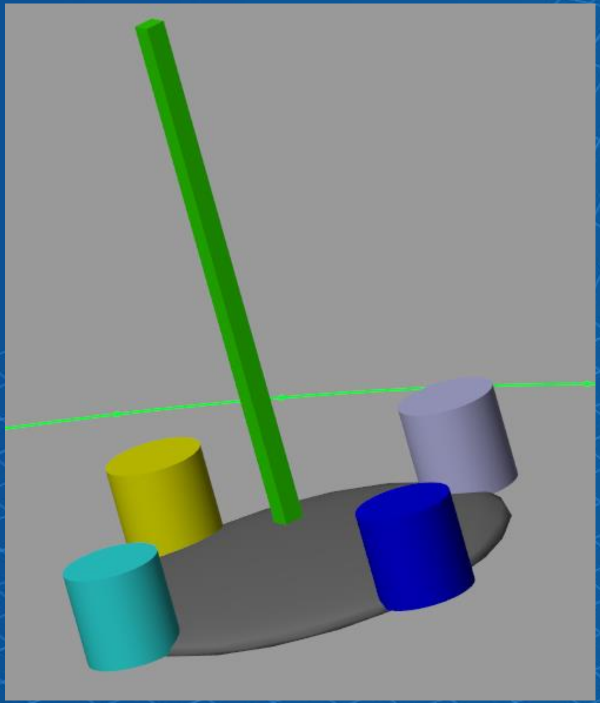
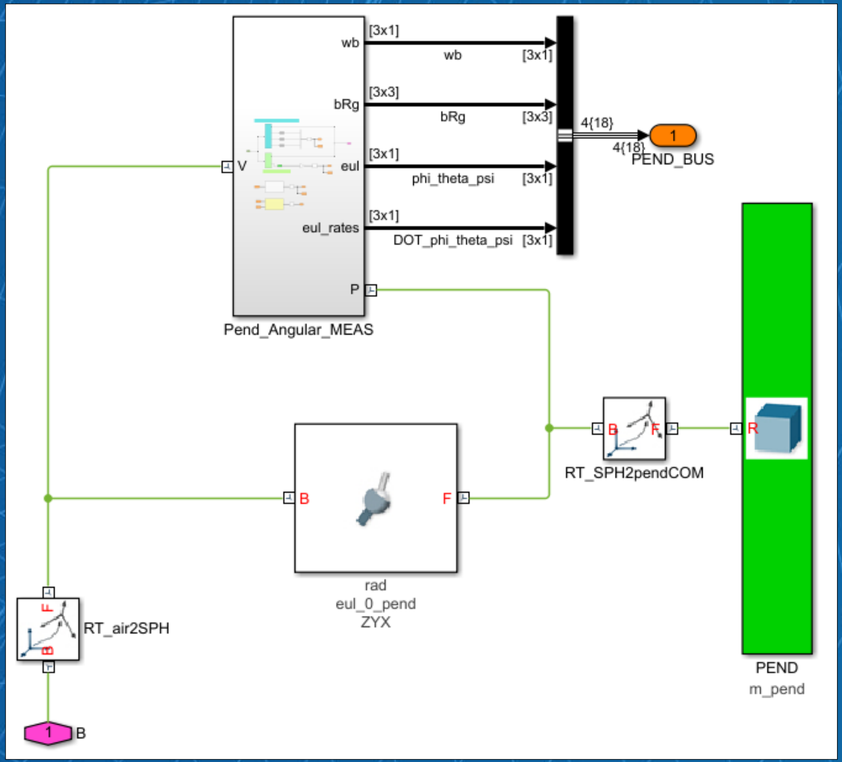
A quadcopter balancing pendulum:

- *How to model and control it*





$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k$$

$$Q_k = \sum_{i=1}^{Nf_{nc}} \left(\vec{F}_i \cdot \frac{\partial \vec{v}_i}{\partial \dot{q}_k} \right) + \sum_{j=1}^{N\tau_{nc}} \left(\vec{\tau}_j \cdot \frac{\partial \vec{\omega}_j}{\partial \dot{q}_k} \right)$$

Brad Horton
Engineer
MathWorks

How do you nurture the **CONFIDENCE** of students ?

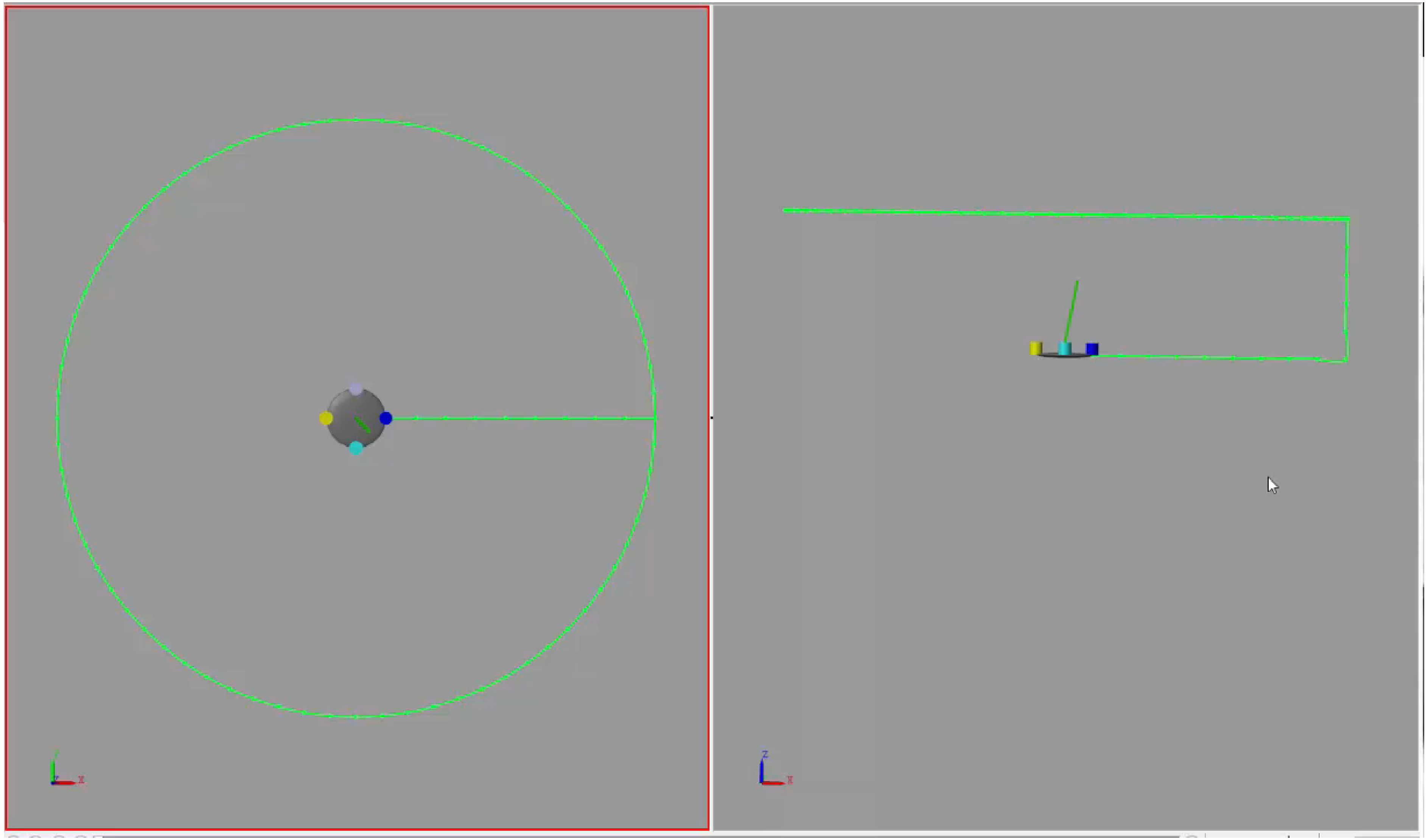
$$\begin{bmatrix}u\\v\\z\end{bmatrix}_L = \begin{bmatrix}\cos(\phi) & \sin(\phi) & 0\\ -\sin(\phi) & \cos(\phi) & 0\\ 0 & 0 & 1\end{bmatrix} . \begin{bmatrix}u\\v\\z\end{bmatrix}_G$$

$$\omega \times r$$

$${}^BF = m.\left({}^B_{-}\dot{v}_C + {}^B_G\omega_B \times {}^B_G v_C \right)$$

$${}^BM = {}^BI \,.\, {}^B_{-}\dot{\omega}_B + {}^B_G\omega_B \times \left({}^BI \,.\, {}^B_G\omega_B \right)$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k$$



Todays agenda:

Phase 1

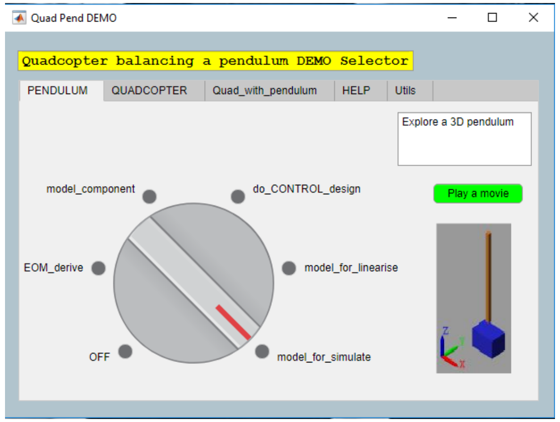
- One of the challenges in Learning Rigid Body Dynamics.
- Computational Thinking – ***Is this the answer ?***

Phase 2

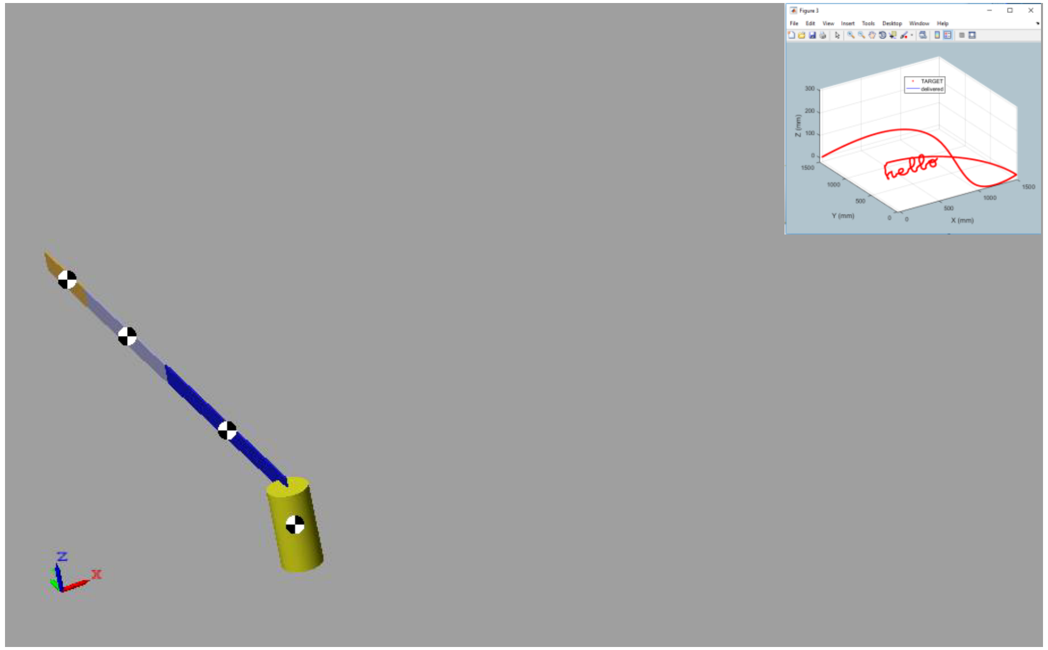
- Applying Computational Thinking
  - **4 Case Studies**

Phase 3

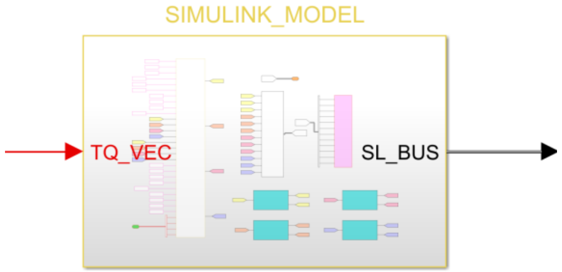
- Questions AND Answers
- How do you get ALL of the examples that you saw today ?



How do you make a rigid body machine move the way you want it to ?



Mathematical model



We need to understand the physics.

Interesting part

We need to apply Lagrange's equation

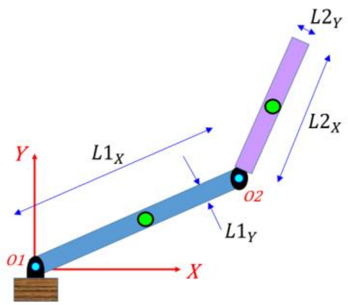
Laborious part

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k$$

$$Q_k = \sum_{i=1}^{Nf_{nc}}\left(\vec{F}_i \, . \, \frac{\partial \vec{v}_i}{\partial \dot{q}_k}\right) + \sum_{j=1}^{N\tau_{nc}}\left(\vec{\tau}_j \, . \, \frac{\partial \vec{\omega}_j}{\partial \dot{q}_k}\right)$$

$$M(q).\ddot{q} + C(\dot{q},q).\dot{q} + K(q).q + g(q) = Q$$

$$\ddot{q} = [M(q)]^{-1}.[Q - C(\dot{q},q).\dot{q} - K(q).q - g(q)]$$



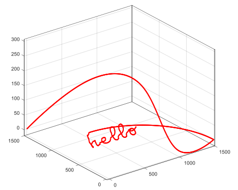
Laborious ?

```
bh_tmp_EOM_file_WILL_BE_DELETED.txt
################################################################
### q = TH1_s
###
### LHS of EOM is:
###
    1        I1G_s*TH1_s_DD
    2   +    I2G_s*TH1_s_DD
    3   +    I2G_s*TH2_s_DD
    4   +    (L1X_s^2*TH1_s_DD*m1_s)/4
    5   +    L1X_s^2*TH1_s_DD*m2_s
    6   +    (L2X_s^2*TH1_s_DD*m2_s)/4
    7   +    (L2X_s^2*TH2_s_DD*m2_s)/4
    8   +    (L1X_s*g_s*m1_s*cos(TH1_s))/2
    9   +    L1X_s*g_s*m2_s*cos(TH1_s)
   10   +    (L2X_s*g_s*m2_s*cos(TH1_s + TH2_s))/2
   11   +    L1X_s*L2X_s*TH1_s_DD*m2_s*cos(TH2_s)
   12   +    (L1X_s*L2X_s*TH2_s_DD*m2_s*cos(TH2_s))/2
   13   +    -(L1X_s*L2X_s*TH2_s_D^2*m2_s*sin(TH2_s))/2
   14   +    -L1X_s*L2X_s*TH1_s_D*TH2_s_D*m2_s*sin(TH2_s)
###
### RHS of EOM is:
    1        Q1_s
################################################################
### q = TH2_s
###
### LHS of EOM is:
###
    1        I2G_s*TH1_s_DD
    2   +    I2G_s*TH2_s_DD
    3   +    (L2X_s^2*TH1_s_DD*m2_s)/4
    4   +    (L2X_s^2*TH2_s_DD*m2_s)/4
    5   +    (L2X_s*g_s*m2_s*cos(TH1_s + TH2_s))/2
    6   +    (L1X_s*L2X_s*TH1_s_DD*m2_s*cos(TH2_s))/2
    7   +    (L1X_s*L2X_s*TH1_s_D^2*m2_s*sin(TH2_s))/2
###
### RHS of EOM is:
    1        Q2_s
```

2-dof

Approx 20 lines

$$\left.\begin{matrix}\ddot{\theta}_1\\ \\ \ddot{\theta}_2\end{matrix}\right\}$$

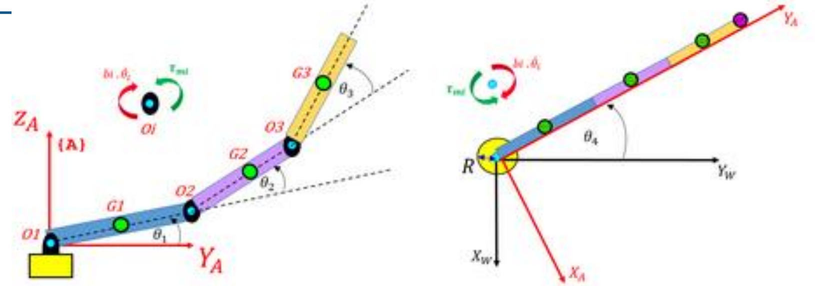


4-dof

Approx 200 lines

$$\left\{\begin{matrix}\ddot{\theta}_1\\ \\ \ddot{\theta}_2\\ \\ \ddot{\theta}_3\\ \\ \ddot{\theta}_4\end{matrix}\right.$$

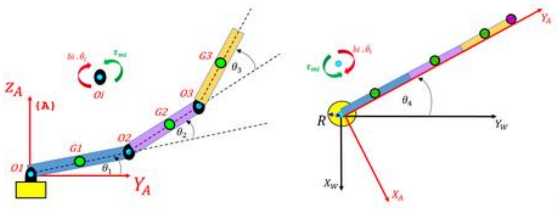
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k$$




```
bh_tmp_EOM_file_WILL_BE_DELETED.txt
################################################################
### q = TH1_s
###
### LHS of EOM is:
###
    1        (L1Y_s^2*TH1_s_DD*m1_s)/3
    2   +    L1Y_s^2*TH1_s_DD*m2_s
    3   +    L1Y_s^2*TH1_s_DD*m3_s
    4   +    (L2Y_s^2*TH1_s_DD*m2_s)/3
    5   +    L2Y_s^2*TH1_s_DD*m3_s
    6   +    (L2Y_s^2*TH2_s_DD*m2_s)/3
    7   +    L2Y_s^2*TH2_s_DD*m3_s
    8   +    (L3Y_s^2*TH1_s_DD*m3_s)/3
    9   +    (L3Y_s^2*TH2_s_DD*m3_s)/3
   10   +    (L3Y_s^2*TH3_s_DD*m3_s)/3
   11   +    (L1Z_s^2*TH1_s_DD*m1_s)/12
   12   +    (L2Z_s^2*TH1_s_DD*m2_s)/12
   13   +    (L2Z_s^2*TH2_s_DD*m2_s)/12
   14   +    (L3Z_s^2*TH1_s_DD*m3_s)/12
   15   +    (L3Z_s^2*TH2_s_DD*m3_s)/12
   16   +    (L3Z_s^2*TH3_s_DD*m3_s)/12
   17   +    (L3Y_s^2*TH4_s_D^2*m3_s*sin(2*TH1_s + 2*TH2_s + 2*TH3_s))/6
   18   +    -(L3Z_s^2*TH4_s_D^2*m3_s*sin(2*TH1_s + 2*TH2_s + 2*TH3_s))/24

   49   +    -(L1Y_s*L3Y_s*TH3_s_D*TH4_s_D*m3_s*sin(2*TH1_s + TH2_s + TH3_s))/2
   50   +    -(L1Y_s*L3Y_s*TH2_s_D*TH4_s_D*m3_s*sin(TH2_s + TH3_s))/2
   51   +    -(L1Y_s*L3Y_s*TH3_s_D*TH4_s_D*m3_s*sin(TH2_s + TH3_s))/2
   52   +    -L2Y_s*L3Y_s*TH1_s_D*TH4_s_D*m3_s*sin(2*TH1_s + 2*TH2_s + TH3_s)
   53   +    -L2Y_s*L3Y_s*TH2_s_D*TH4_s_D*m3_s*sin(2*TH1_s + 2*TH2_s + TH3_s)
   54   +    -(L2Y_s*L3Y_s*TH3_s_D*TH4_s_D*m3_s*sin(2*TH1_s + 2*TH2_s + TH3_s))/2
   55   +    -(L1Y_s*L2Y_s*TH2_s_D*TH4_s_D*m2_s*sin(TH2_s))/2
   56   +    -L1Y_s*L2Y_s*TH2_s_D*TH4_s_D*m3_s*sin(TH2_s)
   57   +    -(L2Y_s*L3Y_s*TH3_s_D*TH4_s_D*m3_s*sin(TH3_s))/2
   58   +    -L1Y_s*L2Y_s*TH1_s_D*TH4_s_D*m2_s*sin(2*TH1_s + TH2_s)
   59   +    -2*L1Y_s*L2Y_s*TH1_s_D*TH4_s_D*m3_s*sin(2*TH1_s + TH2_s)
   60   +    -(L1Y_s*L2Y_s*TH2_s_D*TH4_s_D*m2_s*sin(2*TH1_s + TH2_s))/2
   61   +    -L1Y_s*L2Y_s*TH2_s_D*TH4_s_D*m3_s*sin(2*TH1_s + TH2_s)
###
### RHS of EOM is:
    1        Q4_s
```

Encouraging Deeper Learning engagements in your classroom:



Bigger problems

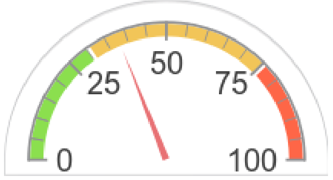
The understanding of the problem physics:

- *3D motion*
- *Inertia matrix*
- *Passive Rotations*
- *Vector sum of angular velocities*

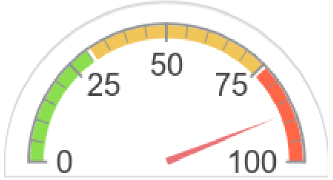
Problem Solving and practice

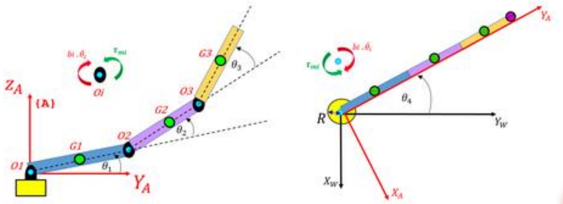
Hand written implementation

BRAIN
Conceptual Difficulty



HAND
Computational Difficulty





Bigger problems

The understanding of the problem physics:

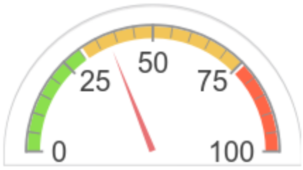
- *3D motion*
- *Inertia matrix*
- *Passive Rotations*
- *Vector sum of angular velocities*

Problem Solving and practice

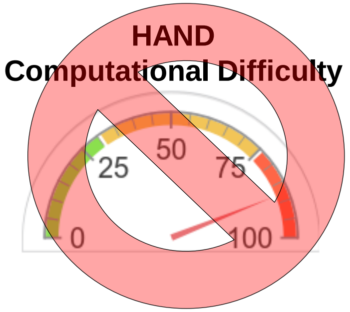
Computational Thinking:

- Brain
- Technology

BRAIN
Conceptual Difficulty



HAND
Computational Difficulty



Enabling Computational Thinking using MATLAB

Problem Solving and practice

Computational Thinking:

- Brain
- Technology

Decomposition

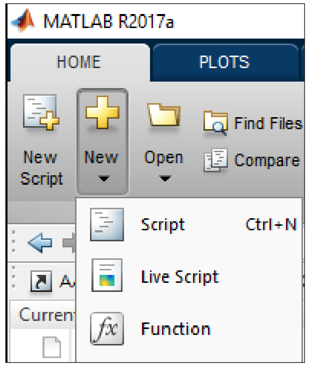
Algorithms + Automation

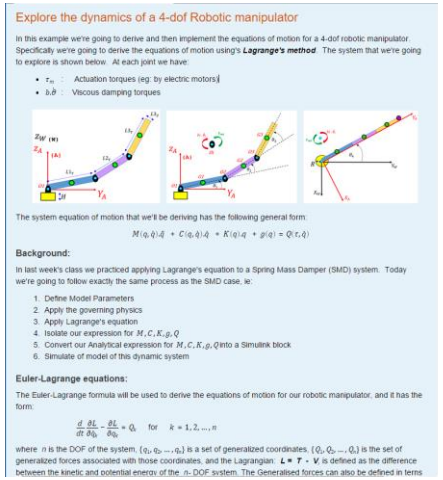
Simulation

?

Decomposition

Live Script





Explore the dynamics of a 4-dof Robotic manipulator

In this example we're going to derive and then implement the equations of motion for a 4-dof robotic manipulator. Specifically we're going to derive the equations of motion using's ***Lagrange's method***. The system that we're going to explore is shown below. At each joint we have:

- τ_m : Actuation torques (eg: by electric motors)
- $b.\dot{\theta}$: Viscous damping torques

The system equation of motion that we'll be deriving has the following general form:

$$M(q,\dot{q}).\ddot{q} + C(q,\dot{q}).\dot{q} + K(q).q + g(q) = Q(\tau,\dot{q})$$

Background:

In last week's class we practiced applying Lagrange's equation to a Spring Mass Damper (SMD) system. Today we're going to follow exactly the same process as the SMD case, ie:

1. Define Model Parameters
2. Apply the governing physics
3. Apply Lagrange's equation
4. Isolate our expression for M, C, K, g, Q
5. Convert our Analytical expression for M, C, K, g, Q into a Simulink block
6. Simulate of model of this dynamic system

Euler-Lagrange equations:

The Euler-Lagrange formula will be used to derive the equations of motion for our robotic manipulator, and it has the form:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k \quad \text{for} \quad k = 1, 2, \ldots, n$$

where n is the DOF of the system, $\{q_1, q_2, \ldots, q_n\}$ is a set of generalized coordinates, $\{Q_1, Q_2, \ldots, Q_n\}$ is the set of generalized forces associated with those coordinates, and the Lagrangian: $L = T - V$, is defined as the difference between the kinetic and potential energy of the n- DOF system. The Generalised forces can also be defined in terms

Algorithms + Automation

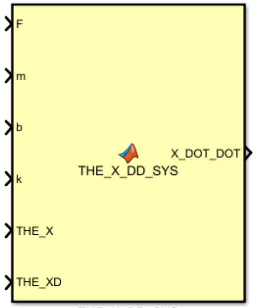
Symbolic Computing

```
>> diff()

>> matlabFunctionBlock()
```

our_EOM(t) =

$$m\frac{\partial^2}{\partial t^2}x(t) + k\,x(t) = F - b\frac{\partial}{\partial t}x(t)$$

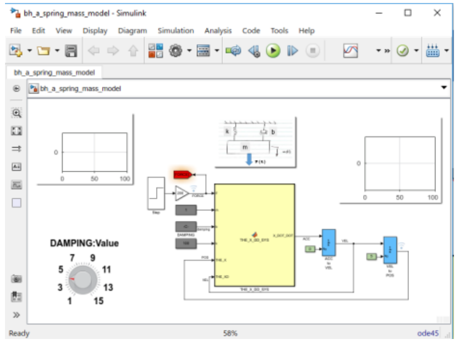


THE_X_DD_SYS

Simulation

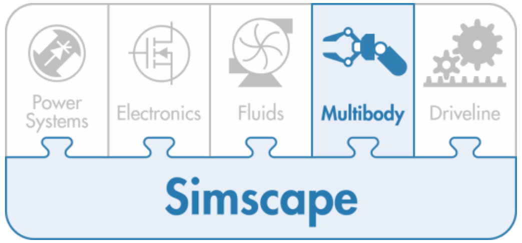
Numeric via Block Diagram

SIMULINK



Simulation (again)

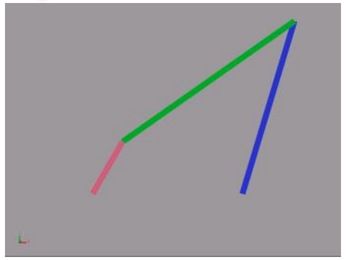
Compare against GOLDEN Reference



Validate

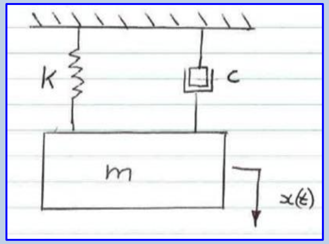
Hand Derivation vs Simscape

Visualization

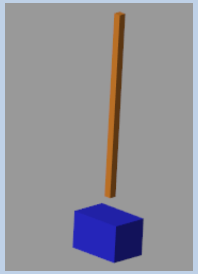


Today's case studies:

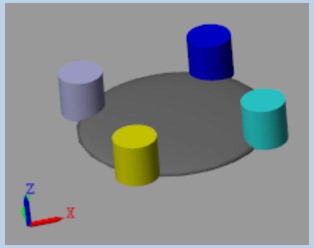
1-dof
(Hand derivation workflow)



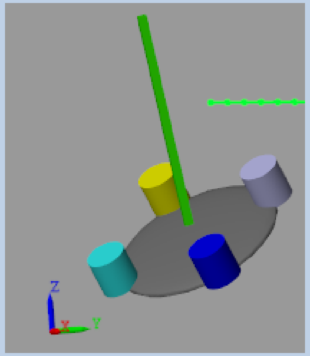
3D inverted Pendulum
(Simscape Multibody)



Quadcopter
(LQI control design)

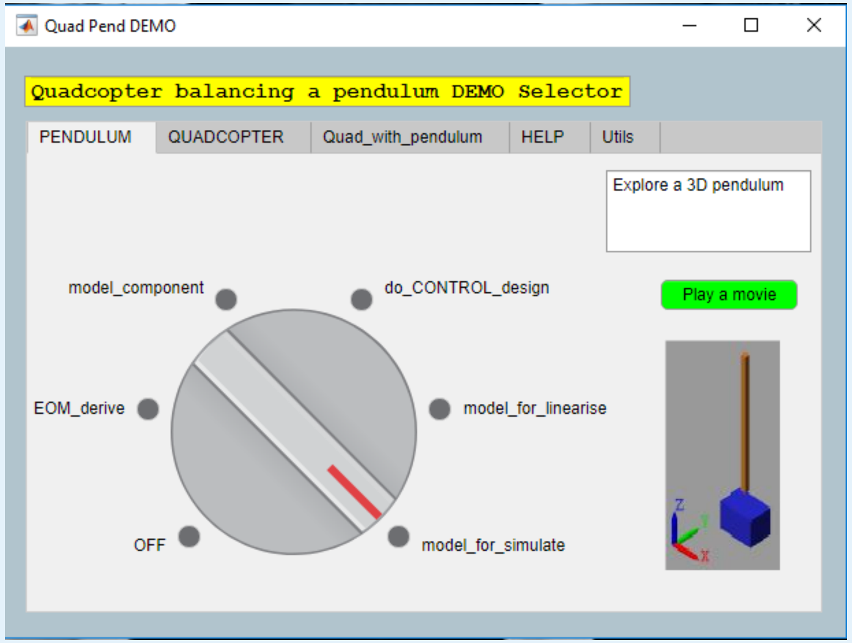


Quadcopter balancing an inverted pendulum



- A partnership that scales
  - Same workflow for small and BIG problems
- Divide and Conquer
  - Capture rationale and implementation
- The Modelling choices
  - Symbolic, Numeric, Block Diagram, Simscape
- The opportunities to explore and Discover
  - Visualization
  - Simulation

Demo these concepts

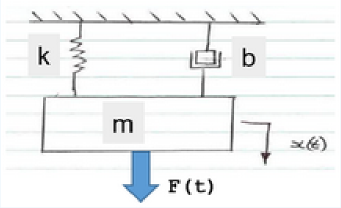


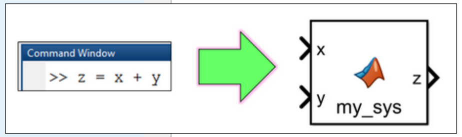
Task: Spring Mass Damper

bh_smd_model_derivation.mlx

Explore the dynamics of a 1-dof Spring Mass Damper

In this example we're going to derive and then implement the equations of motion for 1-dof Spring Mass Damper system. Specifically we're going to derive the equations of motion using's ***Lagrange's method***. The system that we're going to explore is shown below.





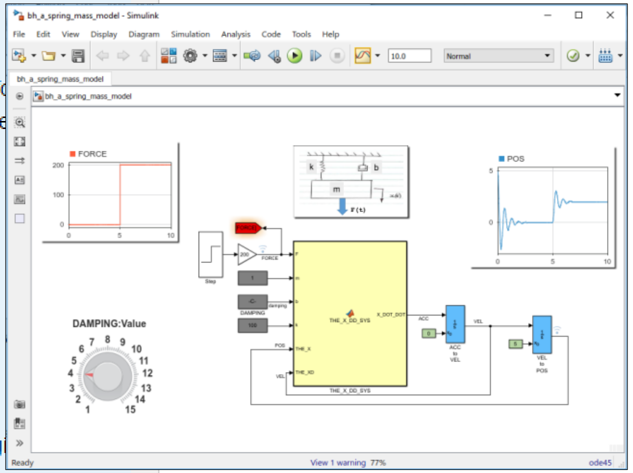
Background:

From our year 1 class in physics and mechanics, we derived using ***Newton's 2nd law***, the equation of motion for the dynamics of a Spring Mass damper system. Recall that it had the following form:

$$m.\ddot{x} + b.\dot{x} + k.x = F(t)$$

Today we'll use the ***Lagrangian approach*** to derive the same equations of motion f
spring mass damper. We're going to break this problem down into the following 6 ste

1. Define Model Parameters
2. Apply the governing physics
3. Apply Lagrange's equation
4. Isolate our expression for $\ddot{x}(t)$
5. Convert our Analytical expression for $\ddot{x}$ into a Simulink block
6. Simulate of model of this dynamic system



Euler-Lagrange equations:

Recall our earlier class where we derived and summarised the fundamental Lagrangi
equations that allow us to derive system equations of motion:

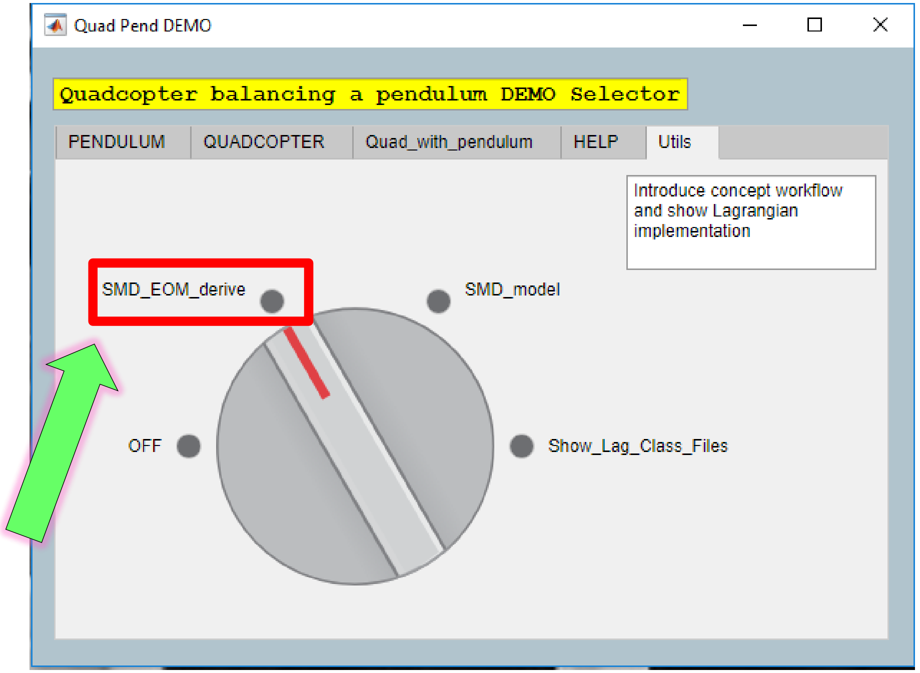
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k \quad \text{where} \quad Q_k = \sum_{i=1}^{N_{fnc}} \left(\vec{F}_i \cdot \frac{\partial \vec{v}_i}{\partial \dot{q}_k} \right) + \sum_{j=1}^{N_{\tau nc}} \left(\vec{\tau}_j \cdot \frac{\partial \vec{\omega}_j}{\partial \dot{q}_k} \right)$$

MathWorks

Live Script:

bh_smd_model_derivation.mlx

Try it:

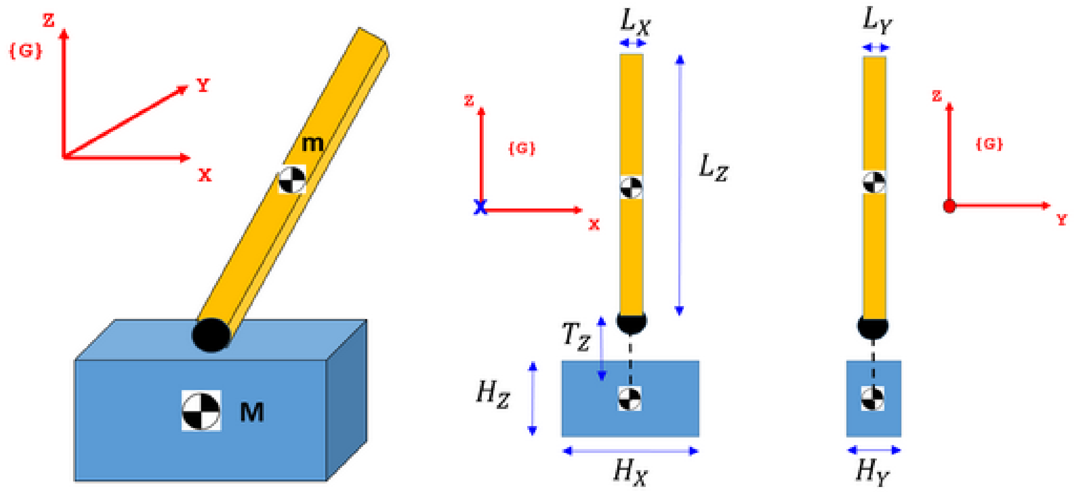


Task: 3-D Inverted Pendulum

bh_derive_LAG_eom_for_3D_pend.mlx

Introducing the 3D pendulum:

Our 3D pendulum looks like this:



- The pendulum is attached to the base cart via a spherical joint.
- The sperical joint permits the pendulum to yaw, pitch and roll.
- The base cart may translate in the inertial X,Y, and Z directions.
- We will be applying forces to the Base cart only
- We will represent the base cart as a point mass.

Defining the model parameters:

```
syms  m_pend  M_cart g
syms  Tv_Z  Lx  Ly  Lz
```

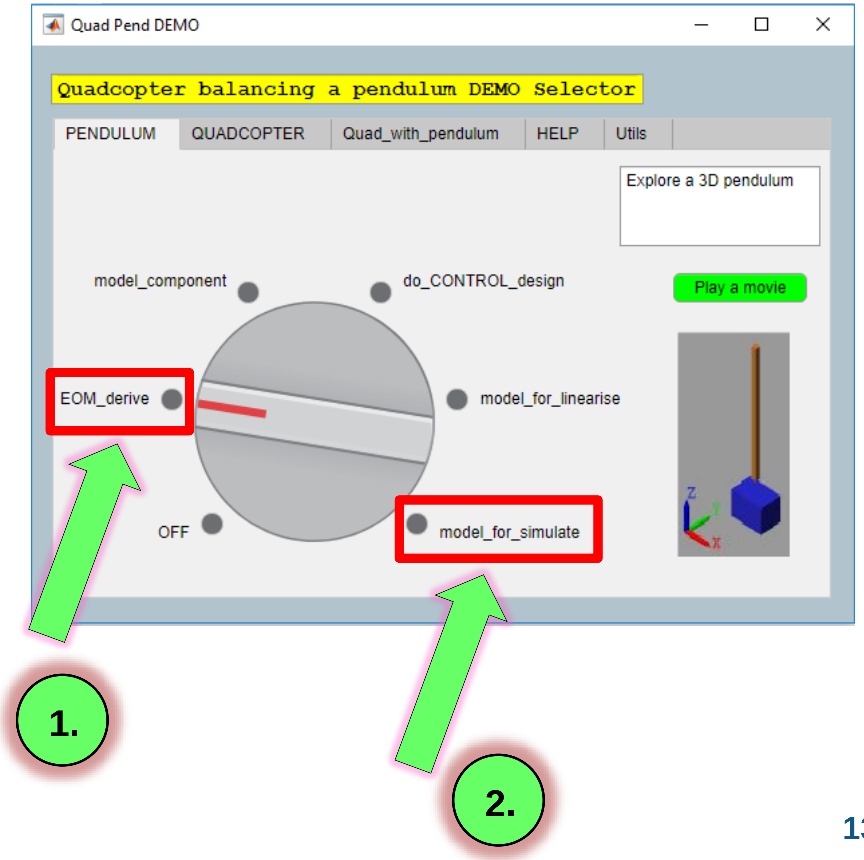
Defining the pendulum INERTIA:

So let's define the INERTIA matrix for the pendulum about its body fixed **center of mass** frame:

Live Script:
bh_derive_LAG_eom_for_3D_pend.mlx

MathWorks®

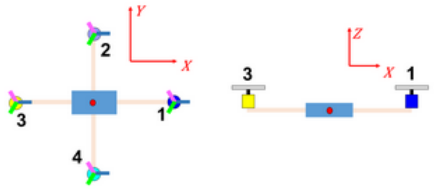
Try it:



Task: Quadcopter

bh_explore_LAG_eom_for_quad_include_spin_props.mlx

Task: **Lagrangian approach for deriving Eoms for Quadcopter**



In this task we're going to look at how the Lagrangian Dynamics approach can be used to derive the equations of motion of a Rigid Body. Steps that we'll take will include:

- What is a PASSIVE rotation matrix ?
- How do i construct a Direction Cosine Matrix (DCM) from a given rotation sequence ?
- What is the relationship between BODy rates and EULER rates ?
- What is the KE and PE of just the airframe ?
- What is the KE and PE of each rotor+Propeller assembly ?
- Apply Lagrange's equation to derive the system EoMs

Euler-Lagrange equations:

The Euler-Lagrange formula will be used to derive the equations of motion for a Rigid Body, and it has the form:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k \quad \text{for} \quad k = 1, 2, \dots, n$$

where n is the DOF of the system, $\{q_1, q_2, \dots, q_n\}$ is a set of generalized coordinates, $\{Q_1, Q_2, \dots, Q_n\}$ is the set of generalized forces associated with those coordinates, and the Lagrangian: ***L = T - V***, is defined as the difference between the kinetic and potential energy of the n- DOF system. The Generalised forces can also be defined in terns of the non conservative forces and torques acting on the multibody system. The formula for the generalised forces acting on the system is:

$$Q_k = \sum_{i=1}^{Nf_{nc}} \left(\vec{F}_i \cdot \frac{\partial \vec{v}_i}{\partial \dot{q}_k} \right) + \sum_{j=1}^{N\tau_{nc}} \left(\vec{\tau}_j \cdot \frac{\partial \vec{\omega}_j}{\partial \dot{q}_k} \right)$$

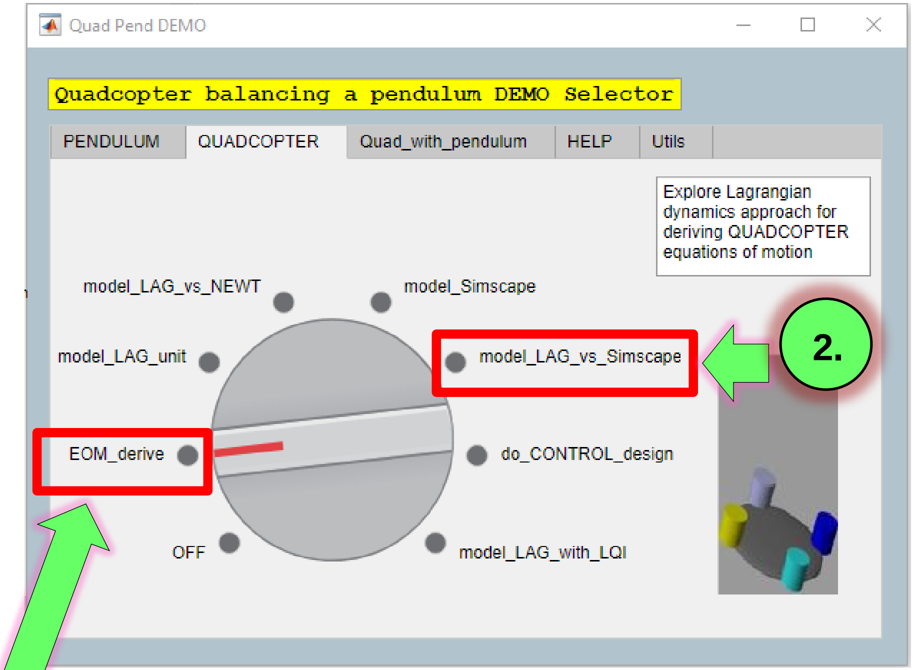
where:

- Q_k : is the generalised force associated with the k^{th} generalised co-ordinate q_k

Live Script:

bh_explore_LAG_eom_for_quad_include_spin_props.mlx

Try it:

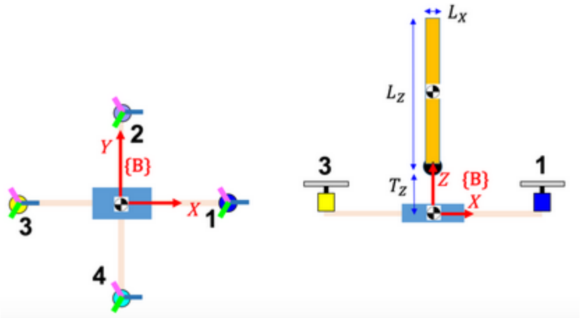


1.

Task: Quadcopter balancing a pendulum

bh_explore_LAG_eom_for_quad_with_PEND.mlx

Task: Lagrangian approach for deriving Eoms for a **Quadcopter** balancing a **pendulum**



In this task we're going to look at how the Lagrangian Dynamics approach can be used to derive the equations of motion of a quadcopter balancing an inverted pendulum. Steps that we'll take will include:

- What is a PASSIVE rotation matrix ?
- How do i construct a Direction Cosine Matrix (DCM) from a given rotation sequence ?
- What is the relationship between BODy rates and EULER rates ?
- What is the KE and PE of just the airframe ?
- What is the KE and PE of each rotor+Propeller assembly ?
- Apply Lagrange's equation to derive the system EoMs

Euler-Lagrange equations:

The Euler-Lagrange formula will be used to derive the equations of motion for a Rigid Body, and it has the form:

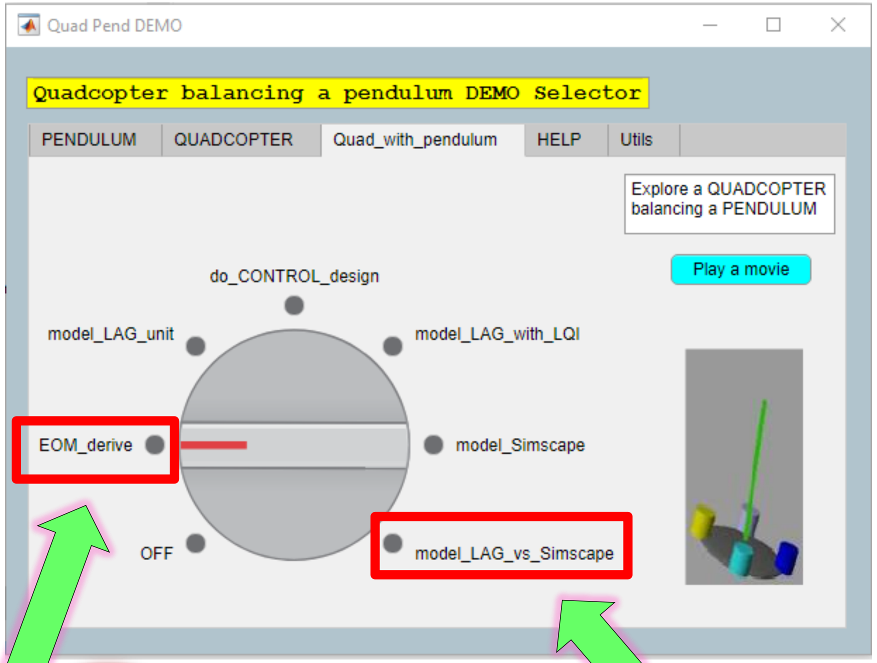
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k \quad \text{for} \quad k = 1, 2, \dots, n$$

where n is the DOF of the system, $\{q_1, q_2, \dots, q_n\}$ is a set of generalized coordinates, $\{Q_1, Q_2, \dots, Q_n\}$ is the set of generalized forces associated with those coordinates, and the Lagrangian: ***L = T - V***, is defined as the difference between the kinetic and potential energy of the n- DOF system. The Generalised forces can also be defined in terns of the non conservative forces and torques acting on the multibody system. The formula for the generalised forces acting on the system is:

Live Script:

bh_explore_LAG_eom_for_quad_with_PEND.mlx

Try it:



1.

Attention: Will take between 5-10mins to run

2.

Wrap up

Enabling Computational Thinking using MATLAB

Problem Solving and practice

Computational Thinking:
- Brain
- Technology

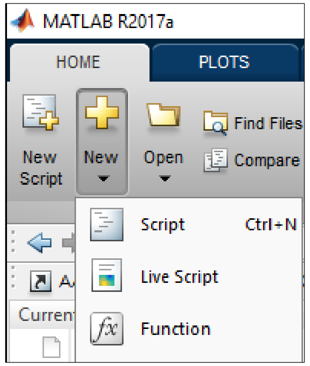
Decomposition

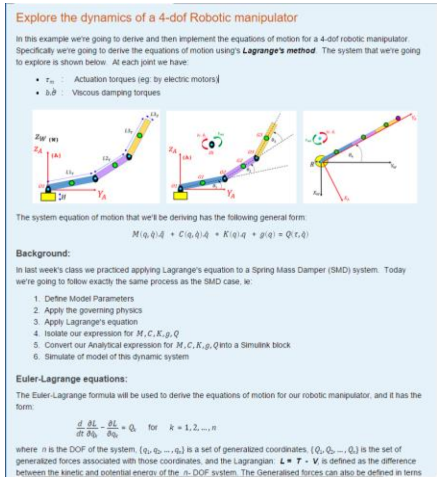
Algorithms + Automation

Simulation

Decomposition

Live Script





Algorithms + Automation

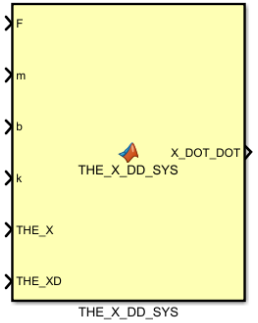
Symbolic Computing

```
>> diff()

>> matlabFunctionBlock()
```

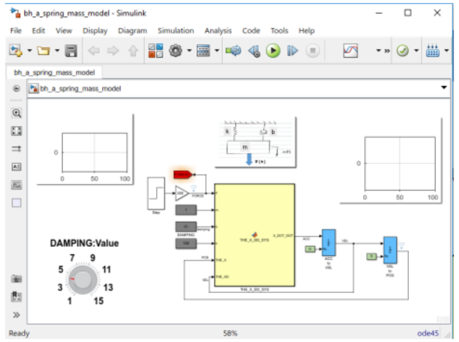
our_EOM(t) =

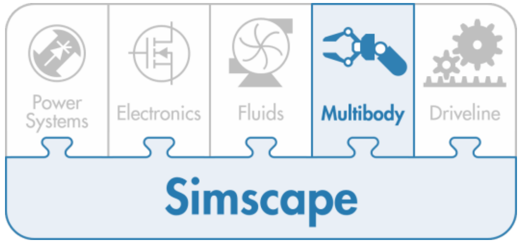
$$m\frac{\partial^2}{\partial t^2}x(t)+k\,x(t)=F-b\frac{\partial}{\partial t}x(t)$$



Simulation

Numeric via Block Diagram





Q/A:

- Are there some questions please ?
- Download the examples that you saw today … and more that you didn't !

R2018a

Problem Solving and practice

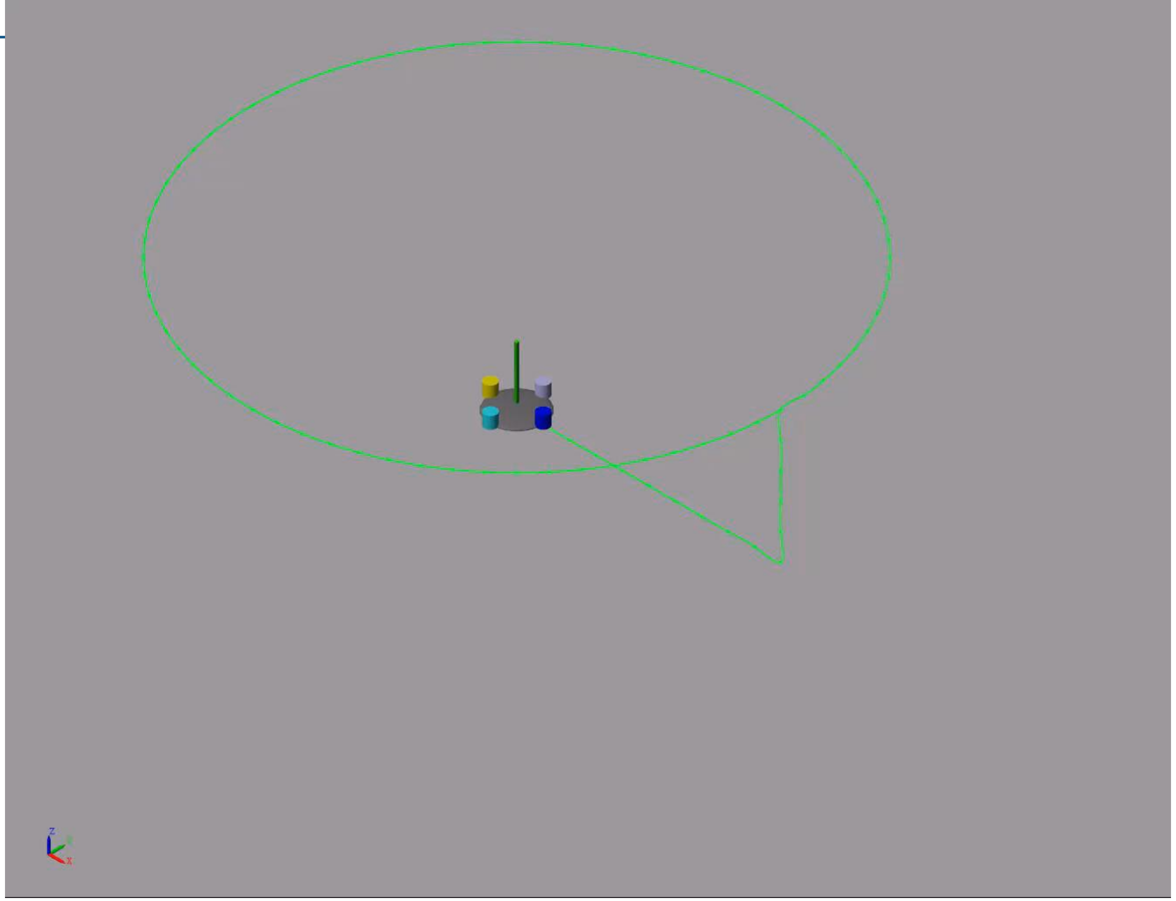
Computational Thinking:

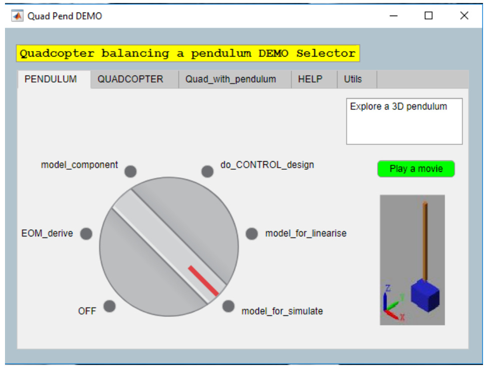
- Brain
- Technology

Decomposition

Algorithms + Automation

Simulation

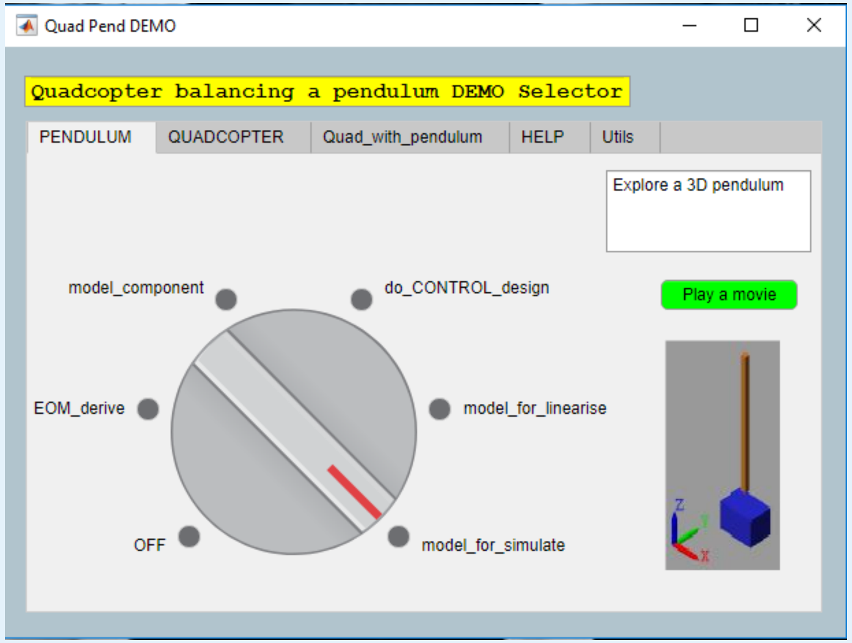





```
>> bh_startup_quad_and_pendulum
```

DEMO requirements

R2018a



What you need to run the demo:

- R2018a (or NEWER)
  – MATLAB
  – Symbolic Maths Toolbox
  – Control System toolbox

  – Simulink
  – Simscape
  – Simscape Multibody
  – Simulink Control Design

R2018a

These products will probably be on your TAH campus license.

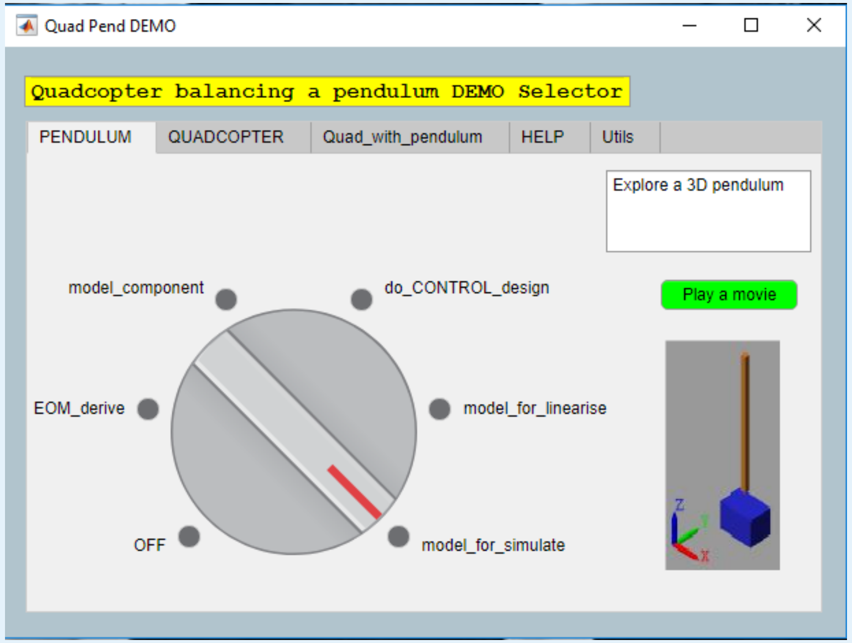
If they're not … then contact the **MathWorks**

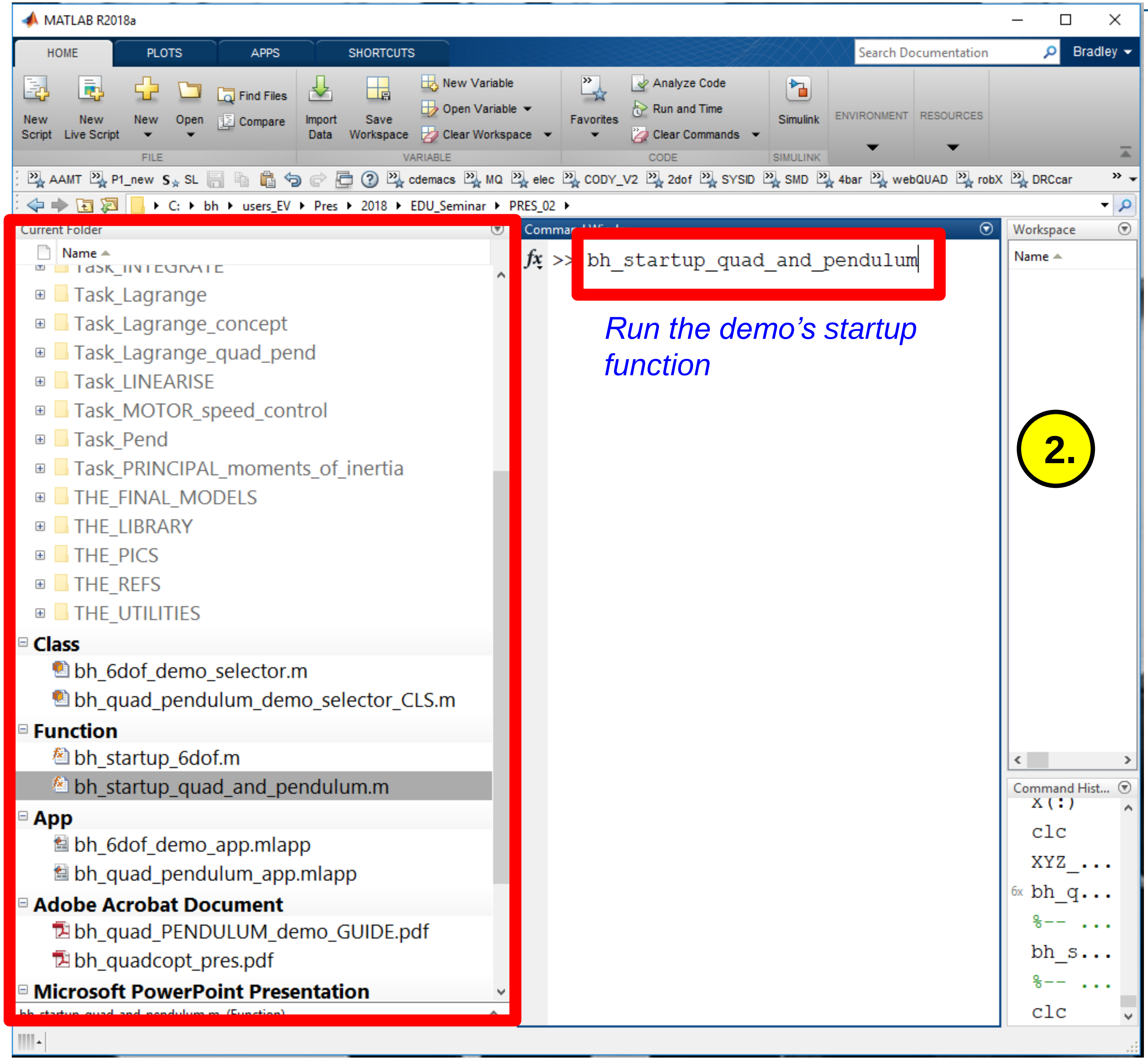
OR

help yourself here:
https://www.mathworks.com/programs/trials/trial_request.html

DEMO setup and launch

R2018a

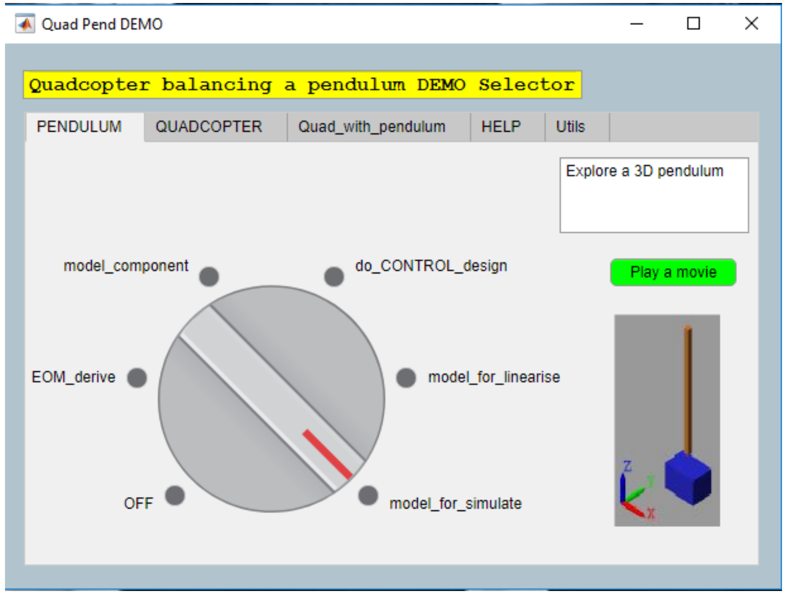




Run the demo's startup function

2.

After running the start-up function the DEMO navigator APP will appear:



1.

In MATLAB, navigate to the DEMO folder