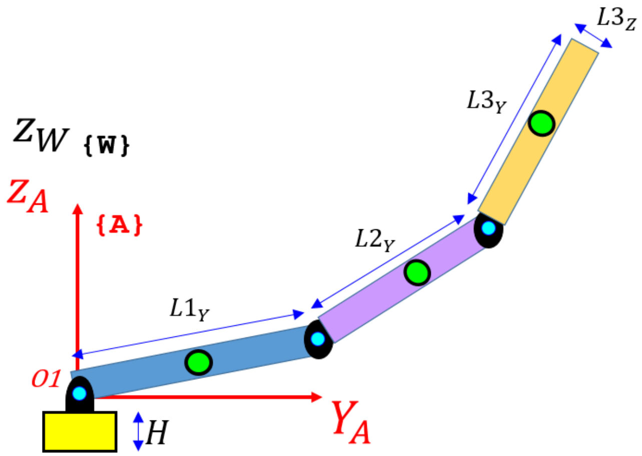
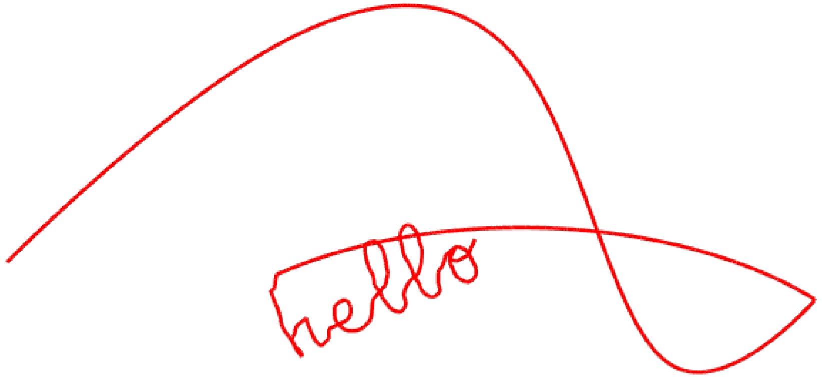
Teaching Rigid Body Dynamics

- a combination of symbolic and numeric computing





Brad Horton
Engineer
MathWorks

T4

Todays agenda:

Phase 1

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

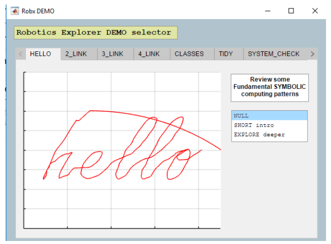
Phase 2

- Applying Computational Thinking
  - **3 Case Studies**

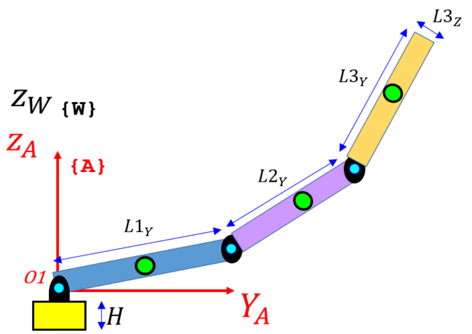
R2017a

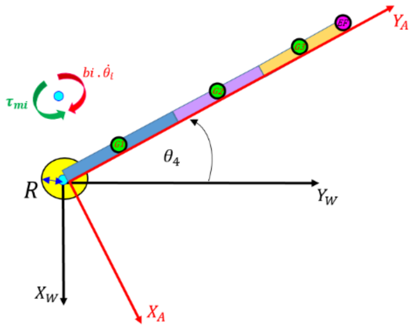
Phase 3

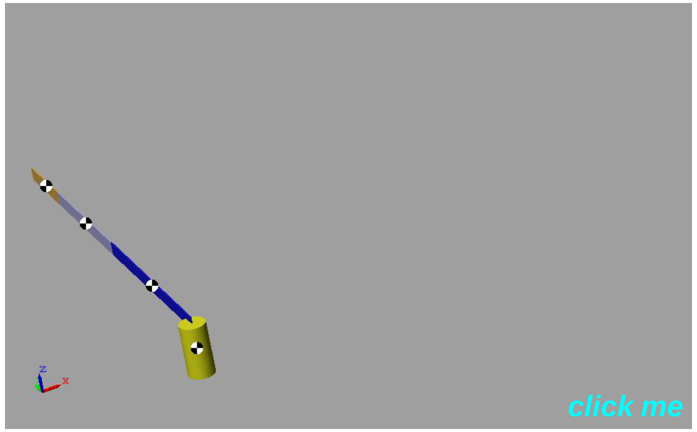
- Questions AND Answers
- How do you get ALL of the examples that you saw today ?

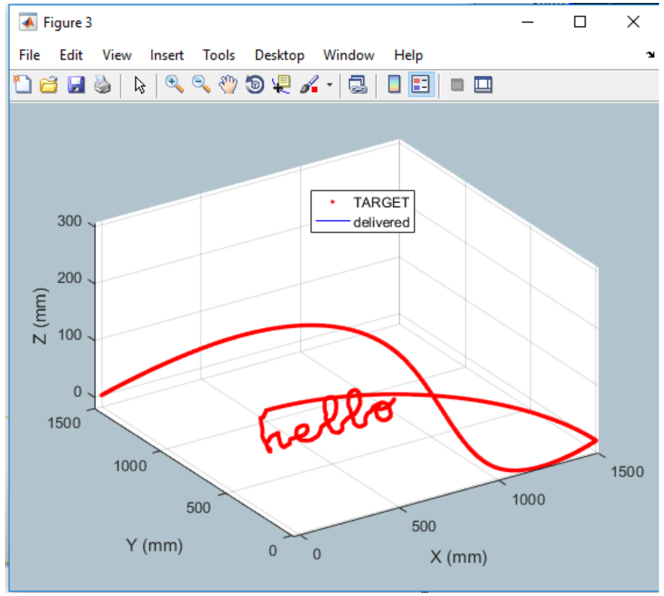


How do you make a robot write hello ?

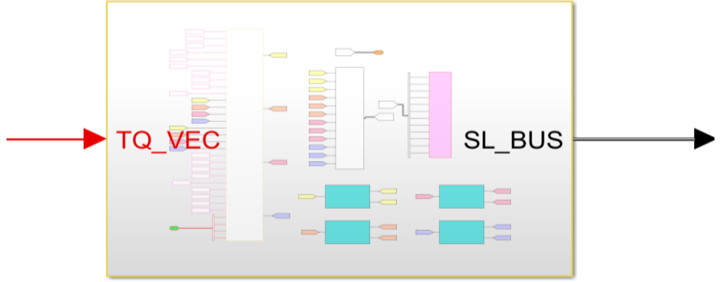








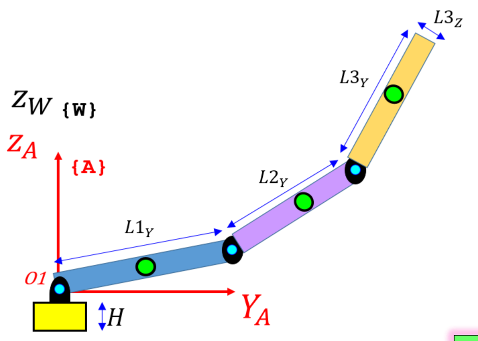
We need a mathematical model

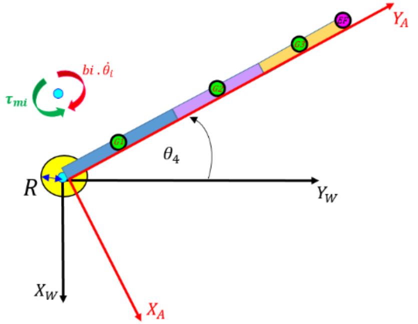


$$M(q).\ddot{q} + C(\dot{q},q).\dot{q} + K(q).q + g(q) = Q$$

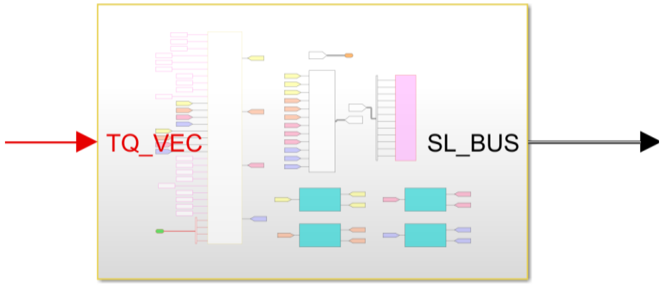
$$\ddot{q} = [M(q)]^{-1}.[Q - C(\dot{q},q).\dot{q} - K(q).q - g(q)]$$

How do you derive the mathematical model?





Mathematical model



We need to understand the physics.

Interesting part

We need to apply Lagrange's equation

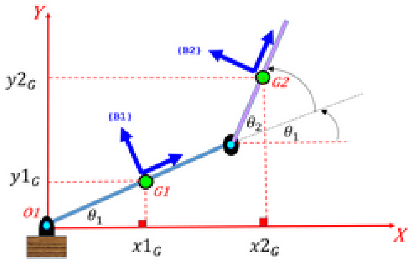
Laborious part

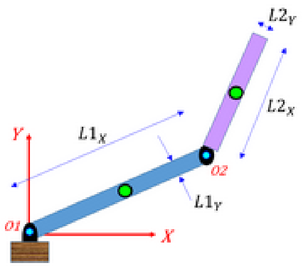
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k$$

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

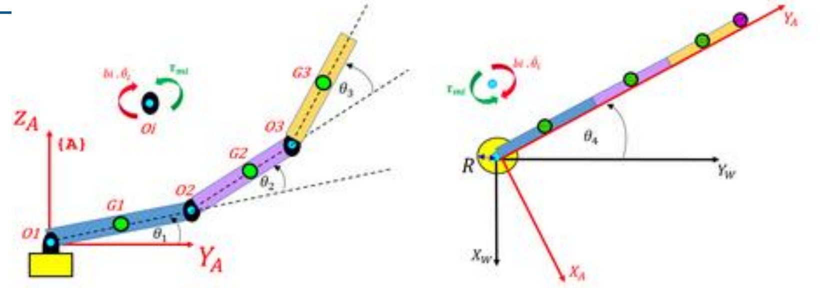
$$M(q).\ddot{q} + C(\dot{q},q).\dot{q} + K(q).q + g(q) = Q$$

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





Laborious ?



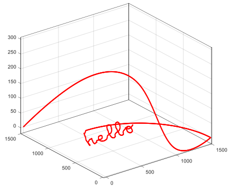
bh_tmp_EOM_file_WILL_BE_DELETED.txt

```
##############################################################
### 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 30 lines

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



4-dof

Approx 200 lines

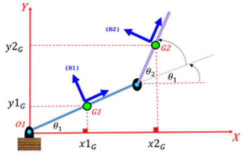
$$\left\{\begin{matrix}\ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \\ \ddot{\theta}_4\end{matrix}\right.$$

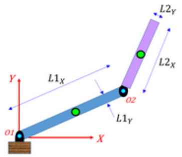
bh_tmp_EOM_file_WILL_BE_DELETED.txt

```
##############################################################
### 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
```

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

Encouraging Deeper Learning engagements in your classroom:





smaller problems

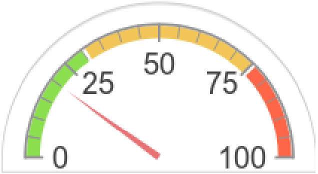
The understanding of the problem physics:

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

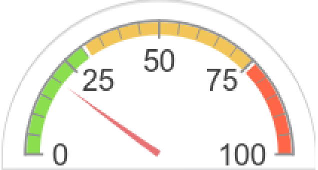
Problem Solving and practice

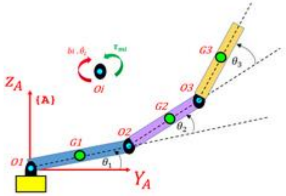
Hand written implementation

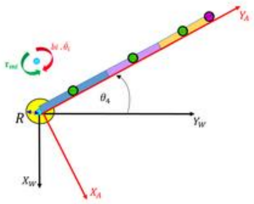
BRAIN
Conceptual Difficulty



HAND
Computational Difficulty







Bigger problems

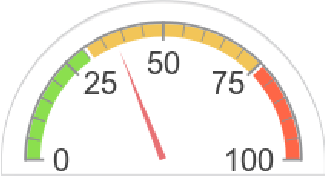
The understanding of the problem physics:

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

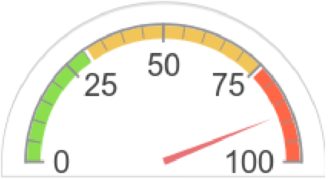
Problem Solving and practice

Hand written implementation

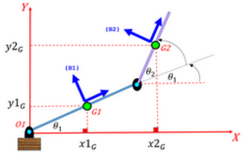
BRAIN
Conceptual Difficulty

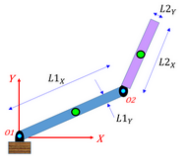


HAND
Computational Difficulty



Encouraging Deeper Learning engagements in your classroom:





smaller problems

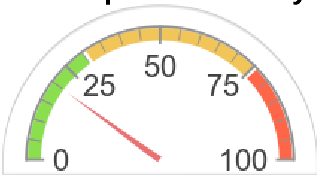
The understanding of the problem physics:

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

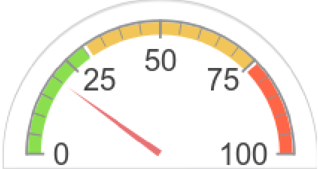
Problem Solving and practice

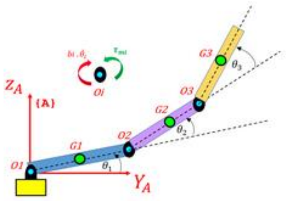
Hand written implementation

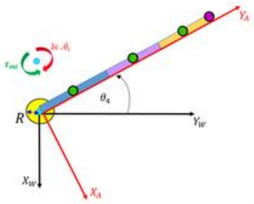
BRAIN
Conceptual Difficulty



HAND
Computational Difficulty







Bigger problems

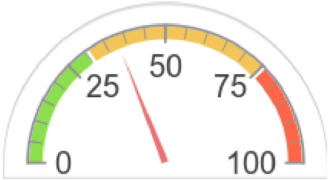
The understanding of the problem physics:

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

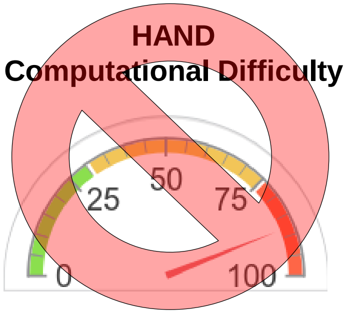
Problem Solving and practice

Computational Thinking:
- Brain
- Technology

BRAIN
Conceptual Difficulty



HAND
Computational Difficulty



Enabling Computational Thinking using MATLAB

Problem Solving and practice

Computational Thinking:

- Brain
- Technology

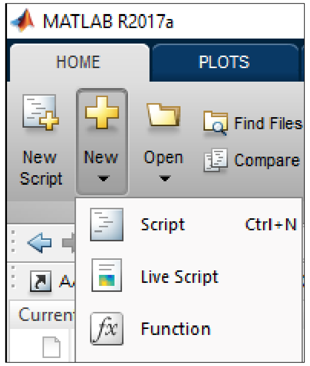
Decomposition

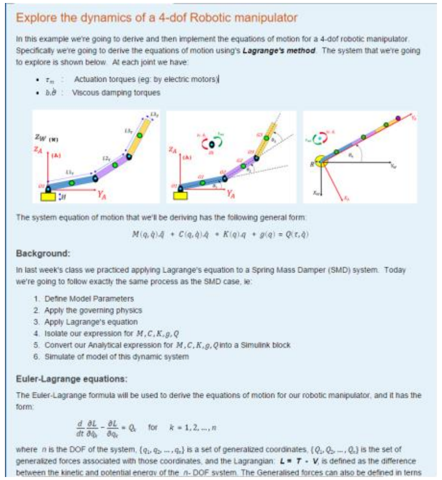
Algorithms + Automation

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'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: $\boldsymbol{L = T - V}$, 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

---

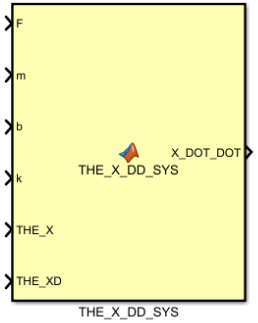
Algorithms + Automation

Symbolic Computing

```
>> diff()

>> matlabFunctionBlock()
```

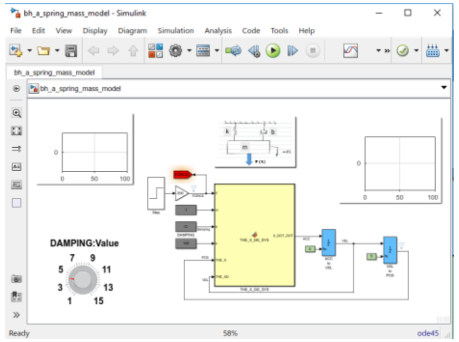
our_EOM(t) =

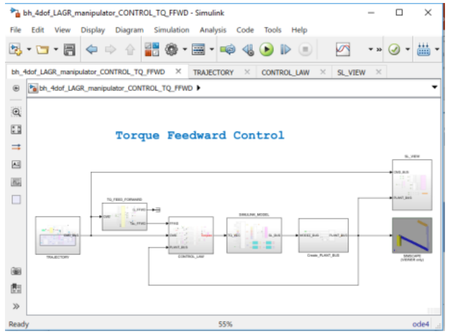
$$m\frac{\partial^2}{\partial t^2}x(t) + k\,x(t) = F - b\frac{\partial}{\partial t}x(t)$$



Simulation

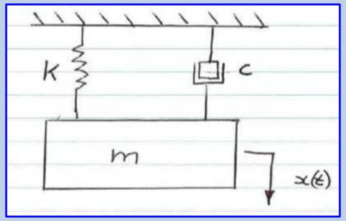
Numeric via Block Diagram



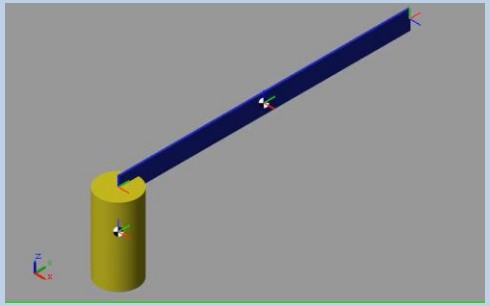


Today's case studies:

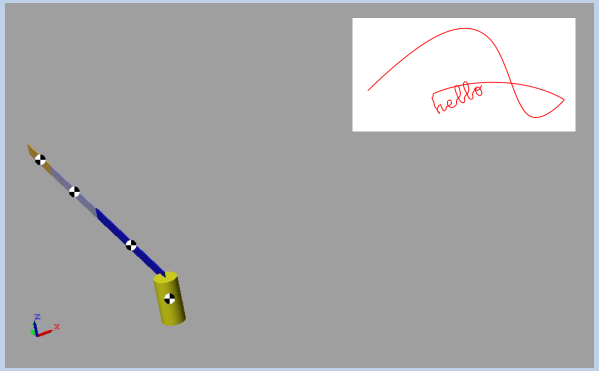
1-dof



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

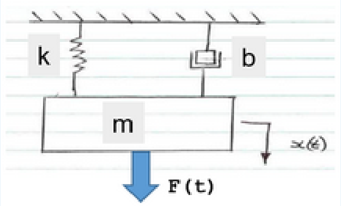
R2017a

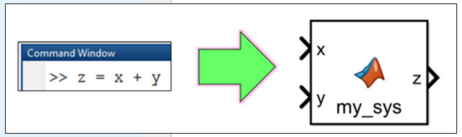
Task: Spring Mass Damper

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.





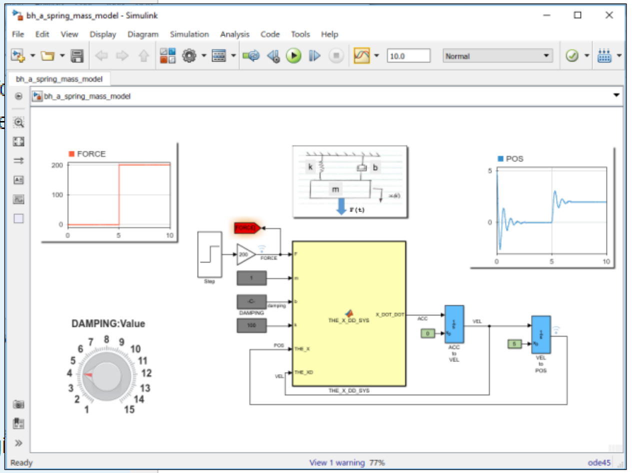
Background:

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:

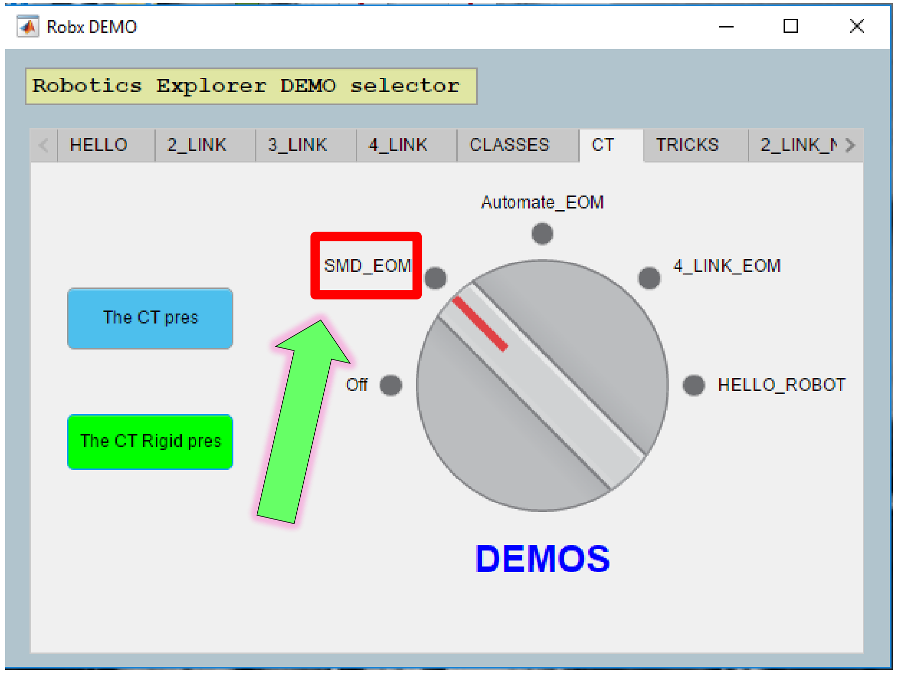
Recall our earlier class where we derived and summarised the fundamental Lagrangi equations that allow us to derive system equations of motion:

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

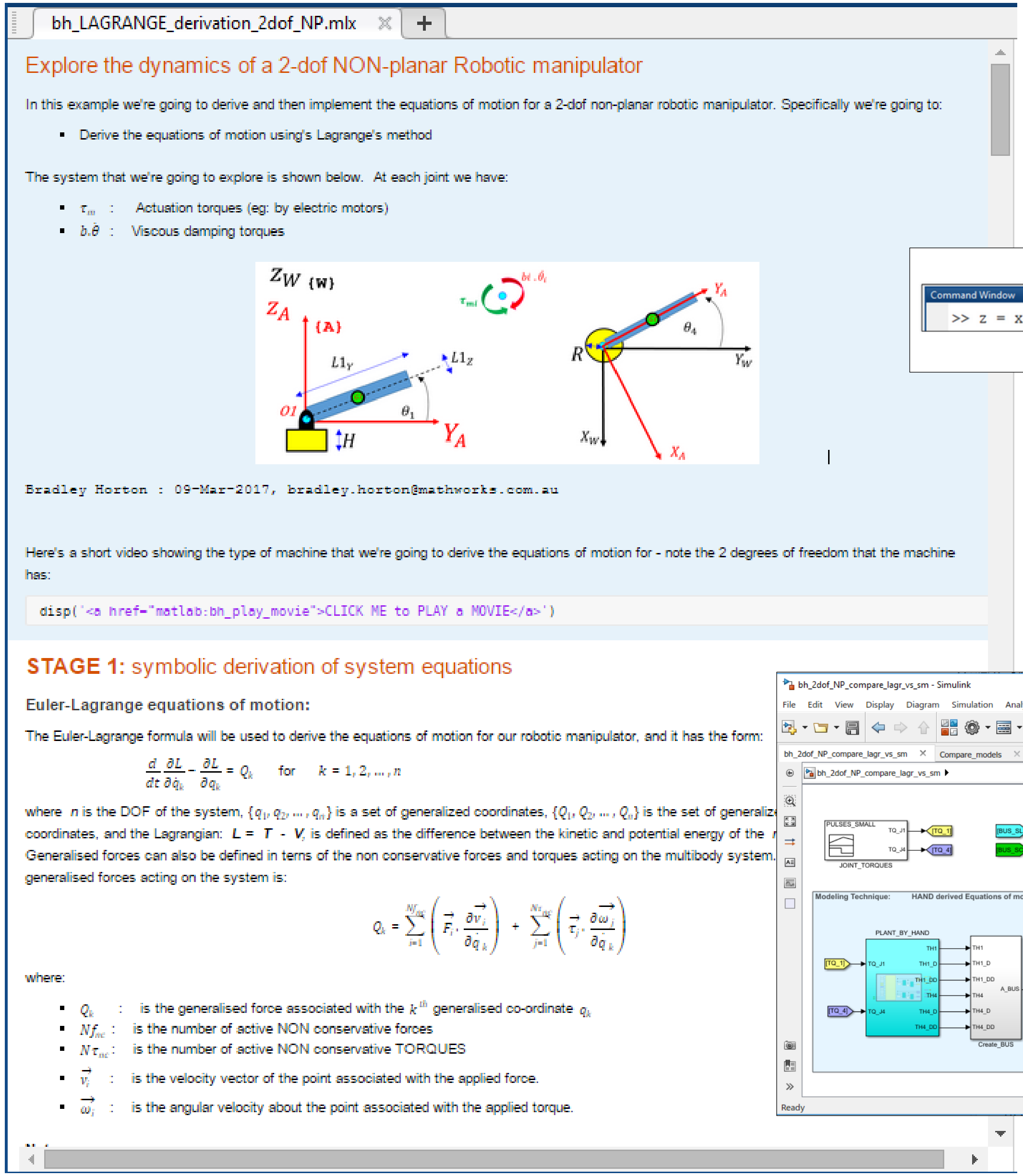
MathWorks

Live Script:
bh_smd_model_derivation.mlx

Try it:



Task: 2-dof Non-planar robotic manipulator



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_m : Actuation torques (eg: by electric motors)
- $b.\dot\theta$: 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:

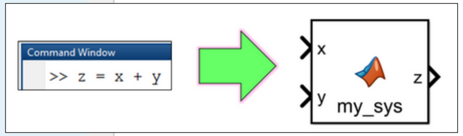
$$\frac{d}{dt}\frac{\partial L}{\partial \dot q_k} - \frac{\partial L}{\partial q_k} = Q_k \quad \text{for} \quad k = 1, 2, \dots, n$$

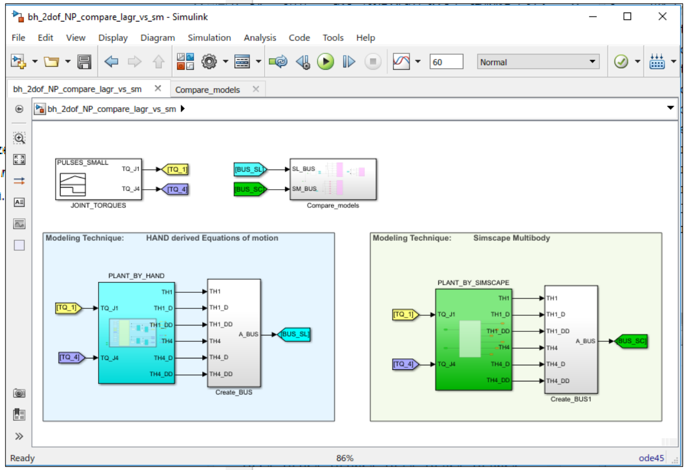
where n is the DOF of the system, $\{q_1, q_2, \dots, q_n\}$ is a set of generalized coordinates, $\{Q_1, Q_2, \dots, Q_n\}$ is the set of generaliz… coordinates, and the Lagrangian: $L = T - V$, is defined as the difference between the kinetic and potential energy of the … Generalised forces can also be defined in terns of the non conservative forces and torques acting on the multibody system… 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
- Nf_{nc} : is the number of active NON conservative forces
- $N\tau_{nc}$: is the number of active NON conservative TORQUES
- $\vec{v}_i$: is the velocity vector of the point associated with the applied force.
- $\vec{\omega}_i$: is the angular velocity about the point associated with the applied torque.

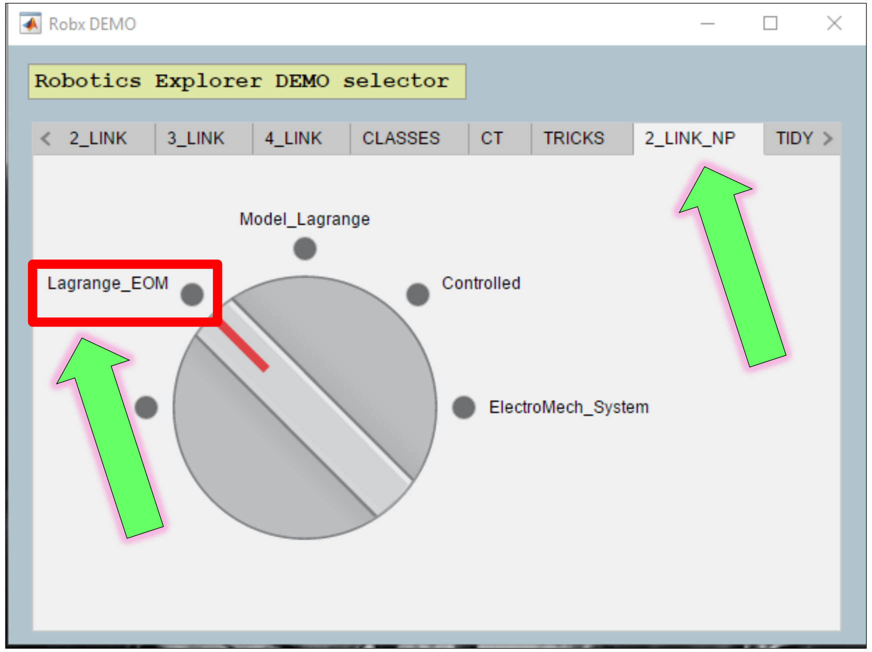




Live Script:

bh_LAGRANGE_derivation_2dof_NP.mlx

Try it:



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	In a MATLAB function	In a MATLAB class
Undergrad Postgrad Lecturer	Undergrad Postgrad Lecturer	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

2. 3.

1.

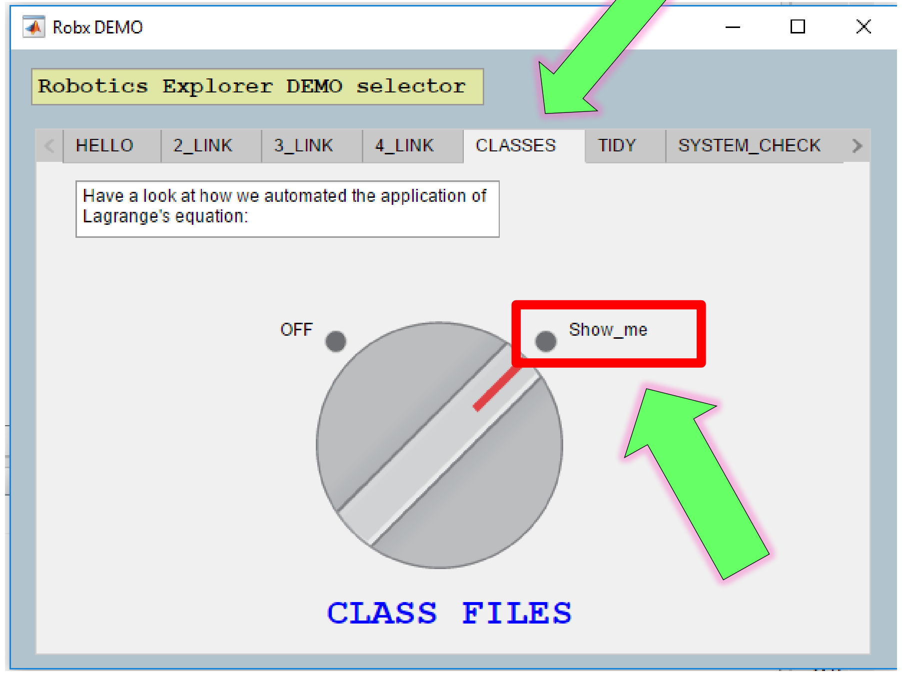
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q}$$

```
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
```

MathWorks

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

Try it:



Task: automating the algorithm

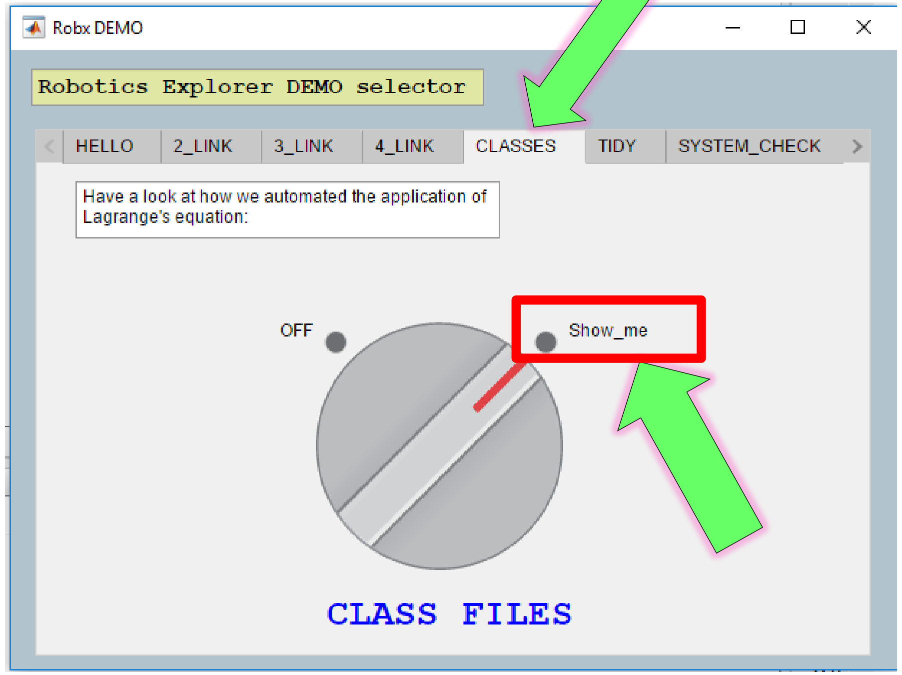
```
N_dof       = OBJ.N_dof;
the_tau_mat = OBJ.THE_CORE.the_tau_mat_holder;
the_w_mat   = OBJ.THE_CORE.the_w_mat_holder;

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
```

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

MathWorks®

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

Try it:



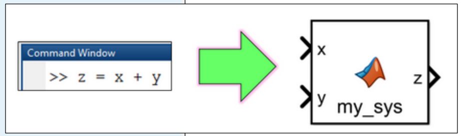
Task: 4-dof Robotic manipulator automate application

bh_LAGRANGE_4dof_manipulator.mlx

Explore the dynamics of a 4-dof Robotic manipulator

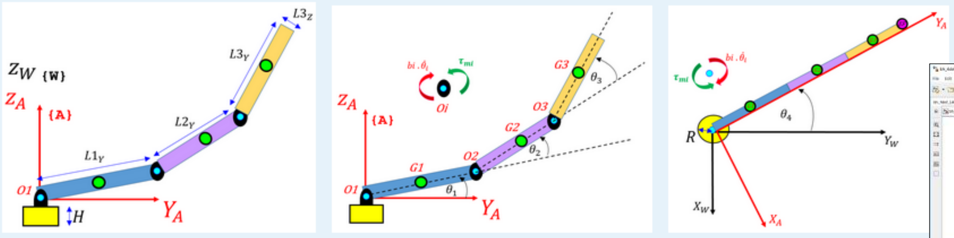
In this example we're going to derive and then implement the equations of motion for a 4-dof robotic manipulator. Specifically we're going to:

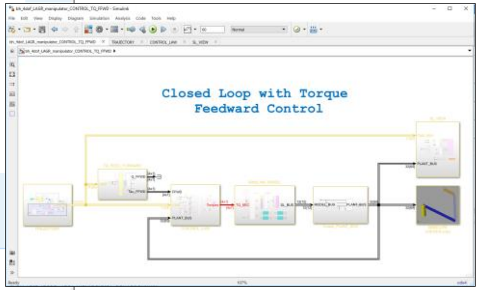
- Derive the equations of motion using's Lagrange's method



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:

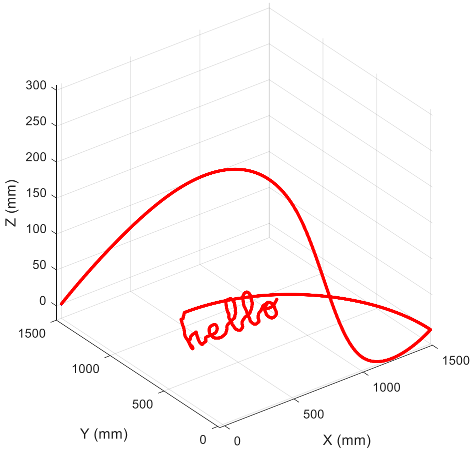
The Euler-Lagrange formula will be used to derive the equations of motion for our robotic manipulator, and it has the form:

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

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 terns of the non conservative forces and torques acting on the multibody system. The formula for the generalised forces acting on the system is:

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

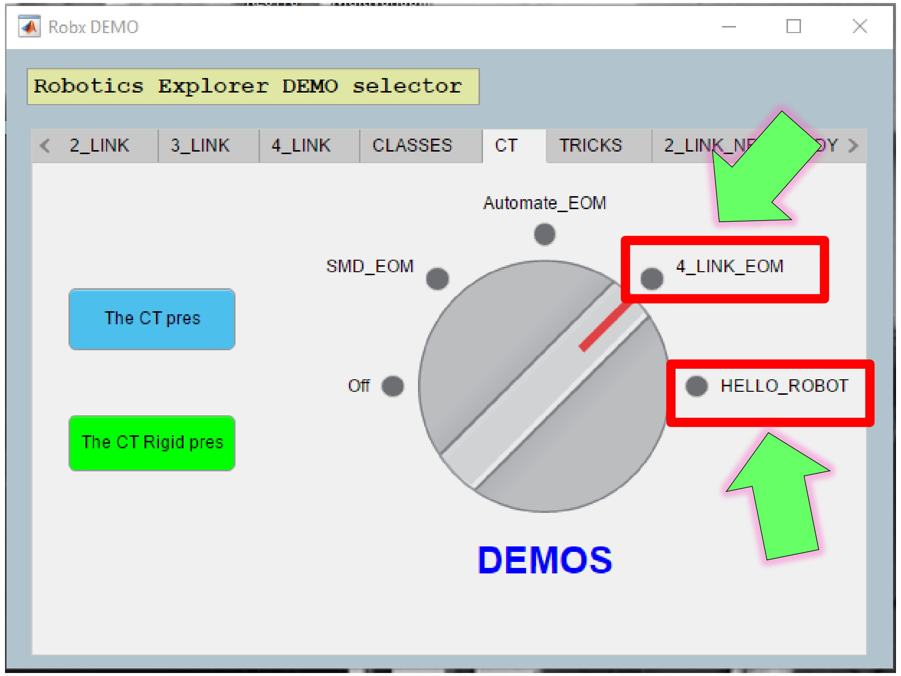
where:



Live Script:

bh_LAGRANGE_4dof_manipulator.mlx

Try it:



Wrap up

The Computational Thinking approach:

Problem Solving and practice

Computational Thinking:

- Brain
- Technology

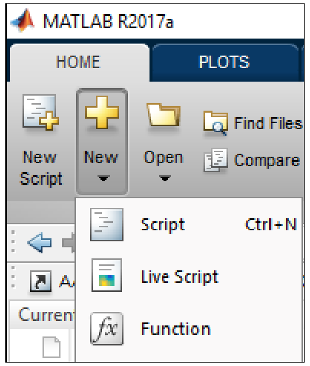
Decomposition

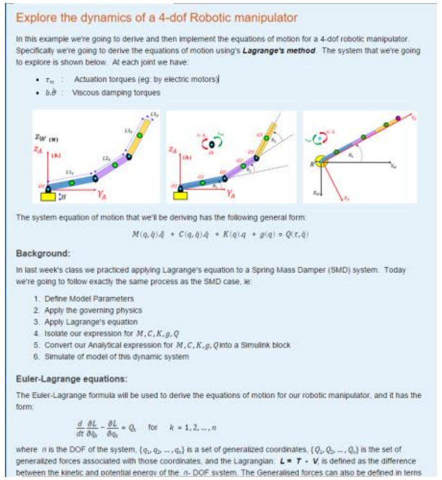
Algorithms + Automation

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'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

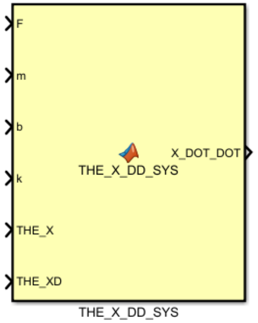
Algorithms + Automation

Symbolic Computing

```
>> diff()

>> matlabFunctionBlock()
```

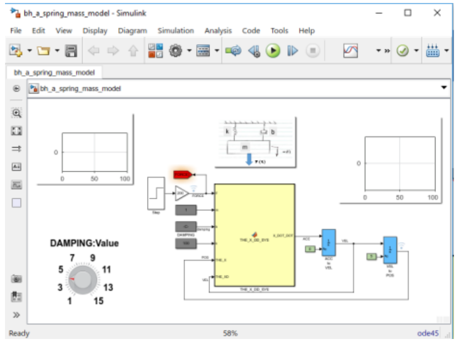
our_EOM(t) =

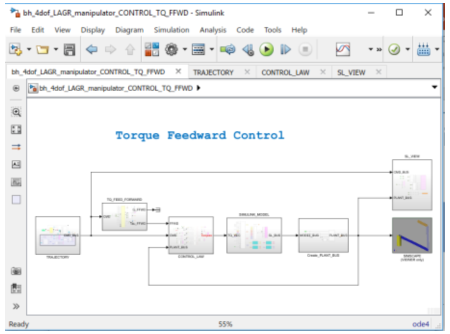
$$m\,\frac{\partial^2}{\partial t^2}\,x(t) + k\,x(t) = F - b\,\frac{\partial}{\partial t}\,x(t)$$



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 !

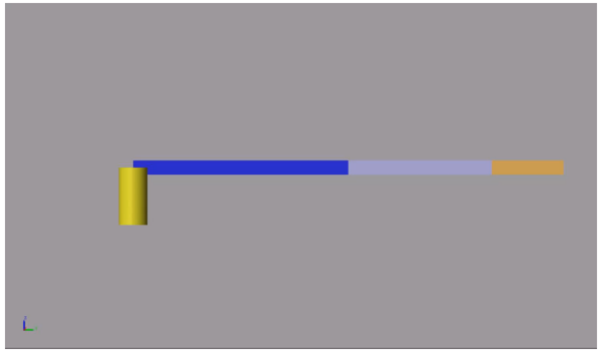
Problem Solving and practice

Computational Thinking:
- Brain
- Technology

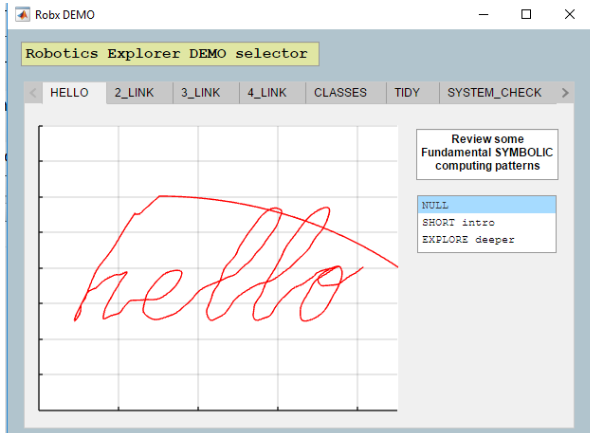
Decomposition

Algorithms + Automation

Simulation



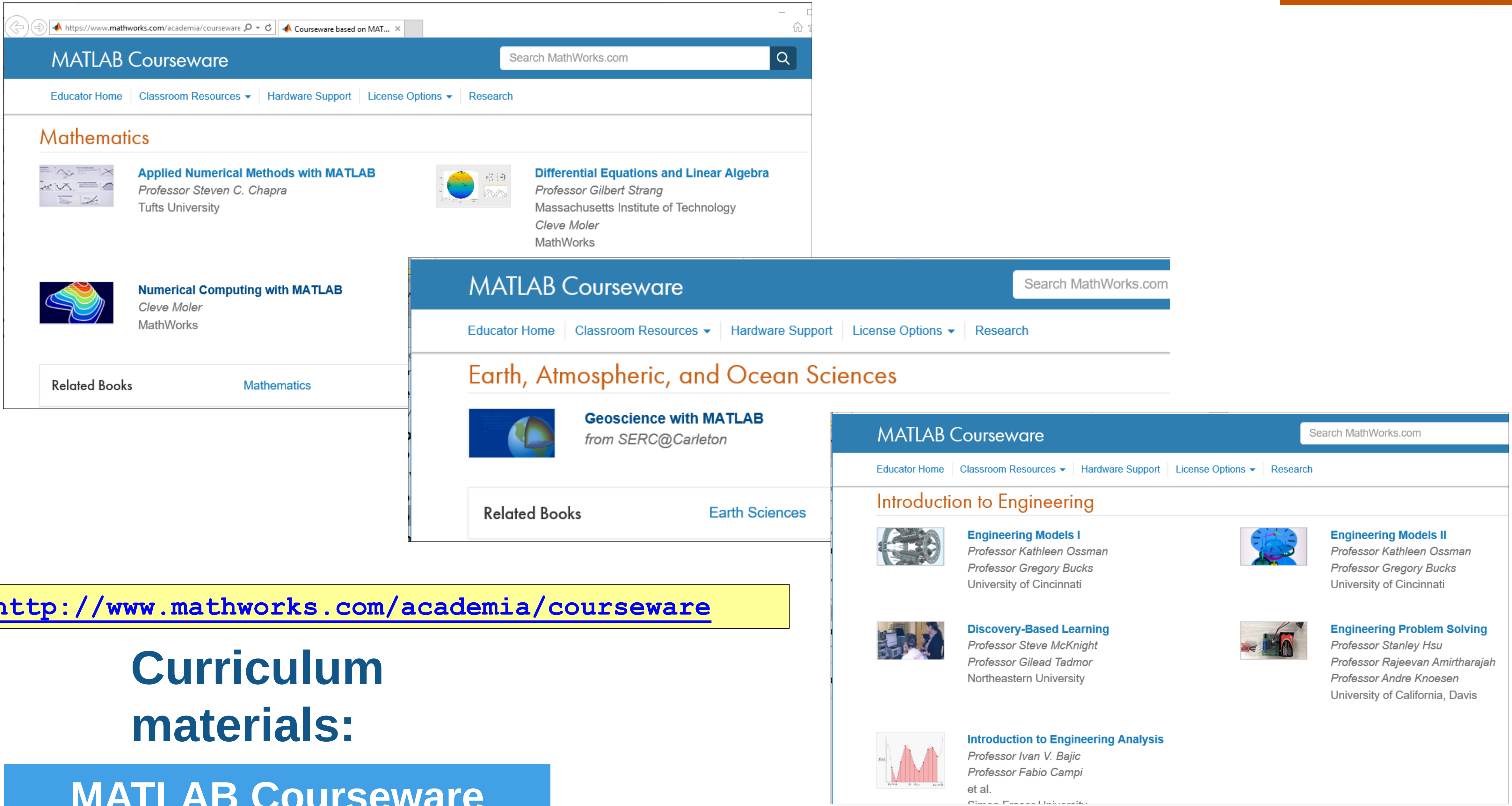
R2017a




```
>> bh_robx_startup
```

Teaching and Learning Resources.

For Educators



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

Curriculum materials:

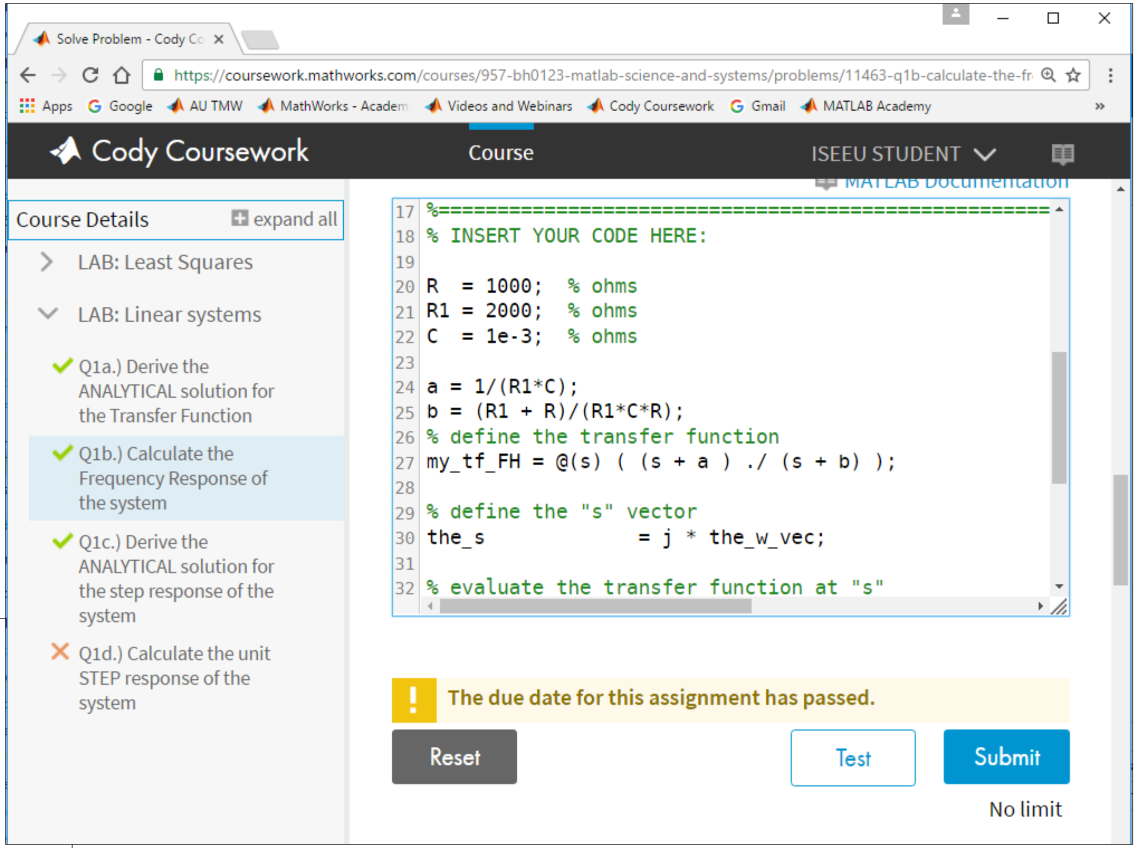
MATLAB Courseware

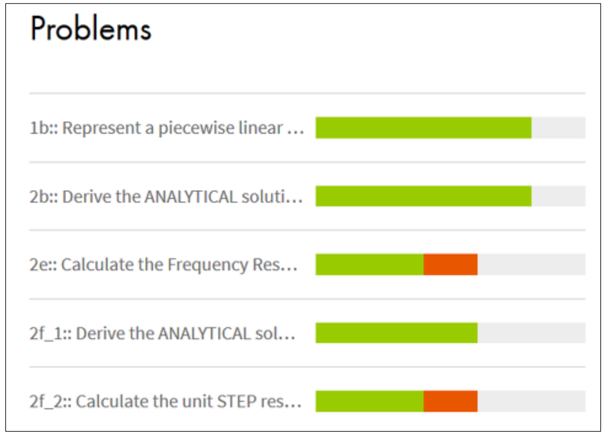
Cody Coursework™

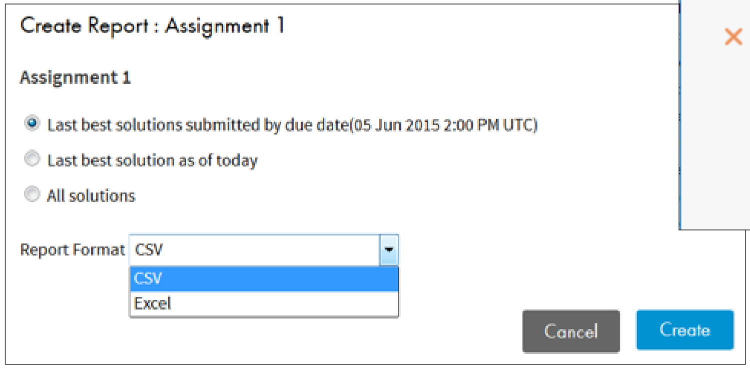
Online automated grading system for MATLAB assignments

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

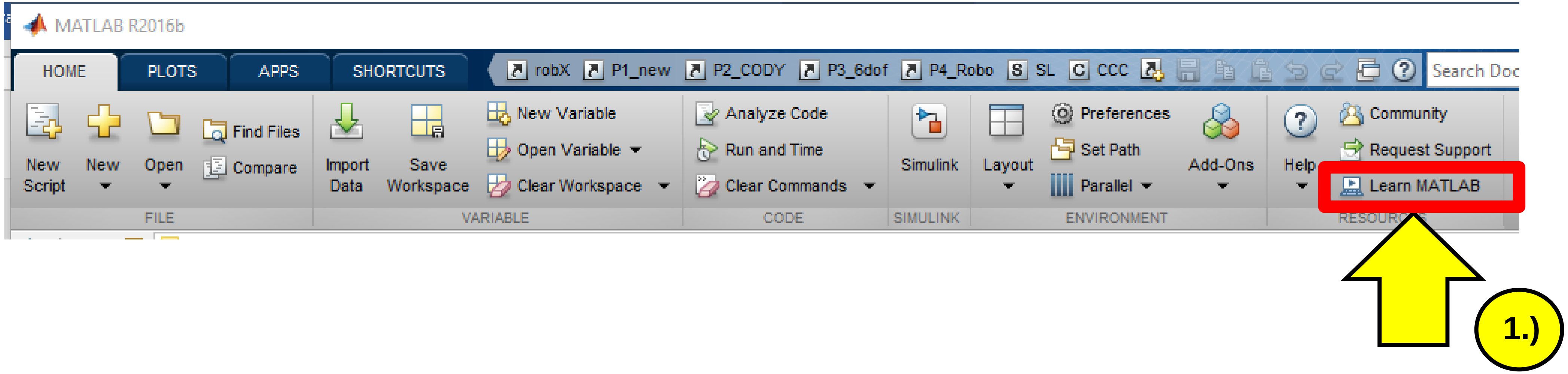


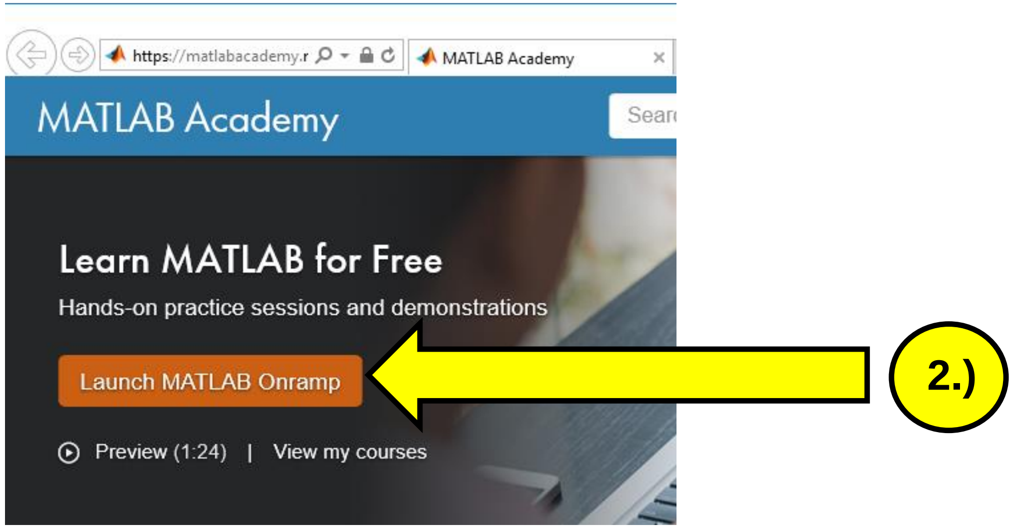




The 1st Stop: …. For students

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



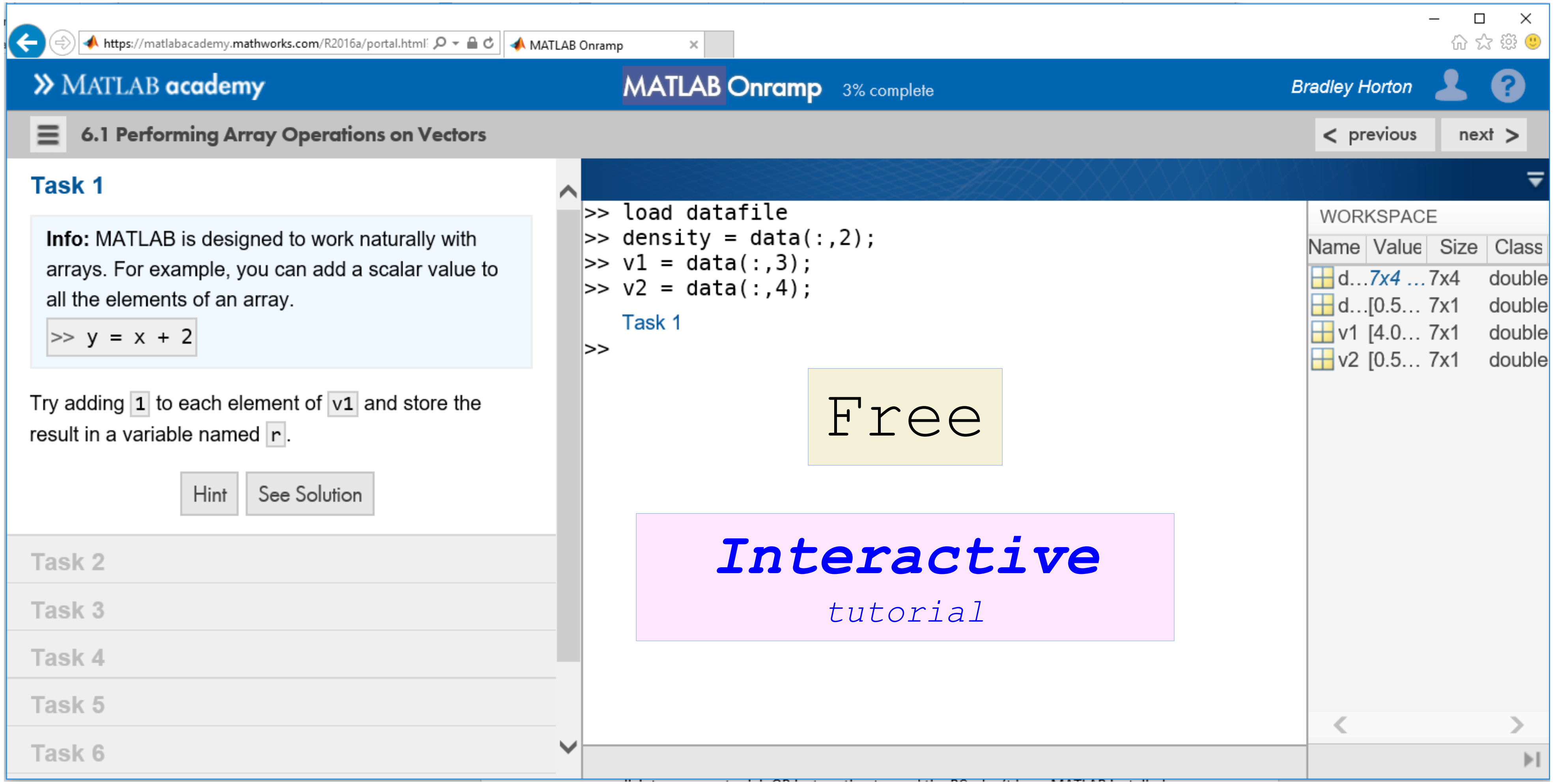


Launch the FREE course called **MATLAB OnRamp**

For Students

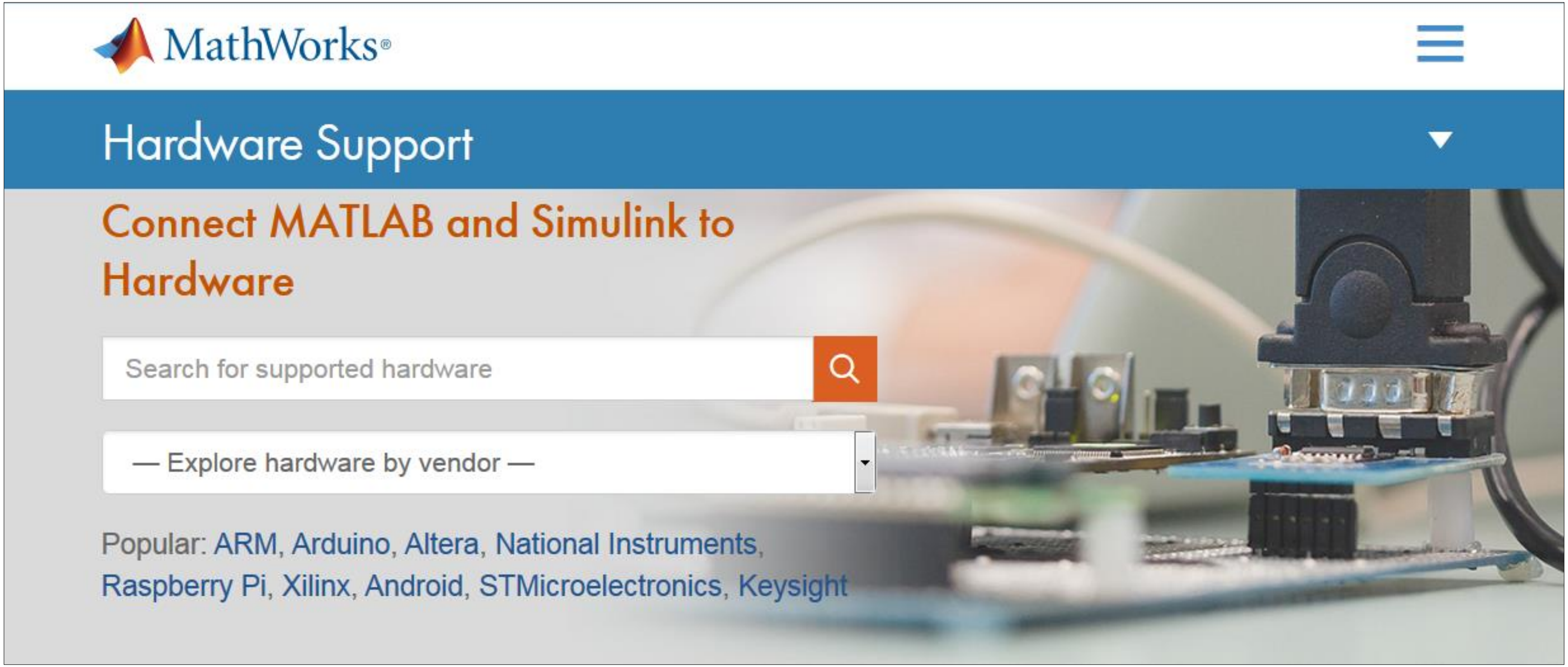
The 1st Stop: …. For students

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



Connecting to Hardware

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



Old slides

MathWorks

Todays 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**

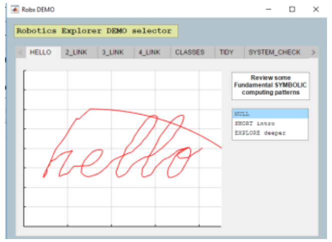
R2017a

Phase 3

- Resources for you and your students

Q/A

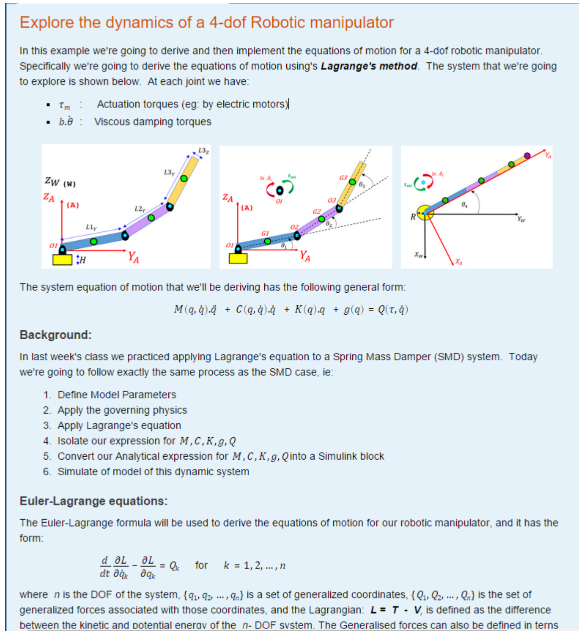
- Questions AND Answers
- How do you get ALL of the examples that you saw today ?



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

Centralization of thought process

MATLAB
Live scripts


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: $\boldsymbol{L = T - V}$, 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

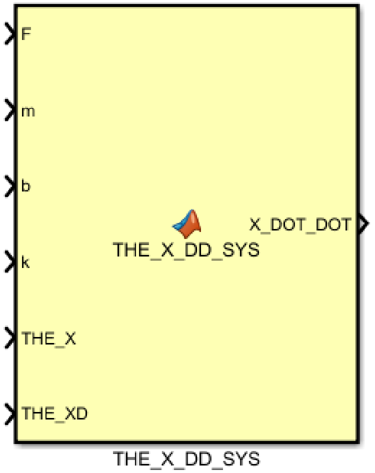
Tedium busters

```
>> diff()

>> matlabFunctionBlock()
```

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

$$dg\_dt(t) =$$

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



Modelling Choices

our_EOM(t) =

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