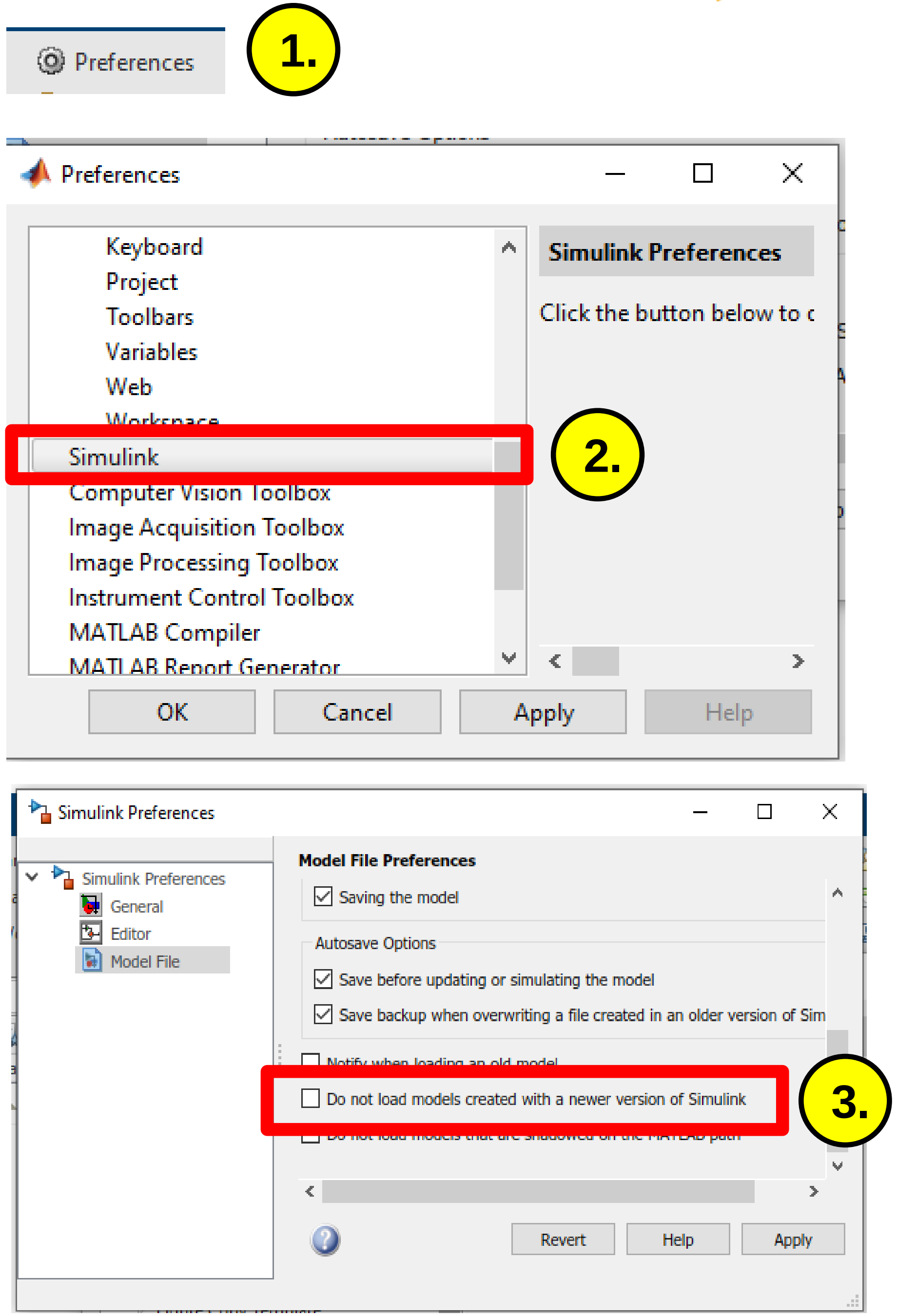
Have you got this ?

Required Resources – `part 2 of 3`

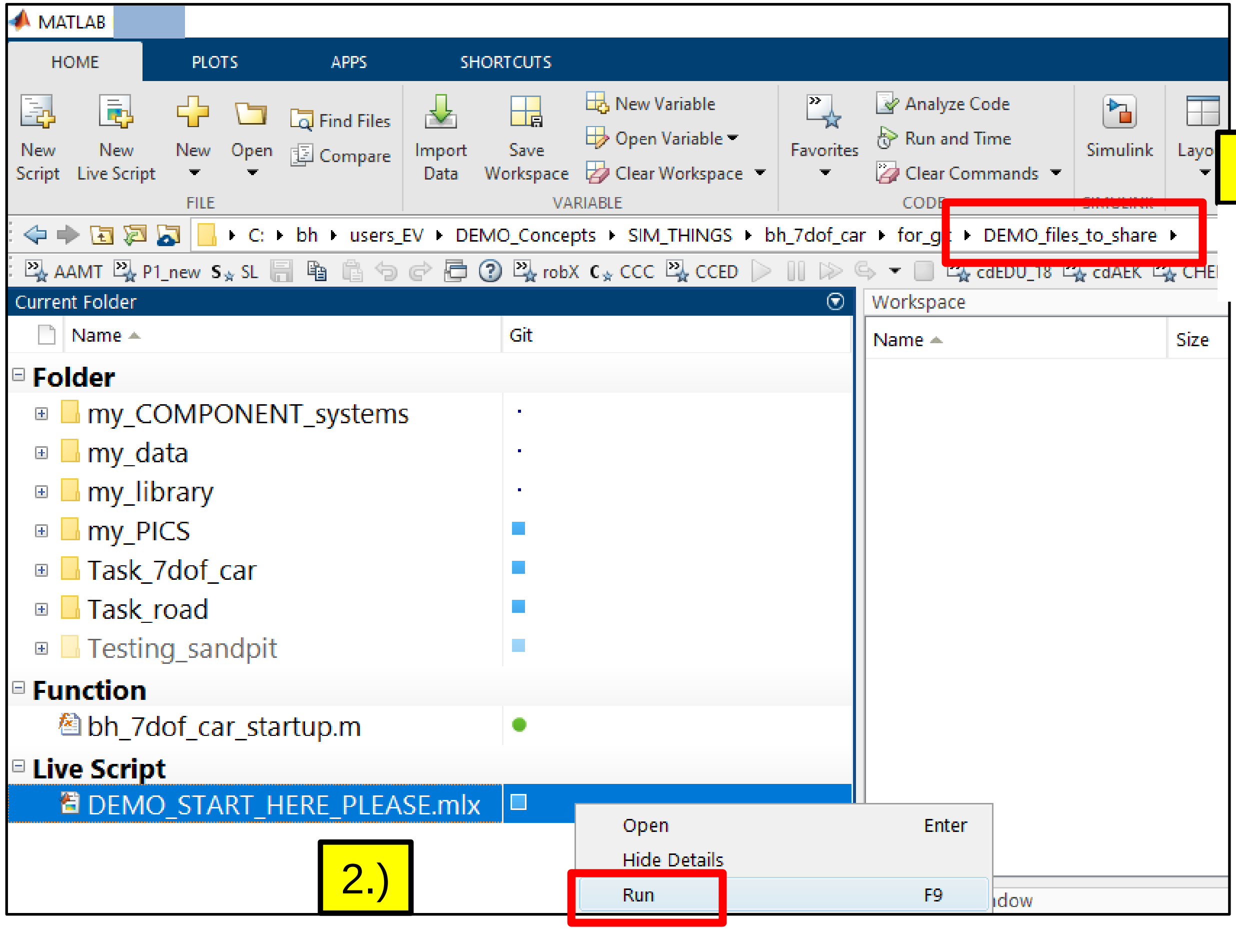
- Do you have the MATLAB License installed ?
  - You need the **R2021b** release installed

    - As a minimum you will need:
      1. *MATLAB*
      2. *Symbolic Math Toolbox*
      3. *Simulink*
      4. *Simscape*
      5. *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



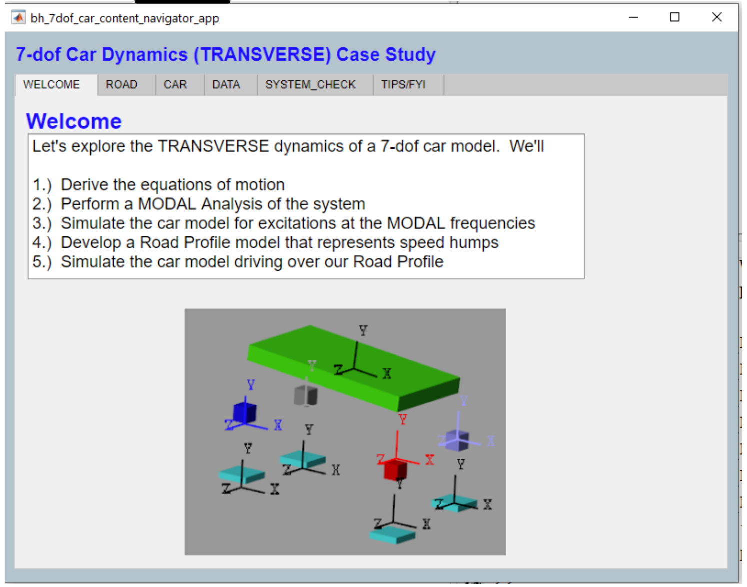
1.)

You are in THIS folder

2.)

Right click THIS file and select Run

3.) *THIS APP should appear*



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 !

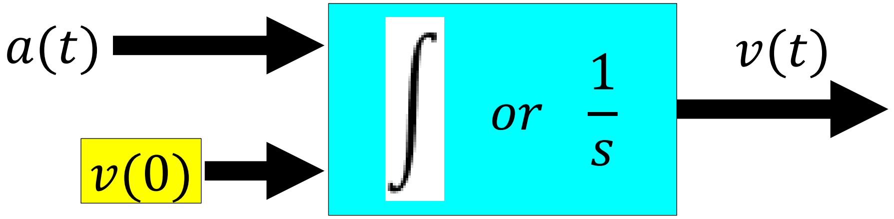
A refresher on modelling dynamic systems in Simulink

The concept of integrating a dynamic signal

Integrate acceleration to get velocity

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

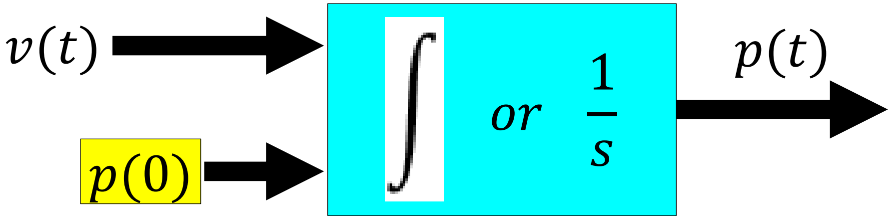
$$v(t) \;=\; v(0) \;+\; \int_0^t a(\tau)\,.\,d\tau$$



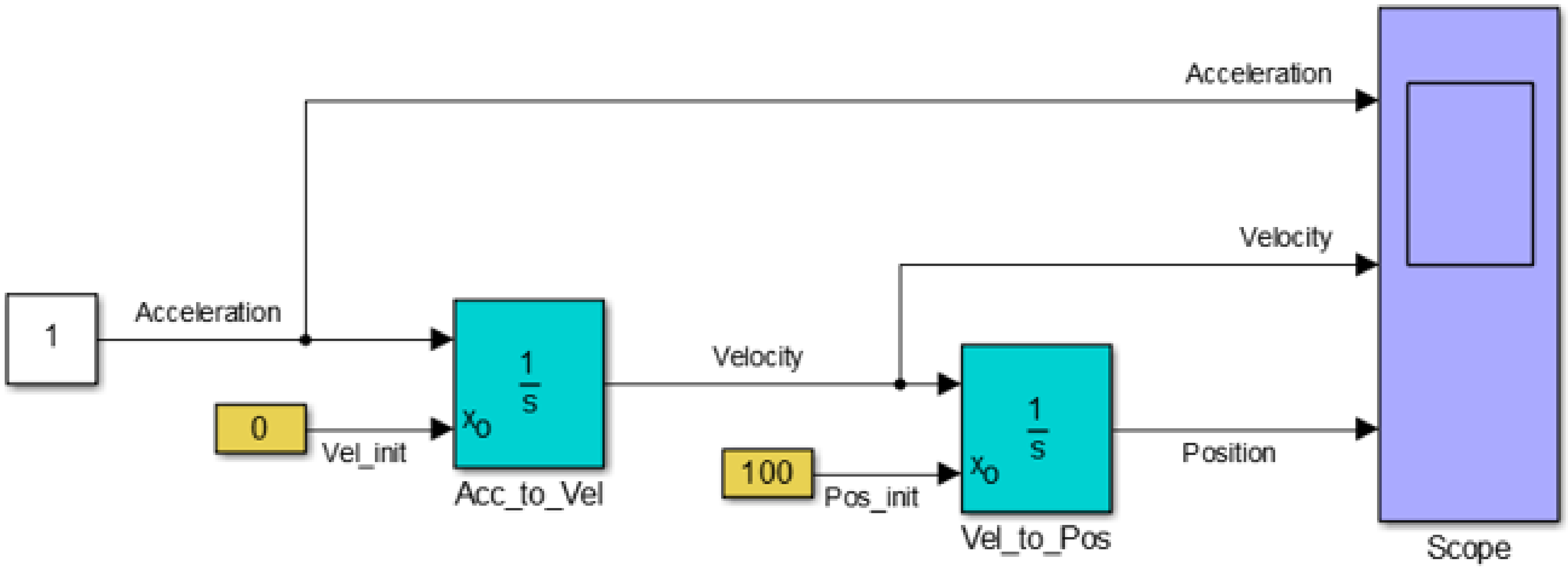
Integrate velocity to get position

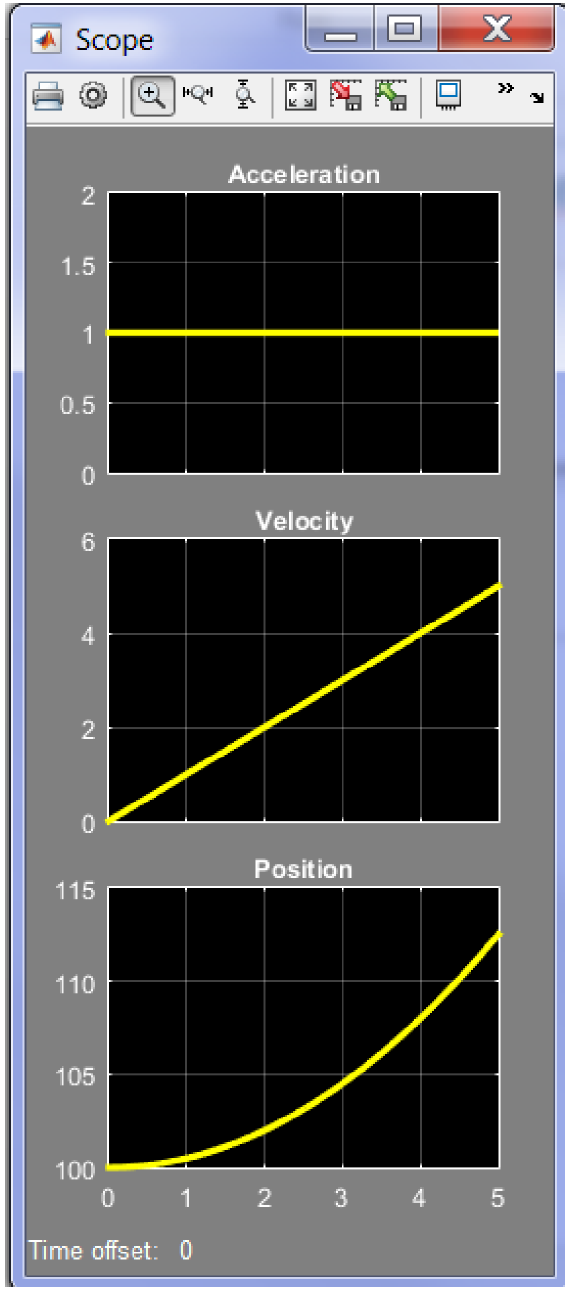
$$\int_0^t v(\tau)\,.\,d\tau \;=\; \int_0^t \frac{dp(\tau)}{d\tau}.\,d\tau \;=\; p(t) - p(0)$$

$$p(t) \;= p(0) \;+\; \int_0^t v(\tau)\,.\,d\tau$$



The concept of integrating a dynamic signal

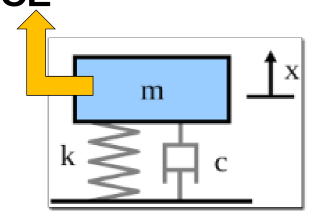




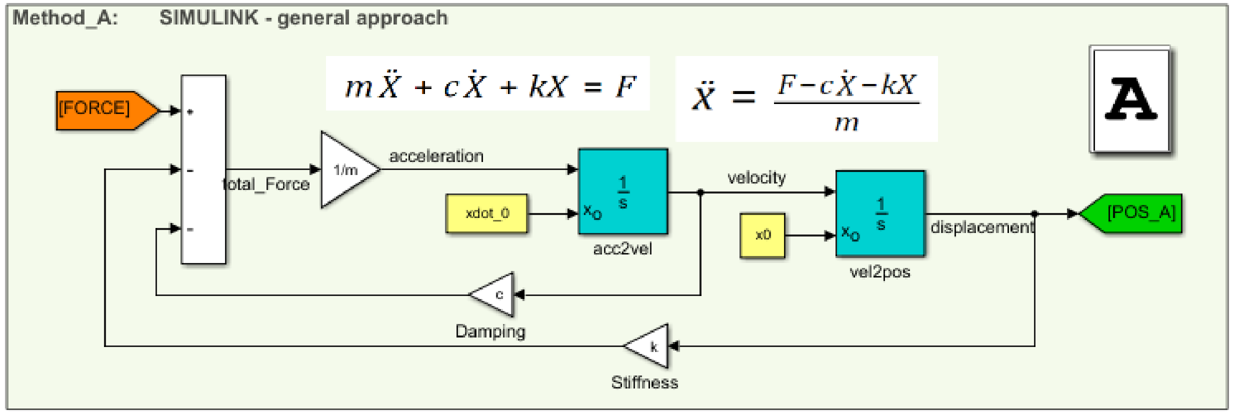
$$v(t) = v(0) + \int_0^t a(\tau)\,.d\tau$$

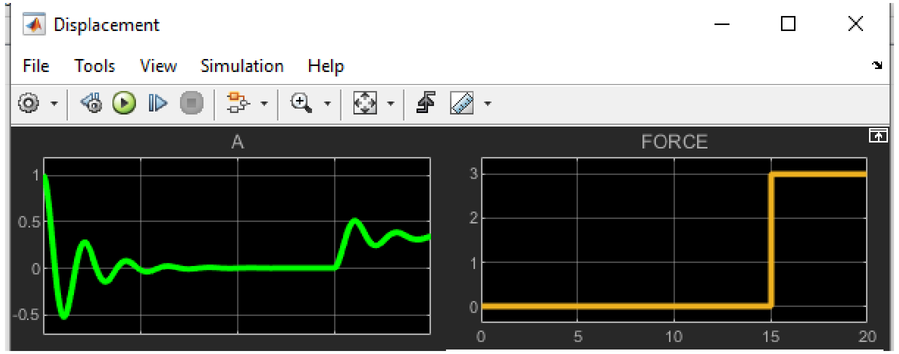
$$p(t) = p(0) + \int_0^t v(\tau)\,.d\tau$$

A Hello World model in Simulink

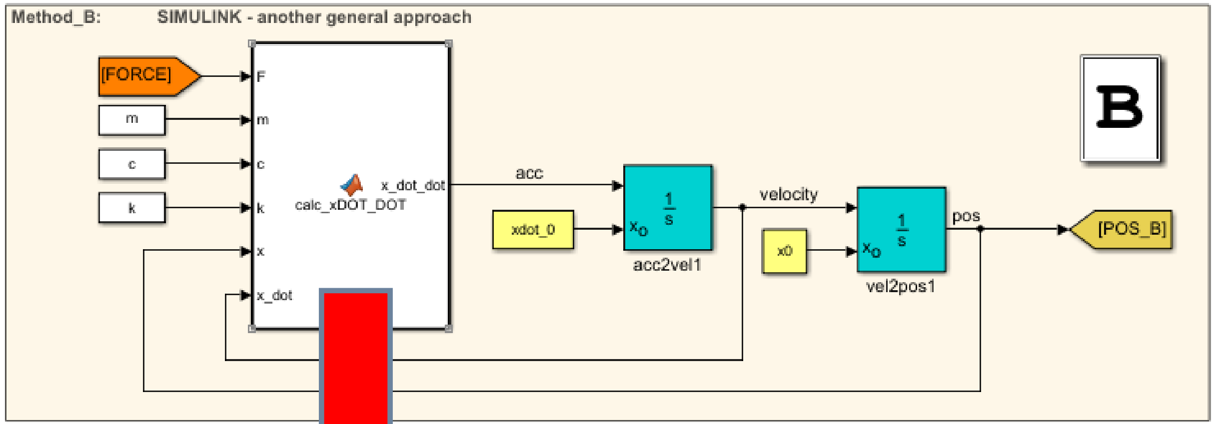


Sometimes you use blocks for everything





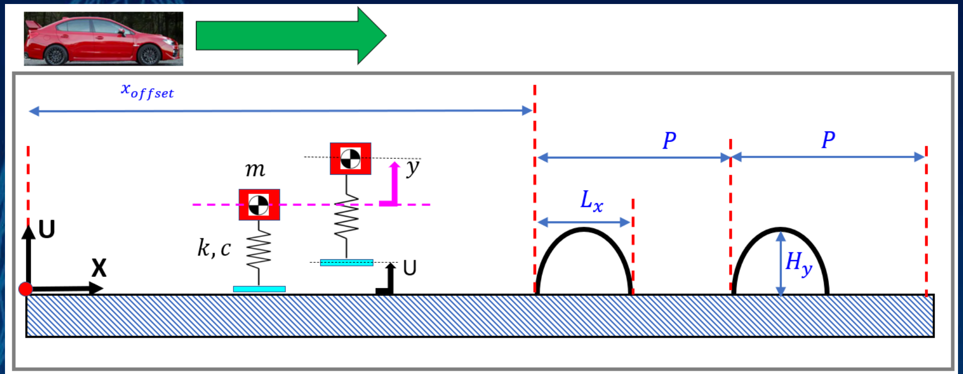
Sometimes you write a MATLAB function for some of it

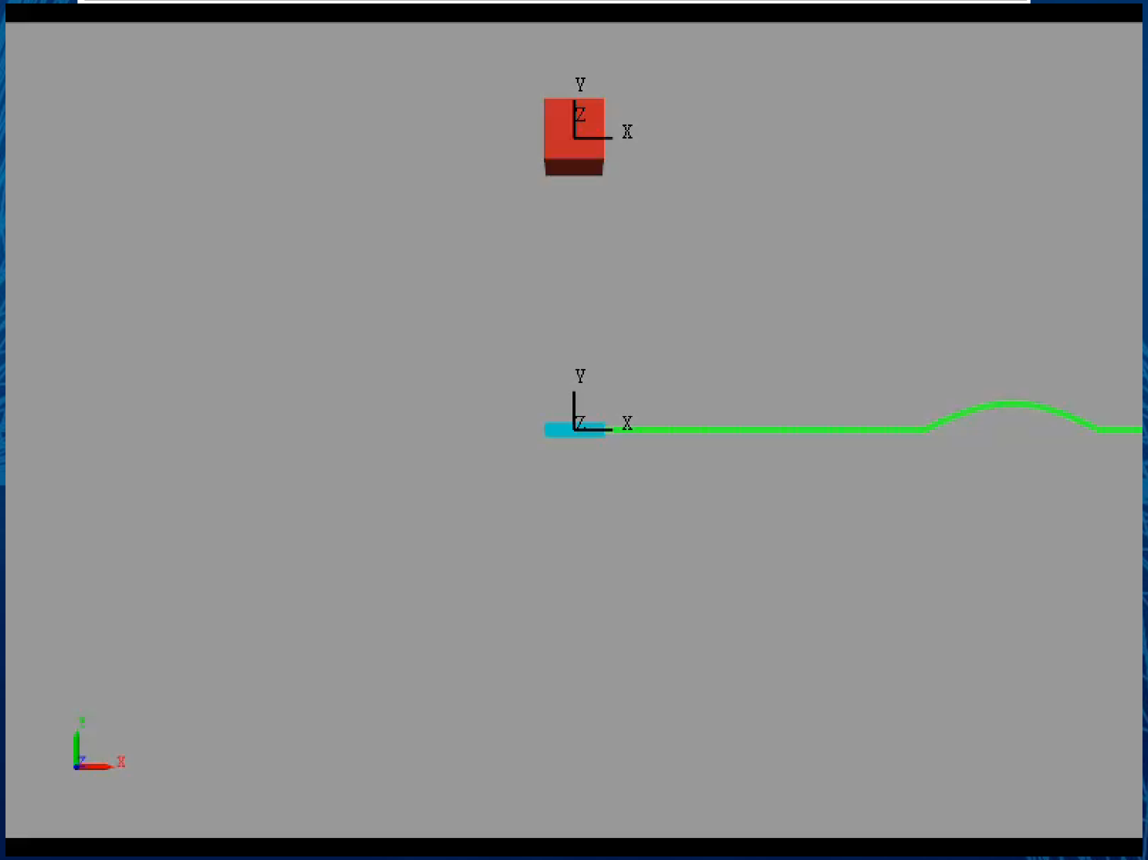


$$\ddot{X} = \frac{F - c.\dot{X} - k.X}{m}$$

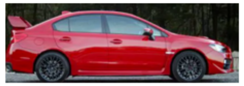
```
function x_dot_dot = calc_xDOT_DOT(F,m,c,k,x,x_dot)

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

end
```

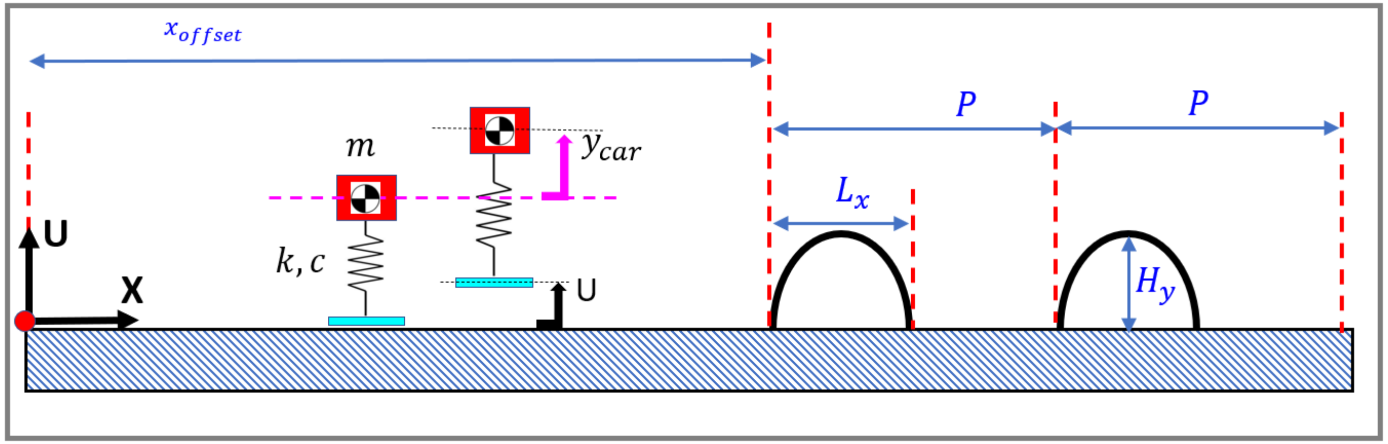
A Road Profile model





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





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

- $U(x) = H_Y.\sin\left(\frac{2.\pi.x}{\lambda}\right) = H_Y.\sin\left(\frac{2.\pi.x}{2.L_X}\right) = H_Y.\sin\left(\frac{\pi.x}{L_X}\right)$

Function INPUTS:

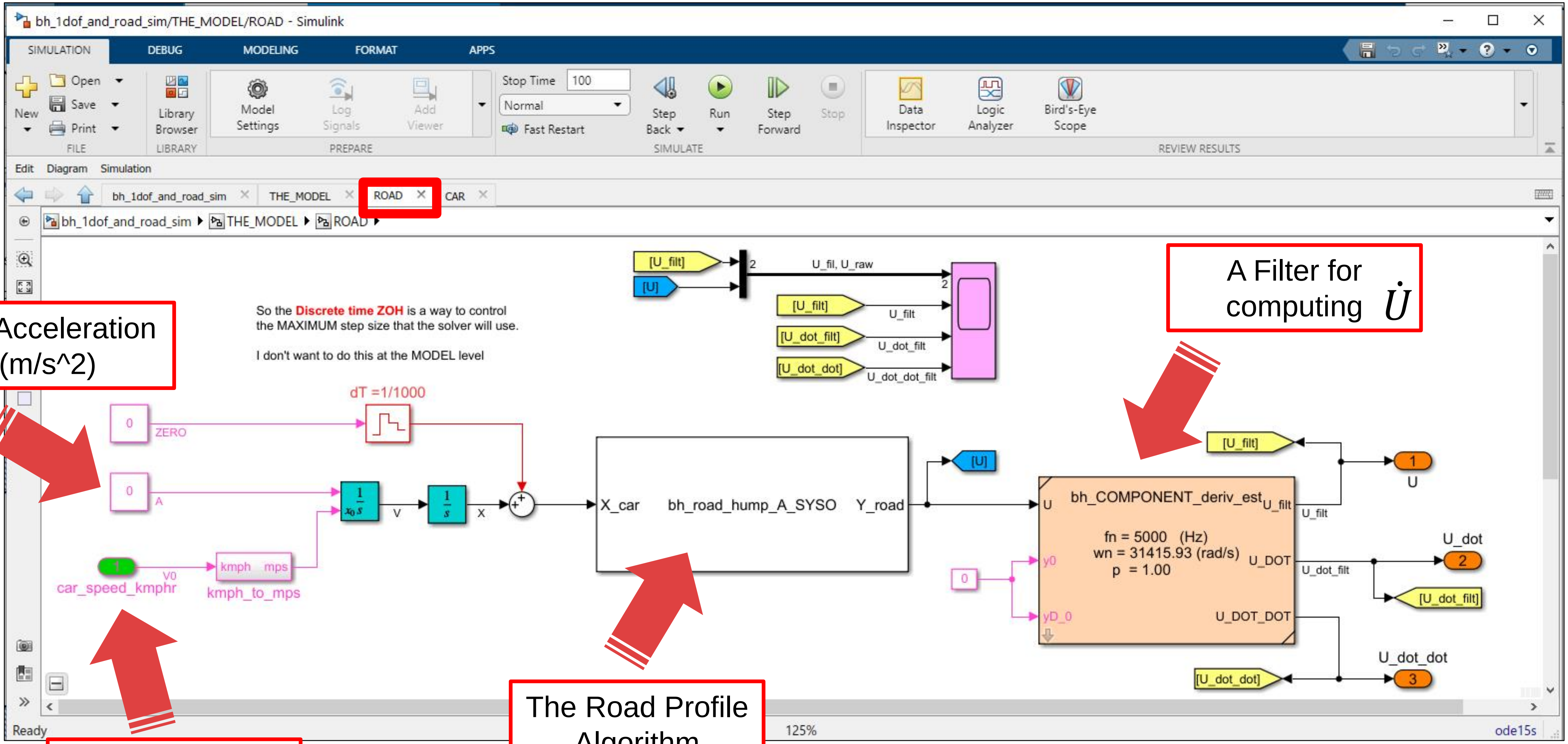
1. `x` : a list of **X-values** at which to compute the height of the road
2. `x_offset` : the horizontal distance at which the FIRST hump begins
3. `Lx` : the **width** of the hump
4. `Hy` : the **Height** of the Hump
5. `P` : the spatial Period of the repeating block

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`



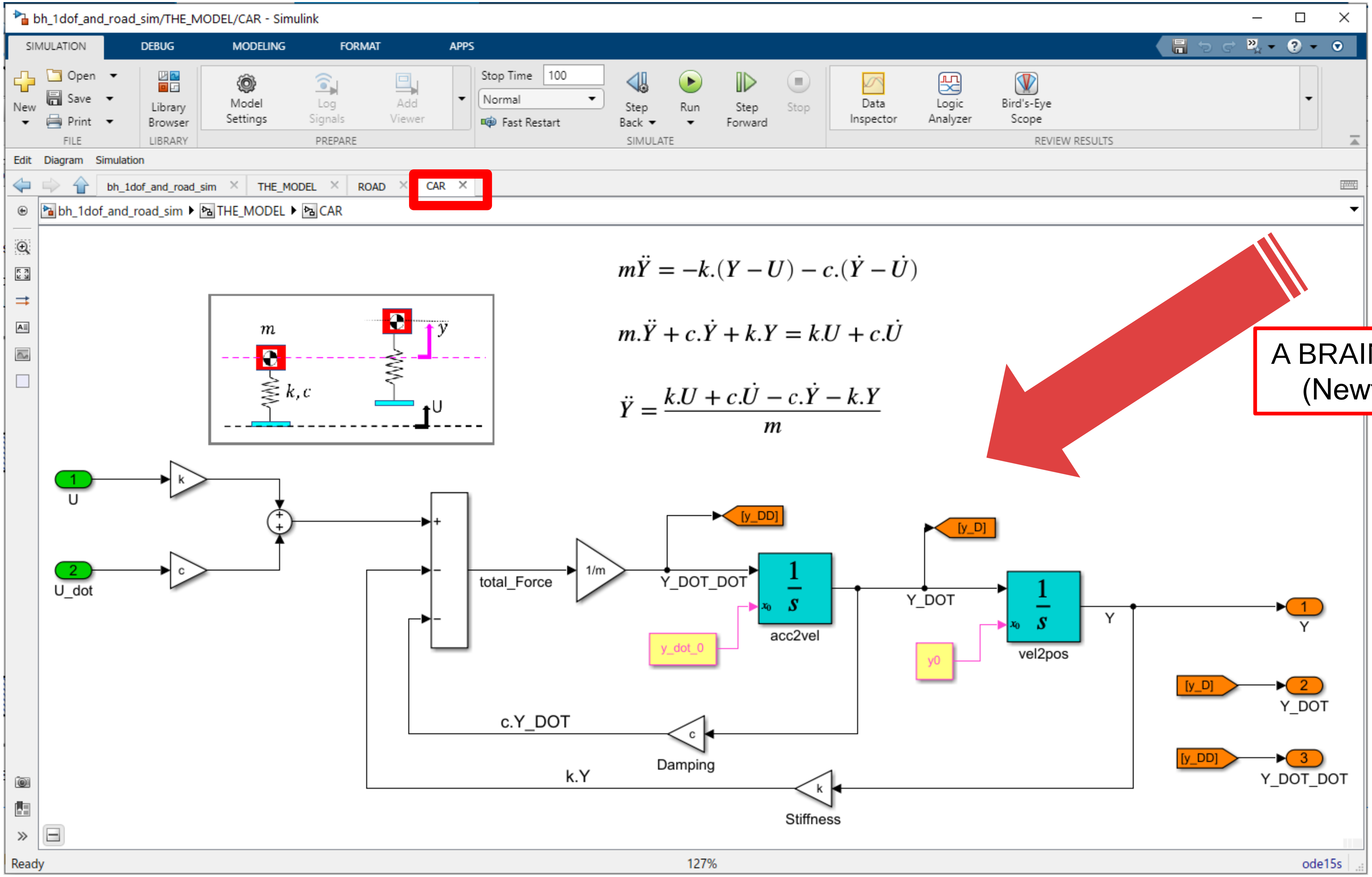
Car Acceleration (m/s^2)

Car **Initial** speed (km/hr)

The Road Profile Algorithm

A Filter for computing $\dot{U}$

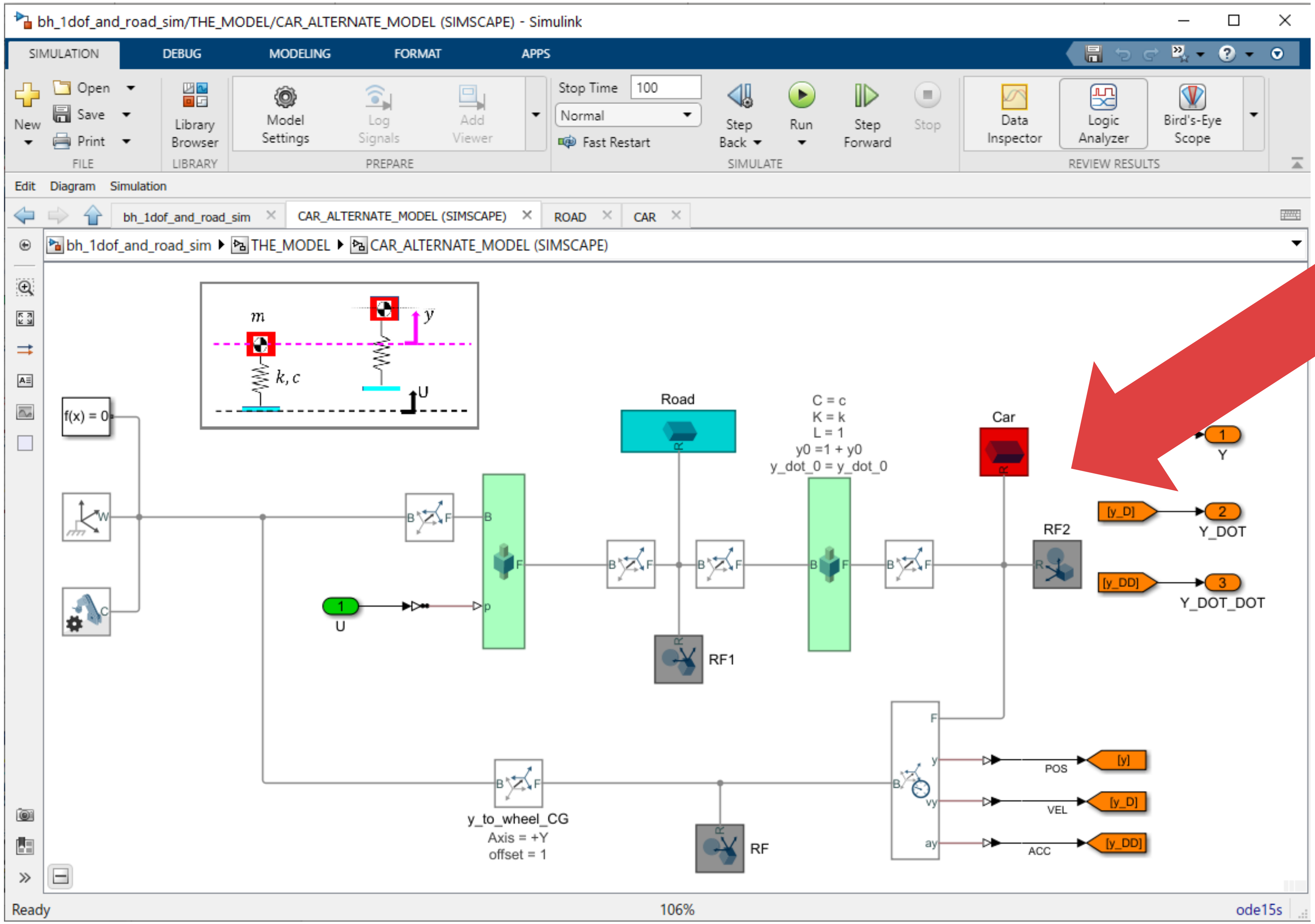
A speed hump Road Profile model – an introduction, Part 3 of 5



Our 1-dof Car dynamics model

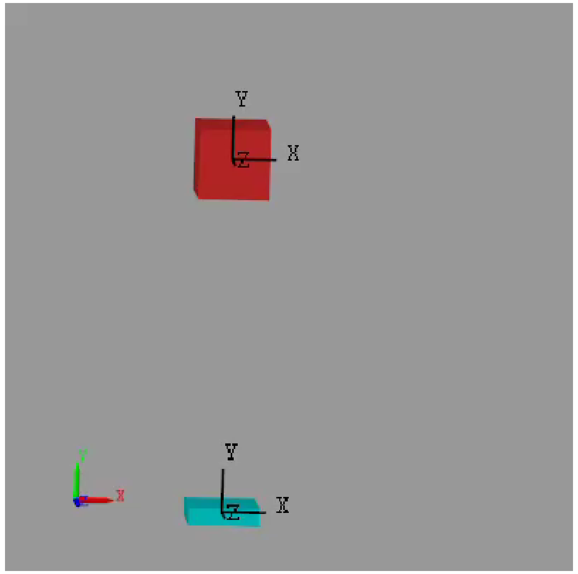
A BRAIN derived model (Newton's 2nd Law)

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



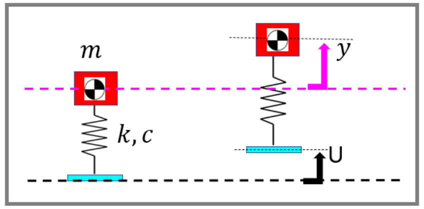
An **Alternate** way of modelling the car dynamics (**Simscape MultiBody**)

The 3D CAD-like viewer appears automatically !



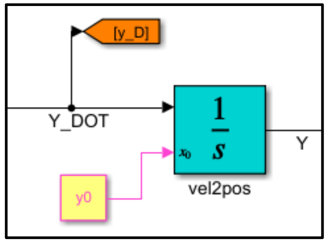
A speed hump Road Profile model – `an introduction, Part 4b of 5`

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



Approach #1

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

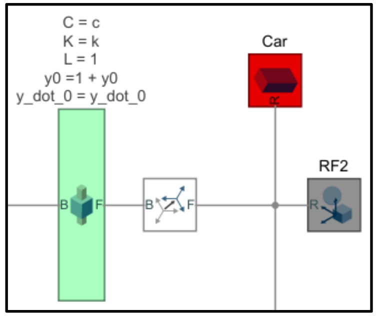


Use integrator blocks

I will show you BOTH techniques

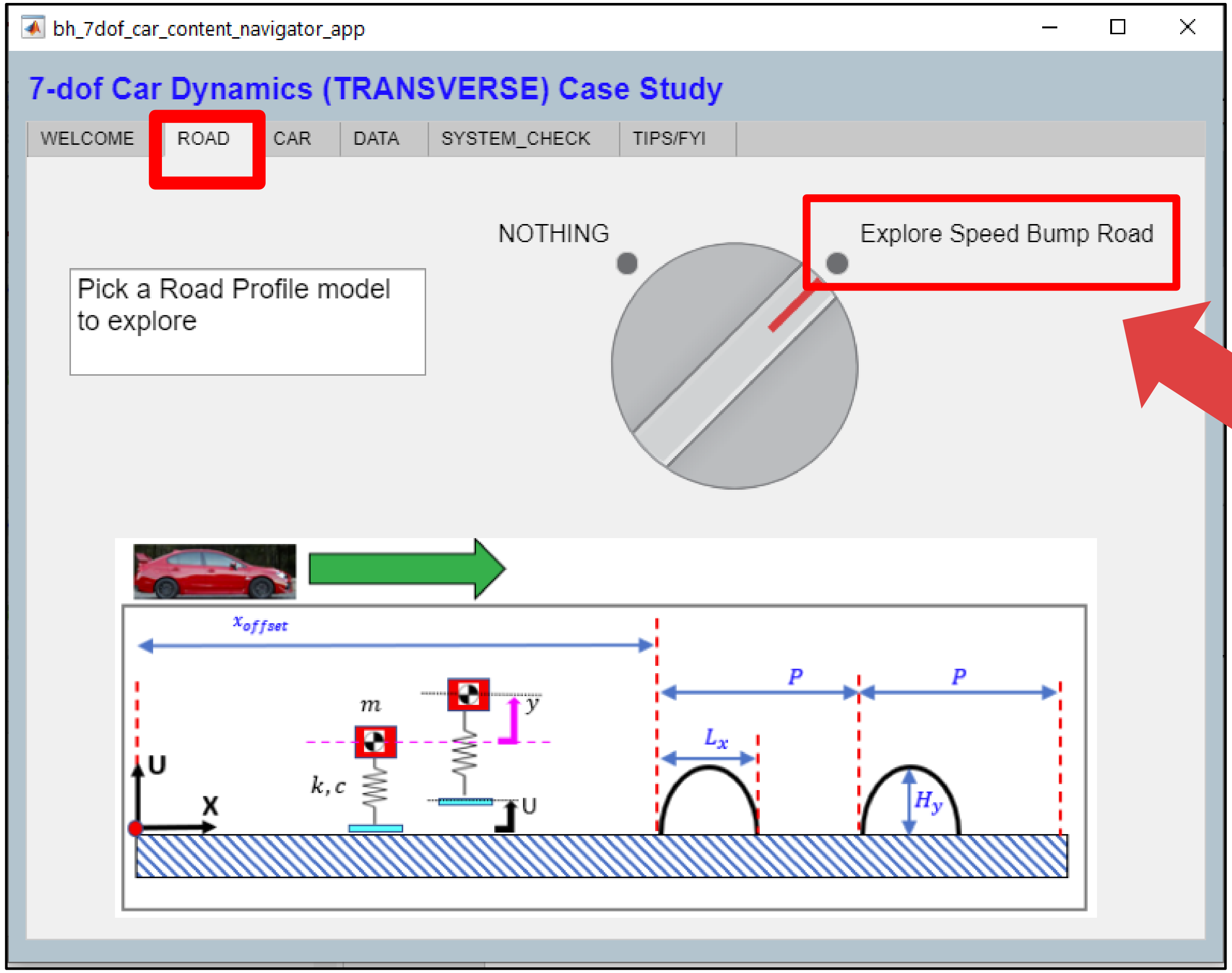
Approach #2

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



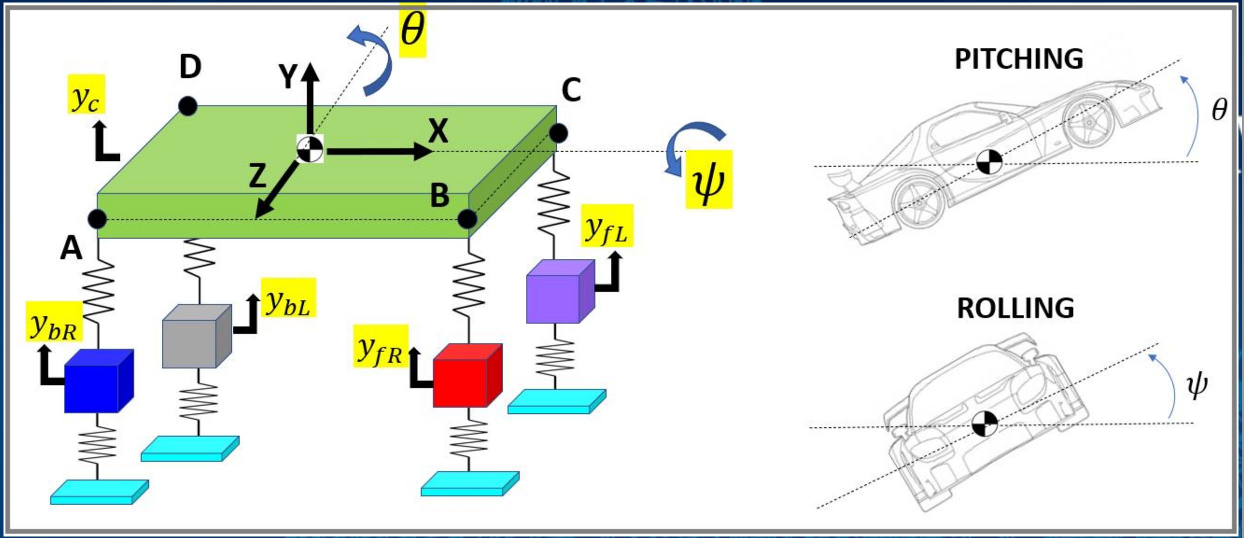
… and a 3D CAD viewer appears automatically !

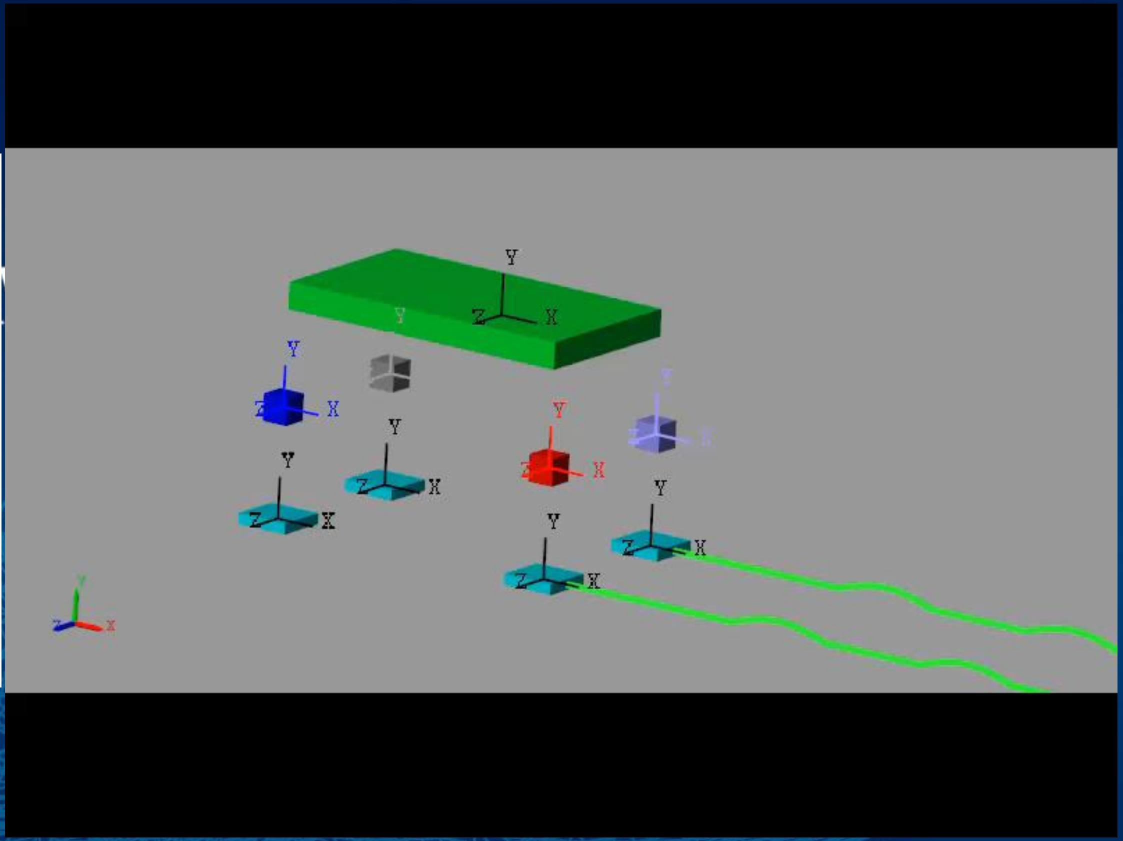
A speed hump Road Profile model – an introduction, Part 5 of 5



Try it !

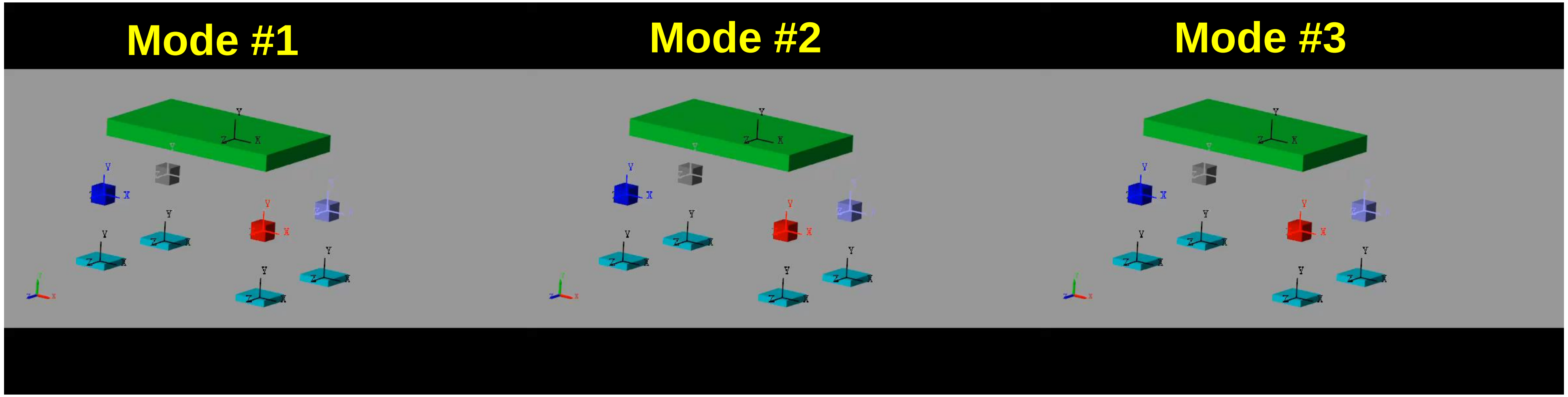
Today's Case Study

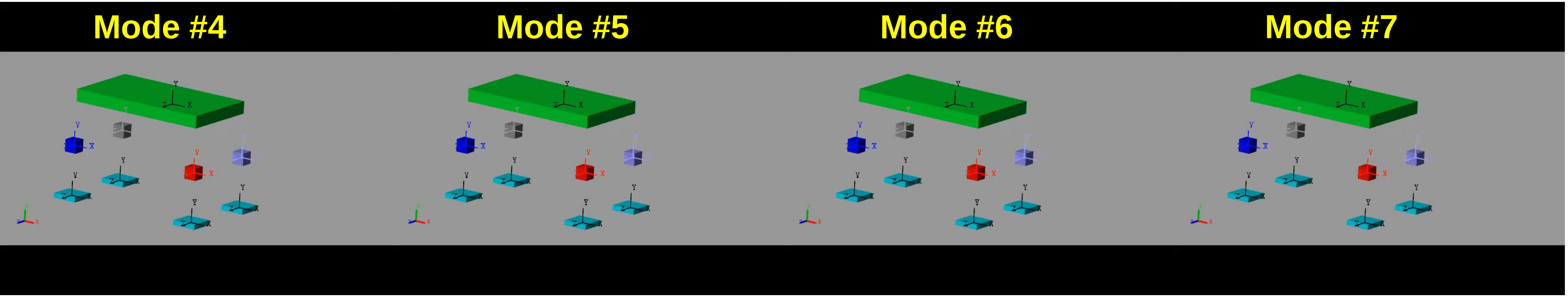




The 7-dof Car Dynamics (Transverse)

The 7 Transverse Undamped MODES

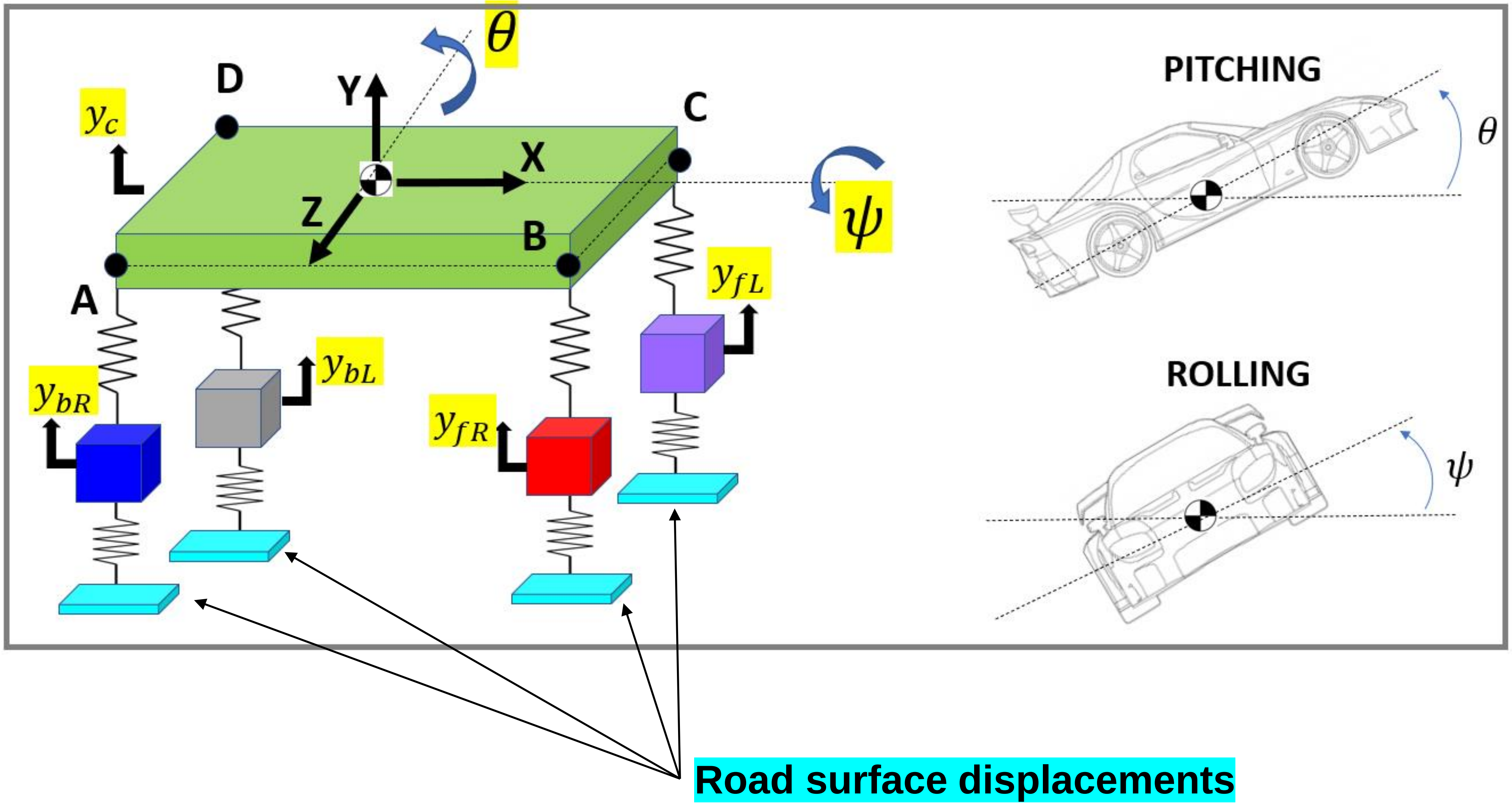




MathWorks

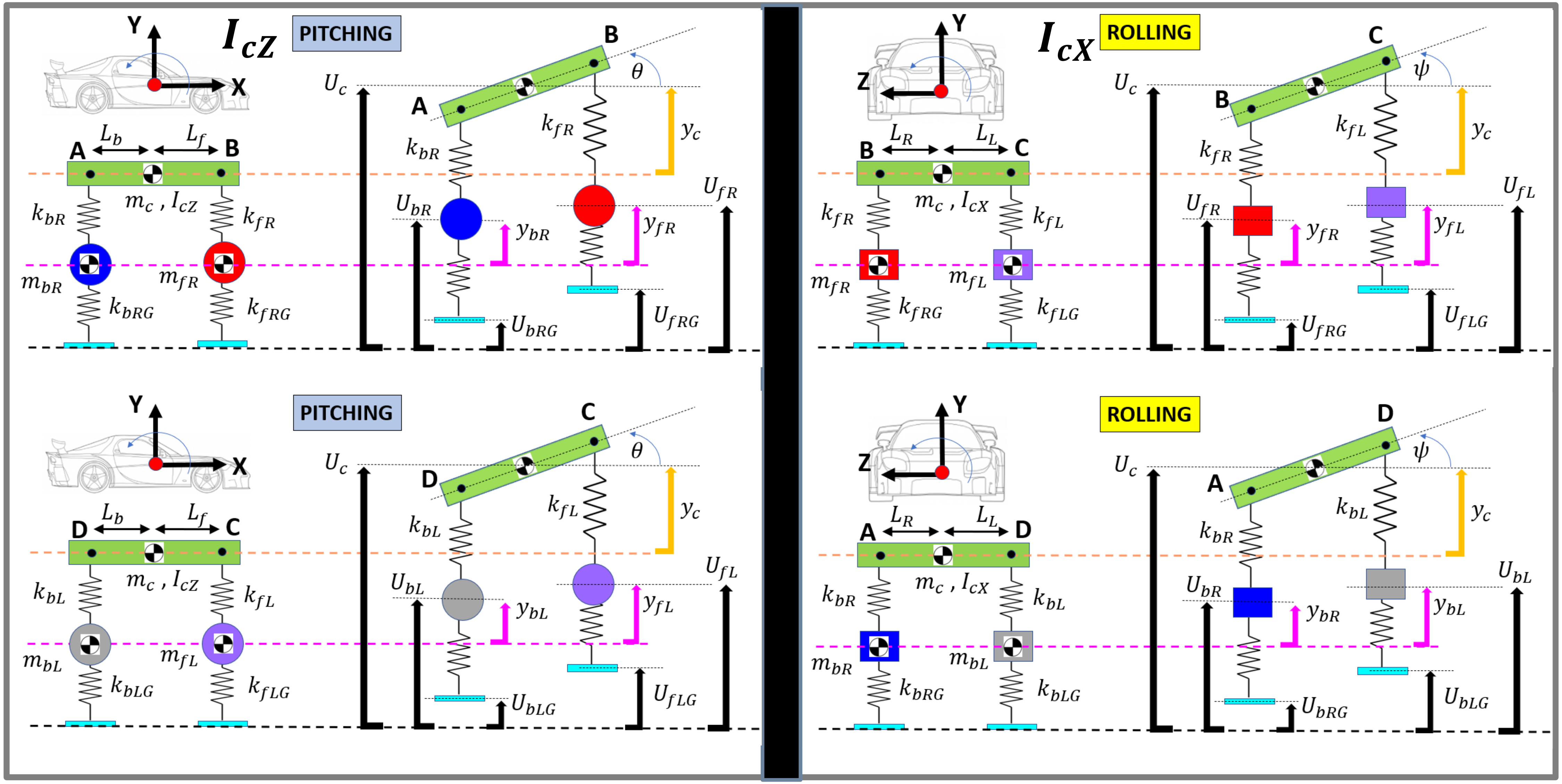
The 7 degrees of freedom

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

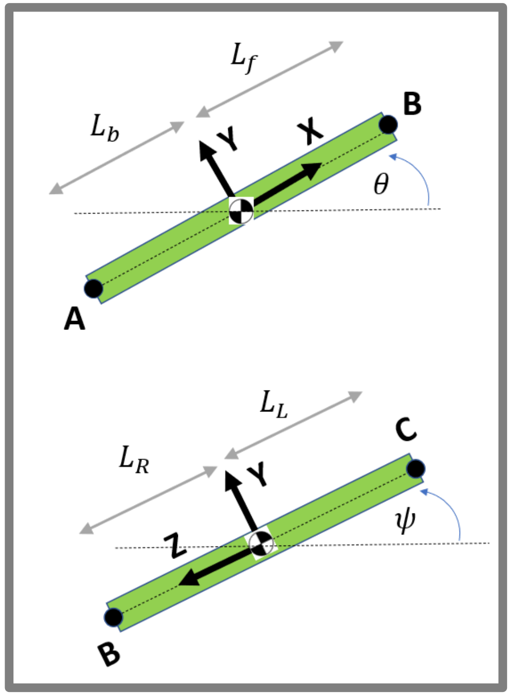


The 7-dof Car – an introduction, Part 2 of 3

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

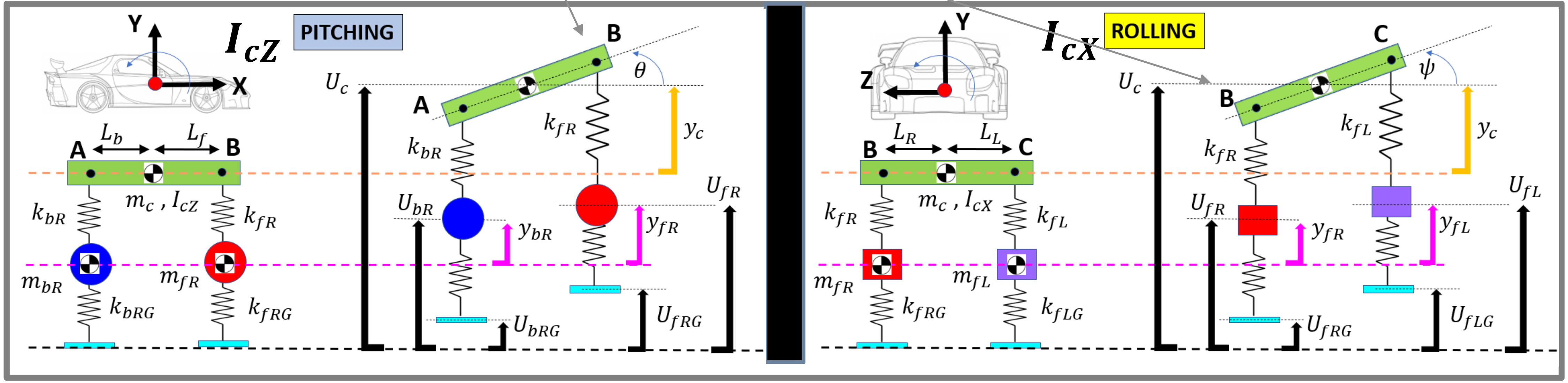


Small angle assumption $\sin(\theta) \approx \theta$
$\sin(\psi) \approx \psi$



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

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



What we'll do: in **MATLAB/Simulink**

DEMO

Try it !

- Derive the equations of motion (Newton's 2nd Law)
  - **Your Engineering BRAIN + MATLAB**

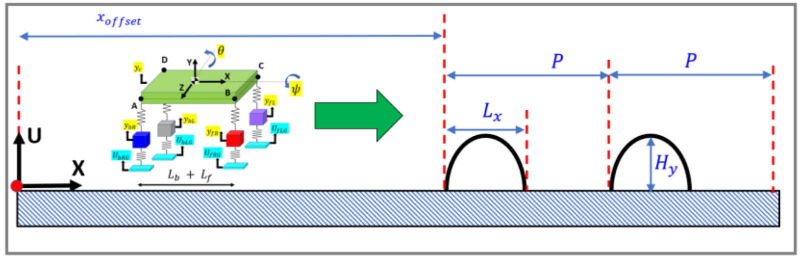
$$M.\ddot{X} + C.\dot{X} + K.X = F$$

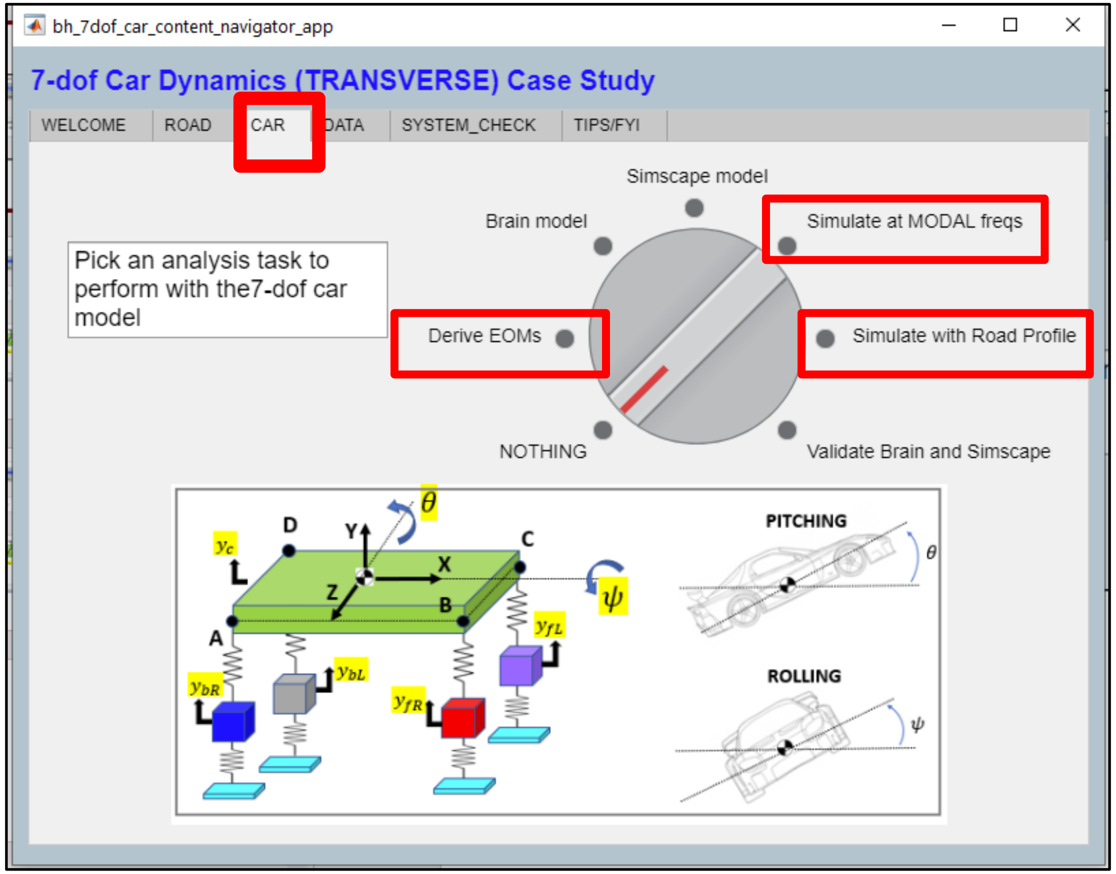
- Perform a MODAL Analysis

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

- 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

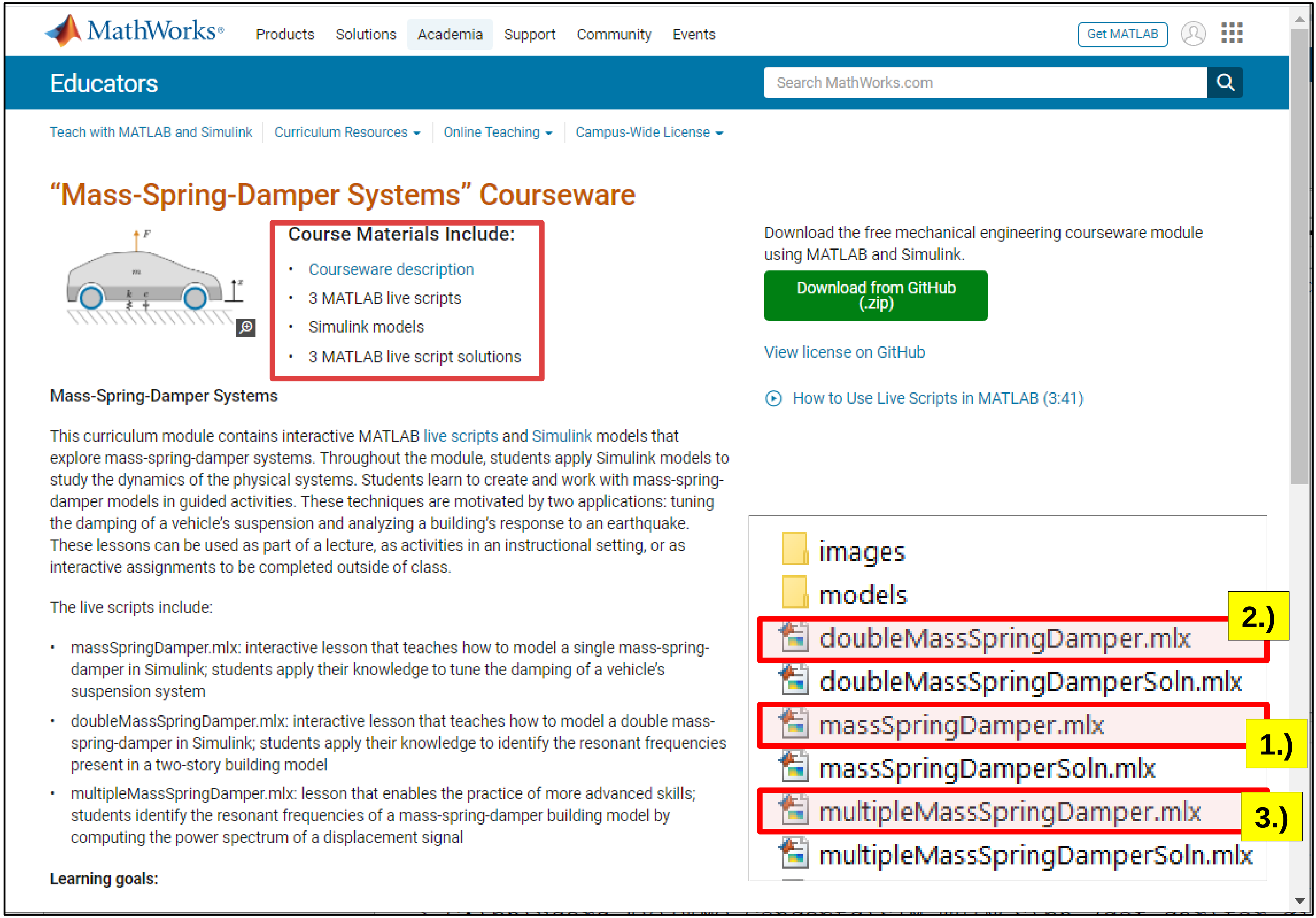




Next Steps

Next Steps:

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

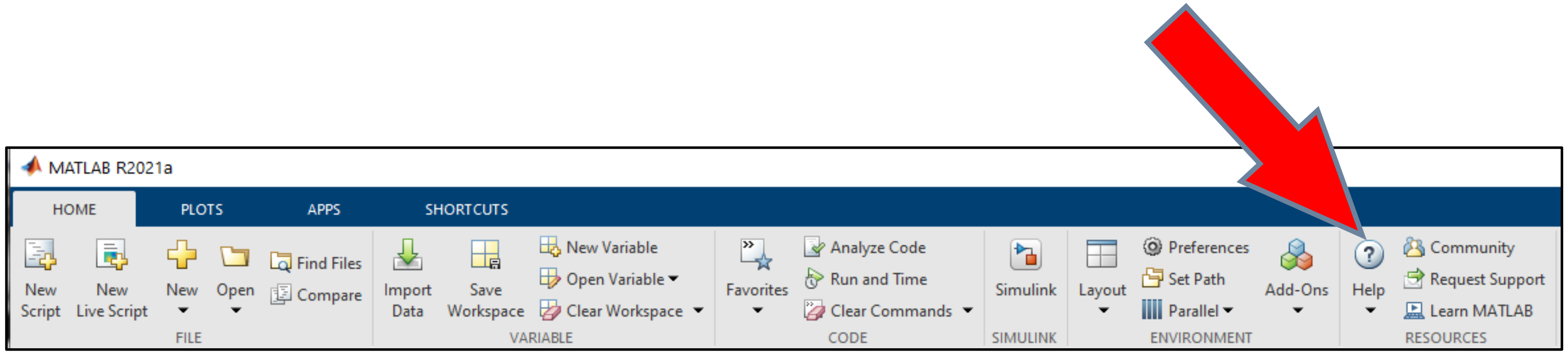


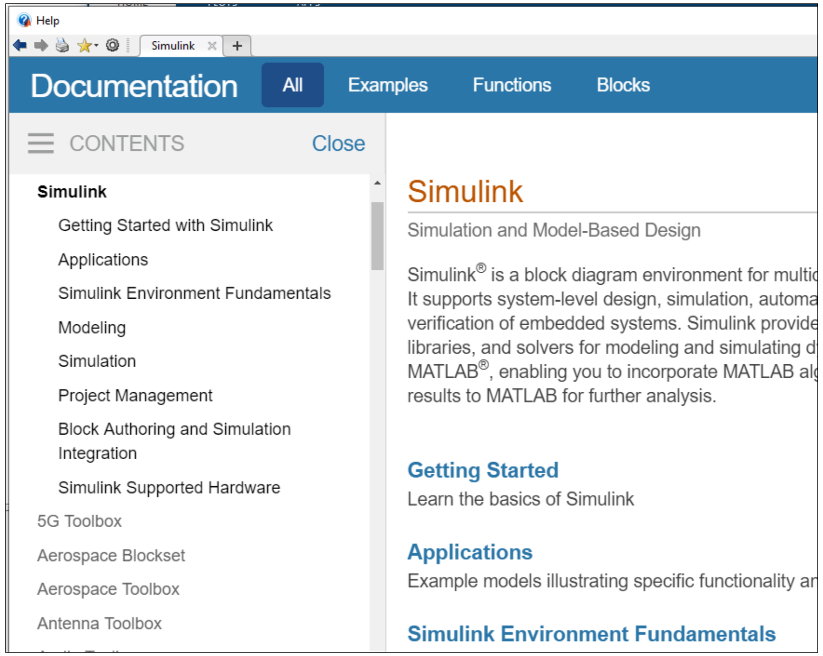
Download our Hands on **(interactive)** Courseware module

Thank you.

(Your BRAIN
Your ENGINEERING) + MATLAB = Very COOL
Things happen

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



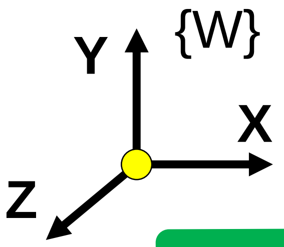


Appendix A:

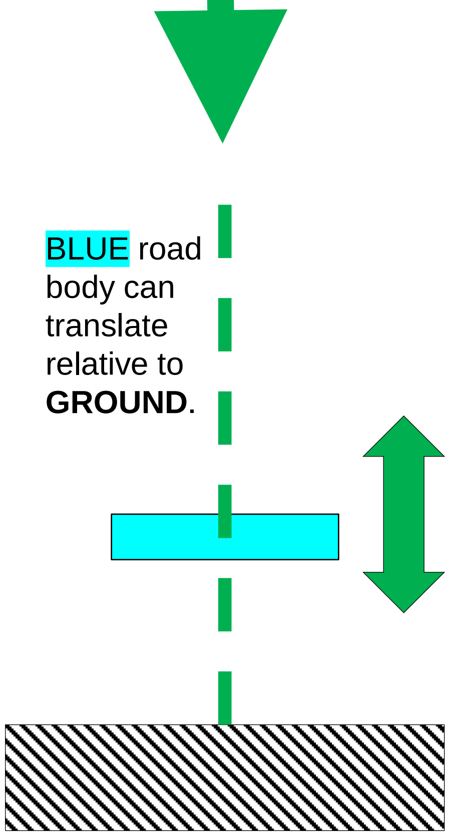
Appendix B:

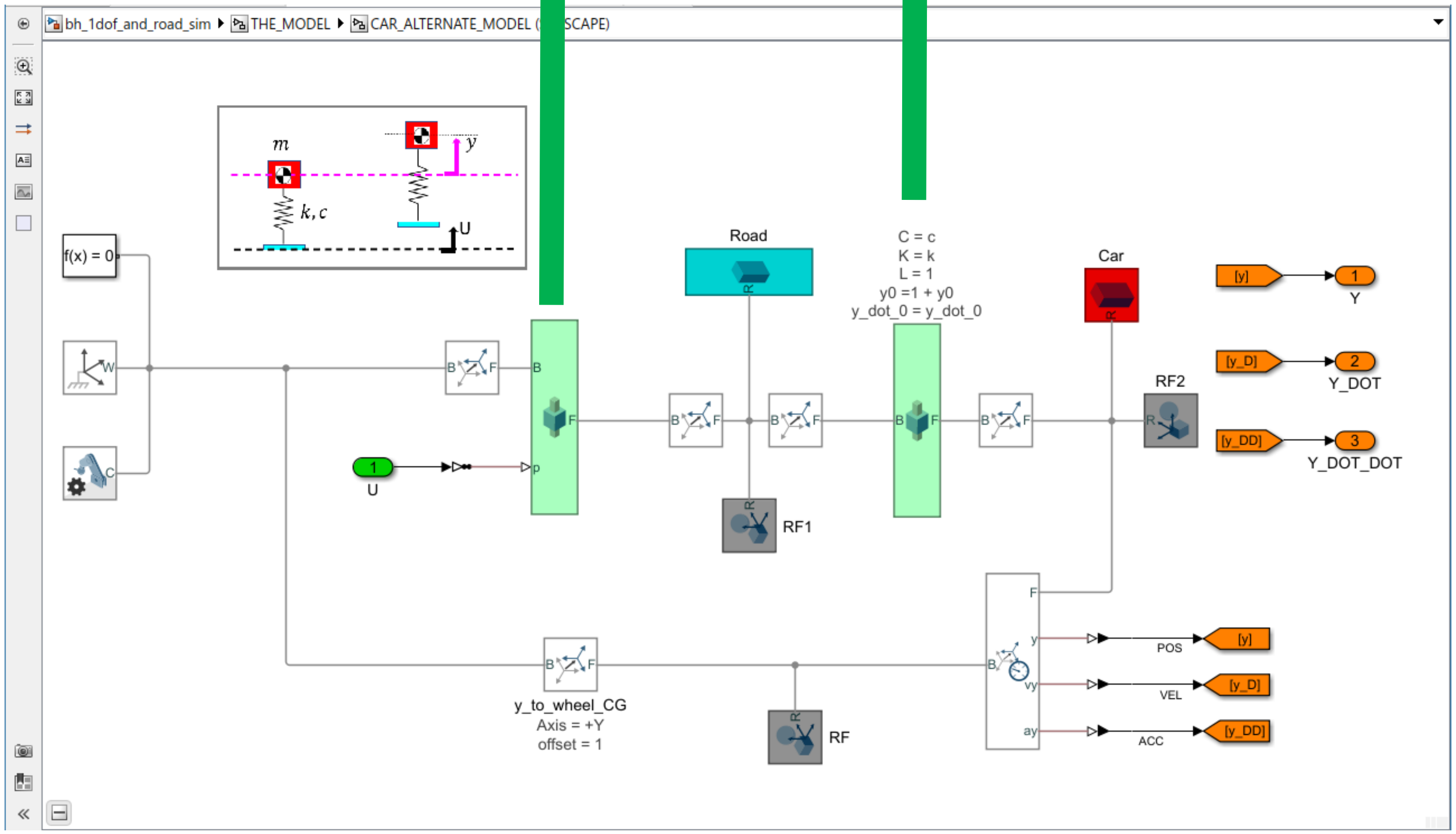
A teardown of the Simscape 1-dof model

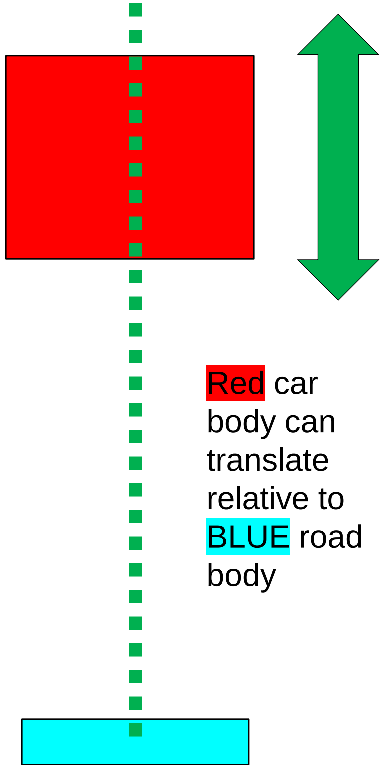
Setting up motion constraints – part 1



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

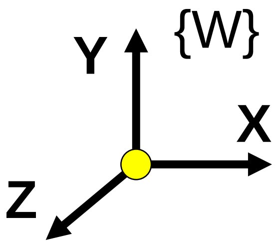


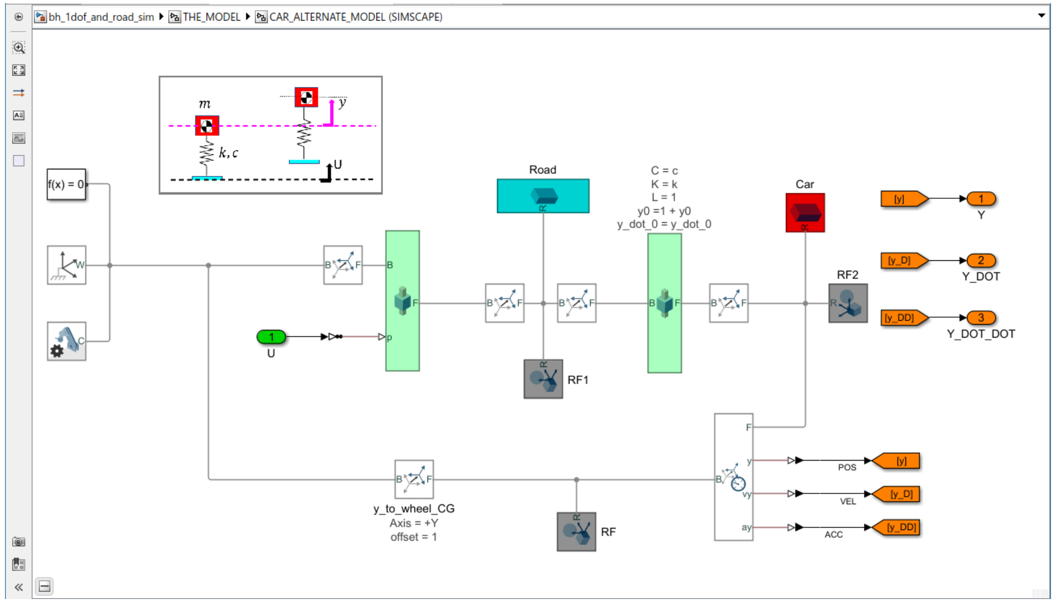


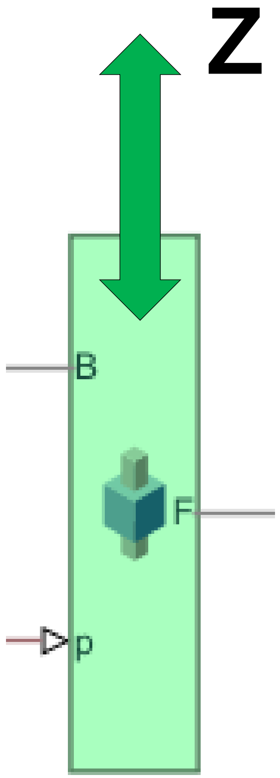


Red car body can translate relative to BLUE road body

Setting up motion constraints – part 2



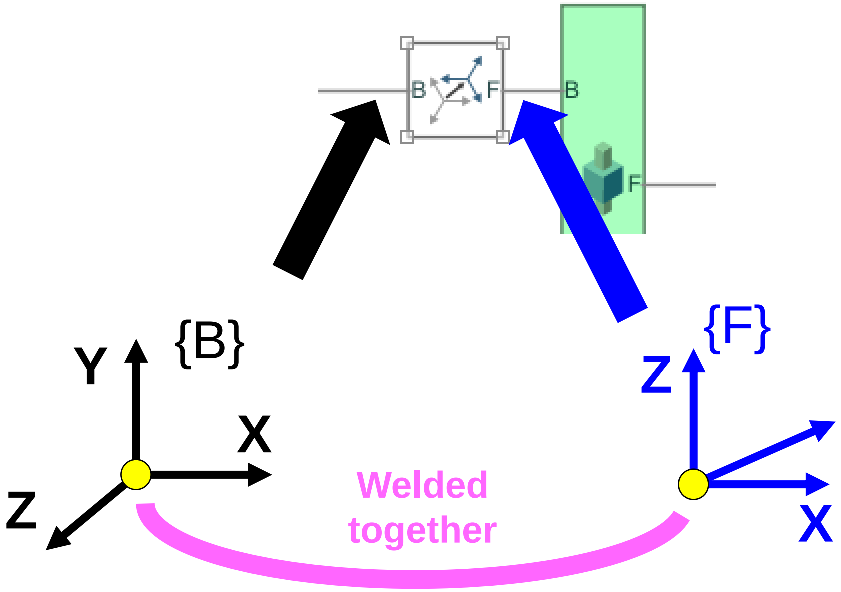




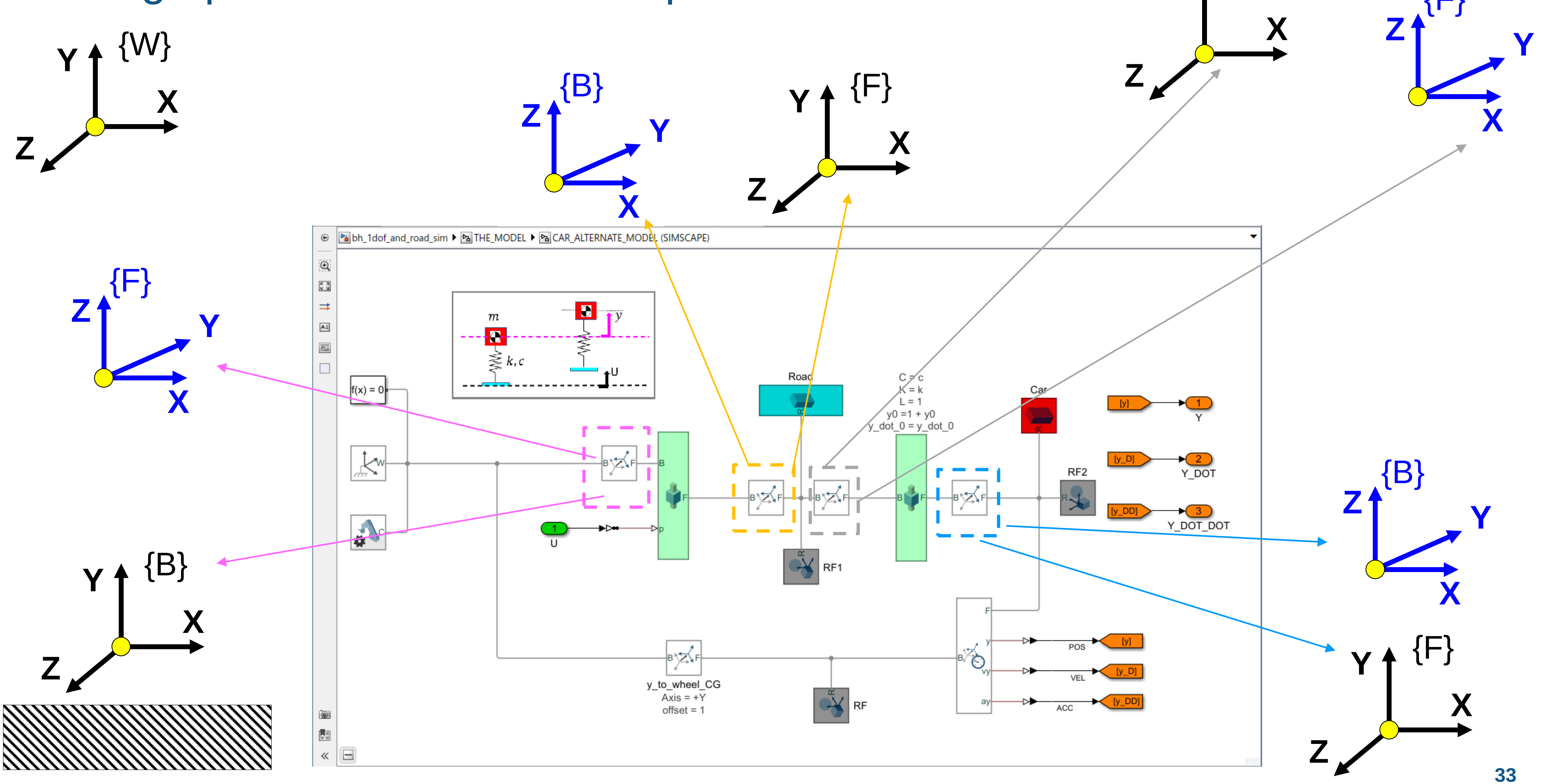
… BUT: the **Prismatic joint** offers Relative motion along "a" Z-axis.

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

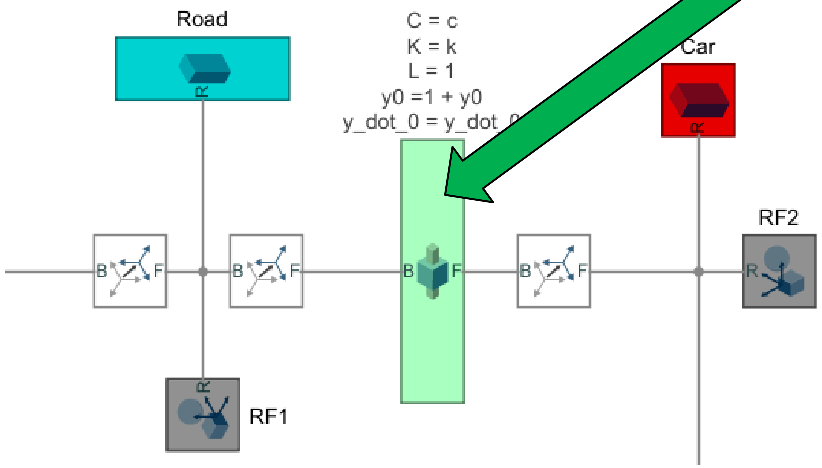
So ? – So we use the **Rigid Transform** Block to create NEW frames that are Orientated according to our needs

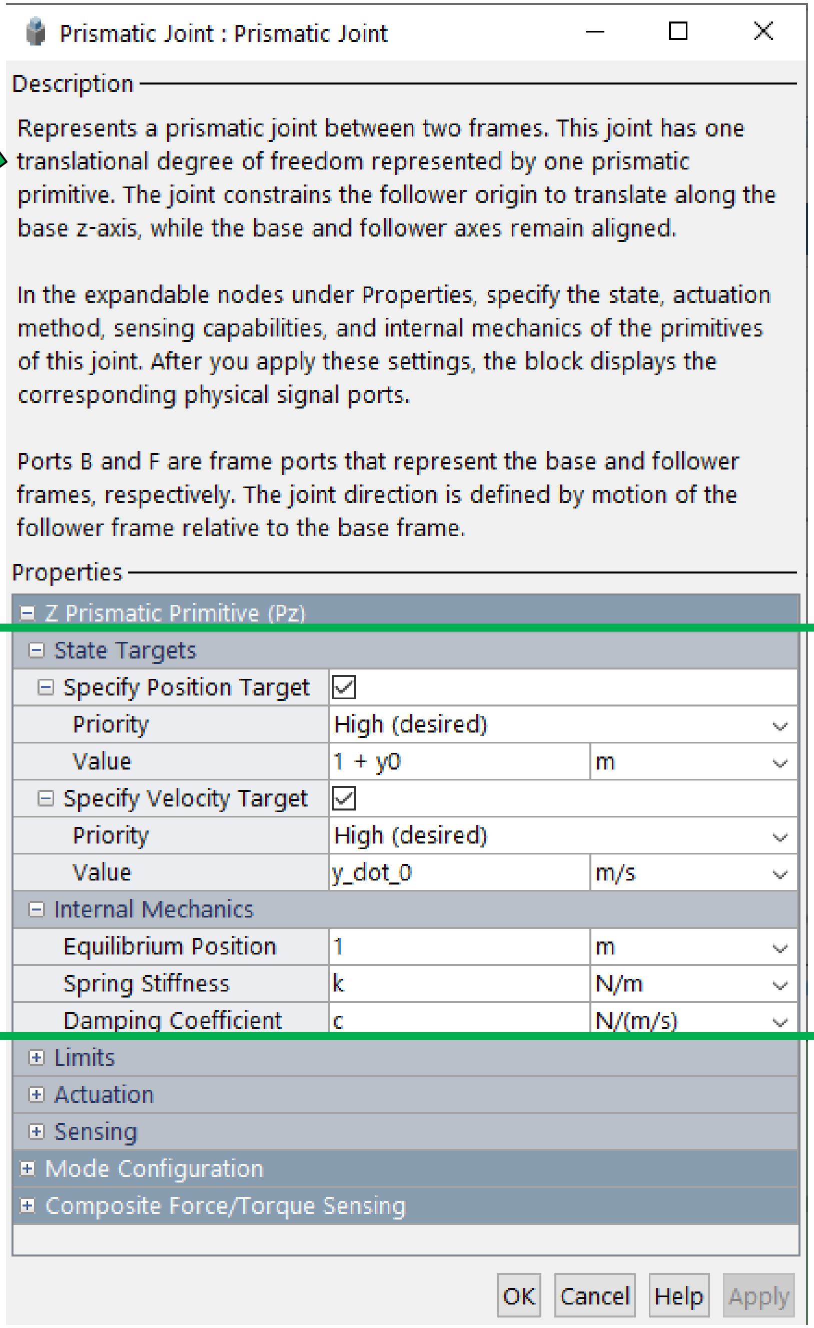


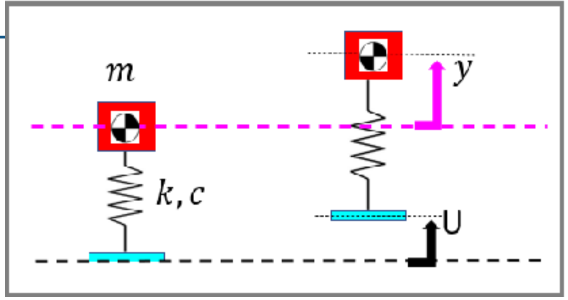
Setting up motion constraints – part 3



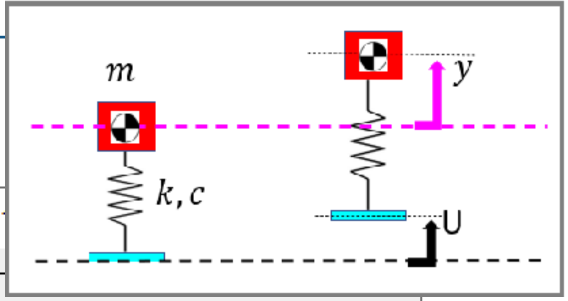
Setting up the **spring**

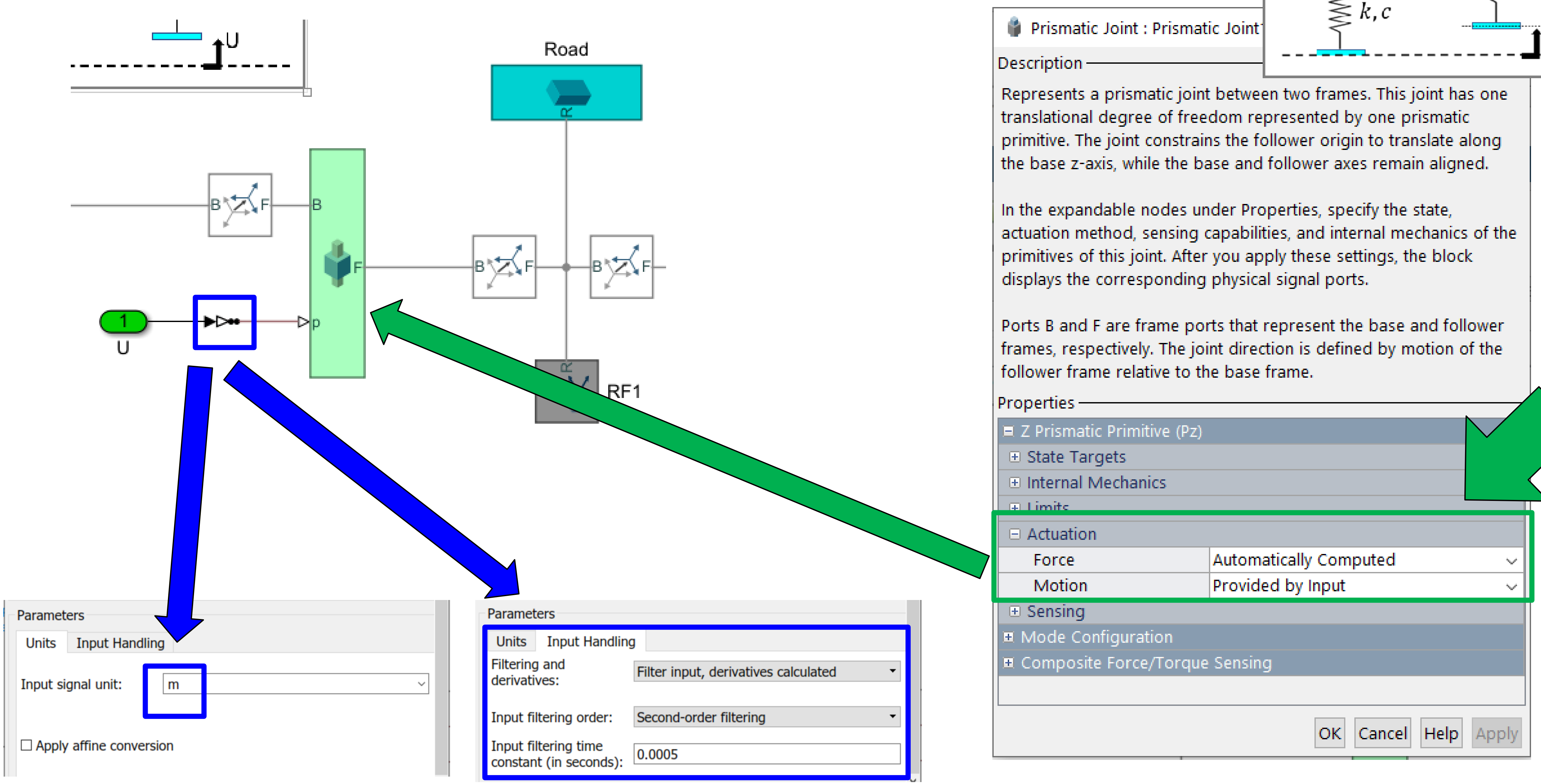




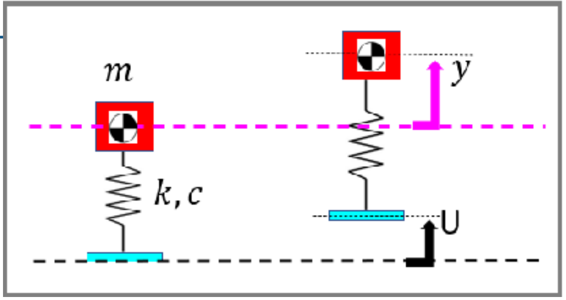


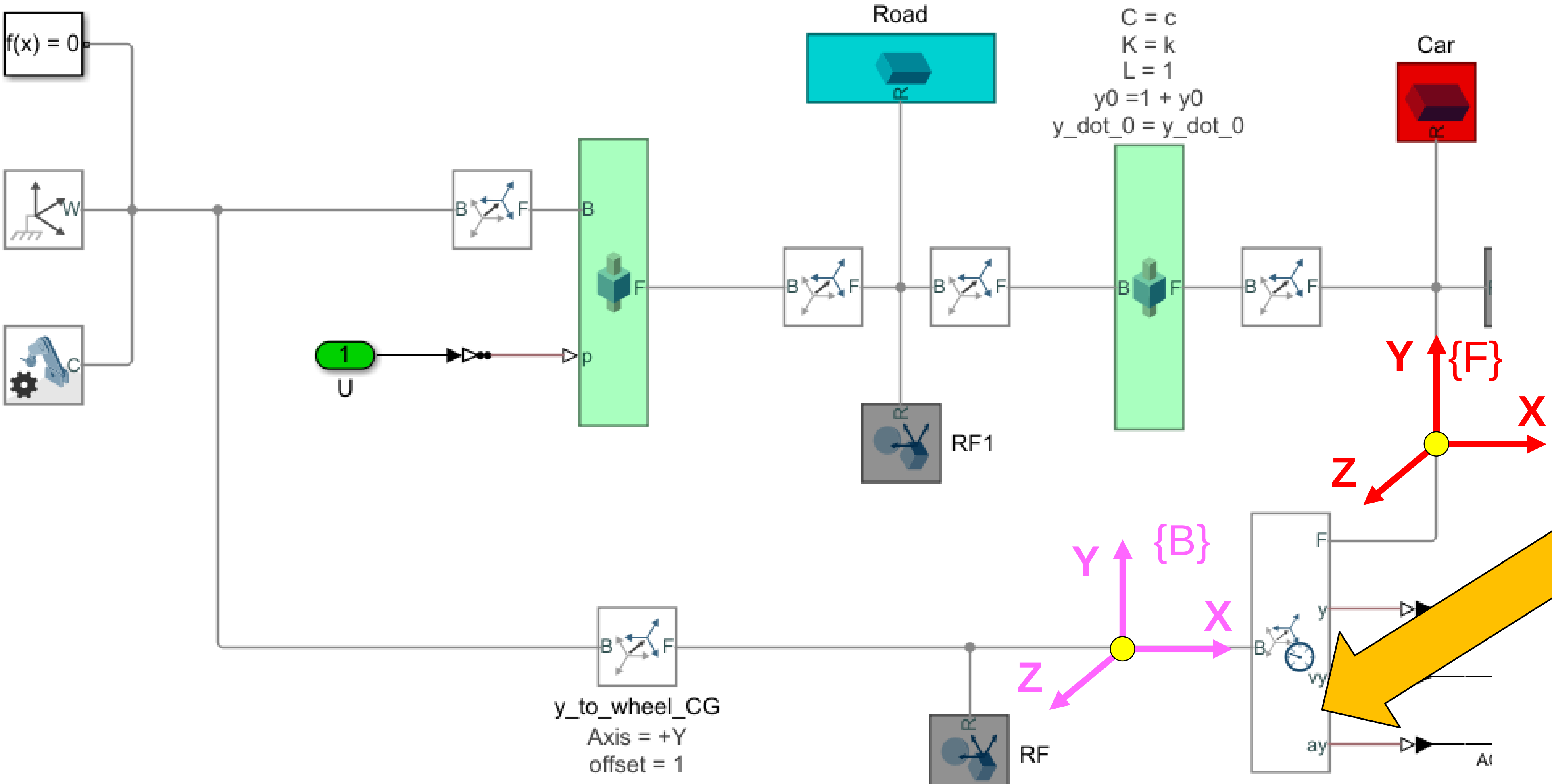
Setting up the **road displacement**





Taking **measurements**





the **Transform Sensor** block Measures the relationship between the FOLLOWER frame relative to the BASE frame