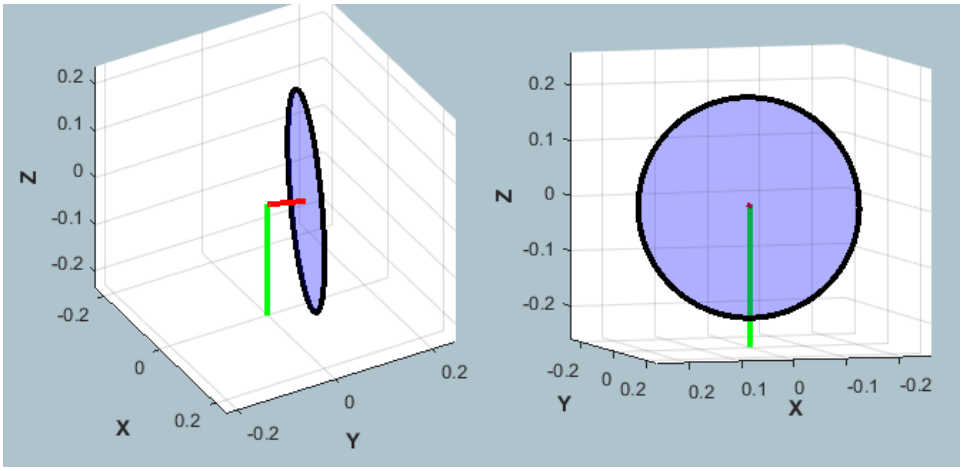
MODIFIED Euler technique:

The wheel on a pole and gyroscopic moments

Let's explore the classic mechanics experiment where we suspend a spinning disc (it's often a bicycle wheel) on a pole (it's often suspended from a rope). In our case we'll use a solid wheel, and the wheel will be connected to a support tower via a "massless" rod. The support tower allows the wheel assembly to yaw(ϕ), pitch (θ) and the wheel can of course spin (Ω).

We'll look specificall at modelling this system using the **MODIFIED** Euler equations.

The system that we'll consider is similar (but NOT identical) to the one shown in this **youtube clip** made by Prof Lewin.



Bradley Horton : 20-Sep-2017, bradley.horton@mathworks.com.au

Change History:

- 20-Sep-2019 - a local **cross()** function is defined as a SUBFUNCTION in this script. This ensures consistent behavior with R2019b

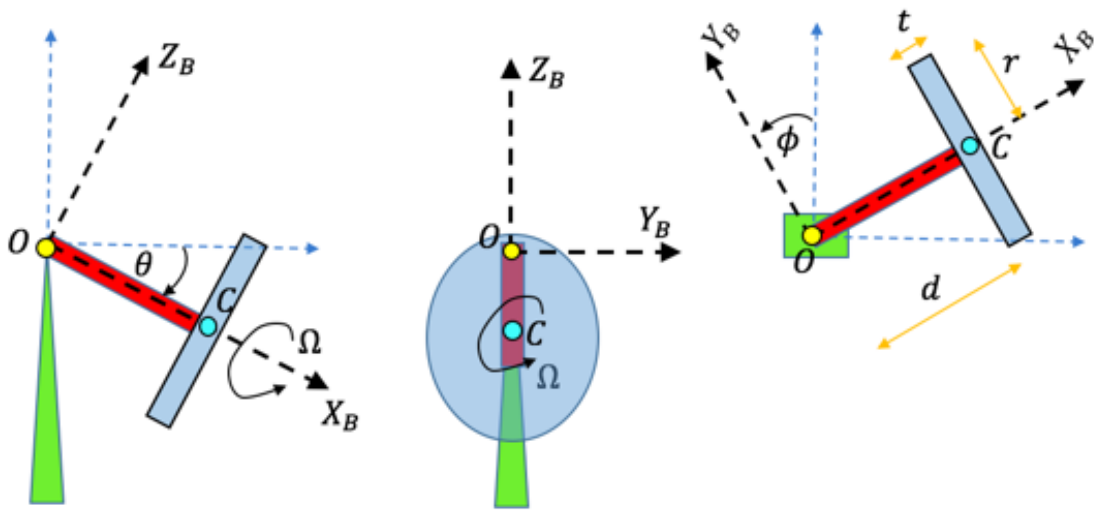
Revision:

Before looking at this case study, review the tutorial document **bh_CONCEPT_Passive_rots_Body_rates_Euler_rates.mlx.** Specifically the sections:

- Review the concept of PASSIVE rotation matrices
- An example of 2 successive PASSIVE rotations

Our system:

OK, the system that we'll consider is shown below:



In the diagram above, note the following:

1. The B-frame is attached at the Pivot point *O*.
2. The B-frame is partially attached to the body - we say "partially", because the B-frame yaws (ϕ) and pitches (θ) **with** the body ... but ... the body is free to spin relative to the ${}^{B}X$ axis.
3. Ω is the spin angular velocity of the body relative to the attached B-frame. The spin velocity occurs along the ${}^{B}X$ axis.
4. We'll use the term "yaw" and "precession" interchangeably.
5. Because of the symmetry of the wheel, the inertia matrix relative to the B-frame stays constant even though the wheel spins (Ω) relative to the B-frame.
6. As stated in the introduction, the (red) rod that attaches the wheel to the pivot point *O*, is a massless rod.

Define some model parameters:

- r : (m), wheel radius
- t : (m), wheel thickness
- m : (kg), wheel mass
- d : (m), rod length , really it's the distance from pivot point O to the CoM of the wheel

Load some parameters needed by the Simulink models - *we'll be opening these shortly*

```
bh_parameters_for_models
```

And here they are:

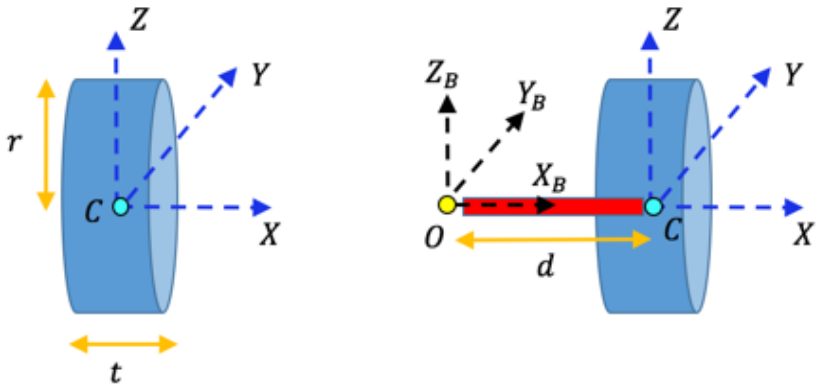
```
whos
```

```
  Name         Size            Bytes  Class          Attributes

  EME_BUS      1x1               394  Simulink.Bus
  d            1x1                 8  double
  m            1x1                 8  double
  mr           1x1                 8  double
  r            1x1                 8  double
  rr           1x1                 8  double
  t            1x1                 8  double
  tr           1x1                 8  double
```

The inertia matrix for solid cylinder:

Let's calculate the inertia matrix of the wheel. First we'll determine the inertia about a frame attached to the center of mass *C* of the wheel, then we'll apply the Parallel axis theorem to calculate the inertia about the pivot point *O*. (consult the following reference for inertias of cylinders https://en.wikipedia.org/wiki/List_of_moments_of_inertia).



Here's the moment of inertia matrix ${}^{B}_{C}I$ about the cylinder's center of mass:

```
Ic = [ (m*r^2)/2,   0,                    0;
       0,           m*(3*r^2 + t^2)/12,   0;
       0,           0,                    m*(3*r^2 + t^2)/12;
     ];

% extract the spin axis Inertia term
Ic_xx_val = Ic(1,1)
```

```
Ic_xx_val = 0.0800
```

And here's the cylinder's moment of inertia matrix ${}^{B}_{O}I$ about the pivot point "O" - note we're simply using the Parallel axis theorem:

```
dx = d;   dy = 0;   dz = 0;

Io = Ic  +  m*[  (dy^2 + dz^2),        (-dx*dy),        (-dx*dz);
                 (-dy*dx),   (dx^2 + dz^2),        (-dy*dz);
                 (-dz*dx),        (-dz*dy),   (dx^2 + dy^2);  ];
```

Note how the Inertia matrix is still (as expected) a diagonal matrix

```
% echo the Inertia matrix
Io
```

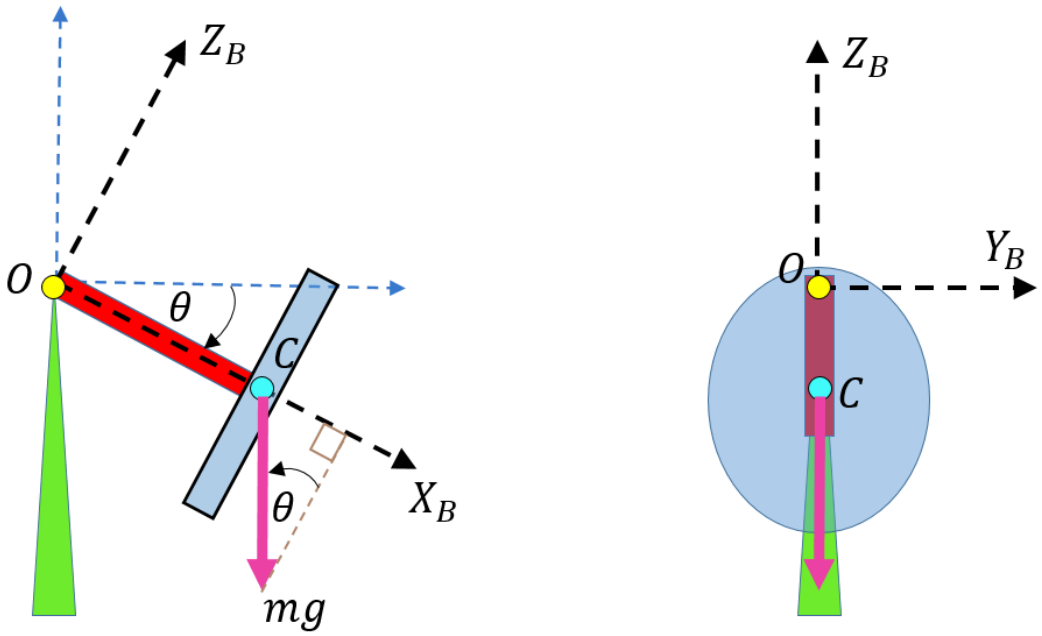
```
Io = 3×3
    0.0800         0         0
         0    0.0597         0
         0         0    0.0597
```

The gravity torque:

REMEMBER: *the {B}-frame will only* yaw(ϕ) and pitch (θ) *with the wheel. The {B}-frame will NOT roll. Instead we are defining* Ω *as the spin angular velocity of the wheel relative to the attached {B}-frame. The spin velocity occurs along the* ${}^{B}X$ *axis.*

The wheel is subjected to a TORQUE due to the wheel's mass and the offset distance "*d*" to the pivot point *O.* In components of the {B}-frame, we can describe the torque applied by gravity as the usual vector ***CROSS PRODUCT*** of the arm length and the applied force, ie:

$$ {}^{B}_{O}\tau \;=\; {}^{B}r \;\times\; {}^{B}\overrightarrow{mg} \;=\; \begin{pmatrix} d \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} mg.Sin(\theta) \\ 0 \\ -mg.Cos(\theta) \end{pmatrix} = \begin{pmatrix} 0 \\ d.mg.Cos(\theta) \\ 0 \end{pmatrix} $$



The Physics of the system:

The equations of motion for this system are defined by the **Modified Euler moment equation** (MEME) for a rigid body - Note that all terms are expressed in components of the body fixed {B}-frame:

- $ {}^{B}_{O}M - ({}^{B}_{C}I.\dot{\Omega} + \omega \times ({}^{B}_{C}I.\Omega)) = {}^{B}_{O}I.\dot{\omega} + \omega \times ({}^{B}_{O}I.\omega) $

Note that on the left hand side of the Euler moment equation we have the additional "gyroscopic" moments(or torques) caused by the wheel's spin velocity Ω.

- Remember that Ω is the spin angular velocity of the body relative to the attached B-frame. The spin velocity occurs along the ${}^{B}X$ axis.

Let's consider the ***specific case*** where (this corresponds to our "**Wheel on a pole**" system):

$\Omega = \begin{pmatrix} \Omega_x \\ 0 \\ 0 \end{pmatrix}$, $\dot{\Omega} = \begin{pmatrix} \dot{\Omega}_x \\ 0 \\ 0 \end{pmatrix}$, ${}^{B}_{C}I = \begin{pmatrix} {}^{B}_{C}I_{xx} & 0 & 0 \\ 0 & {}^{B}_{C}I_{yy} & 0 \\ 0 & 0 & {}^{B}_{C}I_{zz} \end{pmatrix}$, ${}^{B}_{O}I = \begin{pmatrix} {}^{B}_{O}I_{xx} & 0 & 0 \\ 0 & {}^{B}_{O}I_{yy} & 0 \\ 0 & 0 & {}^{B}_{O}I_{zz} \end{pmatrix}$, $\omega = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$ where ω is the angular velocity of the B-frame.

In the following section we're going to substitute the above terms into our **Modified Euler moment equation** (MEME). We'll proceed by first focusing on the LEFT hand side of the MEME, and then we'll focus on the RHS of the EME.

```
% define some symbolic variables
syms Ic_xx Ic_yy Ic_zz                    real
syms Io_xx Io_yy Io_zz                    real
syms omega_x omega_y omega_z              real
syms omega_dot_x omega_dot_y omega_dot_z  real
syms Omega_x  Omega_dot_x                 real
```

Here's our ${}^{B}_{O}I$:

```
soI          = [ Io_xx,     0,     0;
                    0,   Io_yy,     0;
                    0,     0,   Io_zz];
```

Here's our ${}^{B}_{C}I$:

```
scI          = [ Ic_xx,     0,     0;
                    0,   Ic_yy,     0;
                    0,     0,   Ic_zz];
```

Here's our ω and $\dot{\omega}$:

```
 sw          =  [omega_x;         omega_y;      omega_z];
 sw_DOT      =  [omega_dot_x;    omega_dot_y;   omega_dot_z];
```

Here's our Ω and $\dot{\Omega}$:

```
sOmega     =  [Omega_x;        0;            0;];
sOmega_DOT =  [Omega_dot_x;    0;            0;];
```

OK, let's explore the LEFT hand side of our Modified Euler equation:

- ${}^{B}_{O}M - (\ {}^{B}_{C}I.\dot{\Omega} + \omega \times ({}^{B}_{C}I.\Omega) \ ) \ = \ {}^{B}_{O}I.\dot{\omega} + \omega \times ({}^{B}_{O}I.\omega)$

First let's look at our **GYROSCOPIC torque** term

- ${}^{B}_{C}I.\dot{\Omega} + \omega \times ({}^{B}_{C}I.\Omega)$, is:

```
% PART_A and B
part_A =           scI * sOmega_DOT;
part_B = cross(sw, scI * sOmega);

the_GYRO_TQ = part_A + part_B
```

```
the_GYRO_TQ =
```

$$\begin{pmatrix} \mathrm{Ic_{xx}}\,\dot{\Omega}_x \\ \mathrm{Ic_{xx}}\,\Omega_x\,\omega_z \\ -\mathrm{Ic_{xx}}\,\Omega_x\,\omega_y \end{pmatrix}$$

So inserting these gyroscopic moment terms we can write the moment equation as:

- $${}^{B}_{O}M - \begin{pmatrix} {}^{B}_{C}I_{xx} \ . \ \dot{\Omega}_x \\ \omega_z \ . \ {}^{B}_{C}I_{xx} \ . \ \Omega_x \\ -1 \times \omega_y \ . \ {}^{B}_{C}I_{xx} \ . \ \Omega_x \end{pmatrix} \ = \ {}^{B}_{O}I.\dot{\omega} \ + \ \omega \times ({}^{B}_{O}I.\omega)$$

OK, let's explore the RIGHT hand side of our Modified Euler equation:

- ${}^{B}_{O}M - (\ {}^{B}_{C}I.\dot{\Omega} + \omega \times ({}^{B}_{C}I.\Omega) \ ) \ = \ {}^{B}_{O}I.\dot{\omega} + \omega \times ({}^{B}_{O}I.\omega)$

If we now focus on the RHS of our equation of motion, ie:

- ${}^{B}_{O}I.\dot{\omega} \ + \ \omega \times ({}^{B}_{O}I.\omega)$, we get

```
tmp_RHS = soI*sw_DOT + cross(sw, soI*sw)
```

```
tmp_RHS =
```

$$\begin{pmatrix} \mathrm{Io_{xx}}\,\dot{\omega}_x - \mathrm{Io_{yy}}\,\omega_y\,\omega_z + \mathrm{Io_{zz}}\,\omega_y\,\omega_z \\ \mathrm{Io_{yy}}\,\dot{\omega}_y + \mathrm{Io_{xx}}\,\omega_x\,\omega_z - \mathrm{Io_{zz}}\,\omega_x\,\omega_z \\ \mathrm{Io_{zz}}\,\dot{\omega}_z - \mathrm{Io_{xx}}\,\omega_x\,\omega_y + \mathrm{Io_{yy}}\,\omega_x\,\omega_y \end{pmatrix}$$

So our system moment equation can now be written as:

- $$\begin{pmatrix} {}^{B}_{O}M_x \\ {}^{B}_{O}M_y \\ {}^{B}_{O}M_z \end{pmatrix} - \begin{pmatrix} {}^{B}_{C}I_{xx} \;.\; \dot{\Omega}_x \\ \omega_z \;.\; {}^{B}_{C}I_{xx} \;.\; \Omega_x \\ -1 \times \omega_y \;.\; {}^{B}_{C}I_{xx} \;.\; \Omega_x \end{pmatrix} = \begin{pmatrix} {}^{B}_{O}I_{xx}.\dot{\omega_x} + \omega_y\omega_z.({}^{B}_{O}I_{zz} - {}^{B}_{O}I_{yy}) \\ {}^{B}_{O}I_{yy}.\dot{\omega_y} + \omega_x\omega_z.({}^{B}_{O}I_{xx} - {}^{B}_{O}I_{zz}) \\ {}^{B}_{O}I_{zz}.\dot{\omega_z} + \omega_x\omega_y.({}^{B}_{O}I_{yy} - {}^{B}_{O}I_{xx}) \end{pmatrix}$$

So or Modified Euler equation finally becomes:

- $${}^{B}_{O}M - (\; {}^{B}_{C}I.\dot{\Omega} + \omega \times ({}^{B}_{C}I.\Omega) \;) \;=\; {}^{B}_{O}I.\dot{\omega} + \omega \times ({}^{B}_{O}I.\omega)$$

```
syms Mb_x  Mb_y  Mb_z      real

Mb = [Mb_x;  Mb_y;  Mb_z];

EQ = (Mb - the_GYRO_TQ) == (tmp_RHS)
```

```
EQ =
```

$$\begin{pmatrix} \mathrm{Mb}_x - \mathrm{Ic}_{xx}\,\dot{\Omega}_x = \mathrm{Io}_{xx}\,\dot{\omega}_x - \mathrm{Io}_{yy}\,\omega_y\,\omega_z + \mathrm{Io}_{zz}\,\omega_y\,\omega_z \\ \mathrm{Mb}_y - \mathrm{Ic}_{xx}\,\Omega_x\,\omega_z = \mathrm{Io}_{yy}\,\dot{\omega}_y + \mathrm{Io}_{xx}\,\omega_x\,\omega_z - \mathrm{Io}_{zz}\,\omega_x\,\omega_z \\ \mathrm{Mb}_z + \mathrm{Ic}_{xx}\,\Omega_x\,\omega_y = \mathrm{Io}_{zz}\,\dot{\omega}_z - \mathrm{Io}_{xx}\,\omega_x\,\omega_y + \mathrm{Io}_{yy}\,\omega_x\,\omega_y \end{pmatrix}$$

And recall the original symbol list

```
syms Ic_xx Ic_yy Ic_zz
syms Io_xx Io_yy Io_zz
syms omega_x omega_y omega_z
syms omega_dot_x omega_dot_y omega_dot_z
syms Omega_x  Omega_dot_x
```

Substitute Euler Angles into these ODEs:

See **bh_CONCEPT_Passive_rots_Body_rates_Euler_rates.mlx** for a derivation on the relationship between Euler rates and body frame angular velocity, specifically the section:

- An example of 2 successive PASSIVE rotations

Recall that our attached B-frame will yaw(ϕ) and pitch (θ) with the wheel ..... BUT the wheel spins relative to the B-frame with an angular velocity of spin (Ω). Because only 2 Euler angles describe our B-frame motion, we have:

- $$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} -\dot{\phi}.\sin(\theta) \\ \dot{\theta} \\ \dot{\phi}.\cos(\theta) \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} -\ddot{\phi}.\sin(\theta) \;+\; -\dot{\phi}.\dot{\theta}.\cos(\theta) \\ \ddot{\theta} \\ \ddot{\phi}.\cos(\theta) \;-\dot{\phi}.\dot{\theta}.\sin(\theta) \end{bmatrix}$$

So let's substitute these relationships into our Modified Euler equation:

```
syms  phi          theta          real
syms  phi_dot      theta_dot      real
```

```
syms  phi_ddot     theta_ddot      real

my_wx     = -1*phi_dot*sin(theta);
my_wy     = theta_dot;
my_wz     = phi_dot*cos(theta);

my_wx_dot = -1*phi_ddot*sin(theta) +  -1*phi_dot*theta_dot*cos(theta);
my_wy_dot = theta_ddot;
my_wz_dot = phi_ddot*cos(theta)  + -1*phi_dot*theta_dot*sin(theta);

EQ = subs(EQ, [omega_dot_x omega_dot_y omega_dot_z], [my_wx_dot, my_wy_dot, my_wz_dot]);

EQ = subs(EQ, [omega_x omega_y omega_z], [my_wx, my_wy, my_wz])
```

```
EQ =
```

$$\begin{pmatrix} \mathrm{Mb}_x - \mathrm{Ic}_{xx}\,\dot{\Omega}_x = \mathrm{Io}_{zz}\,\dot{\phi}\,\dot{\theta}\cos(\theta) - \mathrm{Io}_{yy}\,\dot{\phi}\,\dot{\theta}\cos(\theta) - \mathrm{Io}_{xx}\left(\ddot{\phi}\sin(\theta) + \dot{\phi}\,\dot{\theta}\cos(\theta)\right) \\ \mathrm{Mb}_y - \mathrm{Ic}_{xx}\,\Omega_x\,\dot{\phi}\cos(\theta) = \mathrm{Io}_{yy}\,\ddot{\theta} - \mathrm{Io}_{xx}\,\dot{\phi}^2\cos(\theta)\sin(\theta) + \mathrm{Io}_{zz}\,\dot{\phi}^2\cos(\theta)\sin(\theta) \\ \mathrm{Mb}_z + \mathrm{Ic}_{xx}\,\Omega_x\,\dot{\theta} = \mathrm{Io}_{zz}\left(\ddot{\phi}\cos(\theta) - \dot{\phi}\,\dot{\theta}\sin(\theta)\right) + \mathrm{Io}_{xx}\,\dot{\phi}\,\dot{\theta}\sin(\theta) - \mathrm{Io}_{yy}\,\dot{\phi}\,\dot{\theta}\sin(\theta) \end{pmatrix}$$

Next, let's solve for:

- $\ddot{\phi}$ and $\ddot{\theta}$ and $\dot{\Omega}$

```
my_sol = solve(EQ, [phi_ddot, theta_ddot, Omega_dot_x])
```

```
Warning: Solutions are only valid under certain conditions. To include parameters and conditions in the
 solution, specify the 'ReturnConditions' value as 'true'.
my_sol = struct with fields:
       phi_ddot: [1×1 sym]
     theta_ddot: [1×1 sym]
    Omega_dot_x: [1×1 sym]
```

Convert the ODEs into a MATLAB function block:

Should I make a MATLAB function block

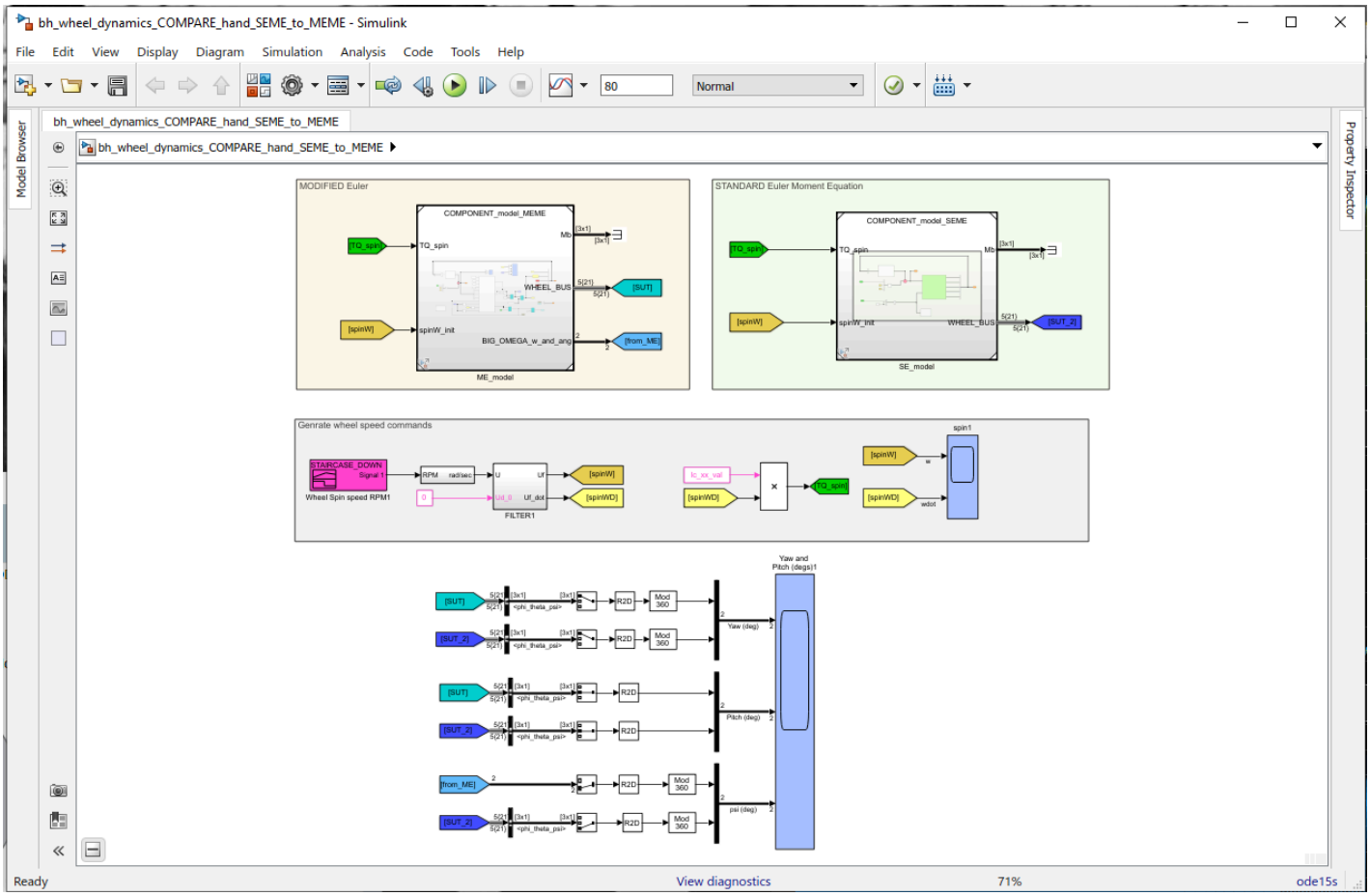
```
bh_tf_make_MLFblock = true;
```

OK then do it:

```
if(true == bh_tf_make_MLFblock)
    my_block = 'bh_tmp_model/tmp_block';
    new_system('bh_tmp_model');
    open_system('bh_tmp_model')
    matlabFunctionBlock(my_block, ...
                        my_sol.phi_ddot, my_sol.theta_ddot, my_sol.Omega_dot_x, ...
                        'Outputs', {'phi_DOT_DOT','theta_DOT_DOT', 'BIG_Omega_DOT'} ...
                )
end
```

The Simulink model:

Our system equation of motion is a beautiful thing ..... but how do we make this mathematics come alive? How do we get confidence that what we see in the real world (... and on youtube) is explained by our maths ? Well, the easiest way to solve our moment equations is to model/represent these equations in Simulink. Open and explore the following Simulink model **<bh_wheel_dynamics_COMPARE_hand_SEME_to_MEME>** .

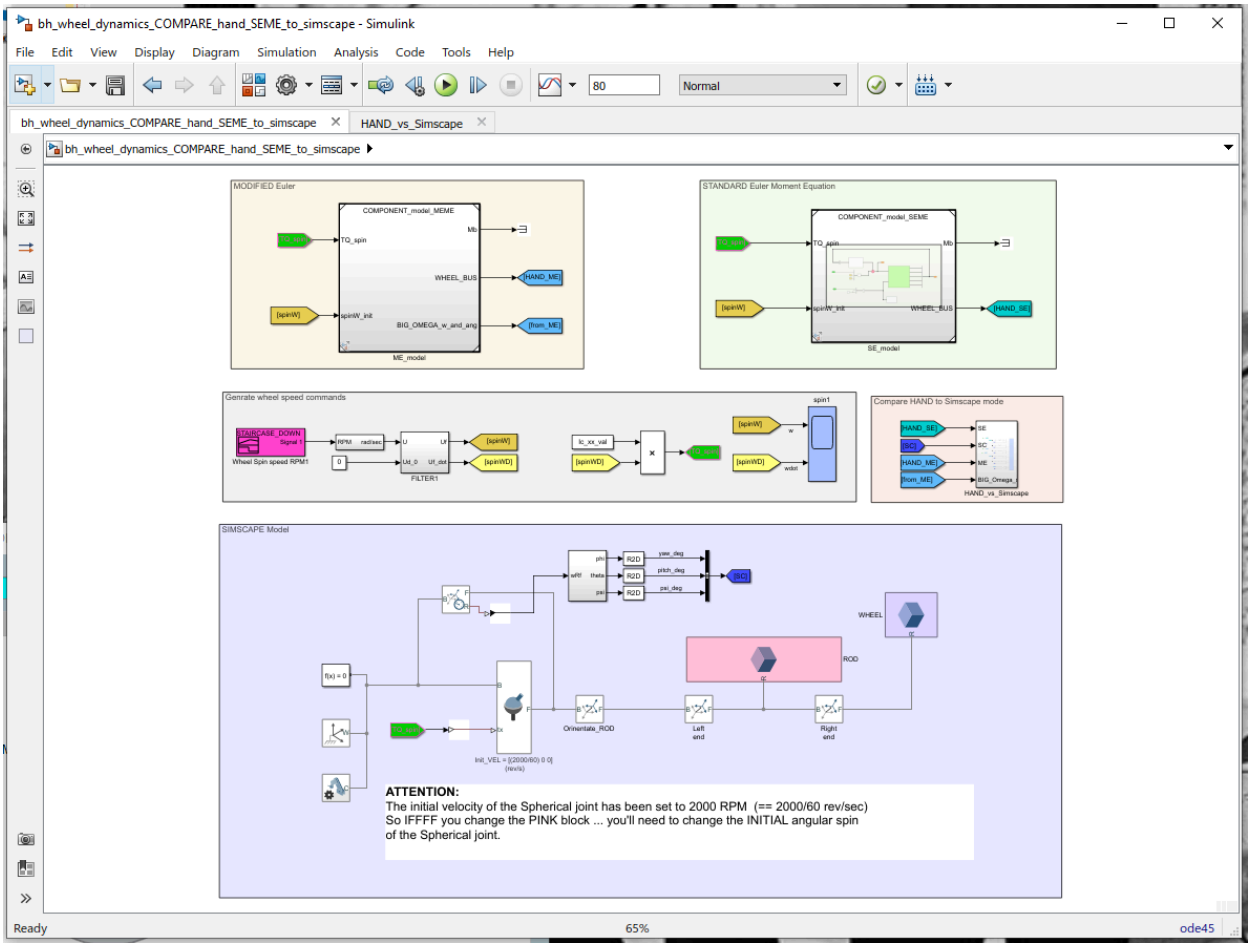


Open and run the Simulink model:

```
% open_system('bh_wheel_dynamics_COMPARE_hand_SEME_to_MEME')
```

APPENDIX_A: Here's another way to model this system.

Review the model called <**bh_wheel_dynamics_COMPARE_hand_SEME_to_simscape**>. Here we've modelled the same "wheel on a pole" system ... but we've used Simscape to do it. Simscape uses model building elements such as "joints, bodies, sensors": And it has a great visualization tool called *Mechanics Explorer* to help you view your assembled machine.



Local functions only beyond this point:

The vector cross product

```
function c = cross(a,b)

    c1 = a(2).*b(3)-a(3).*b(2);
    c2 = a(3).*b(1)-a(1).*b(3);
    c3 = a(1).*b(2)-a(2).*b(1);

    c  = [c1;c2;c3];

end
```