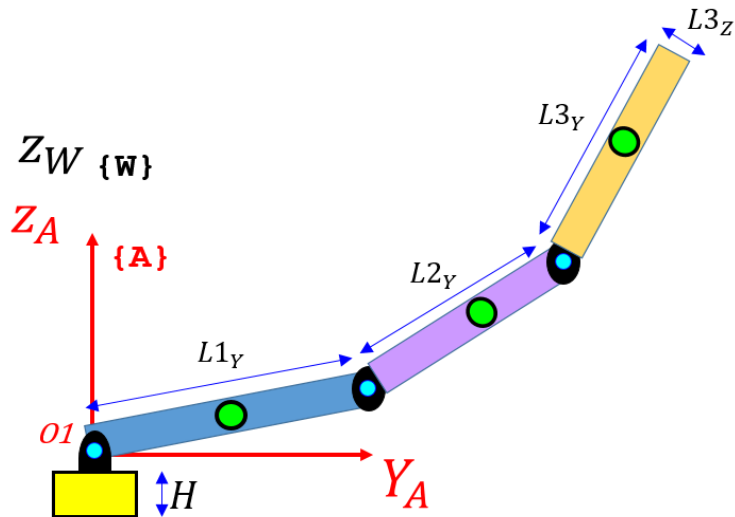


Teaching Lagrangian Dynamics

- a combination of symbolic and numeric computing

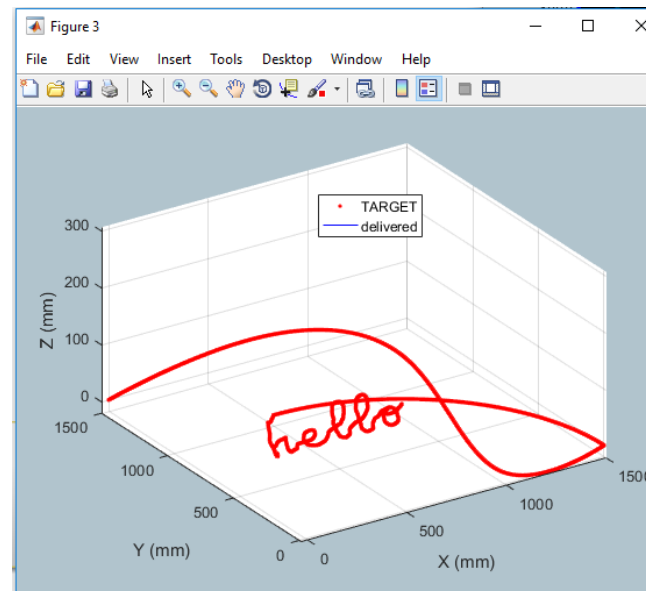
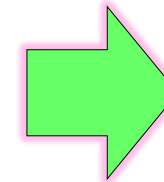
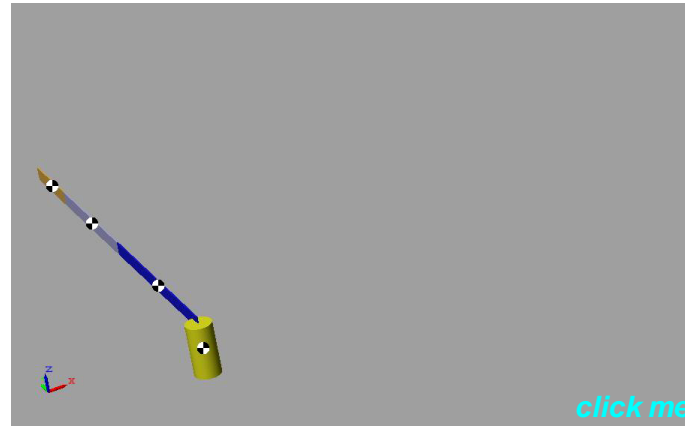
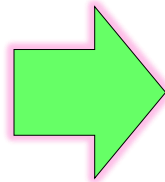
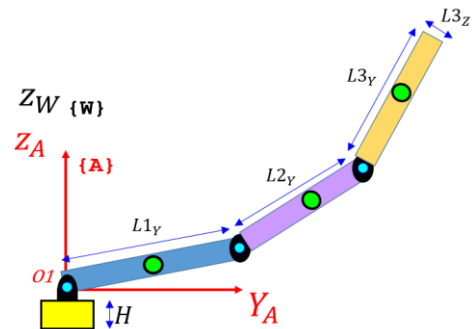


$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k$$

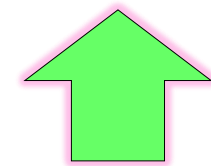
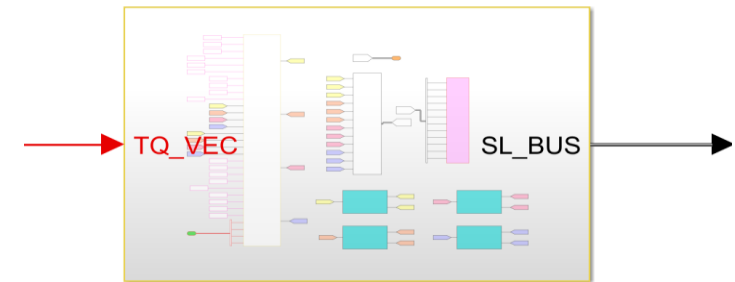
$$Q_k = \sum_{i=1}^{Nf_{nc}} \left(\vec{F}_i \cdot \frac{\partial \vec{v}_i}{\partial \dot{q}_k} \right) + \sum_{j=1}^{N\tau_{nc}} \left(\vec{\tau}_j \cdot \frac{\partial \vec{\omega}_j}{\partial \dot{q}_k} \right)$$



How do you make a robot write hello ?



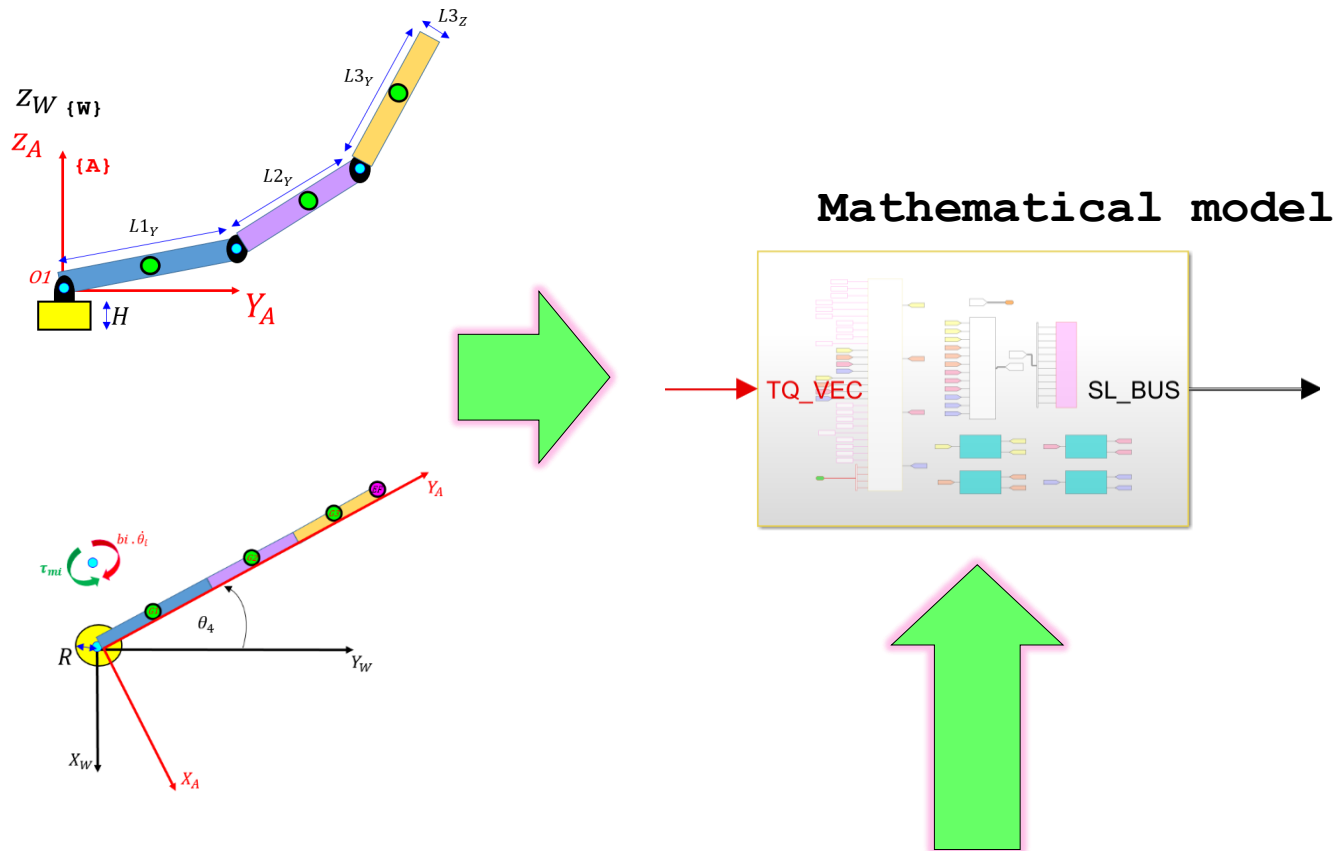
We need a mathematical model



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

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

How do you derive the mathematical model?



We need to understand the physics.

Interesting part

We need to apply Lagrange's equation

Laborious part

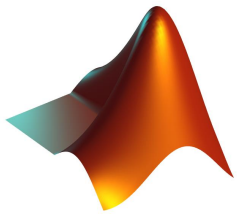
$$M(q) \cdot \ddot{q} + C(\dot{q}, q) \cdot \dot{q} + K(q) \cdot q + g(q) = Q$$

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

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

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

How do you derive the Mathematical model in MATLAB ?



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

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 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)
- k, b : Viscous damping torques

Bradley Horton : 13-Sep-2016, bhorton@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:

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

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: $L = T - V$ 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:

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

where:

- Q_k : is the generalised force associated with the k^{th} generalised co-ordinate q_k
- N_{nc} : is the number of active NON conservative forces
- N_{nt} : is the number of active NON conservative TORQUES
- \vec{r}_i : 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.

Note :

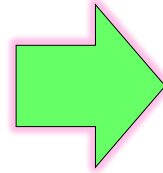
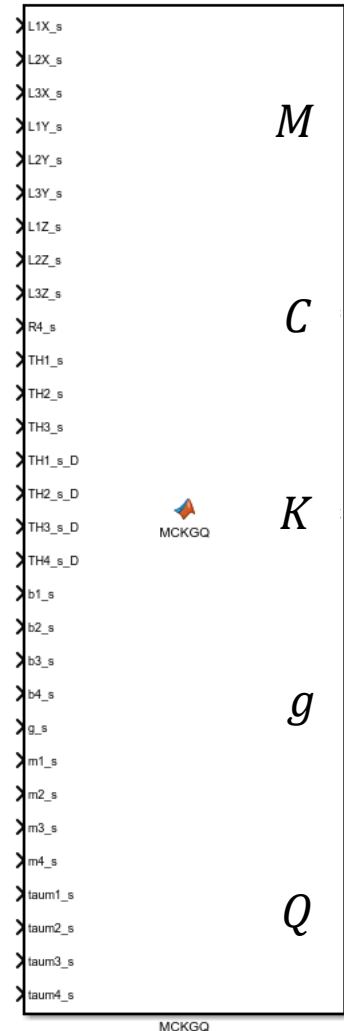
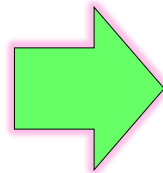
- For our system, we will choose $q = [\theta_1, \theta_2, \theta_3, \theta_4]$
- This is a 4 degree of freedom system, and as such there are four 2nd order ODEs that can be derived.

Defining Model parameters and geometry:

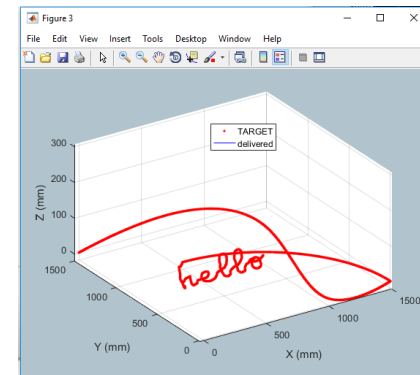
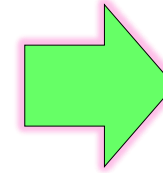
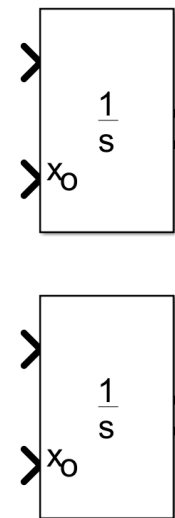
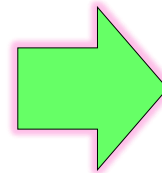
Here are some model parameters:

```

syms m2 s m1 s m3 s m4 s % masses
syms b1 s b2 s b3 s b4 s % damping
syms tau1 s tau2 s tau3 s tau4 s % motor torques
    
```

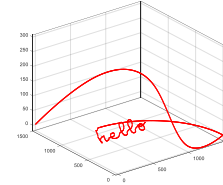
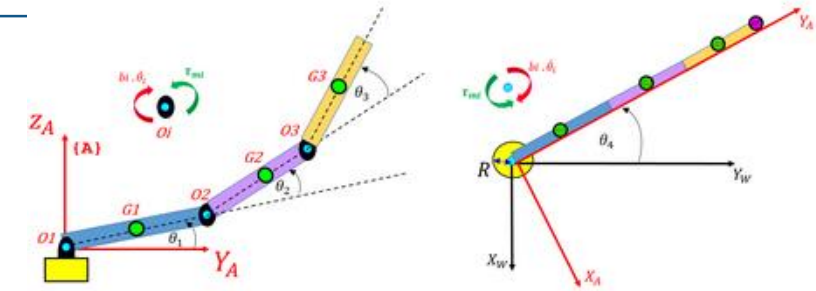
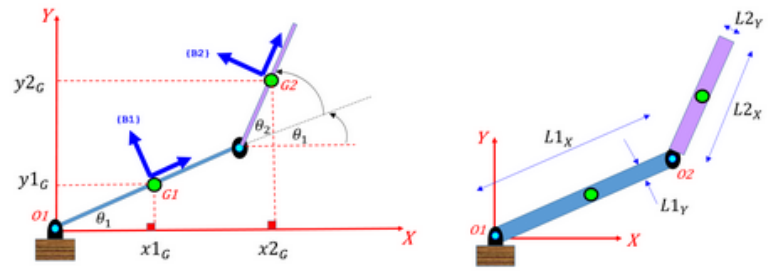


$$\ddot{q}$$



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

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

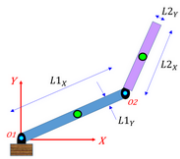
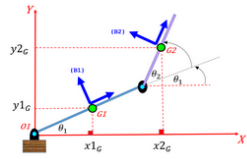
```

```

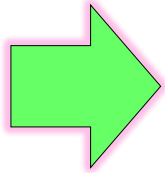
bh_tmp_EOM_file_WILL_BE_DELETED.txt
1 #####
2 ### q = TH1_s
3 ###
4 ### LHS of EOM is:
5 ###
6 1 (L1Y_s^2*TH1_s_DD*m1_s)/3
7 2 + L1Y_s^2*TH1_s_DD*m2_s
8 3 + L1Y_s^2*TH1_s_DD*m3_s
9 4 + (L2Y_s^2*TH1_s_DD*m2_s)/3
10 5 + L2Y_s^2*TH1_s_DD*m3_s
11 6 + (L2Y_s^2*TH2_s_DD*m2_s)/3
12 7 + L2Y_s^2*TH2_s_DD*m3_s
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

```

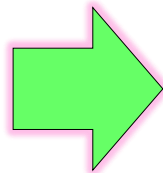
The role of Symbolic computing in your classroom:



smaller problems



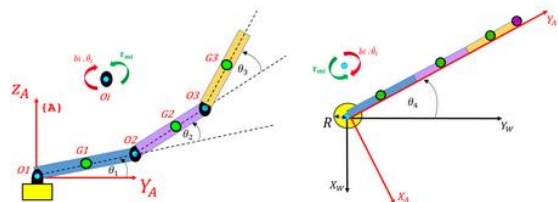
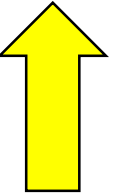
The understanding of the problem physics



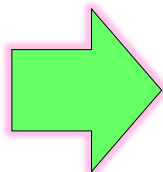
Hand written implementation

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

fundamental concepts



Bigger problems



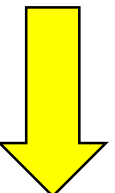
Symbolic computing implementation

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

Manual implementation

Automated implementation

interesting applications

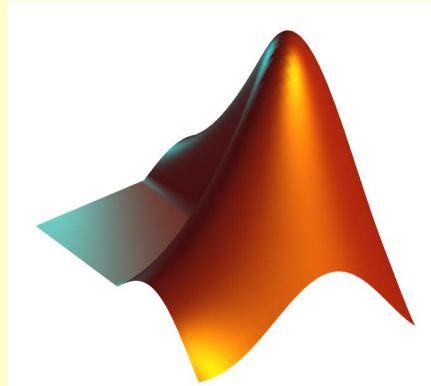


The role of **Symbolic computing with MATLAB** in your classroom:

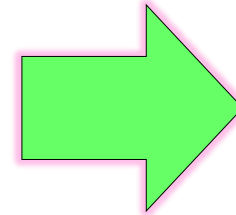
$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k$$

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

MATLAB



**Technical Computing
Environment**



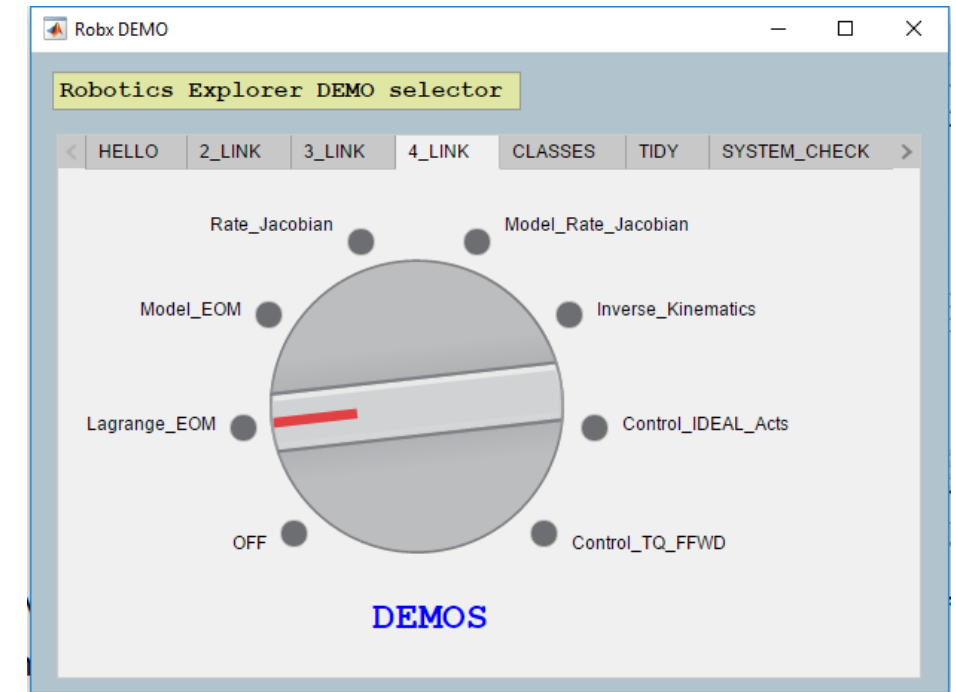
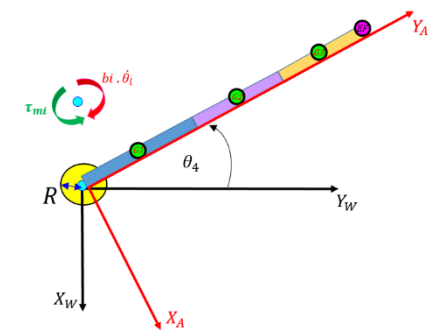
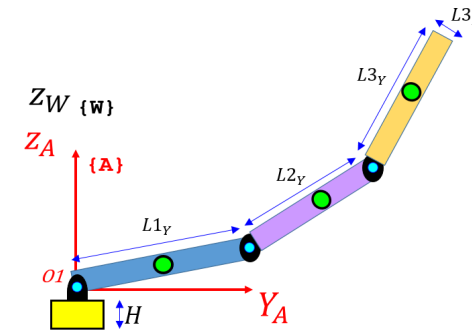
Enhancement of Understanding

- Build upon existing skills and experiences
- Provide choices on how to solve.
- Decompose BIG problems into several smaller problems.
- Provide self serve support

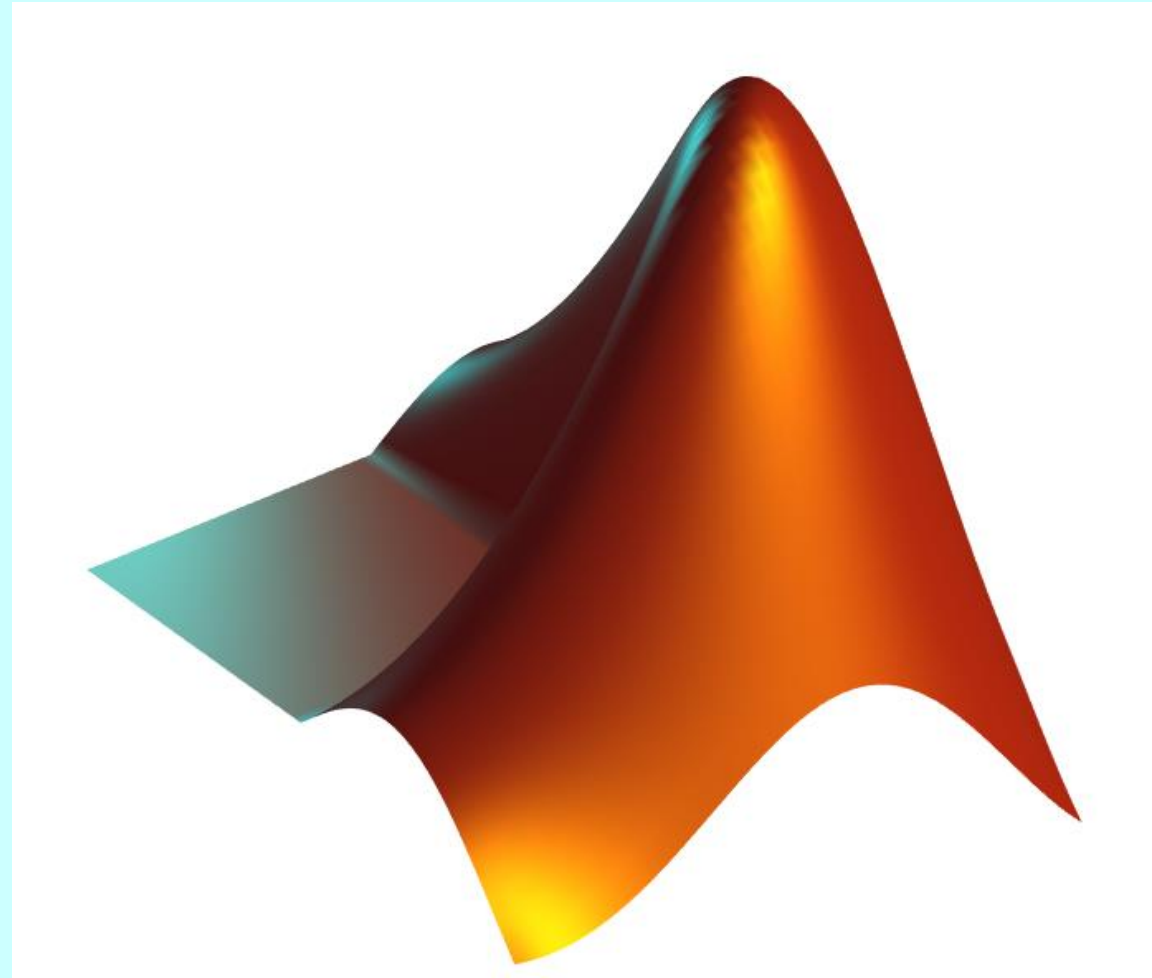
Agenda

Today's Agenda:

- **Symbolic Computing**
 - Review of some fundamental patterns in MATLAB
- Lagrangian Dynamics – part 1
 - **Manual** application
 - 2 LINK robot
- Lagrangian Dynamics – part 2
 - How to **automate** the application
 - 2 LINK robot
- Lagrangian Dynamics – part 3
 - Make a robot write hello
 - 4 LINK robot
- **BONUS session: Inverse kinematics**
 - Symbolic computing AND a constrained optimization problem
- Where can you find teaching resources ?
- Q/A
 - Would you like ALL of the examples that you've seen today ?

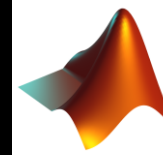


Demo these concepts



R2016b

Task: Symbolic computing patterns



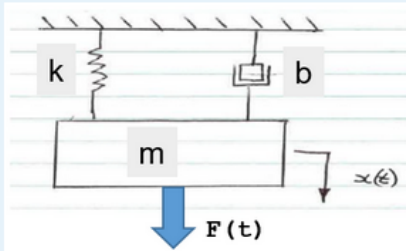
Live Script:

bh_short_intro_fundamental_symbolic_patterns.mlx

A Case Study - part 1: Deriving the equations of motion

Look at a simple application of Lagrange's equation ... say for simple spring, mass mechanical system:

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



```
clear; clc
```

Define some Symbolic variables that we can play with:

```
syms t x(t) m k b F
syms THE_X THE_XD THE_XDD
actual_list = [ x, diff(x,t), diff(x,t,2)];
HOLDER_list = [ THE_X, THE_XD, THE_XDD];
```

Define our system Lagrangian and Generalised force:

```
v = diff(x,t); % velocity
KE = 0.5*m*v^2; % KINETIC energy
PE = 0.5*k*x^2; % POTENTIAL energy
Q = F - b*v; % total NON conservative forces for deltaX
L = KE - PE % our Lagrangian
```

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

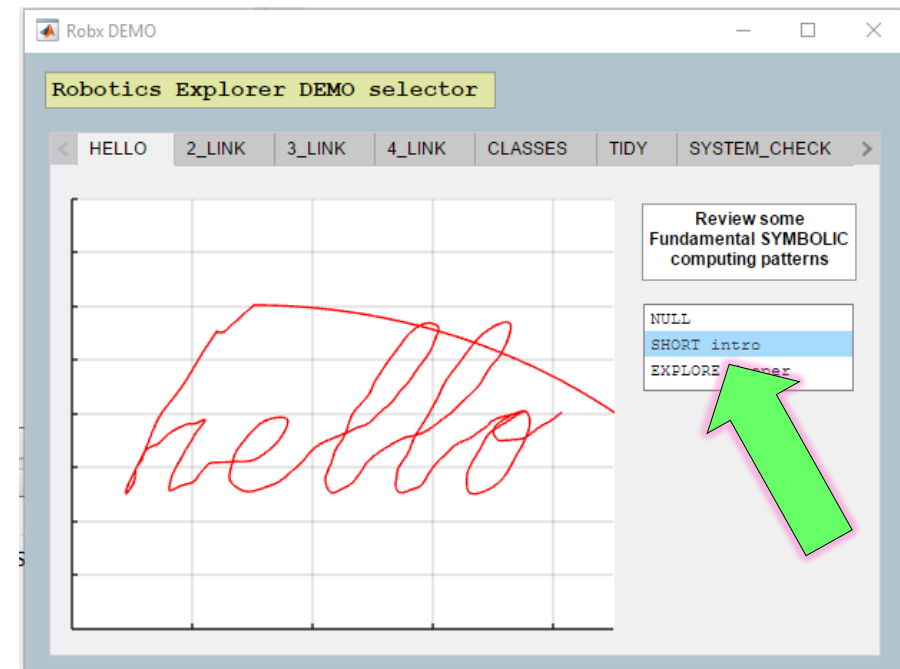
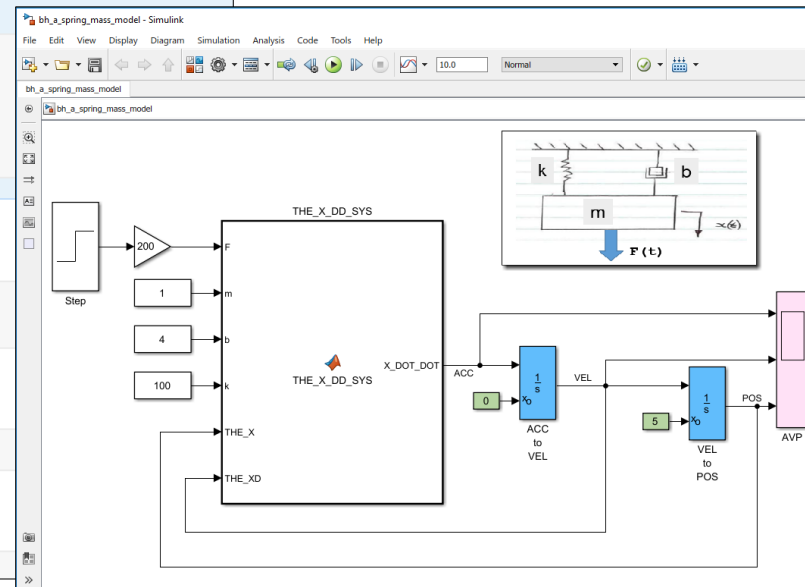
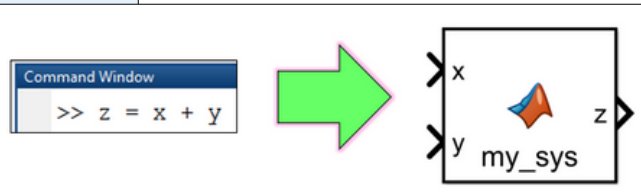
```
% OLD_LIST    NEW_LIST
L_new = subs(L, actual_list, HOLDER_list)
```

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

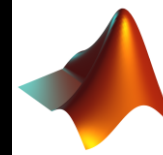
```
dLdx = diff(L_new, THE_X)
```

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

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



Task: 2-LINK manual application



Live Script:

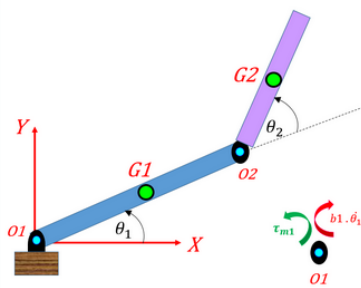
bh_LAGRANGE_double_PEND_MANUAL_derivation.mlx

Explore the dynamics of a DOUBLE compound Pendulum

In this example we're going to derive and then implement the equations of motion for a DOUBLE compound pendulum. 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:



STAGE 1: symbolic derivation of system equations

Euler-Lagrange equations of motion

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

where n is the DOF of the system, $\{q_1, q_2, \dots, q_n\}$ is a set of generalized coordinates associated with those coordinates, and the Lagrangian: $L = T - V$, is defined as the energy of the n -DOF system.

Note :

- For our system, we will choose $q = \{\theta_1, \theta_2\}$
- This is a 2 degree of freedom system, and as such there are two 2nd order

$$\text{Derive the } \theta_2 \text{ EOM: } \frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_2} - \frac{\partial L}{\partial \theta_2} = Q_2$$

Recall our variable mapping:

$$\begin{array}{cccc} \theta_1(t) & \dot{\theta}_1(t) & \theta_2(t) & \dot{\theta}_2(t) \\ \downarrow & \downarrow & \downarrow & \downarrow \\ q_1 & q_{1p} & q_2 & q_{2p} \end{array}$$

$$L = \text{subs}(L_ORIGINAL, [\text{thetal}(t), \text{diff}(\text{thetal}(t), t), \text{theta2}(t), \text{diff}(\text{theta2}(t), t)], [q_1, q_{1p}, q_2, q_{2p}]);$$

$$\text{Now let's do this part: } \frac{\partial L}{\partial \dot{\theta}_2} \Rightarrow \frac{\partial L}{\partial (q_{2p})}$$

$$\begin{aligned} \text{dLdq2p} &= \text{diff}(L, q_{2p}); \\ \text{dLdq2p} &= \text{subs}(\text{dLdq2p}, [\text{thetal}(t), \text{diff}(\text{thetal}(t), t), \text{theta2}(t), \text{diff}(\text{theta2}(t), t)], [q_1, q_{1p}, q_2, q_{2p}]); \end{aligned}$$

$$\text{Next, let's take the derivative w.r.t time: } \frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_2} \Rightarrow \frac{d}{dt} \frac{\partial L}{\partial (q_{2p})}$$

$$\text{der_dt_of_dLdq2p} = \text{diff}(\text{dLdq2p}, t);$$

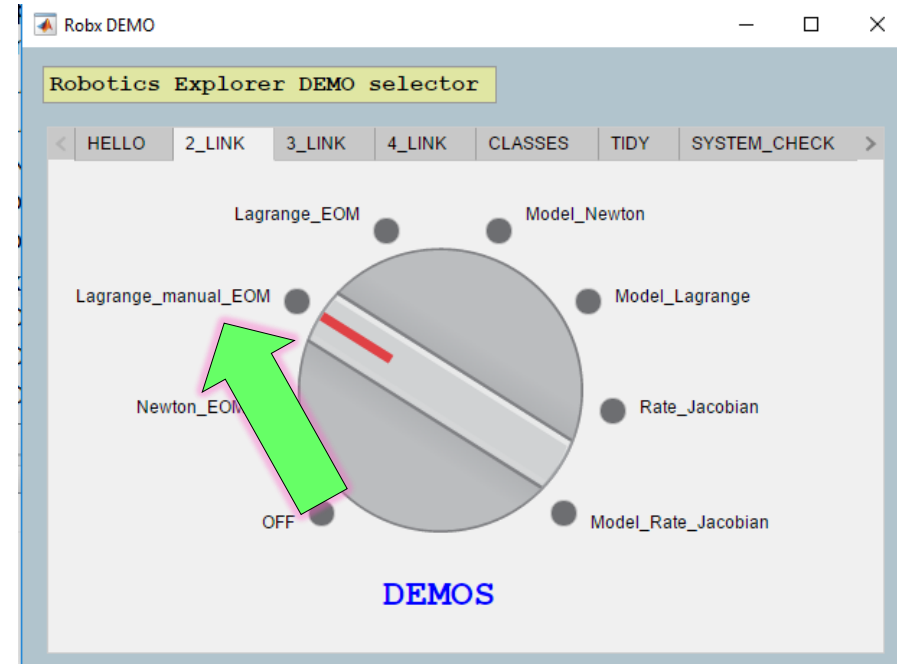
$$\text{Next, let's do this part: } \frac{\partial L}{\partial \theta_2} \Rightarrow \frac{\partial L}{\partial (q_2)}$$

$$\begin{aligned} \text{dLdq2} &= \text{diff}(L, q_2); \\ \text{dLdq2} &= \text{subs}(\text{dLdq2}, [\text{thetal}(t), \text{diff}(\text{thetal}(t), t), \text{theta2}(t), \text{diff}(\text{theta2}(t), t)], [q_1, q_{1p}, q_2, q_{2p}]); \end{aligned}$$

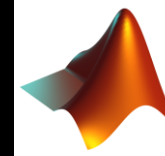
So our second equation of motion is:

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_2} - \frac{\partial L}{\partial \theta_2} = Q_2$$

```
syms Q2_s
EOM_TH2_LHS = der_dt_of_dLdq2p - dLdq2;
EOM_TH2_LHS = simplify(EOM_TH2_LHS);
EOM_TH2_RHS = Q2_s;
EOM_TH2 = EOM_TH2_LHS == EOM_TH2_RHS;
```



Task: automating the application



```

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);

    dLdq = diff(L, THE_qp);

    % OLD          NEW
    dLdq = subs(dLdq, states_holder_list, states_actual_list);
    der_dt_of_dLdq = diff(dLdq, t);

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

    eom_LHS = der_dt_of_dLdq - 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 );

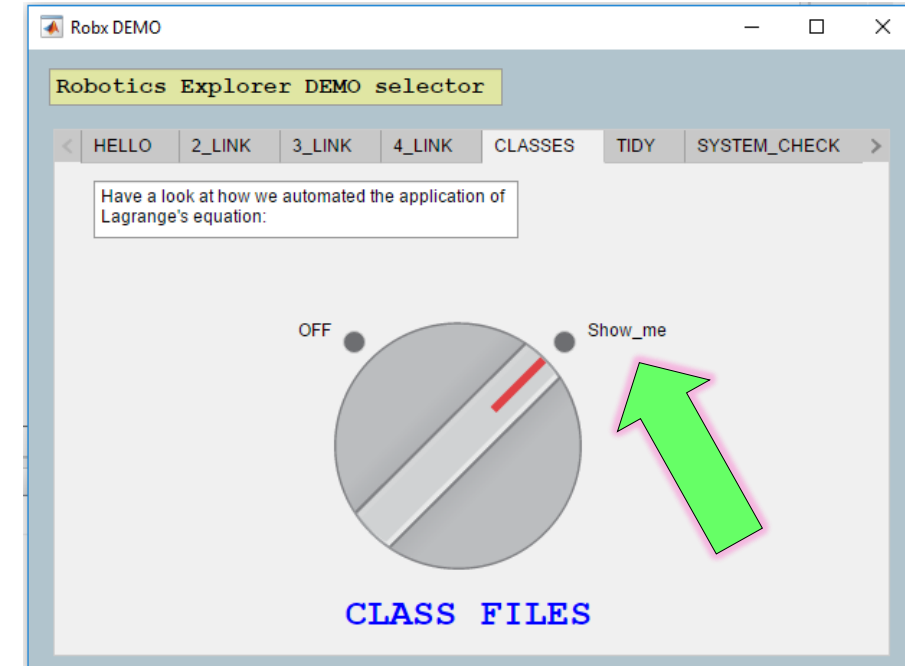
    % now store into a struct array
    EOM(kk).actual_eom_LHS = eom_LHS;
    EOM(kk).actual_eom_RHS = eom_RHS;
    EOM(kk).actual_eom_EQ = eom_LHS == eom_RHS;

    % 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
    
```

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

Class

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



Task: automating the application

```
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;
```

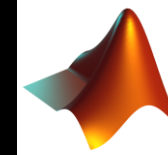
```
                OBJ.holder_list_SYM_vel;
```

```
                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

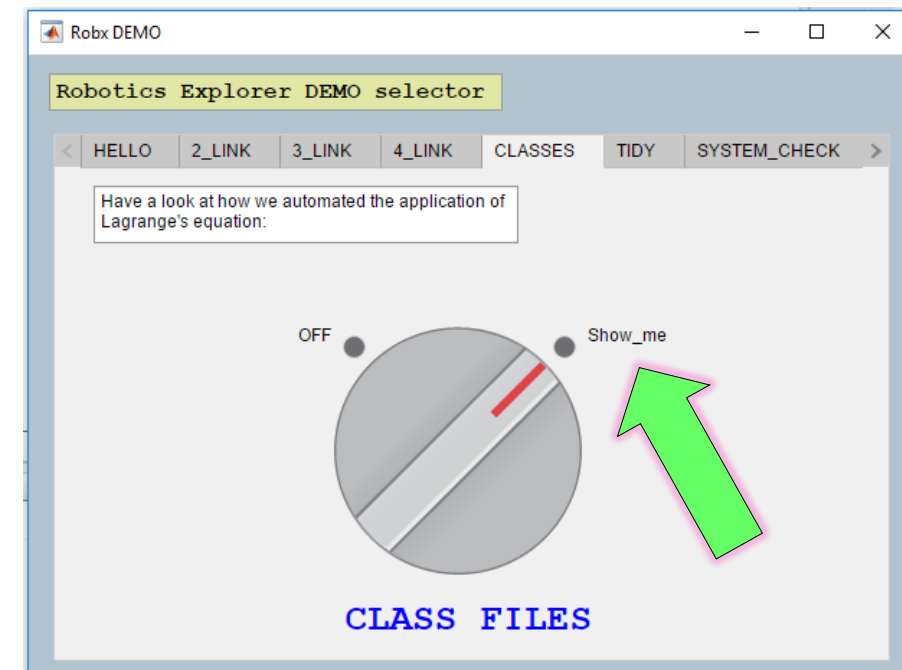
bh_eom_CLS.m

bh_genF4manips_CLS.m

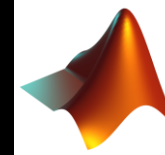
bh_lagr4manips_CLS.m

bh_MCKGQ_CLS.m

bh_qman4manips_CLS.m



Task: 2-LINK automate application



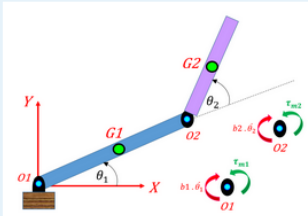
Explore the dynamics of a DOUBLE compound Pendulum

In this example we're going to derive and then implement the equations of motion for a DOUBLE compound pendulum. 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 : 01-Aug-2016, bradley.horton@mathworks.com

STAGE 1: symbolic derivation of sy

Euler-Lagrange equations of motion

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

where n is the DOF of the system $\{q_1, q_2, \dots, q_n\}$ is a set

Apply Lagrange's equation:

So let's now apply Lagrange's equation:

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

First we'll define our generalised co-ordinates. I'm going to use 2 sets of these generalised co-ordinates:

- the **ACTUAL** set of symbols are our "proper" set of symbols
- the **HOLDER** set are for easier expression manipulation

```
actual_list_SYM_pos = formula( [ theta1, theta2 ] );  
holder_list_SYM_pos = [ TH1_s, TH2_s ];
```

Automate

OK: let's create a Lagrangian object using the class <bh_lagr4manips_CLS>

```
lag_OBJ = bh_lagr4manips_CLS( KE, PE, actual_list_SYM_pos, holder_list_SYM_pos );
```

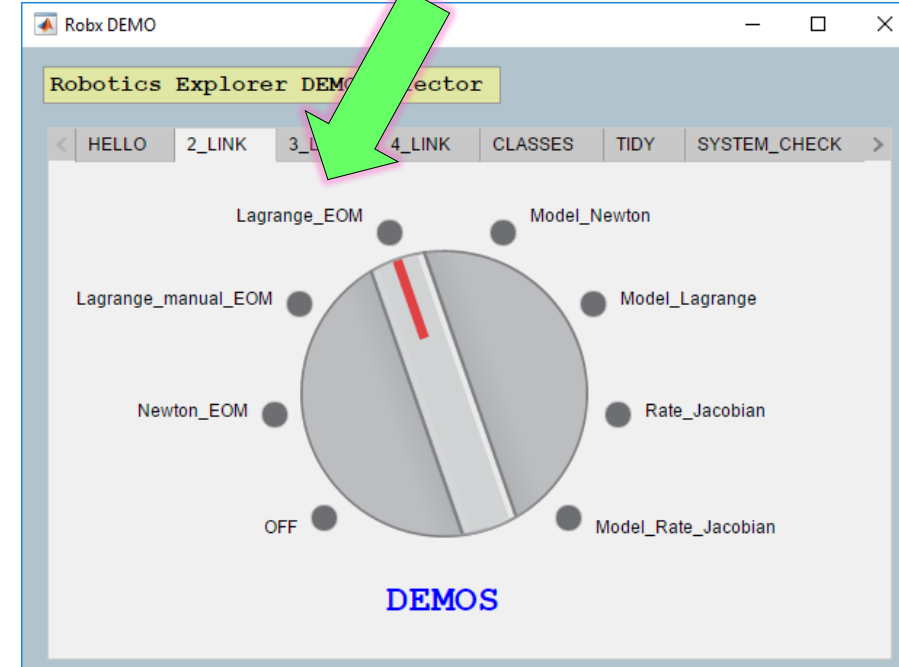
And let's compute the system's equations of motion:

```
lag_OBJ = lag_OBJ.calc_eom()
```

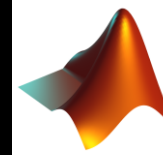
So what do the equations of motion actually look like ?

Live Script:

bh_LAGRANGE_double_PEND.mlx



Task: 4-LINK automate application



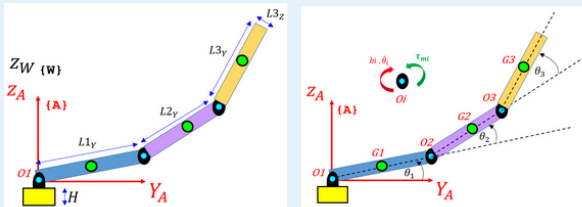
Explore the dynamics of a 4-dof Robotic manipulator

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)
- $b \cdot \dot{\theta}$: 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 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, $\{F_i, \frac{\partial v_i}{\partial q_k}\}$ are generalized forces associated with those coordinates, and the Lagrangian: $L = T - V$ between the kinetic and potential energy of the n - DOF system. The Generalised of the non conservative forces and torques acting on the multibody system. The force acting on the system is:

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

where:

Now let's get the M,C,K,G,Q matrices:

We can express our system equations of motion in the following form:

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

```
lag_OBJ = lag_OBJ.create_MCKGQ();
```

Retrieve the MCKGQ struct:

```
res_T = lag_OBJ.get_MCKGQ();
```

And let's have a look at each of these terms:

Here's M:

```
res_T.M
```

Here's C:

```
res_T.C
```

Here's K:

```
res_T.K
```

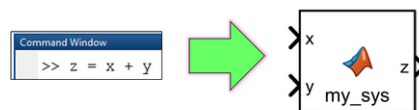
Here's G:

```
res_T.G
```

Here's Q:

```
res_T.Q
```

Now create the MATLAB function blocks for Simulink:

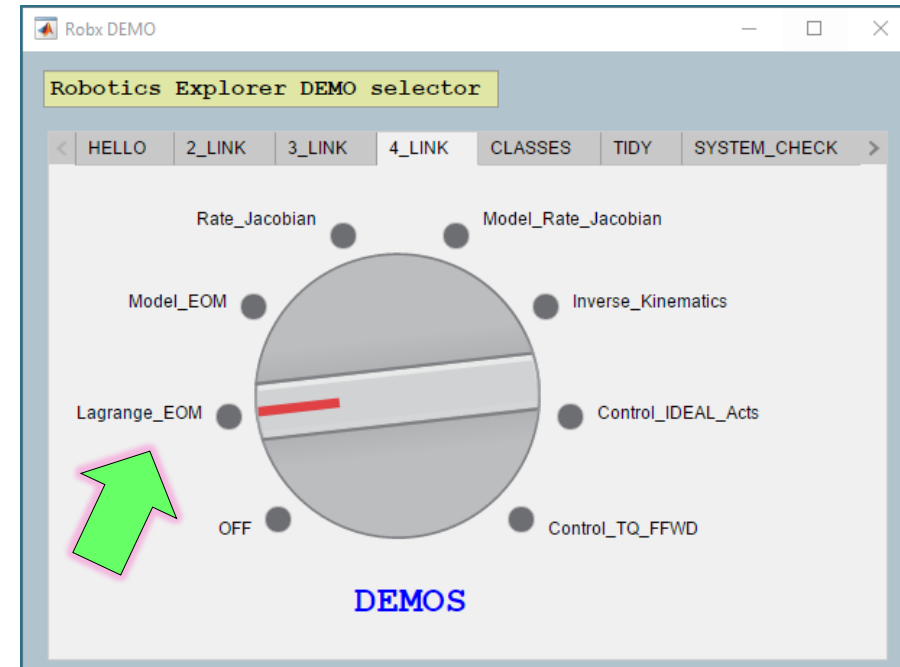


To use/solve these derived equations of motion we'll create a MATLAB Function block that can be used inside Simulink:

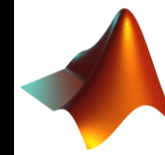
```
lag_OBJ.create_MLF_blocks();
```

Live Script:

bh_LAGRANGE_4dof_manipulator.mlx



Task: Inverse kinematics



Live Script:

bh_invKIN_4dof_manipulator_NUMERICAL_OPTIM.mlx

Solving the Inverse KINEMATIC problem numerically - part 2

So we're going to formulate and then solve a constrained optimization problem. The "general" form of a constrained optimization problem is shown below:

$$\min_x J(x) \text{ such that } \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub, \end{cases}$$

- $J(x)$ is a Cost function: it is the thing that we need to minimize. In our case we have:
- $J(x) = \| \text{DESIRED_XYZ} - \text{FORWARD_KINEMATICS}(\theta_1, \theta_2, \theta_3, \theta_4) \|$, where: $\| \vec{p} \| = \sqrt{p_x^2 + p_y^2 + p_z^2}$
- " x " is a vector of design variables, ie: the things that we need to determine in order to minimize the cost function. In our case we have:
- $x = \{ \theta_1, \theta_2, \theta_3, \theta_4 \}$

Now we're going to use the `fmincon()` function to solve our optimization problem. The format that we need to package our problem into is this:

```
x = fmincon( J_fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, opts)
```

So let's start setting up the problem to solve.

```
opts_T = optimset;
opts_T.Display = 'off';
```

! Here are the lower and upper bounds for our 4 joint angles $\{ \theta_1, \theta_2, \theta_3, \theta_4 \}$ - **NOTE**, how we're asking for a solution where $\theta_1 \geq 0$

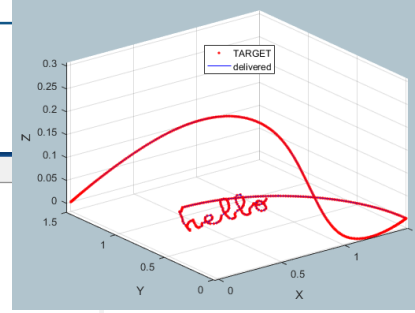
```
qa_lb = [ 0; -pi; -pi; -pi]; % LOWER bounds for angles
qa_ub = [ pi; pi; pi; pi]; % UPPER bounds for angles
```

Here's an initial guess for what we think the solution is:

```
qa = [ 0.5; 0.5; 0.5; 0.5]; % initial guess
```

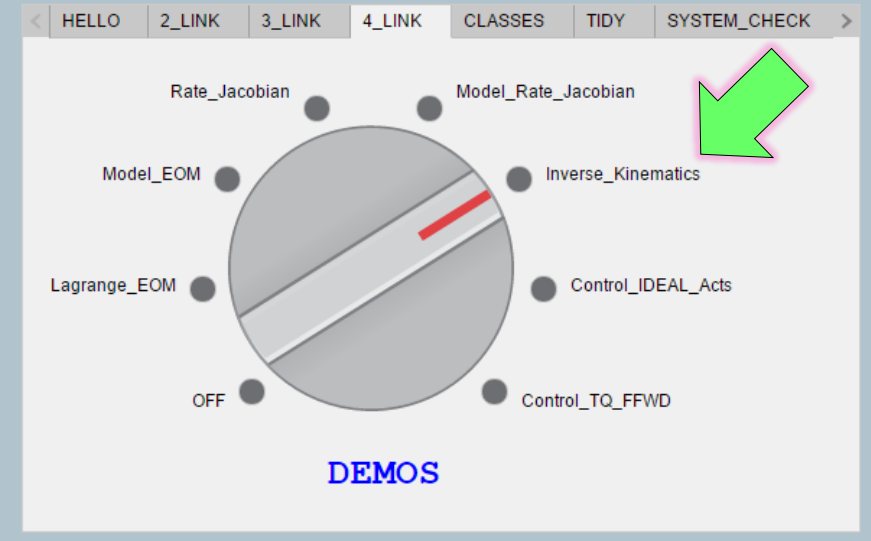
Now at the heart of this approach is the utilization of our FORWARD KINEMATICS function that we derived earlier:

- `XYZ_EFF = bh_xyz_for_4dof_manip(L1Y_s, L2Y_s, L3Y_s, theta1, theta2, theta3, theta4)`
- Forward Kinematic Solution
 $(\theta_1, \theta_2, \theta_3, \theta_4) \rightarrow f(\theta_1, \theta_2, \theta_3, \theta_4) \rightarrow (X_E, Y_E, Z_E)$



Robx DEMO

Robotics Explorer DEMO selector

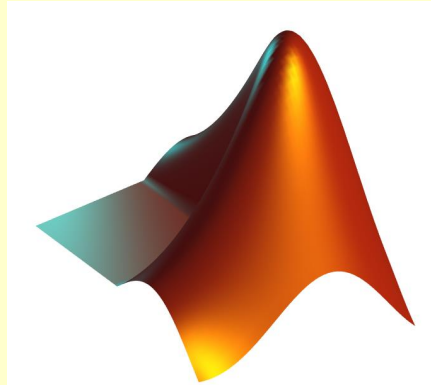


The role of **Symbolic computing with MATLAB** in your classroom:

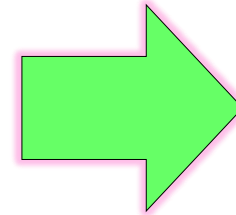
$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k$$

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

MATLAB



**Technical Computing
Environment**



Enhancement of Understanding

- Build upon existing skills and experiences
- Provide choices on how to solve.
- Decompose BIG problems into several smaller problems.
- Provide self serve support

Teaching and Learning Resources.


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

MATLAB Courseware

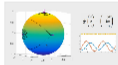
Search MathWorks.com

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.

<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>

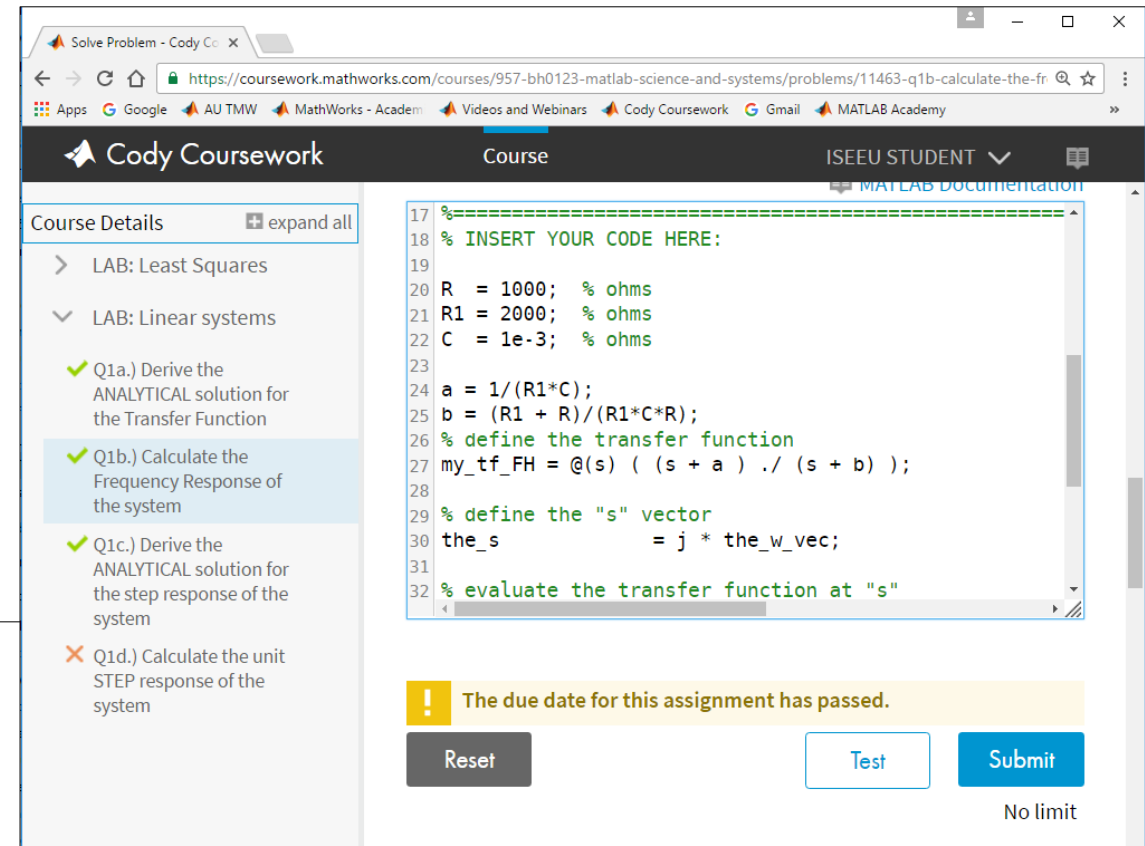
Create Report : Assignment 1

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 for a MATLAB assignment. The browser address bar displays the URL: <https://coursework.mathworks.com/courses/957-bh0123-matlab-science-and-systems/problems/11463-q1b-calculate-the-fr>. The page title is "Solve Problem - Cody Co...". The course name is "Cody Coursework" and the user is "ISEEU STUDENT".

The left sidebar shows the "Course Details" section with a list of problems:

- LAB: Least Squares
- LAB: Linear systems
- Q1a.) Derive the ANALYTICAL solution for the Transfer Function
- Q1b.) Calculate the Frequency Response of the system
- Q1c.) Derive the ANALYTICAL solution for the step response of the system
- Q1d.) Calculate the unit STEP response of the system

The main content area displays the MATLAB code for the assignment:

```

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"
    
```

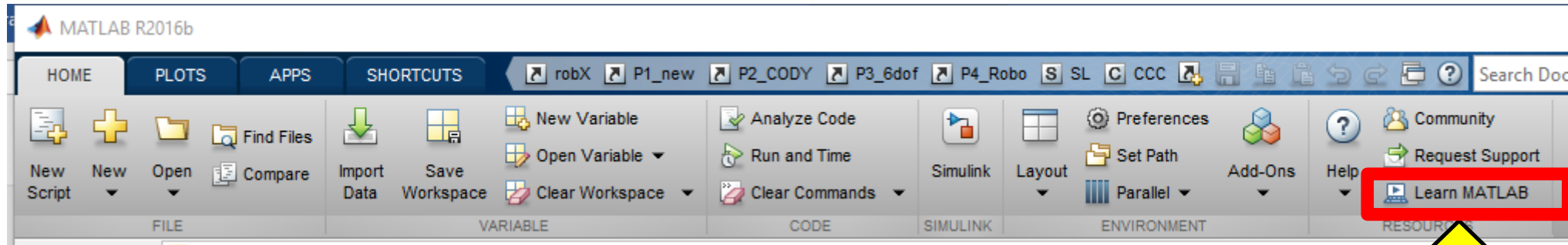
Below the code editor, there is a warning message: "The due date for this assignment has passed." with buttons for "Reset", "Test", and "Submit". The "Submit" button is disabled, and the text "No limit" is shown below it.

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`.

Buttons: `Hint`, `See Solution`

Task list on the left: Task 2, Task 3, Task 4, Task 5, Task 6

Code Editor:

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

Task 1
>>
```

Workspace:

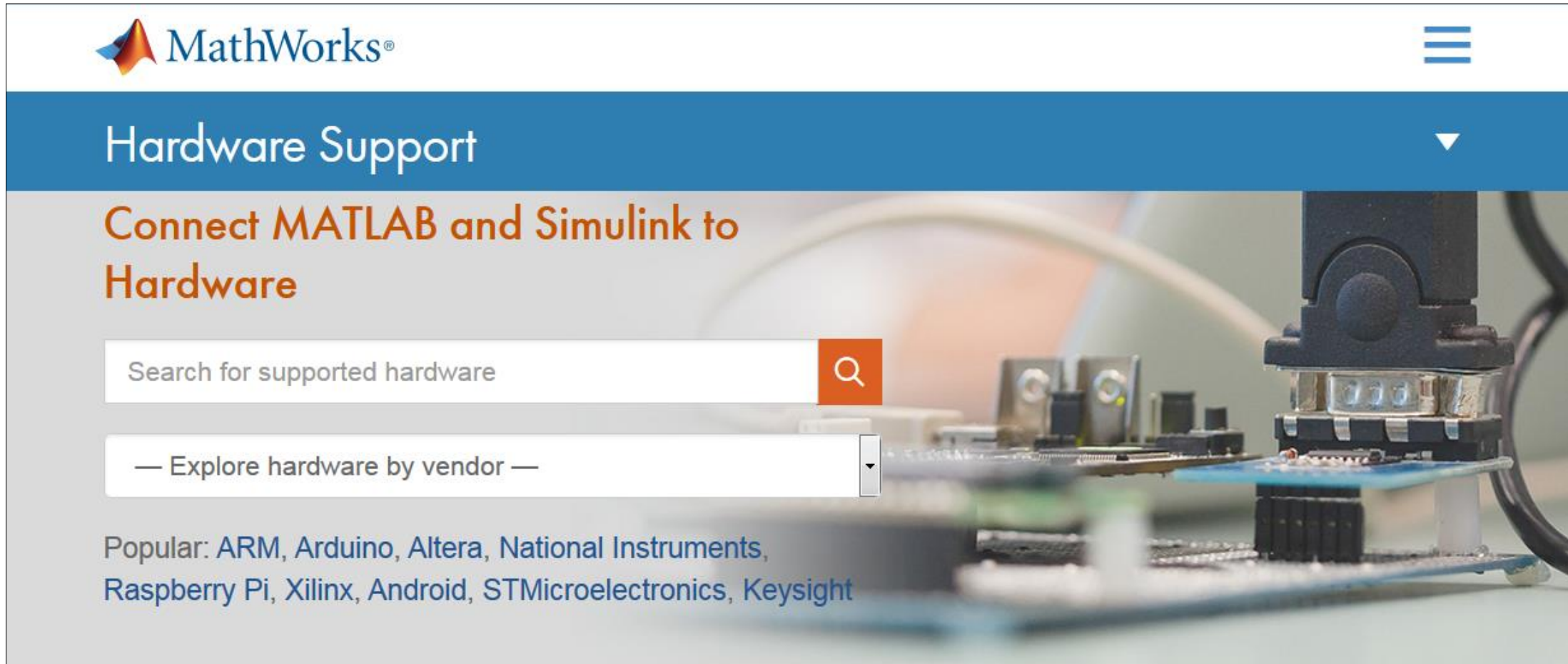
Name	Value	Size	Class
d...	7x4 ...	7x4	double
d...	[0.5... 7x1	7x1	double
v1	[4.0... 7x1	7x1	double
v2	[0.5... 7x1	7x1	double

Overlaid text boxes:

- A yellow box with the word "Free".
- A pink box with the text "Interactive tutorial".

Connecting to Hardware

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



The screenshot shows the MathWorks Hardware Support page. At the top is the MathWorks logo and a hamburger menu icon. Below this is a blue header with the text "Hardware Support" and a downward arrow. The main content area has a background image of a circuit board being tested by a probe. The text "Connect MATLAB and Simulink to Hardware" is prominently displayed in orange. Below this is a search bar with the placeholder text "Search for supported hardware" and a magnifying glass icon. Under the search bar is a dropdown menu with the text "— Explore hardware by vendor —". At the bottom, a list of popular hardware vendors is shown: ARM, Arduino, Altera, National Instruments, Raspberry Pi, Xilinx, Android, STMicroelectronics, and 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

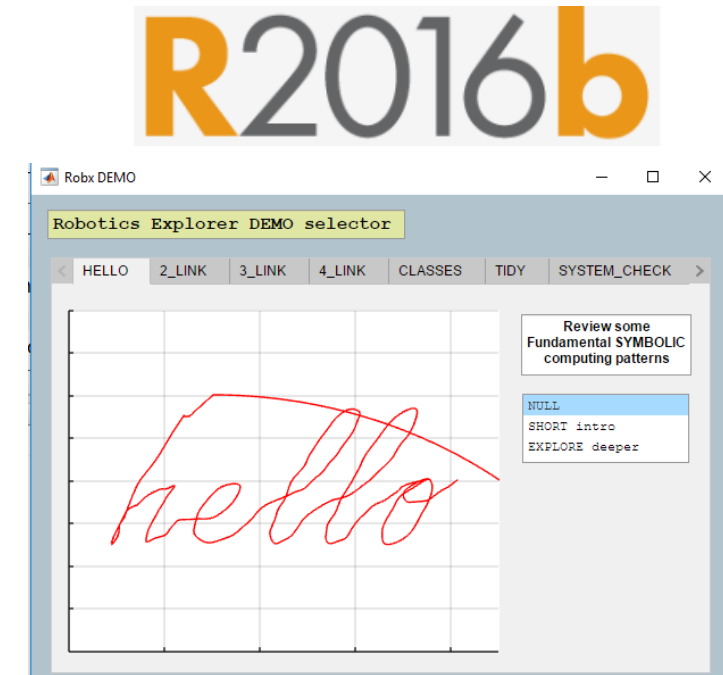
Wrap up

Q/A:

- Are there some questions please ?
- Download the examples that you saw today ... and more that you didn't !
- .. And I have 1 question for you*

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

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



>> bh_robx_startup