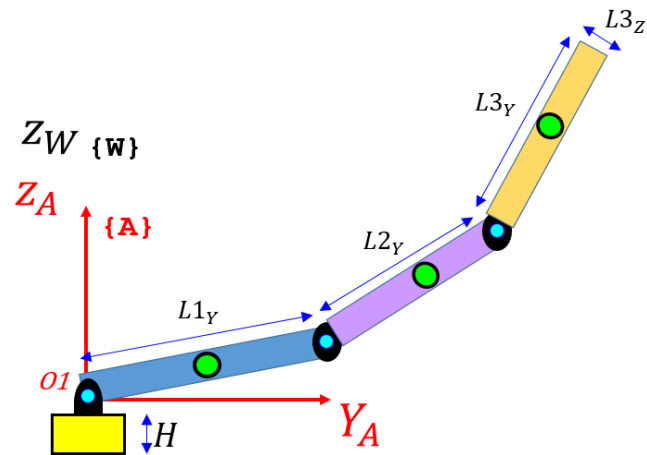


Teaching Rigid Body Dynamics

- *a combination of symbolic and numeric computing*



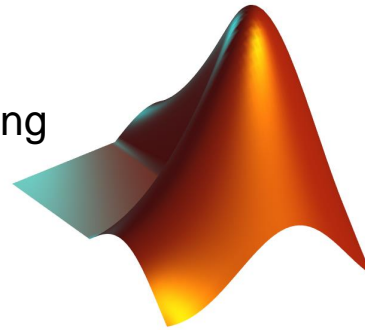
Today's agenda:

Phase 1

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

Phase 2

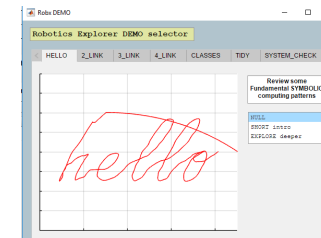
- Applying Computational Thinking
 - 3 Case Studies



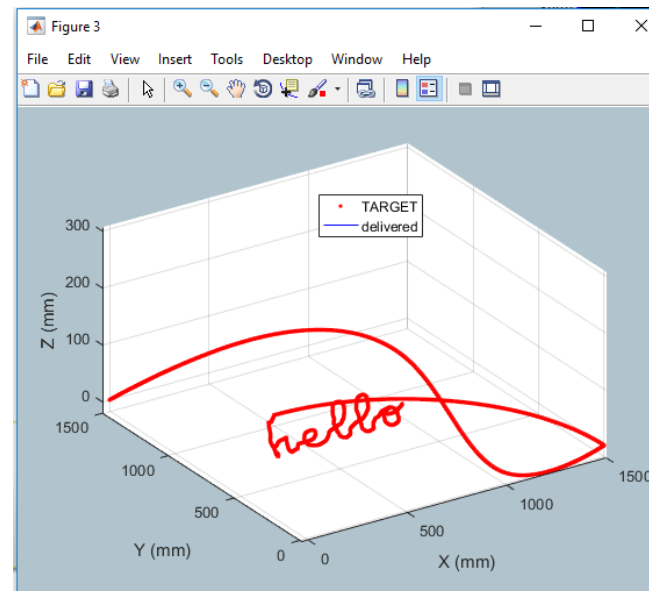
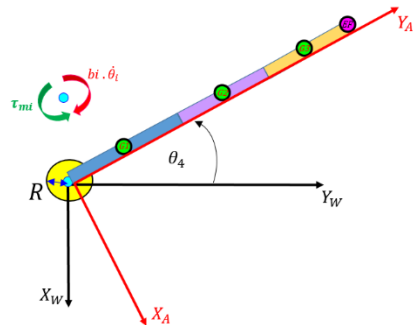
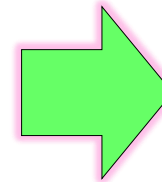
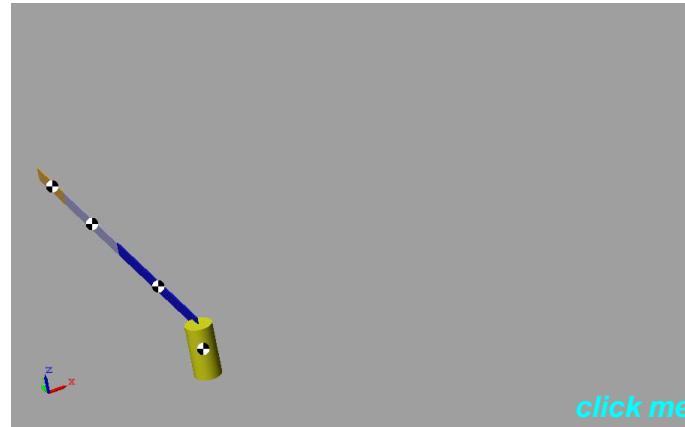
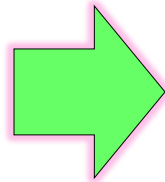
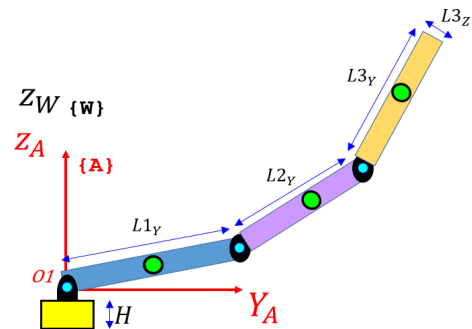
R2017a

Phase 3

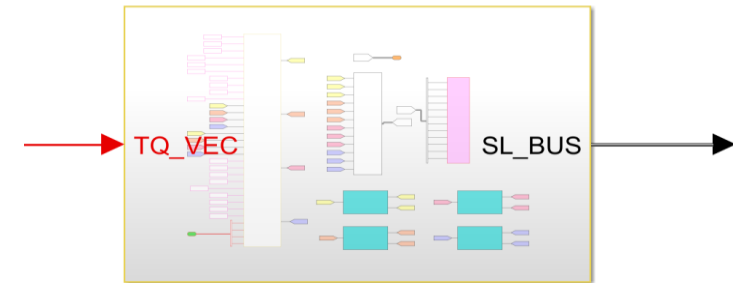
- Questions AND Answers
- How do you get ALL of the examples that you saw today ?



How do you make a robot write hello ?



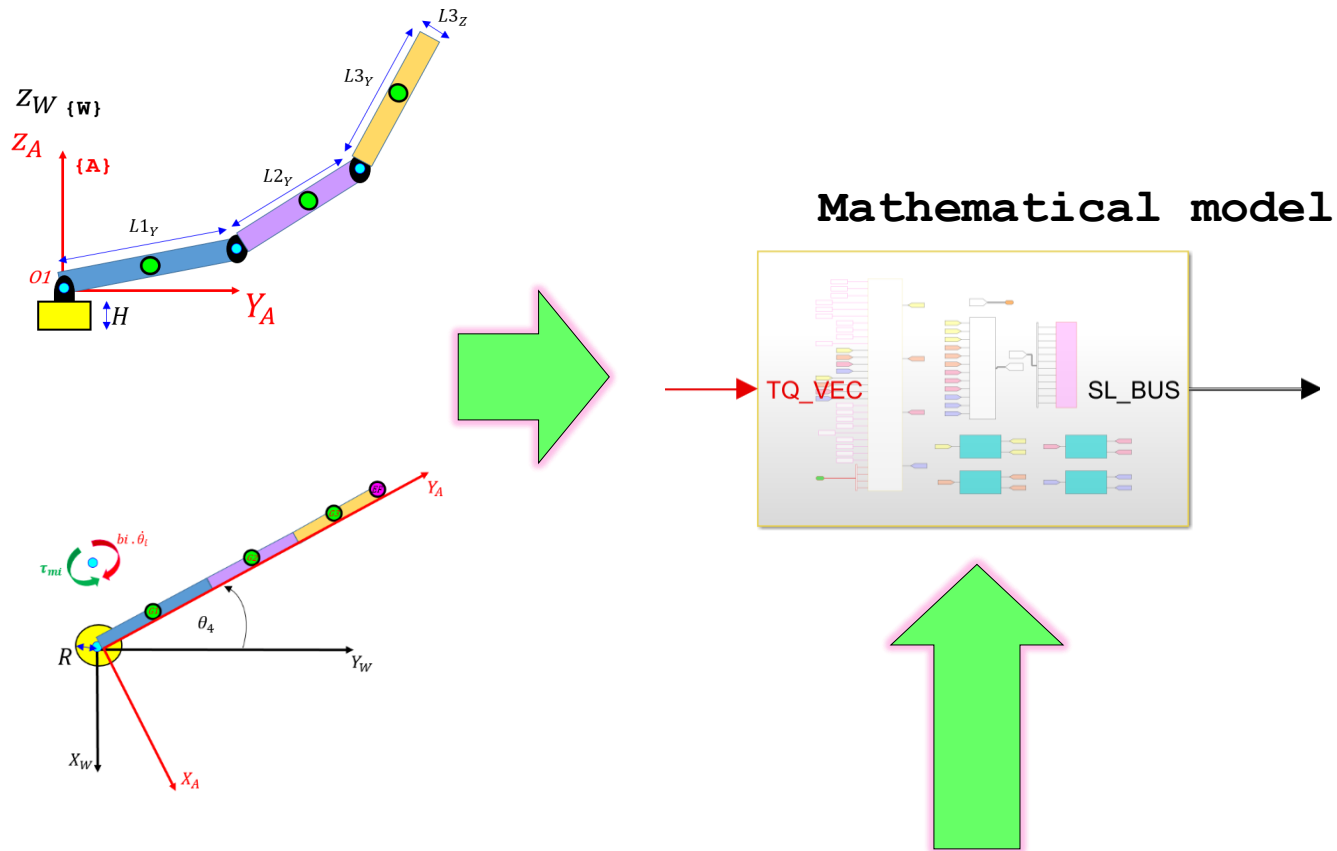
We need a mathematical model



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

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

How do you derive the mathematical model?



We need to understand the physics.

Interesting part



We need to apply Lagrange's equation

Laborious part

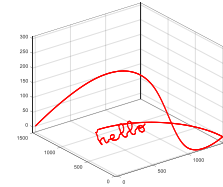
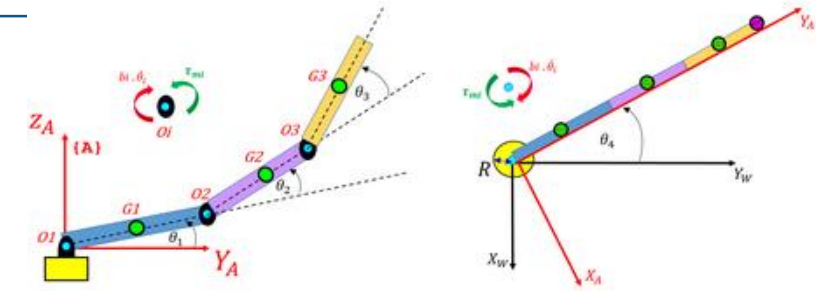
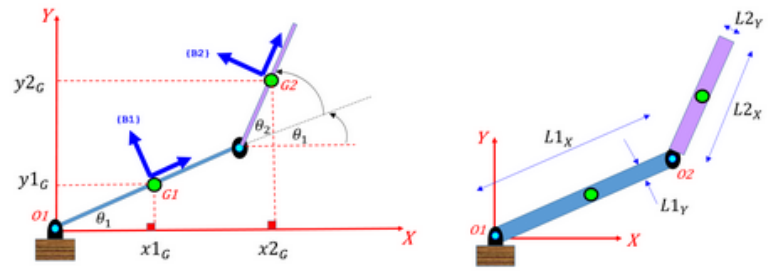
$$M(q) \cdot \ddot{q} + C(\dot{q}, q) \cdot \dot{q} + K(q) \cdot q + g(q) = Q$$

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

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

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

Laborious ?



2-dof

Approx
30 lines

$\ddot{\theta}_1$
 $\ddot{\theta}_2$

4-dof

Approx
200 lines

$\ddot{\theta}_1$
 $\ddot{\theta}_2$
 $\ddot{\theta}_3$
 $\ddot{\theta}_4$

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

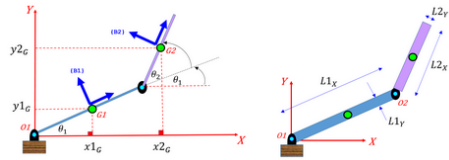
```

bh_tmp_EOM_file_WILL_BE_DELETED.txt
1 #####
2 ### q = TH1_s
3 ###
4 ### LHS of EOM is:
5 ###
6 1 I1G_s*TH1_s_DD
7 2 + I2G_s*TH1_s_DD
8 3 + I2G_s*TH2_s_DD
9 4 + (L1X_s^2*TH1_s_DD*m1_s)/4
10 5 + L1X_s^2*TH1_s_DD*m2_s
11 6 + (L2X_s^2*TH1_s_DD*m2_s)/4
12 7 + (L2X_s^2*TH2_s_DD*m2_s)/4
13 8 + (L1X_s*g_s*m1_s*cos(TH1_s))/2
14 9 + L1X_s*g_s*m2_s*cos(TH1_s)
15 10 + (L2X_s*g_s*m2_s*cos(TH1_s + TH2_s))/2
16 11 + L1X_s*L2X_s*TH1_s_DD*m2_s*cos(TH2_s)
17 12 + (L1X_s*L2X_s*TH2_s_DD*m2_s*cos(TH2_s))/2
18 13 + -(L1X_s*L2X_s*TH2_s_D^2*m2_s*sin(TH2_s))/2
19 14 + -L1X_s*L2X_s*TH1_s_D*TH2_s_D*m2_s*sin(TH2_s)
20 ###
21 ### RHS of EOM is:
22 1 Q1_s
23 #####
24 ### q = TH2_s
25 ###
26 ### LHS of EOM is:
27 ###
28 1 I2G_s*TH1_s_DD
29 2 + I2G_s*TH2_s_DD
30 3 + (L2X_s^2*TH1_s_DD*m2_s)/4
31 4 + (L2X_s^2*TH2_s_DD*m2_s)/4
32 5 + (L2X_s*g_s*m2_s*cos(TH1_s + TH2_s))/2
33 6 + (L1X_s*L2X_s*TH1_s_DD*m2_s*cos(TH2_s))/2
34 7 + (L1X_s*L2X_s*TH1_s_D^2*m2_s*sin(TH2_s))/2
35 ###
36 ### RHS of EOM is:
37 1 Q2_s
    
```

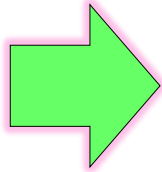
```

bh_tmp_EOM_file_WILL_BE_DELETED.txt
1 #####
2 ### q = TH1_s
3 ###
4 ### LHS of EOM is:
5 ###
6 1 (L1Y_s^2*TH1_s_DD*m1_s)/3
7 2 + L1Y_s^2*TH1_s_DD*m2_s
8 3 + L1Y_s^2*TH1_s_DD*m3_s
9 4 + (L2Y_s^2*TH1_s_DD*m2_s)/3
10 5 + L2Y_s^2*TH1_s_DD*m3_s
11 6 + (L2Y_s^2*TH2_s_DD*m2_s)/3
12 7 + L2Y_s^2*TH2_s_DD*m3_s
13 8 + (L3Y_s^2*TH1_s_DD*m3_s)/3
14 9 + (L3Y_s^2*TH2_s_DD*m3_s)/3
15 10 + (L3Y_s^2*TH3_s_DD*m3_s)/3
16 11 + (L1Z_s^2*TH1_s_DD*m1_s)/12
17 12 + (L2Z_s^2*TH1_s_DD*m2_s)/12
18 13 + (L2Z_s^2*TH2_s_DD*m2_s)/12
19 14 + (L3Z_s^2*TH1_s_DD*m3_s)/12
20 15 + (L3Z_s^2*TH2_s_DD*m3_s)/12
21 16 + (L3Z_s^2*TH3_s_DD*m3_s)/12
22 17 + (L3Y_s^2*TH4_s_D^2*m3_s*sin(2*TH1_s + 2*TH2_s + 2*TH3_s))/6
23 18 + -(L3Z_s^2*TH4_s_D^2*m3_s*sin(2*TH1_s + 2*TH2_s + 2*TH3_s))/24
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
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)
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108 ###
109 ### RHS of EOM is:
110 1 Q4_s
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
    
```

Encouraging Deeper Learning engagements in your classroom:

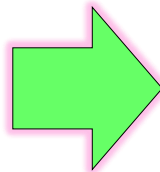


smaller problems



The understanding of the problem physics:

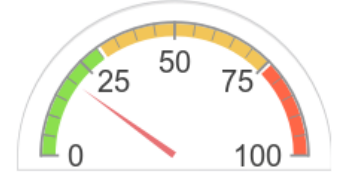
- *Motion in a plane*
- *Inertia about an axis*
- *Virtual Work*



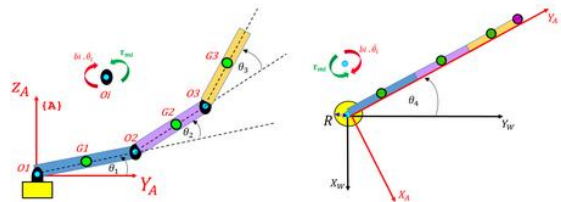
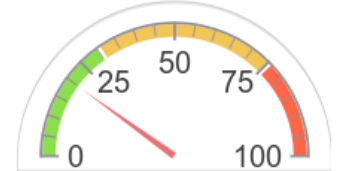
Problem Solving and practice

Hand written implementation

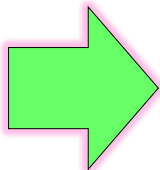
BRAIN
Conceptual Difficulty



HAND
Computational Difficulty



Bigger problems



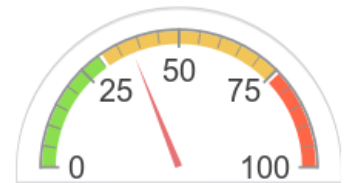
The understanding of the problem physics:

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

Problem Solving and practice

Hand written implementation

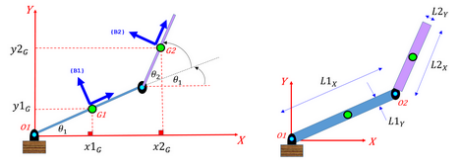
BRAIN
Conceptual Difficulty



HAND
Computational Difficulty



Encouraging Deeper Learning engagements in your classroom:



smaller problems

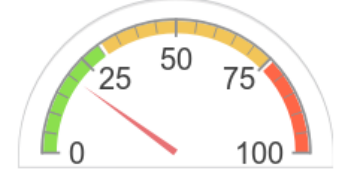
The understanding of the problem physics:

- *Motion in a plane*
- *Inertia about an axis*
- *Virtual Work*

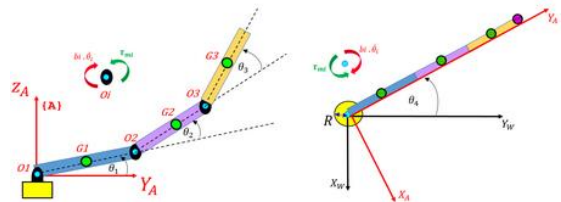
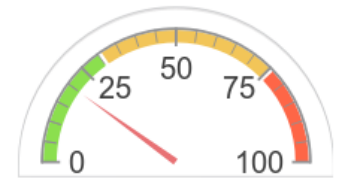
Problem Solving and practice

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*

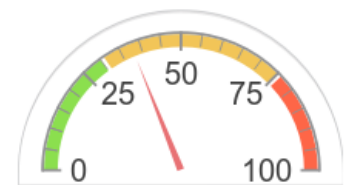
Problem Solving and practice

Computational Thinking:

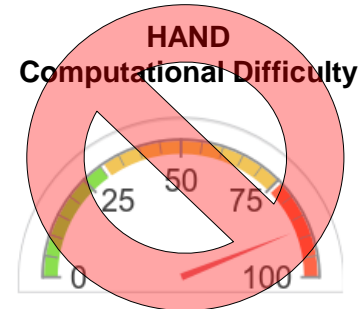
- Brain
- Technology



BRAIN
Conceptual Difficulty



HAND
Computational Difficulty



Enabling Computational Thinking using MATLAB

Problem Solving
and practice

Computational Thinking:

- Brain
- Technology

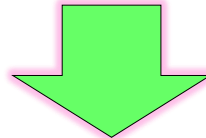


Decomposition

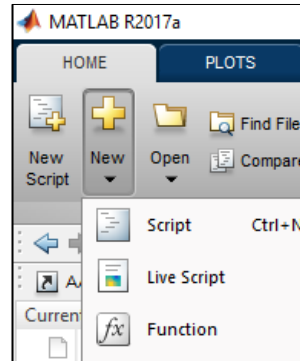
Algorithms
+
Automation

Simulation

Decomposition



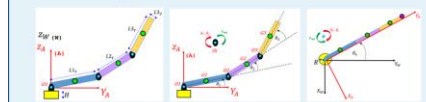
Live Script



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 the **Lagrange's method**. The system that we're going to explore is shown below. At each point 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:

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

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, τ
5. Convert our analytical expression for M, C, K, g, τ into a Simulink block
6. Simulate our model of this dynamic system

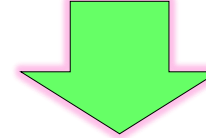
Euler-Lagrange equations:

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

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

where: n is the DOF of the system, (q_1, q_2, \dots, q_n) is a set of generalized coordinates, $(\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n)$ is the set of generalized velocities associated with those coordinates, and the Lagrangian: $L = T - V$. V is defined as the difference between the kinetic and potential energies of the n -DOF system. The Generalised forces can also be defined in terms

Algorithms
+
Automation



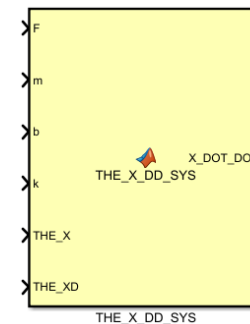
Symbolic Computing

```
>> diff()
```

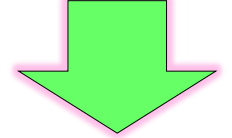
```
>> matlabFunctionBlock()
```

our_EOM(t) =

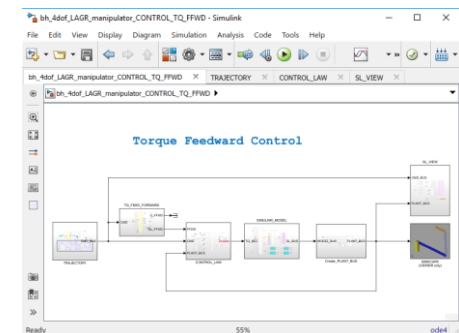
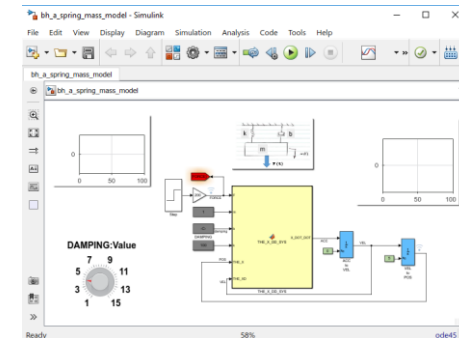
$$m \frac{\partial^2}{\partial t^2} x(t) + k x(t) = F - b \frac{\partial}{\partial t} x(t)$$



Simulation

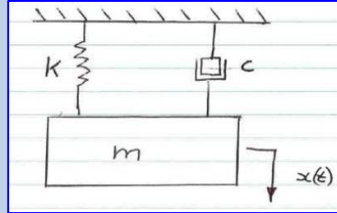


Numeric via Block Diagram

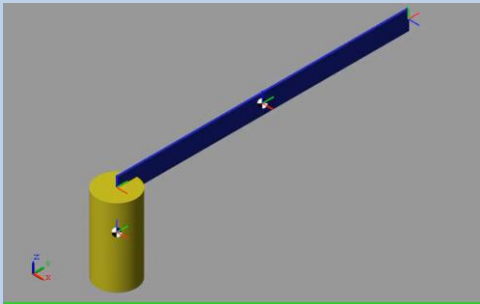


Today's case studies:

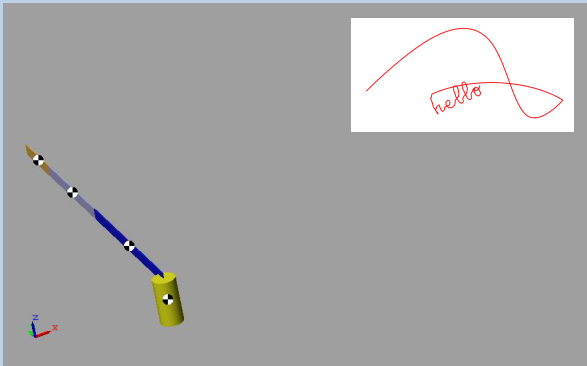
1-dof



2-dof
(non planar)



4-dof
(non planar)



The
understanding
of the problem
physics

**Problem Solving
and practice**

Computational Thinking:

- Brain
- Technology

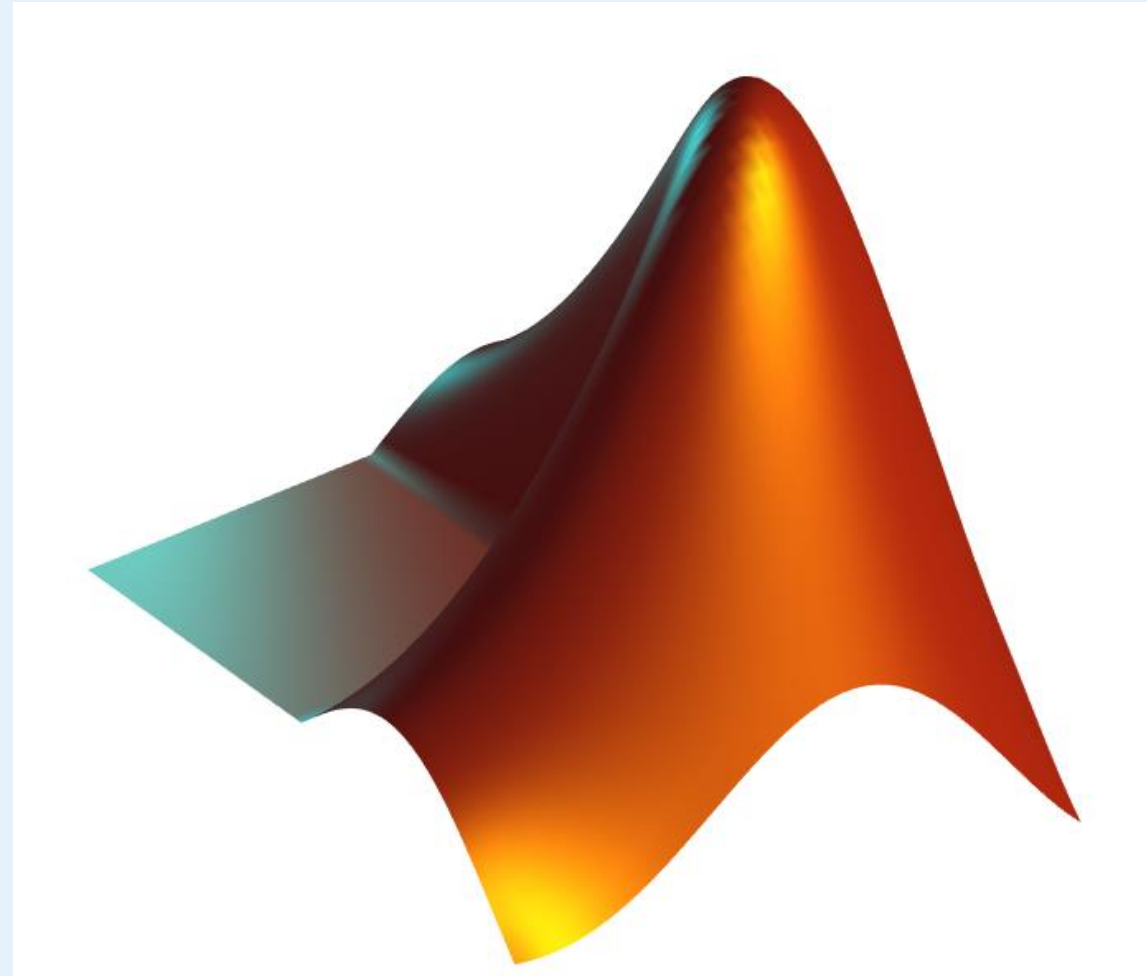


Decomposition

**Algorithms
+
Automation**

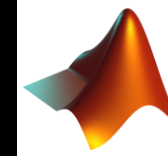
Simulation

Demo
these
concepts



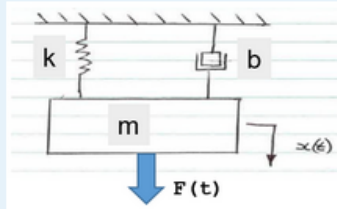
R2017a

Task: Spring Mass Damper



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:

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

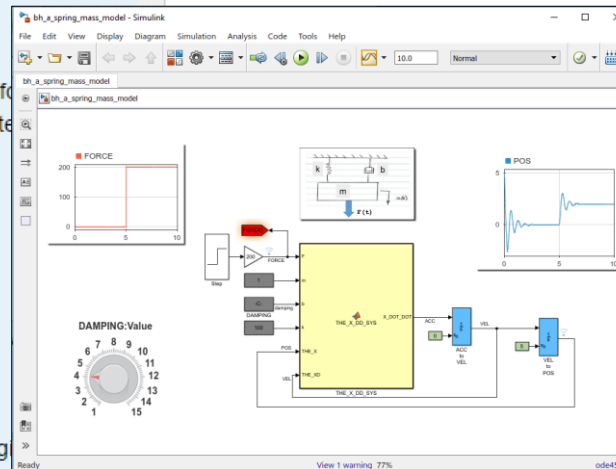
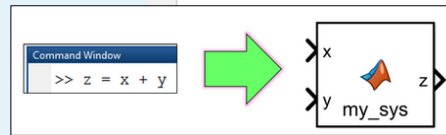
Today we'll use the **Lagrangian approach** to derive the same equations of motion for a spring mass damper. We're going to break this problem down into the following 6 steps:

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 Lagrangian equations that allow us to derive system equations of motion:

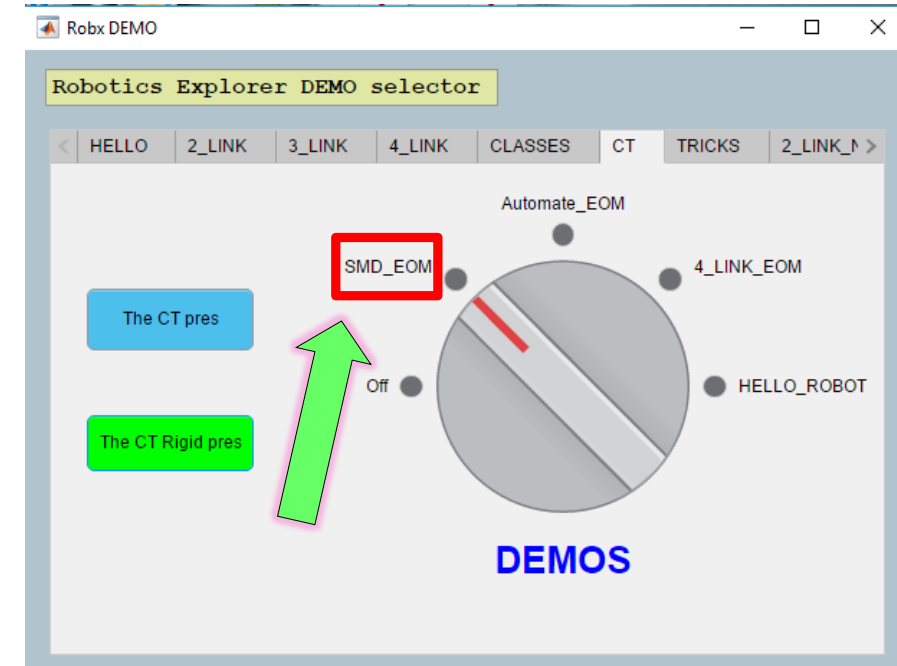
$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k \quad \text{where} \quad Q_k = \sum_{i=1}^{N_{fnc}} \left(\vec{F}_i \cdot \frac{\partial \vec{v}_i}{\partial \dot{q}_k} \right) + \sum_{j=1}^{N_{\tau nc}} \left(\vec{\tau}_j \cdot \frac{\partial \vec{\omega}_j}{\partial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



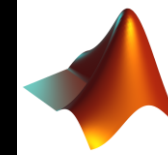
Live Script:

bh_smd_model_derivation.mlx

Try it:



Task: 2-dof Non-planar robotic manipulator



Live Script:

bh_LAGRANGE_derivation_2dof_NP.mlx

bh_LAGRANGE_derivation_2dof_NP.mlx

Explore the dynamics of a 2-dof NON-planar Robotic manipulator

In this example we're going to derive and then implement the equations of motion for a 2-dof non-planar 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_{in} : Actuation torques (eg: by electric motors)
- $b \cdot \dot{\theta}$: Viscous damping torques

Bradley Horton : 09-Mar-2017, bradley.horton@mathworks.com.au

Here's a short video showing the type of machine that we're going to derive the equations of motion for - note the 2 degrees of freedom that the machine has:

```
disp('a href="matlab:bh_play_movie">CLICK ME to PLAY a MOVIE</a>')
```

STAGE 1: symbolic derivation of system equations

Euler-Lagrange equations of motion:

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

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

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, and the Lagrangian: $L = T - V$, is defined as the difference between the kinetic and potential energy of the system. Generalised forces can also be defined in terms of the non conservative forces and torques acting on the multibody system. generalised forces acting on the system is:

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

where:

- Q_k : is the generalised force associated with the k^{th} generalised co-ordinate q_k
- $N_{f_{nc}}$: is the number of active NON conservative forces
- $N_{\tau_{nc}}$: is the number of active NON conservative TORQUES
- \vec{v}_j : is the velocity vector of the point associated with the applied force.
- $\vec{\omega}_j$: is the angular velocity about the point associated with the applied torque.

bh_2dof_NP_compare_lagr_vs_sm - Simulink

Try it:

Robx DEMO

Robotics Explorer DEMO selector

< 2_LINK 3_LINK 4_LINK CLASSES CT TRICKS 2_LINK_NP TIDY >

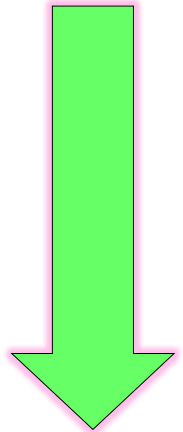
Task: automating the algorithm

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = Q$$

We should be able to automate
this
for a MULTI dof system

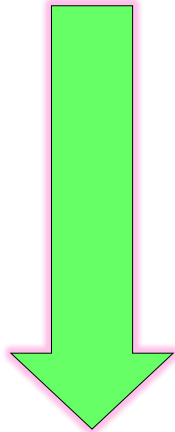
Choices:

In a
MATLAB
script



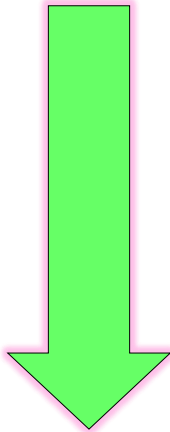
- Undergrad
- Postgrad
- Lecturer

In a
MATLAB
function



- Undergrad
- Postgrad
- Lecturer

In a
MATLAB
class



- Postgrad
- Lecturer

STEP_3: Apply Lagrange's equation - PART 1 of 3

Could be
Automated

Now let's start applying Lagranges equation $\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x}$:

```
% OLD_LIST      NEW_LIST
L_new = subs(L, actual_list, HOLDER_list);
```

1. Our 1st piece is: $\frac{\partial L}{\partial x}$

```
dLdx = diff(L_new, THE_X);
```

2. Our 2nd piece is: $\frac{\partial L}{\partial \dot{x}}$

```
dLdxdot = diff(L_new, THE_XD);
```

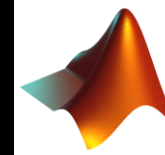
3. Our 3rd piece is: $\frac{d}{dt} \frac{\partial L}{\partial \dot{x}}$

```
% OLD_LIST      NEW_LIST
dLdxdot = subs(dLdxdot, HOLDER_list, actual_list);
dt_of_dLdxdot = diff(dLdxdot, t);
```

Now put it all together: $\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x}$

```
our_EOM_LHS = dt_of_dLdxdot - dLdx;
our_EOM_LHS = subs(our_EOM_LHS, HOLDER_list, actual_list)
```

Task: automating the algorithm



Class

- bh_eom_CLS.m
- bh_genF4manips_CLS.m
- bh_lagr4manips_CLS.m
- bh_MCKGQ_CLS.m
- bh_qman4manips_CLS.m

```
for kk=1:OBJ.N_dof

    L_ORIGINAL = OBJ.L;

    % OLD      NEW
    L = subs(L_ORIGINAL, states_actual_list, states_holder_list );

    THE_q = OBJ.holder_list_SYM_pos(kk);
    THE_qp = OBJ.holder_list_SYM_vel(kk);

    dLdqp = diff(L, THE_qp);

    % OLD      NEW
    dLdqp = subs(dLdqp, states_holder_list, states_actual_list);
    der_dt_of_dLdqp = diff(dLdqp, t);

    dLdq = diff(L, THE_q);
    dLdq = subs(dLdq, states_holder_list, states_actual_list);

    eom_LHS = der_dt_of_dLdqp - dLdq;
    eom_LHS = simplify( eom_LHS );

    THE_Q = OBJ.Qk_list(kk); % actual
    eom_RHS = simplify( THE_Q );

    eom_LHS = formula( eom_LHS );
    eom_RHS = formula( eom_RHS );

    % now store into a struct array
    EOM(kk).actual_eom_LHS = eom_LHS;
    EOM(kk).actual_eom_RHS = eom_RHS;
    EOM(kk).actual_eom_EQ = eom_LHS == eom_RHS;

    % store a few other useful things
    EOM(kk).actual_SYM_pos = OBJ.actual_list_SYM_pos(kk);
    EOM(kk).actual_SYM_vel = OBJ.actual_list_SYM_vel(kk);
    EOM(kk).actual_SYM_acc = OBJ.actual_list_SYM_acc(kk);

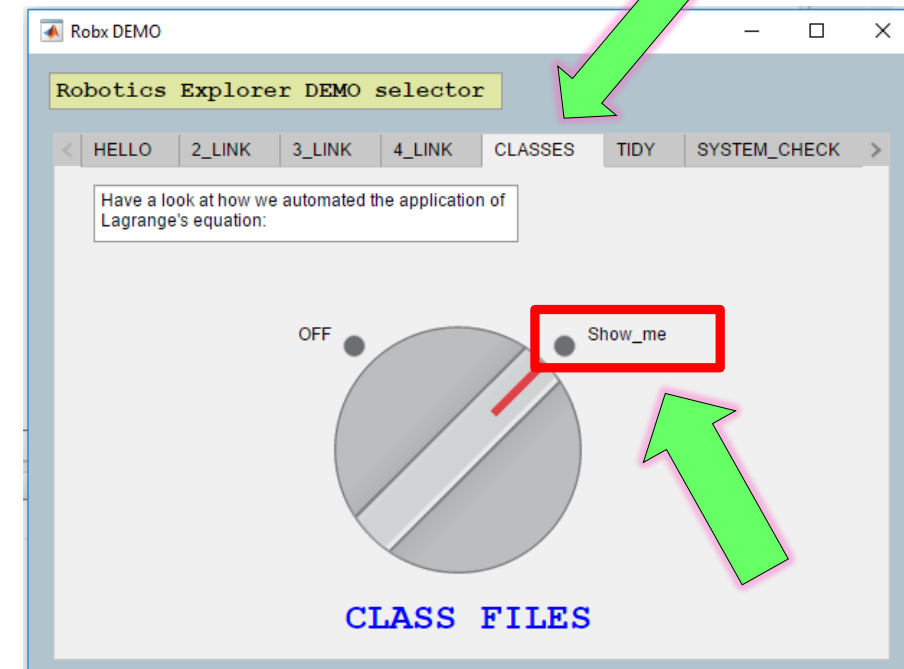
end % for kk=1:OBJ.N_dof
```

2. 3.

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q}$$

1.

Try it:



Task: automating the algorithm

```

N_dof      = OBJ.N_dof;
the_tau_mat = OBJ.THE_CORE.the_tau_mat_holder;
the_w_mat   = OBJ.THE_CORE.the_w_mat_holder;

for kk = 1:N_dof

    the_qdot_sym = OBJ.holder_list_SYM_vel(kk);

    % initialise the Qk
    the_Q        = sym(0);

    for jj=1:size(the_tau_mat,2)
        the_tau_col = the_tau_mat(:,jj);
        the_w_col   = the_w_mat(:,jj);

        the_dwdq_col = diff(the_w_col, the_qdot_sym);

        % now do the DOT product
        this_Q        = sum( the_tau_col.* the_dwdq_col );

        % accumulate
        the_Q = the_Q + this_Q;
    end % jj

    % assign the final holder result
    the_holder_eom_Q(kk,1) = the_Q;

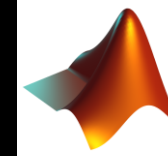
    % create and assign the ACTUAL symbol result
    act_list = [ OBJ.actual_list_SYM_pos;
                 OBJ.actual_list_SYM_vel;
                 OBJ.actual_list_SYM_acc ];

    hol_list = [ OBJ.holder_list_SYM_pos;
                 OBJ.holder_list_SYM_vel;
                 OBJ.holder_list_SYM_acc ];

    the_actual_eom_Q(kk,1) = subs( the_holder_eom_Q(kk), ...
                                   hol_list, act_list);

end % kk
    
```

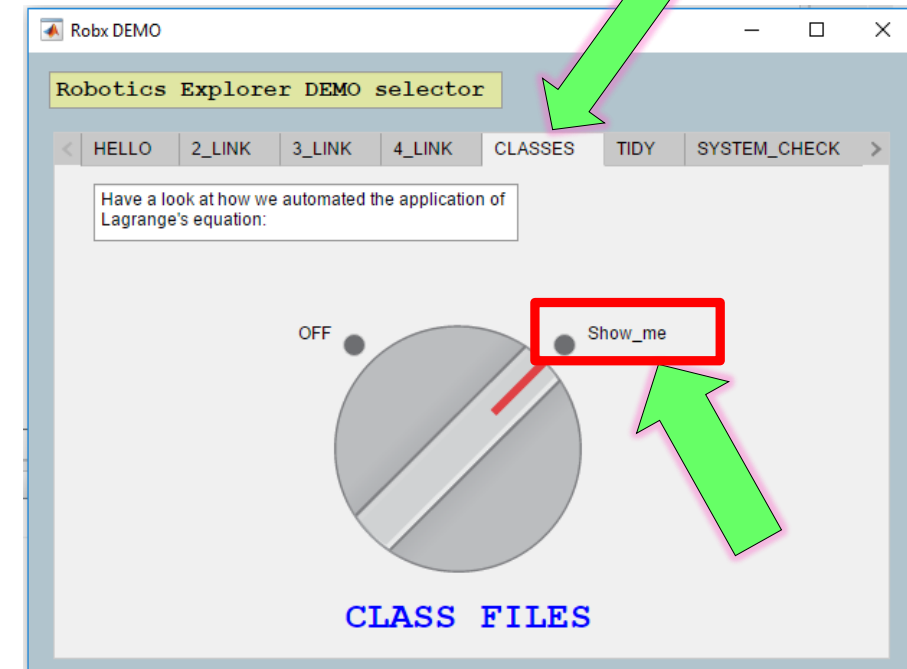
$$Q_k = \sum_{i=1}^{N_{f_{nc}}} \left(\vec{F}_i \cdot \frac{\partial \vec{v}_i}{\partial \dot{q}_k} \right) + \sum_{j=1}^{N_{\tau_{nc}}} \left(\vec{\tau}_j \cdot \frac{\partial \vec{\omega}_j}{\partial \dot{q}_k} \right)$$



Class

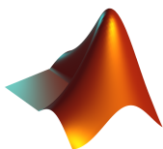
- bh_eom_CLS.m
- bh_genF4manips_CLS.m
- bh_lagr4manips_CLS.m
- bh_MCKGQ_CLS.m
- bh_qman4manips_CLS.m

Try it:



CLASS FILES

Task: 4-dof Robotic manipulator automate application



Live Script:

bh_LAGRANGE_4dof_manipulator.mlx

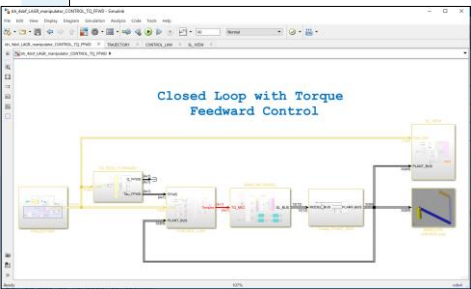
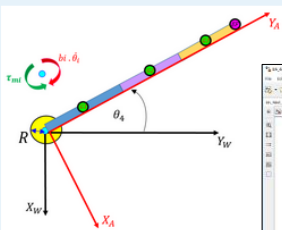
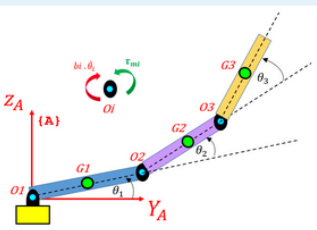
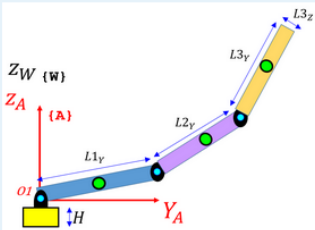
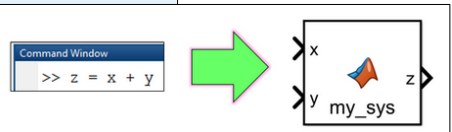
Explore the dynamics of a 4-dof Robotic manipulator

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



Bradley Horton : 13-Sep-2016, bradley.horton@mathworks.com.au

STAGE 1: symbolic derivation of system equations

Euler-Lagrange equations of motion:

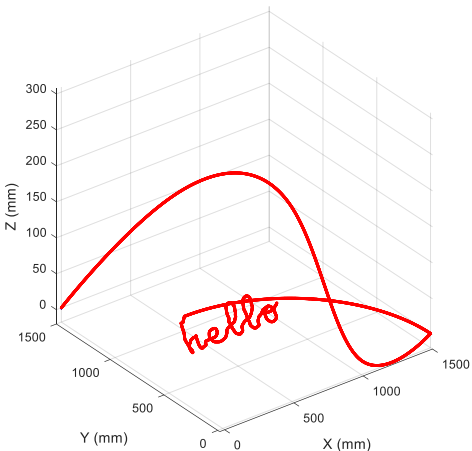
The Euler-Lagrange formula will be used to derive the equations of motion for our robotic manipulator, and it has the form:

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

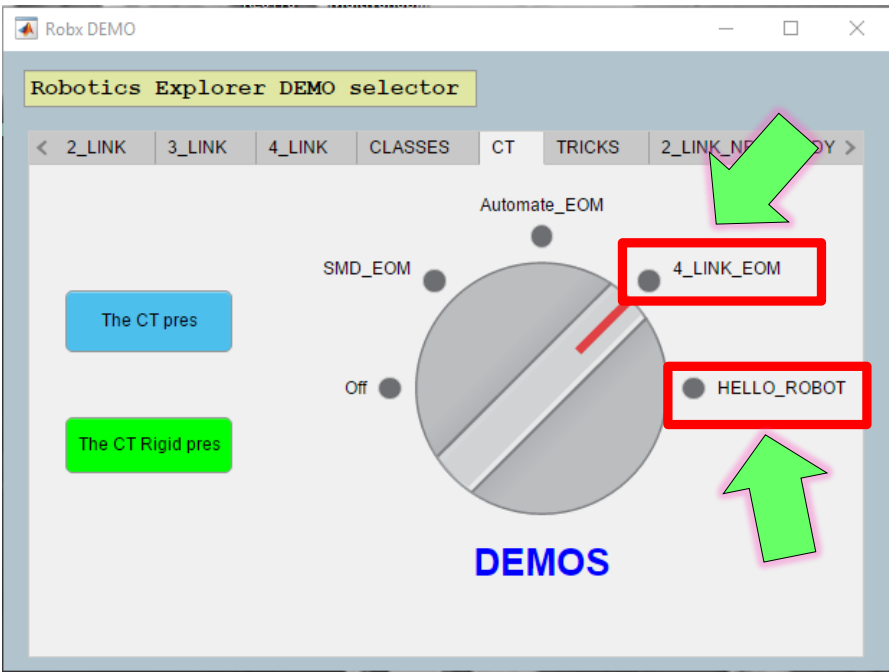
where n is the DOF of the system, $\{q_1, q_2, \dots, q_n\}$ is a set of generalized coordinates, $\{Q_1, Q_2, \dots, Q_n\}$ is the set of generalized forces associated with those coordinates, and the Lagrangian: $L = T - V$, is defined as the difference between the kinetic and potential energy of the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

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

where:



Try it:



Wrap up

The Computational Thinking approach:

Problem Solving
and practice

Computational Thinking:

- Brain
- Technology

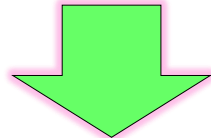


Decomposition

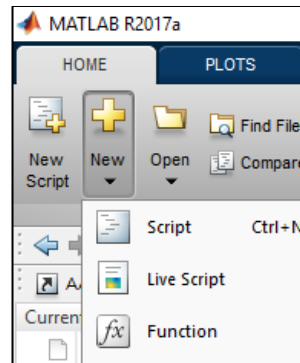
Algorithms
+
Automation

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 the **Lagrange's method**. The system that we're going to explore is shown below. At each point 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:

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

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 model of this dynamic system

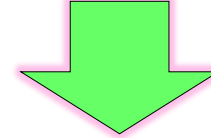
Euler-Lagrange equations:

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

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

where: n is the DOF of the system, (q_1, q_2, \dots, q_n) is a set of generalized coordinates, $(\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n)$ is the set of generalized velocities associated with those coordinates, and the Lagrangian: $L = T - V$. V is defined as the difference between the kinetic and potential energies of the n -DOF system. The Generalised forces can also be defined in terms

Algorithms
+
Automation



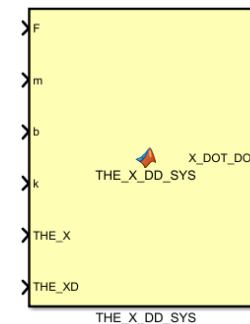
Symbolic Computing

```
>> diff()
```

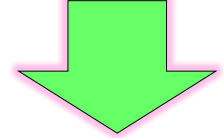
```
>> matlabFunctionBlock()
```

our_EOM(t) =

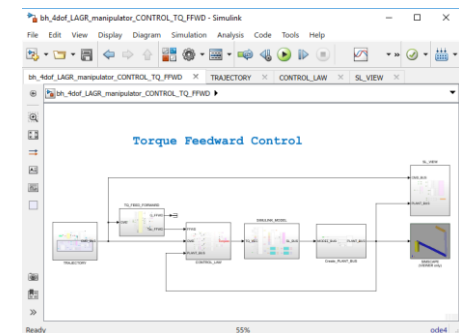
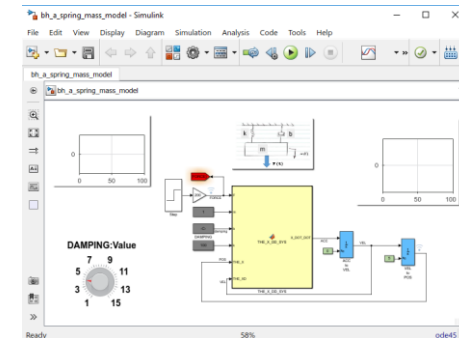
$$m \frac{\partial^2}{\partial t^2} x(t) + k x(t) = F - b \frac{\partial}{\partial t} x(t)$$



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 !

Problem Solving and practice

Computational Thinking:

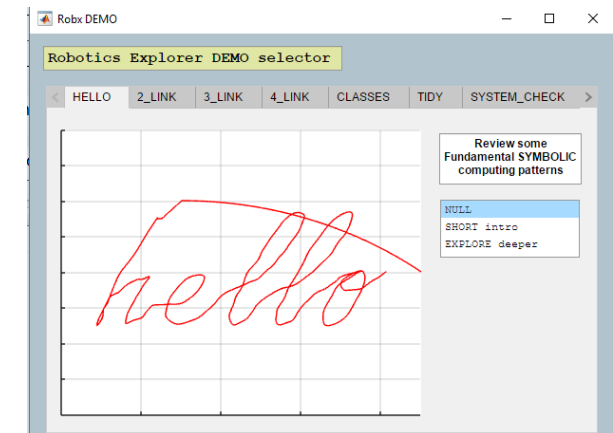
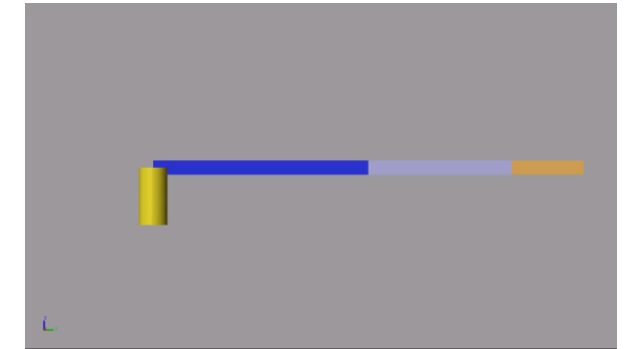
- Brain
- Technology



Decomposition

**Algorithms
+
Automation**

Simulation



>> bh_robx_startup

Teaching and Learning Resources.


https://www.mathworks.com/academia/courseware Courseware based on MAT...

MATLAB Courseware

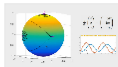
Search MathWorks.com

Educator Home Classroom Resources Hardware Support License Options Research


Mathematics



Applied Numerical Methods with MATLAB
Professor Steven C. Chapra
Tufts University



Differential Equations and Linear Algebra
Professor Gilbert Strang
Massachusetts Institute of Technology
Cleve Moler
MathWorks



Numerical Computing with MATLAB
Cleve Moler
MathWorks


Related Books Mathematics

MATLAB Courseware

Search MathWorks.com

Educator Home Classroom Resources Hardware Support License Options Research

Earth, Atmospheric, and Ocean Sciences



Geoscience with MATLAB
from SERC@Carleton


Related Books Earth Sciences

MATLAB Courseware

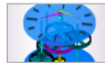
Search MathWorks.com

Educator Home Classroom Resources Hardware Support License Options Research


Introduction to Engineering




Engineering Models I
Professor Kathleen Ossman
Professor Gregory Bucks
University of Cincinnati



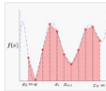
Engineering Models II
Professor Kathleen Ossman
Professor Gregory Bucks
University of Cincinnati



Discovery-Based Learning
Professor Steve McKnight
Professor Gilead Tadmor
Northeastern University



Engineering Problem Solving
Professor Stanley Hsu
Professor Rajeevan Amirtharajah
Professor Andre Knoesen
University of California, Davis



Introduction to Engineering Analysis
Professor Ivan V. Bajic
Professor Fabio Campi
et al.
Cine Free University

<http://www.mathworks.com/academia/courseware>

Curriculum
materials:

MATLAB Courseware

Cody Coursework™

Online automated grading system for MATLAB assignments

<http://mathworks.com/help/coursework/cody-coursework-for-instructors.html>

- Create online private courses and assignments
- Students **execute MATLAB code on the web**
- Control the visibility of the test suites from students.
- Visualize solution results using MATLAB graphics
- Download all student attempts and **report on grading data**

Problems

1b:: Represent a piecewise linear ...	<div><div></div></div>
2b:: Derive the ANALYTICAL soluti...	<div><div></div></div>
2e:: Calculate the Frequency Res...	<div><div></div></div>
2f_1:: Derive the ANALYTICAL sol...	<div><div></div></div>
2f_2:: Calculate the unit STEP res...	<div><div></div></div>

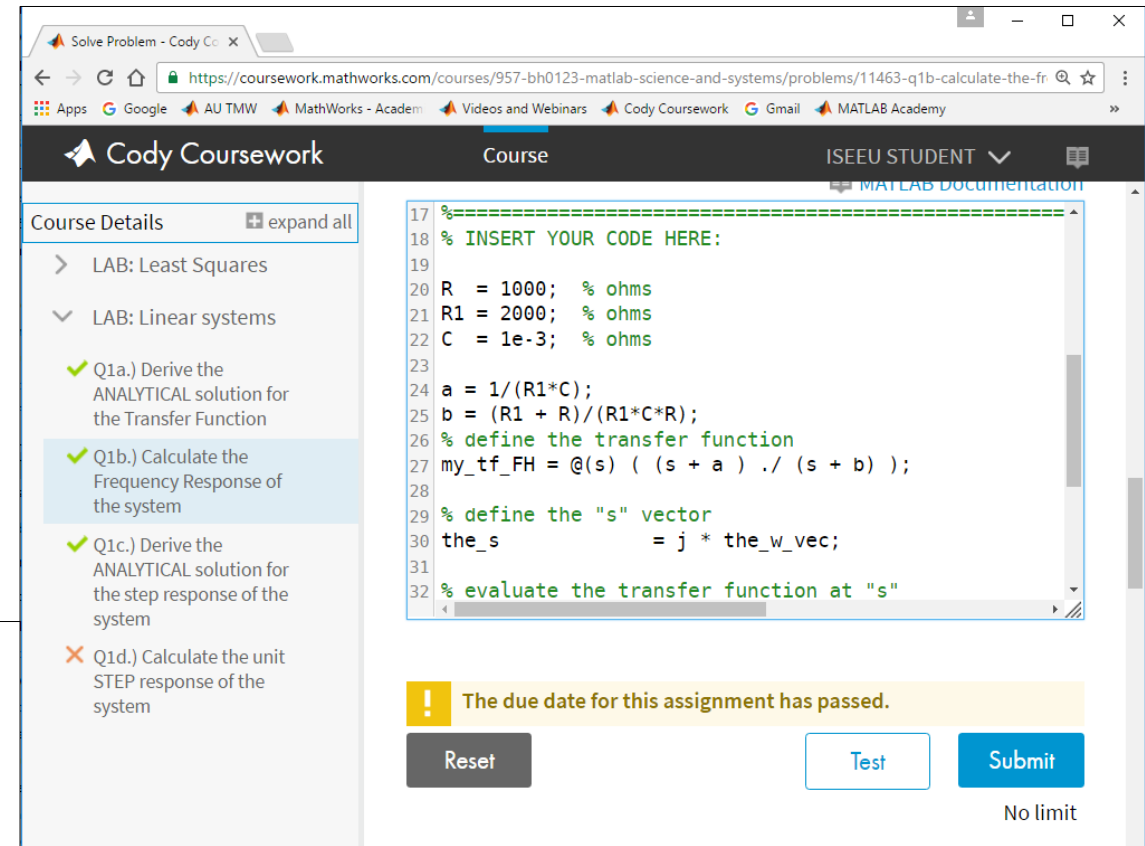
Create Report : Assignment 1

Assignment 1

- ☒ Last best solutions submitted by due date(05 Jun 2015 2:00 PM UTC)
☐ Last best solution as of today
☐ All solutions

Report Format: CSV
 CSV
 Excel

Cancel Create



The screenshot shows the Cody Coursework interface. On the left, a sidebar lists course details under 'LAB: Linear systems', including three questions (Q1a, Q1b, Q1c) marked with green checkmarks and one question (Q1d) marked with a red X. The main area displays a MATLAB code editor with the following code:

```

17 %=====
18 % INSERT YOUR CODE HERE:
19
20 R = 1000; % ohms
21 R1 = 2000; % ohms
22 C = 1e-3; % ohms
23
24 a = 1/(R1*C);
25 b = (R1 + R)/(R1*C*R);
26 % define the transfer function
27 my_tf_FH = @(s) ( (s + a) ./ (s + b) );
28
29 % define the "s" vector
30 the_s = j * the_w_vec;
31
32 % evaluate the transfer function at "s"
    
```

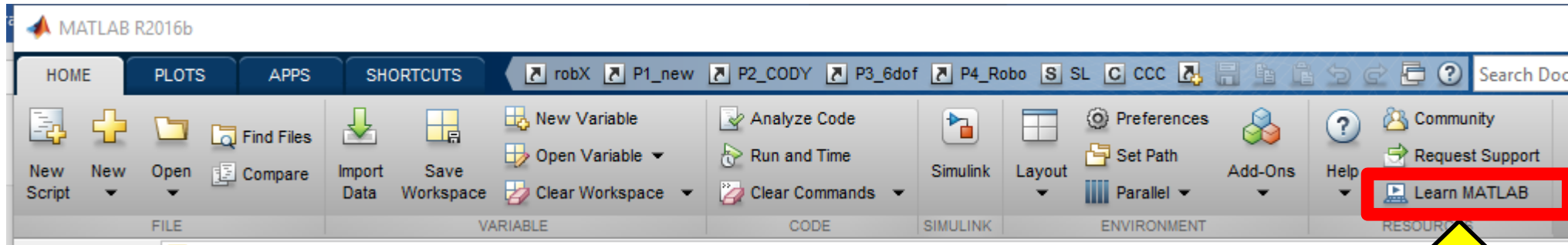
Below the code editor, a yellow warning box states: "The due date for this assignment has passed." At the bottom right, there are buttons for 'Reset', 'Test', and 'Submit', along with the text 'No limit'.

The 1st Stop: For students

For Students



- **MATLAB ACADEMY (the portal)**
 - Access a free interactive training course called **MATLAB Onramp**



1.)



2.)

Launch the *FREE*
course called
MATLAB OnRamp

The 1st Stop: For students

For Students



- **MATLAB Onramp**
 - Provided through your web browser
 - Introduction of programming concepts
 - Students answer questions ... and get IMMEDIATE feedback

The screenshot displays the MATLAB Onramp web interface. The browser address bar shows the URL `https://matlabacademy.mathworks.com/R2016a/portal.html`. The page header includes the MATLAB academy logo, the title "MATLAB Onramp" with a "3% complete" progress indicator, and the user name "Bradley Horton". The main content area is titled "6.1 Performing Array Operations on Vectors" and contains "Task 1".

Task 1

Info: MATLAB is designed to work naturally with arrays. For example, you can add a scalar value to all the elements of an array.

```
>> y = x + 2
```

Try adding `1` to each element of `v1` and store the result in a variable named `r`.

[Hint](#) [See Solution](#)

Task 2
Task 3
Task 4
Task 5
Task 6

The central workspace area shows the following MATLAB code:

```
>> load datafile
>> density = data(:,2);
>> v1 = data(:,3);
>> v2 = data(:,4);
```

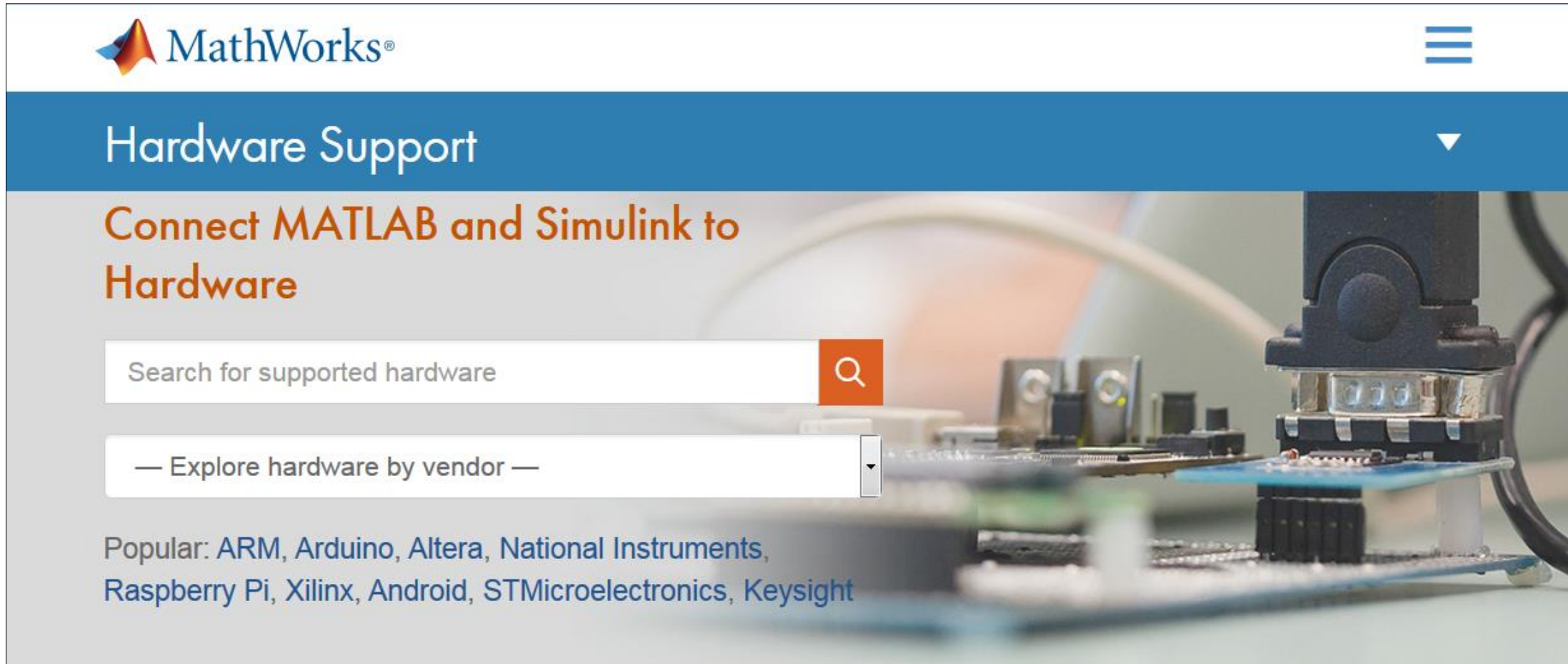
Below the code, the text "Task 1" is displayed. Overlaid on this area are two text boxes: a yellow box with the word "Free" and a pink box with the words "Interactive" and "tutorial" in blue italicized font.

Workspace

Name	Value	Size	Class
d...	7x4 ...	7x4	double
d...	[0.5... 7x1	7x1	double
v1	[4.0... 7x1	7x1	double
v2	[0.5... 7x1	7x1	double

Connecting to Hardware

<http://www.mathworks.com/hardware-support/home.html>



The screenshot shows the MathWorks Hardware Support page. At the top left is the MathWorks logo. To its right is a hamburger menu icon. Below the logo is a blue header bar with the text "Hardware Support" and a white downward-pointing triangle. Below the header bar is a large image of a circuit board with a microscope. Overlaid on the left side of the image is a search bar with the placeholder text "Search for supported hardware" and an orange search button with a white magnifying glass icon. Below the search bar is a dropdown menu with the text "— Explore hardware by vendor —". At the bottom left of the image, there is a list of popular hardware vendors: "Popular: ARM, Arduino, Altera, National Instruments, Raspberry Pi, Xilinx, Android, STMicroelectronics, Keysight".

MathWorks®

Hardware Support

Connect MATLAB and Simulink to Hardware

Search for supported hardware

— Explore hardware by vendor —

Popular: ARM, Arduino, Altera, National Instruments, Raspberry Pi, Xilinx, Android, STMicroelectronics, Keysight

Old slides

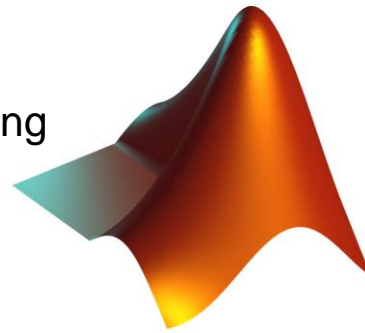
Today's agenda:

Phase 1

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

Phase 2

- Applying Computational Thinking
 - 3 Case Studies



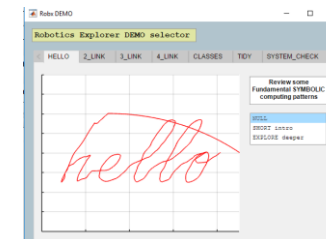
R2017a

Phase 3

- Resources for you and your students

Q/A

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

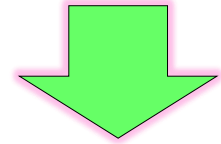


Using Computational Thinking and **MATLAB** to foster learning curiosity

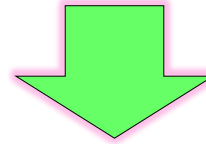
Centralization of thought process

Tedium busters

Modelling Choices

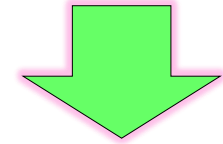


MATLAB
Live scripts



```
>> diff()
```

```
>> matlabFunctionBlock()
```



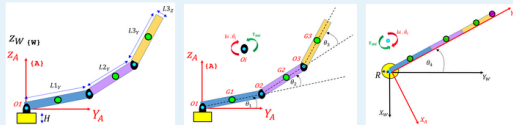
our_EOM(t) =

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

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:

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

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 terms

$$g(t) = \sin(z(t))^2$$

$$dg_dt(t) =$$

$$2 \cos(z(t)) \sin(z(t)) \frac{\partial}{\partial t} z(t)$$

