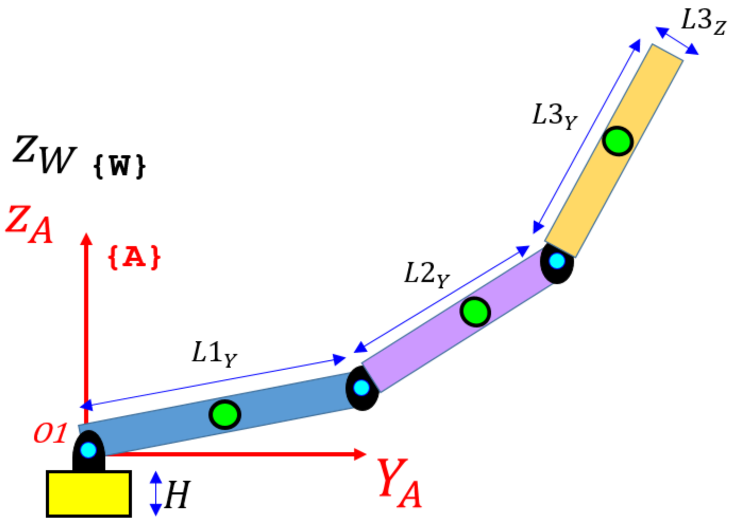
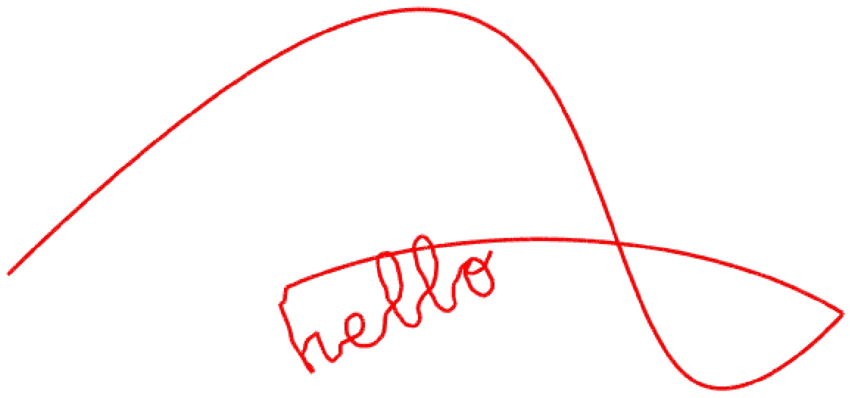
Teaching Lagrangian Dynamics

- a combination of symbolic and numeric computing



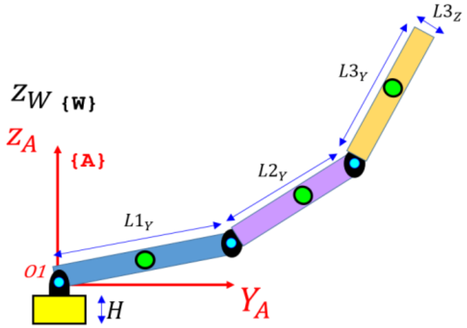
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k$$

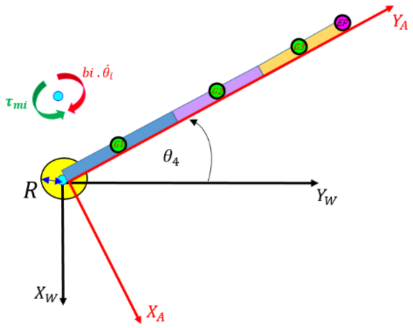
$$Q_k = \sum_{i=1}^{Nf_{nc}} \left(\vec{F}_i \cdot \frac{\partial \vec{v}_i}{\partial \dot{q}_k} \right) + \sum_{j=1}^{N\tau_{nc}} \left(\vec{\tau}_j \cdot \frac{\partial \vec{\omega}_j}{\partial \dot{q}_k} \right)$$

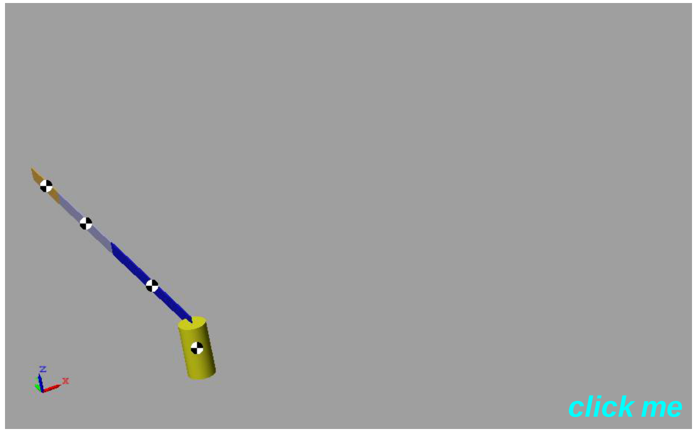


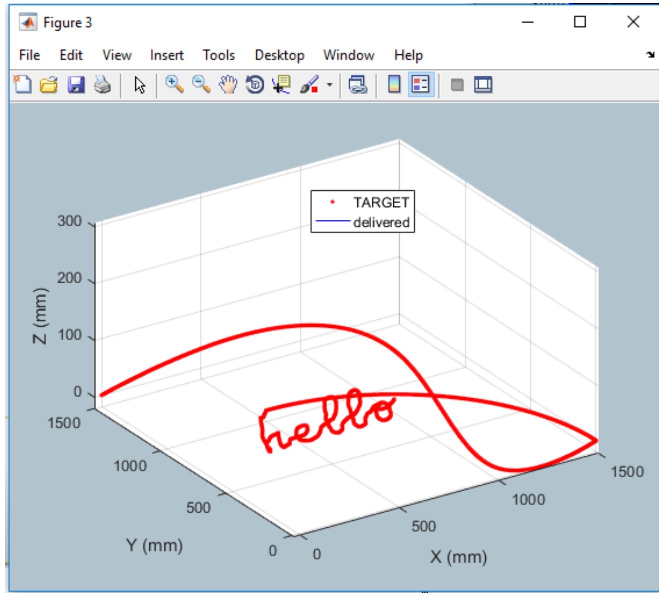
Brad Horton
Engineer
MathWorks

How do you make a robot write hello ?

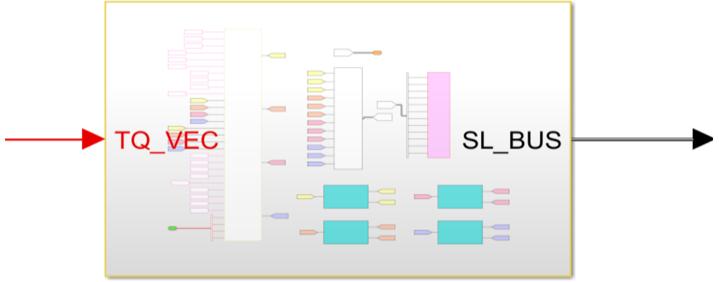








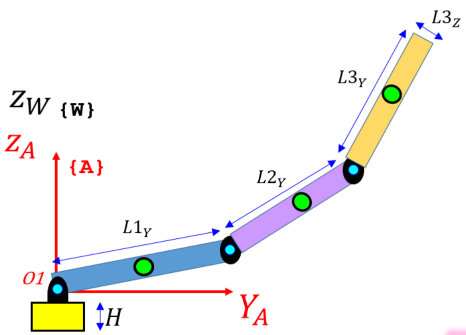
We need a mathematical model

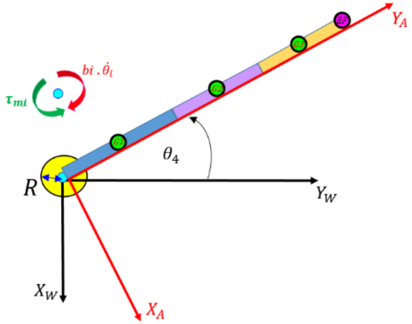


$$M(q).\ddot{q} + C(\dot{q},q).\dot{q} + K(q).q + g(q) = Q$$

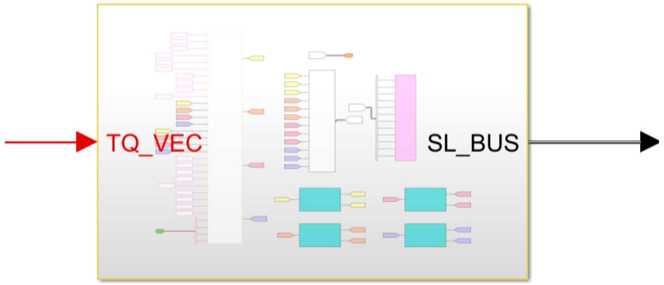
$$\ddot{q} = [M(q)]^{-1}.[Q - C(\dot{q},q).\dot{q} - K(q).q - g(q)]$$

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

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

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

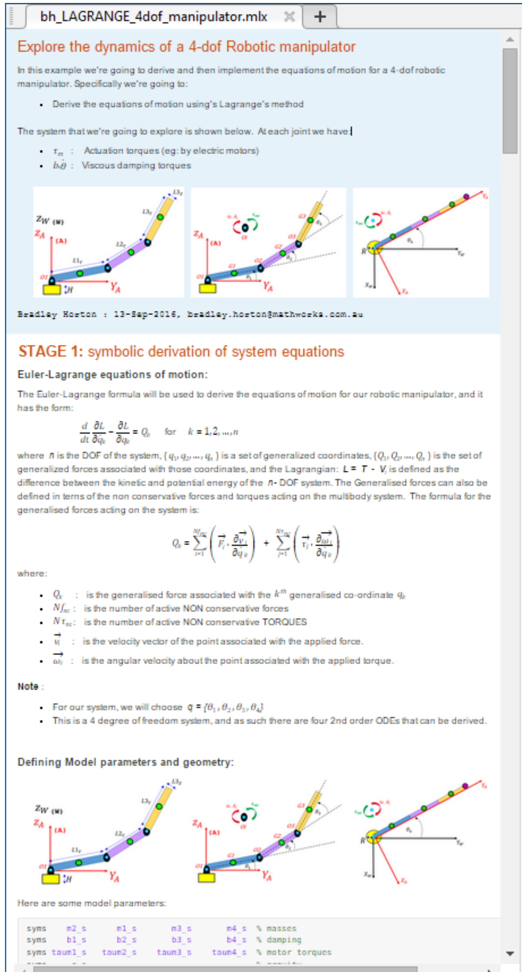
$$M(q).\ddot{q} + C(\dot{q},q).\dot{q} + K(q).q + g(q) = Q$$

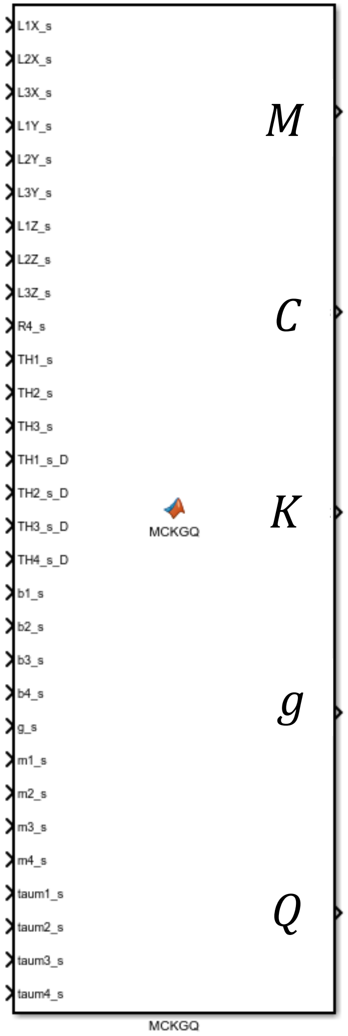
$$\ddot{q} = [M(q)]^{-1}.[Q - C(\dot{q},q).\dot{q} - K(q).q - g(q)]$$

How do you derive the Mathematical model …… in MATLAB ?

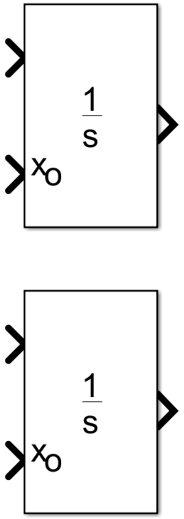
DEMO

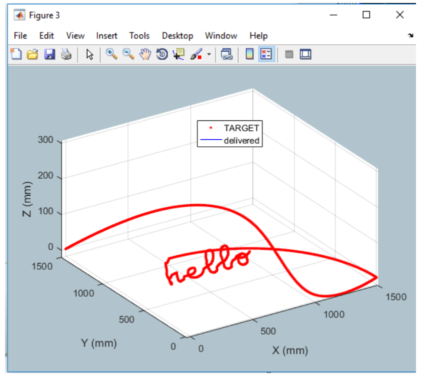
$$M(q).\ddot{q} \ + \ C(\dot{q},q).\dot{q} \ + \ K(q).q \ + \ g(q) \ = \ Q(\tau,\dot{q})$$



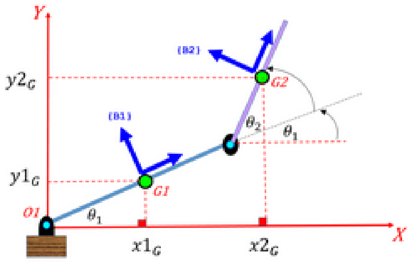


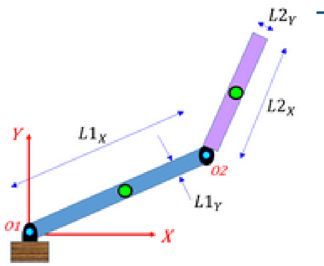
$\ddot{q}$



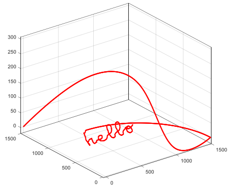


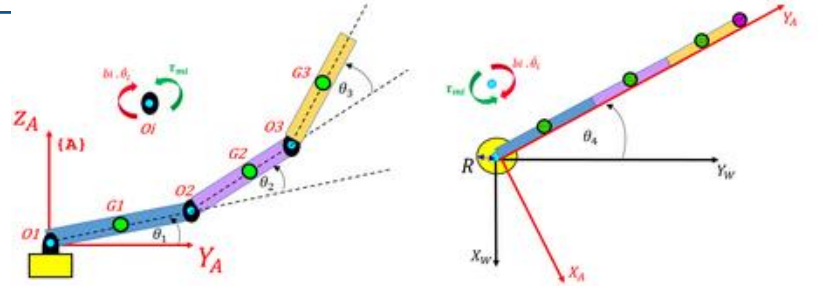
$$\ddot{q} \ = \ [M(q)]^{-1}.[Q(\tau,\dot{q}) - C(\dot{q},q).\dot{q} \ - K(q).q \ - g(q) \ ]$$





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

$$\left.\begin{array}{l}\ddot{\theta}_1\\[1em]\ddot{\theta}_2\end{array}\right\}$$

4-dof

Approx
200 lines

$$\left\{\begin{array}{l}\ddot{\theta}_1\\[1em]\ddot{\theta}_2\\[1em]\ddot{\theta}_3\\[1em]\ddot{\theta}_4\end{array}\right.$$

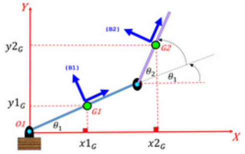
bh_tmp_EOM_file_WILL_BE_DELETED.txt

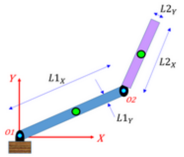
```
################################################################
### q = TH1_s
###
### LHS of EOM is:
###
    1        (L1Y_s^2*TH1_s_DD*m1_s)/3
    2   +    L1Y_s^2*TH1_s_DD*m2_s
    3   +    L1Y_s^2*TH1_s_DD*m3_s
    4   +    (L2Y_s^2*TH1_s_DD*m2_s)/3
    5   +    L2Y_s^2*TH1_s_DD*m3_s
    6   +    (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
```

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

The role of Symbolic computing in your classroom:





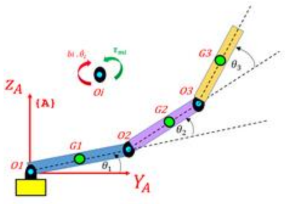
smaller problems

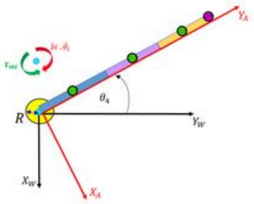
The understanding of the problem physics

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:

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

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

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

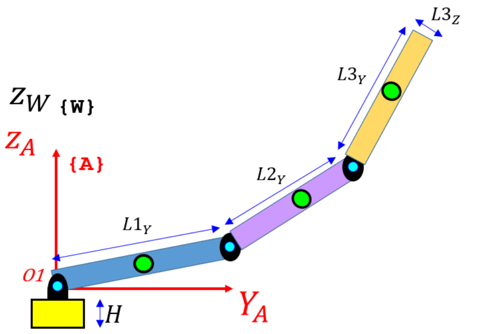
- Lagrangian Dynamics – part 2
  - How to ***automate*** the application
    - 2 LINK robot

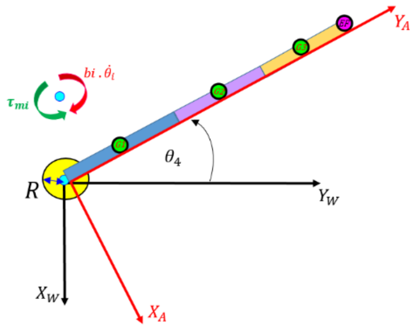
- Lagrangian Dynamics – part 3
  - Make a robot write hello
    - 4 LINK robot

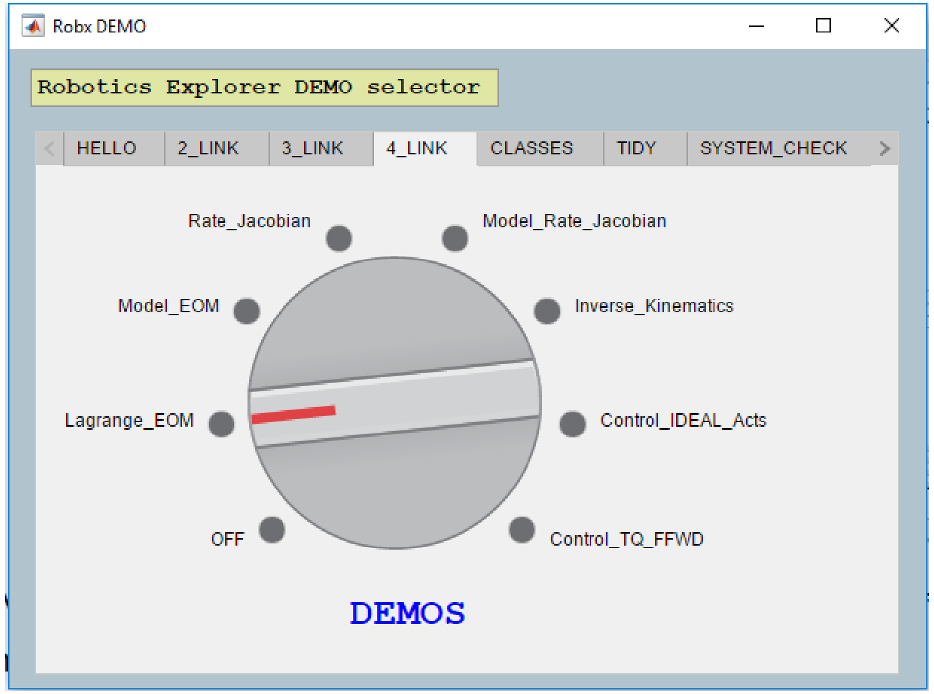
- ***BONUS session: Inverse kinematics***
  - Symbolic computing AND a constrained optimization problem

- Where can you find teaching resources ?

- Q/A
  - Would you like ALL of the examples that you’ve seen today ?







Demo these concepts

R2016b

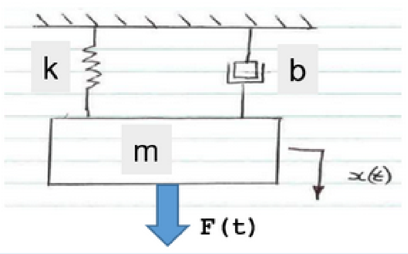
Task: Symbolic computing patterns

bh_short_intro_fundamental_symbolic_patterns.mlx

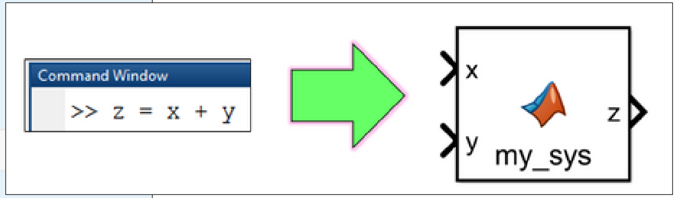
A Case Study - part 1: Deriving the equations of motion

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 Lagranges equation $\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x}$:

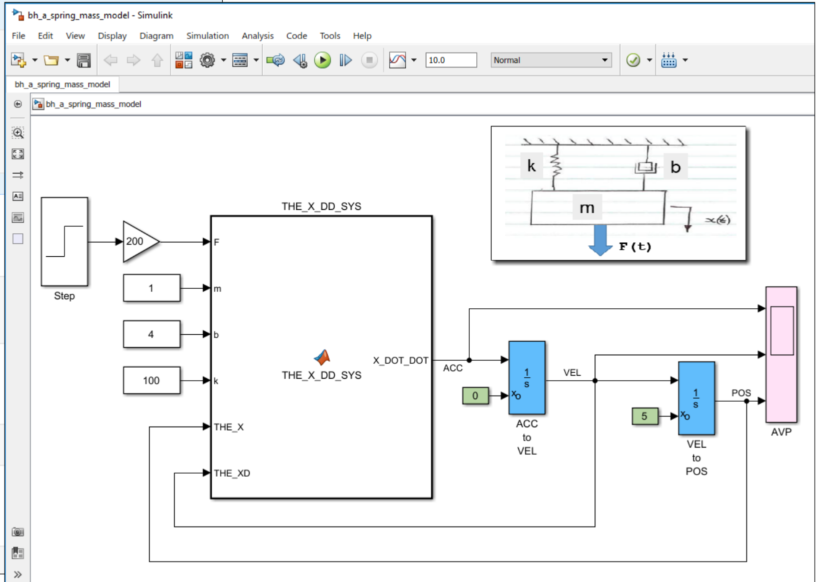
```
                    % OLD_LIST        NEW_LIST
L_new    = subs(L, actual_list,    HOLDER_list)
```

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

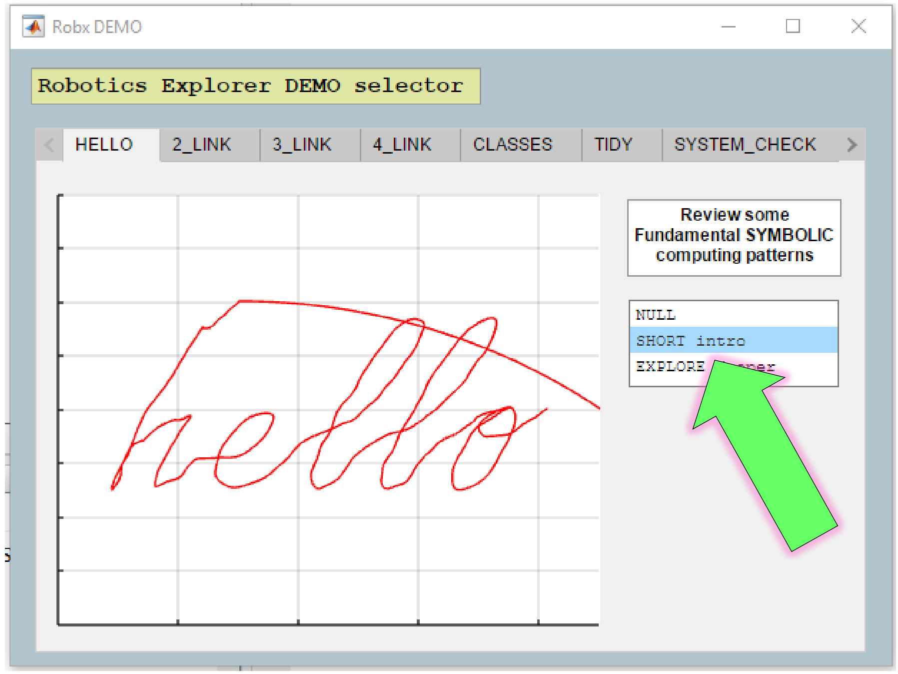
```
dLdx     = diff(L_new, THE_X)
```

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

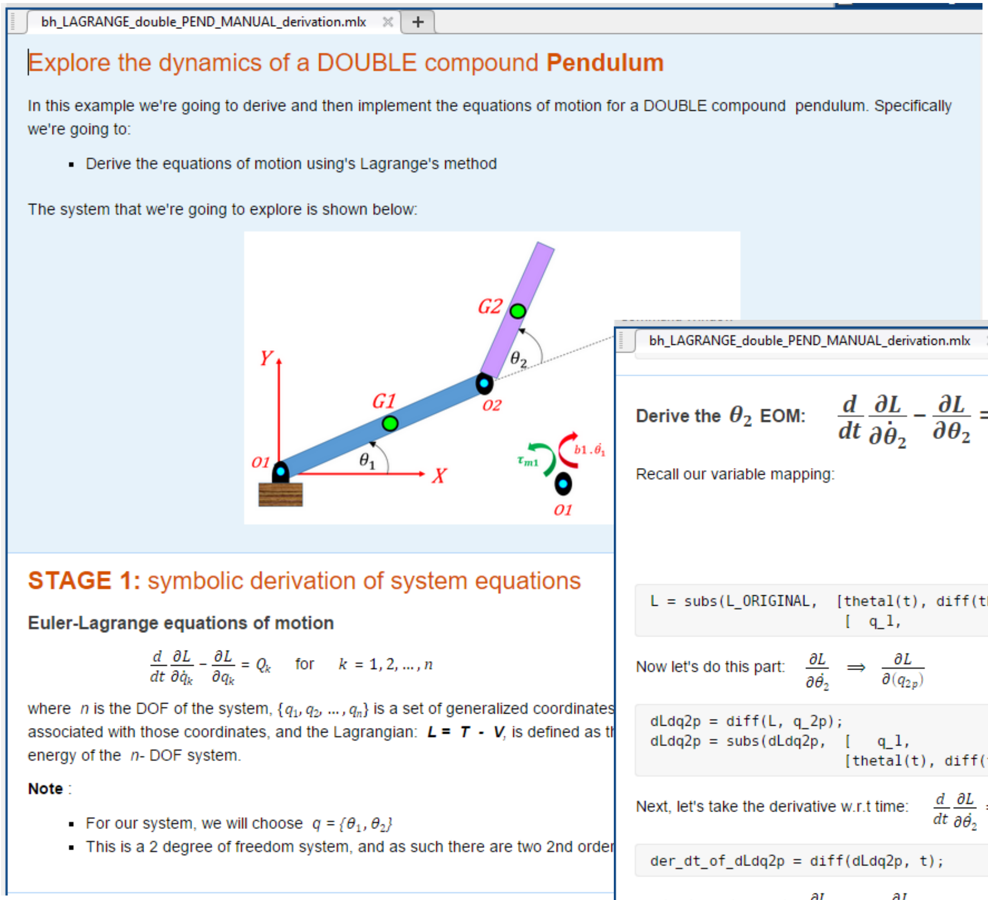
```
dLdxdot =  diff(L_new, THE_XD)
```



Live Script:
bh_short_intro_fundamental_symbolic_patterns.mlx



Task: 2-LINK manual application



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

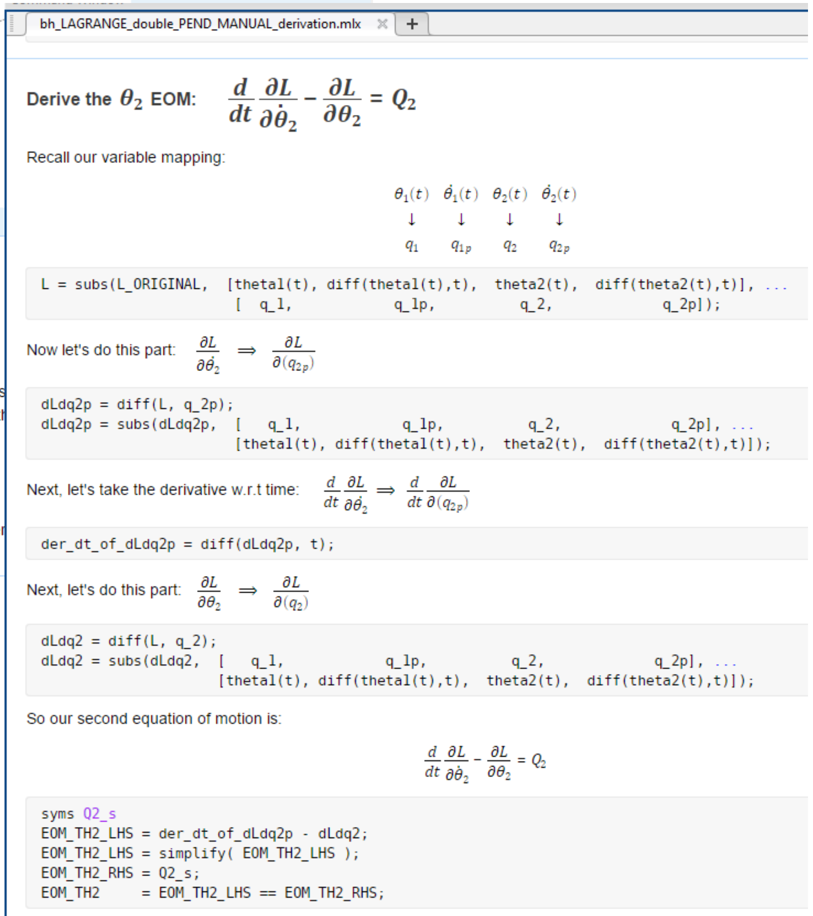
Euler-Lagrange equations of motion

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

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



bh_LAGRANGE_double_PEND_MANUAL_derivation.mlx

Derive the θ_2 EOM: $\dfrac{d}{dt}\dfrac{\partial L}{\partial \dot{\theta}_2} - \dfrac{\partial L}{\partial \theta_2} = Q_2$

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 = subs(L_ORIGINAL,  [theta1(t), diff(theta1(t),t),  theta2(t),  diff(theta2(t),t)], ...
                      [  q_1,          q_1p,            q_2,            q_2p]);
```

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

```
dLdq2p = diff(L, q_2p);
dLdq2p = subs(dLdq2p,  [   q_1,          q_1p,          q_2,           q_2p], ...
                       [theta1(t), diff(theta1(t),t),  theta2(t),  diff(theta2(t),t)]);
```

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

```
der_dt_of_dLdq2p = diff(dLdq2p, t);
```

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

```
dLdq2 = diff(L, q_2);
dLdq2 = subs(dLdq2,  [   q_1,          q_1p,          q_2,           q_2p], ...
                     [theta1(t), diff(theta1(t),t),  theta2(t),  diff(theta2(t),t)]);
```

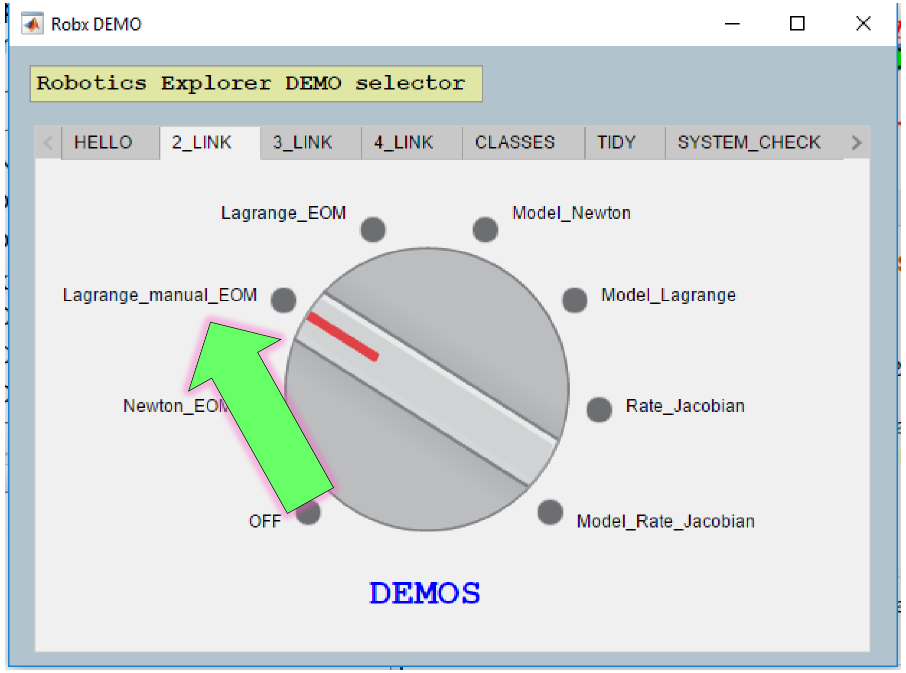
So our second equation of motion is:

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

```
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;
```

Live Script:

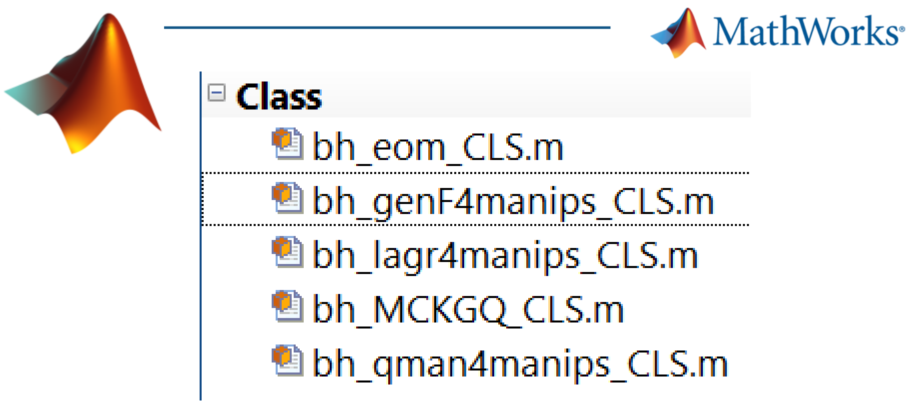
bh_LAGRANGE_double_PEND_MANUAL_derivation.mlx

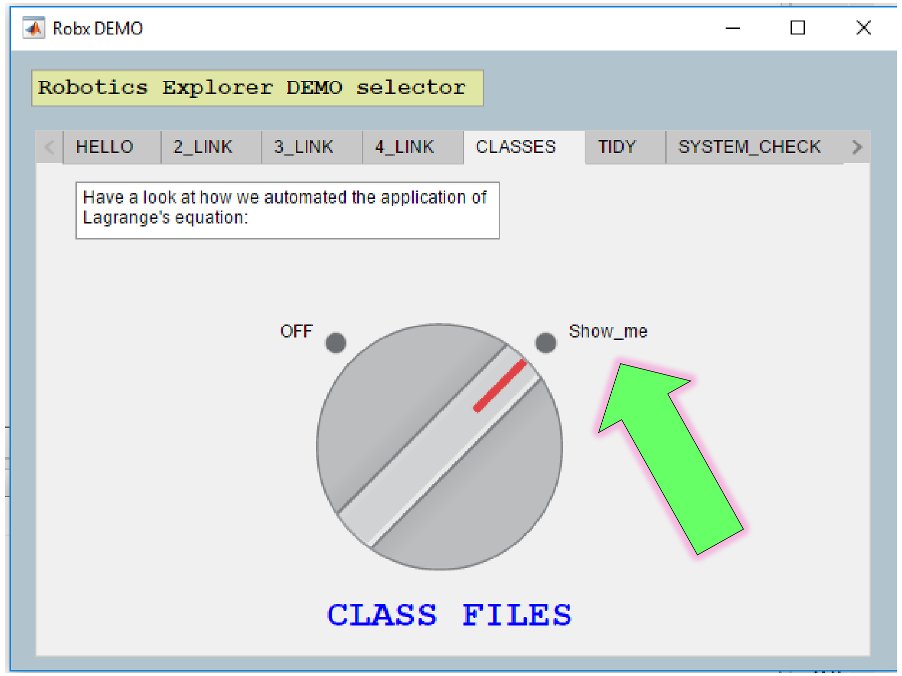


Task: automating the application

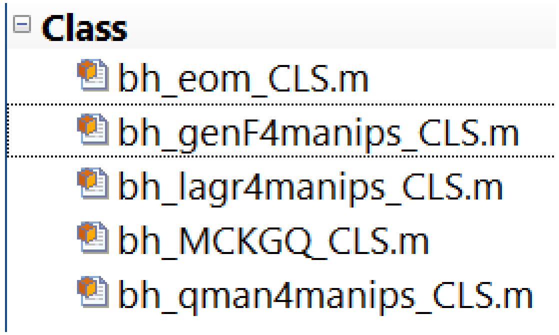
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = Q$$

```
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
```



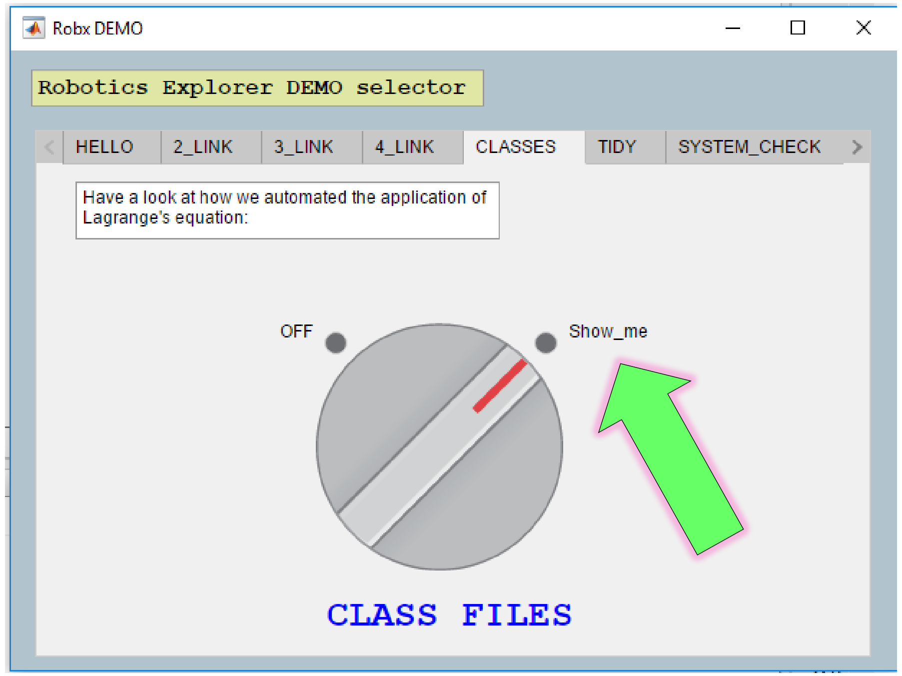


Task: automating the application




```matlab
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
```

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



Task: 2-LINK automate application

bh_LAGRANGE_double_PEND.mlx

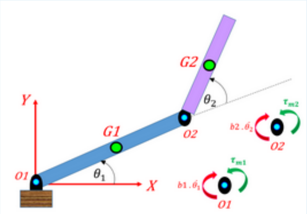
Explore the dynamics of a DOUBLE compound **Pendulum**

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. 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.hort
```

STAGE 1: symbolic derivation of sy

Euler-Lagrange equations of motion

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

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

bh_LAGRANGE_double_PEND.mlx

Apply Lagrange's equation:

So let's now apply Lagrange's equation:

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

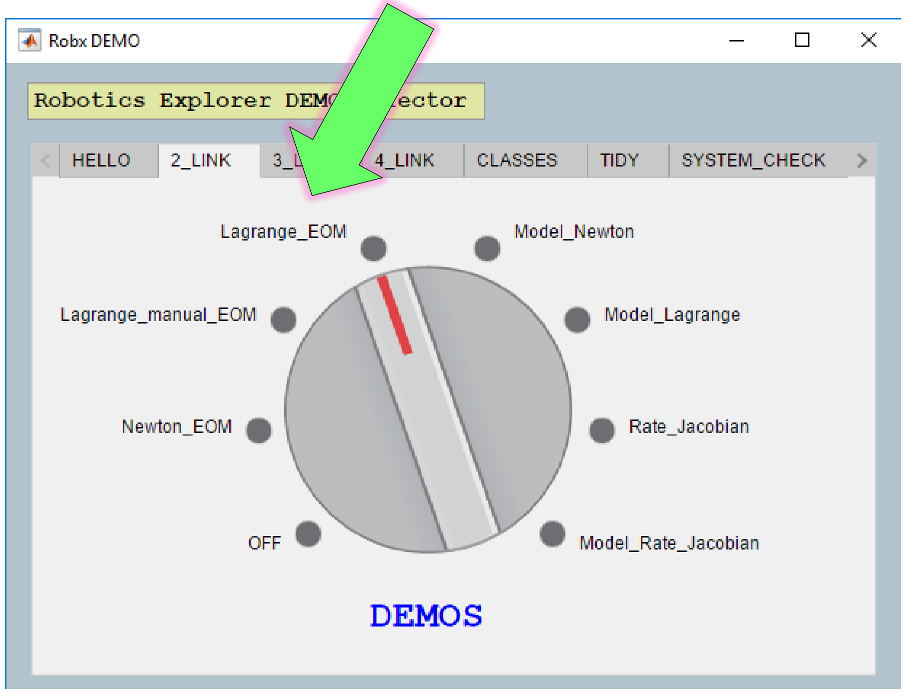
And let's compute the system's equations of motion:

```
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

bh_LAGRANGE_4dof_manipulator.mlx

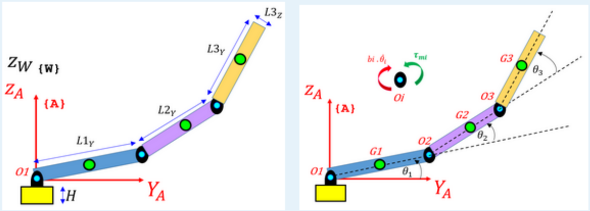
Explore the dynamics of a 4-dof Robotic manipulator

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.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 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 coordinat generalized forces associated with those coordinates, and the Lagrangian: $L = T$ 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 f 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:

bh_LAGRANGE_4dof_manipulator.mlx

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

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

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

```
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
```

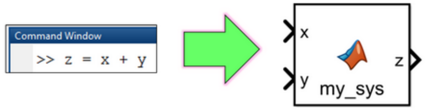
Here's **G**:

```
res_T.G
```

Here's **Q**:

```
res_T.Q
```

Now create the MATLAB function blocks for Simulink:



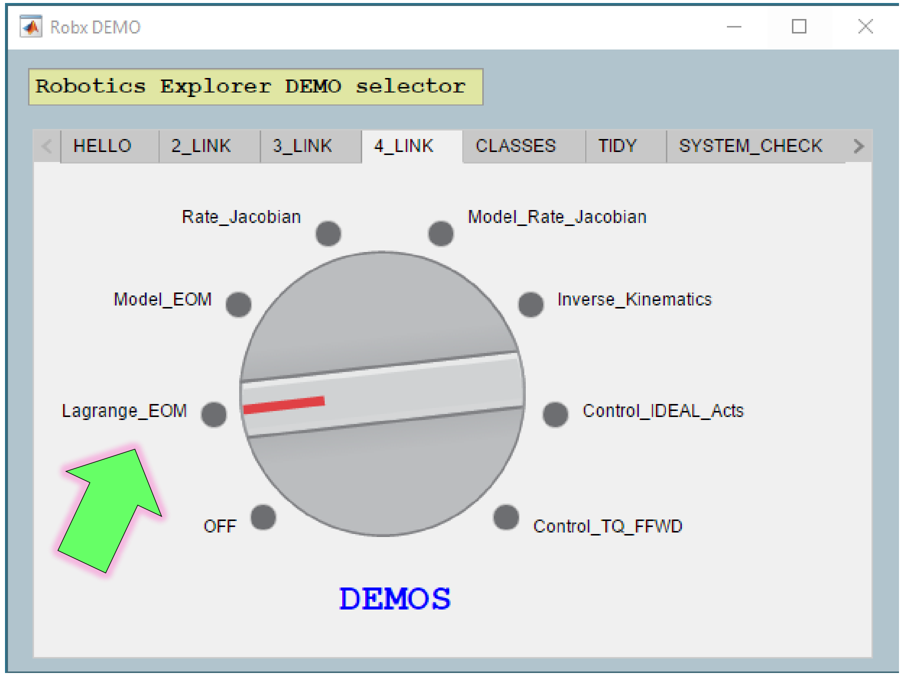
To use/solve these derived equations of motion we'll create a MATLAB Function block that can be used inside Simulink:

```
lag_OBJ.create_MLF_blocks()
```

MathWorks

Live Script:

bh_LAGRANGE_4dof_manipulator.mlx



Task: Inverse kinematics

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:

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

- $J(x)$ 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 $\theta_1 \ge 0$

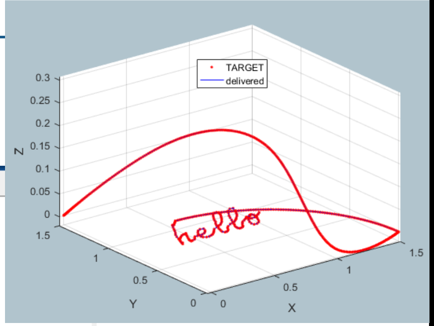
```
qa_lb = [   0;  -pi;   -pi;    -pi]; % LOWER bounds for angles
qa_ub = [  pi;   pi;    pi;     pi]; % UPPER bounds for angles
```

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

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

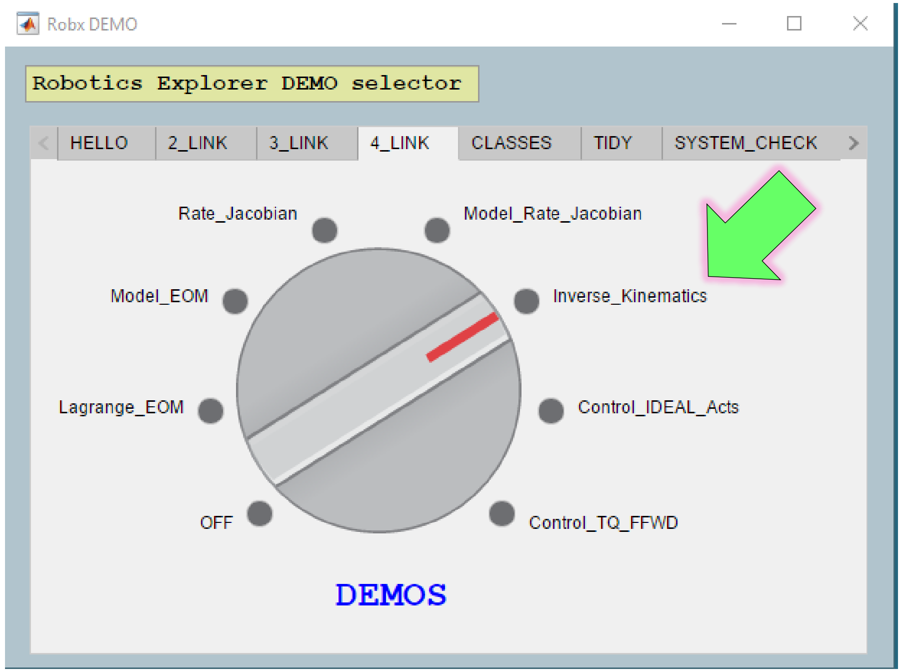
Now at the heart of this approach is the utilization of our FORWARD KINEMATICS function that we derived earlier:

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



MathWorks

Live Script:
bh_invKIN_4dof_manipulator_NUMERICAL_OPTIM.mlx

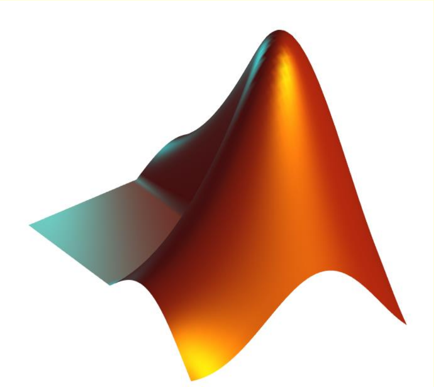


The role of **Symbolic computing with MATLAB** in your classroom:

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

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

MATLAB

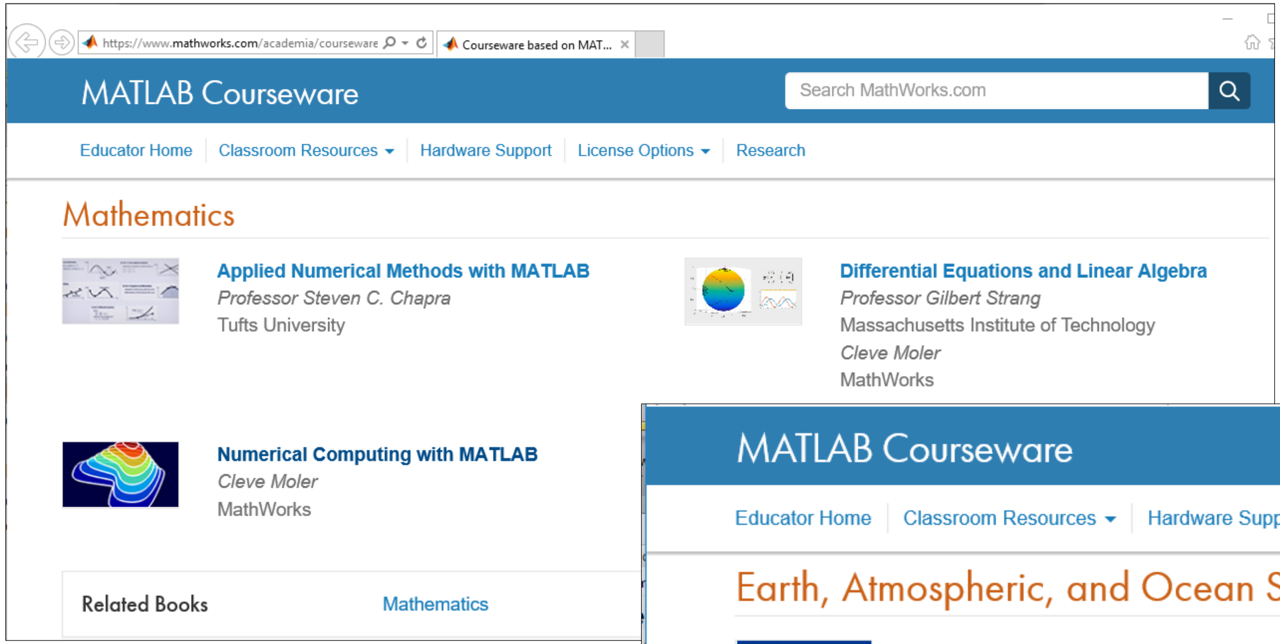


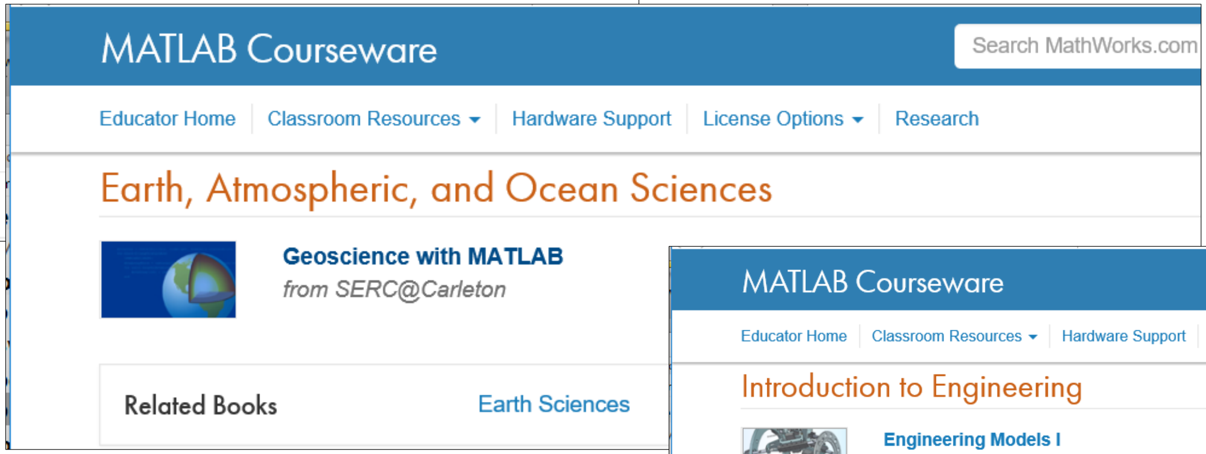
Technical Computing Environment

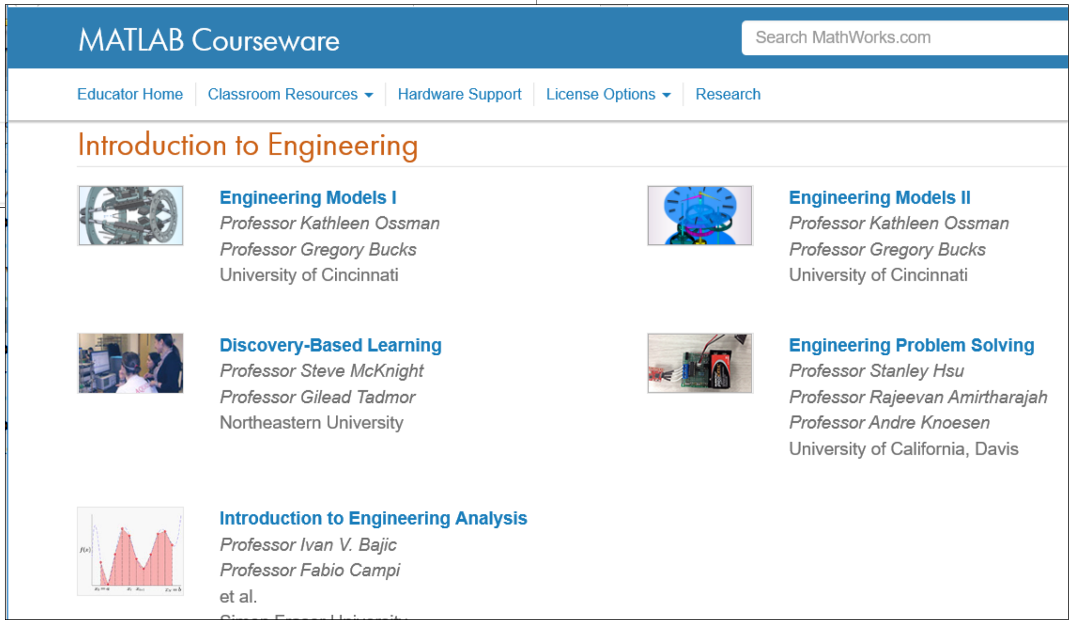
Enhancement of Understanding

- 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.







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

Curriculum materials:

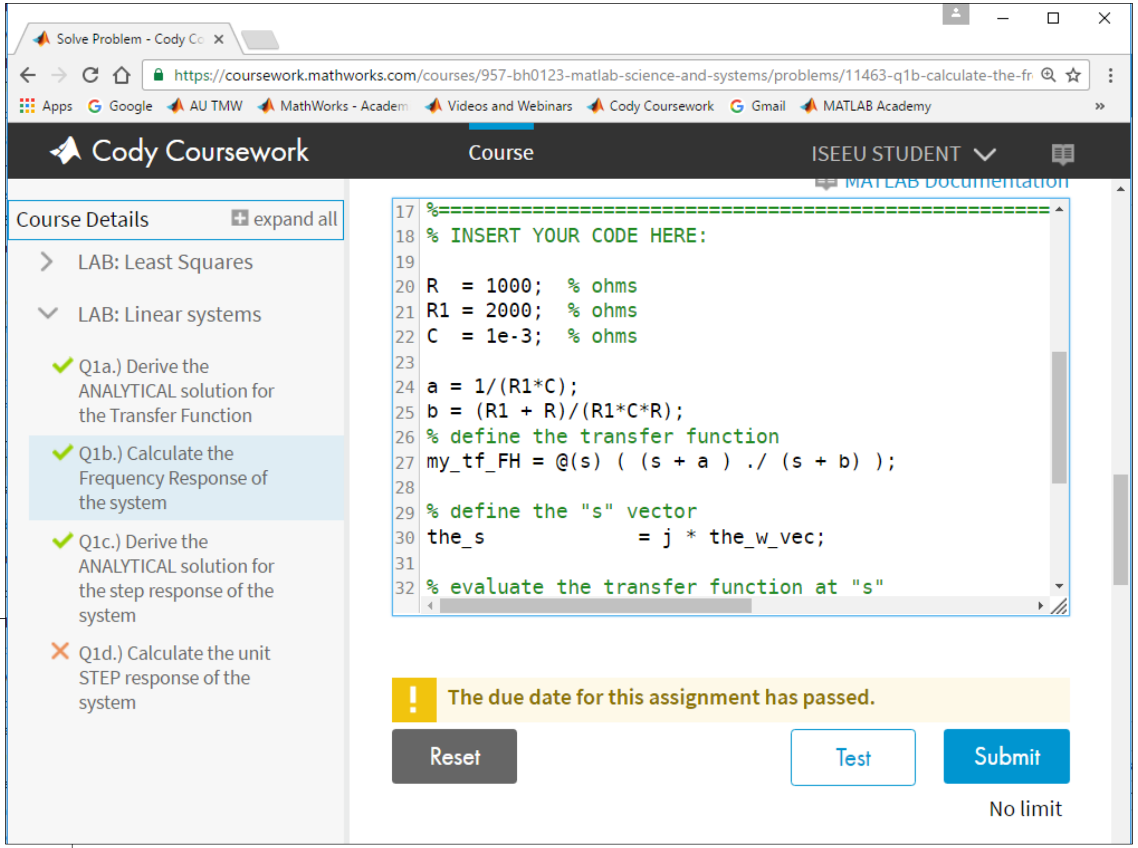
MATLAB Courseware

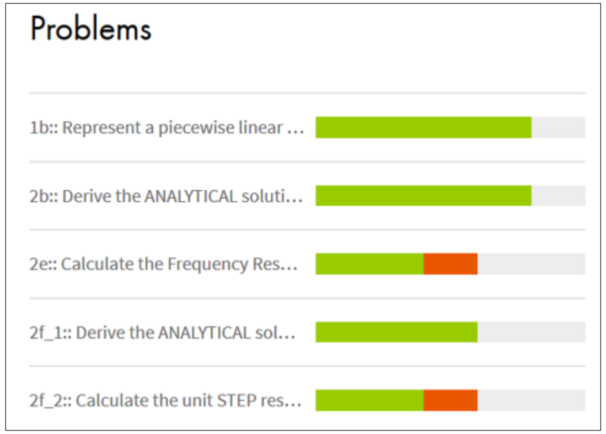
Cody Coursework™

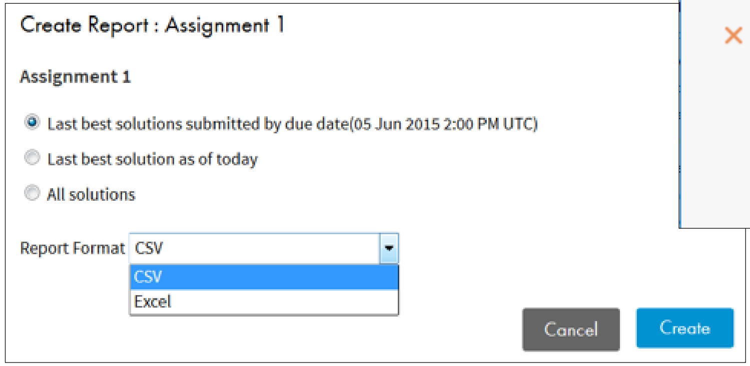
Online automated grading system for MATLAB assignments

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



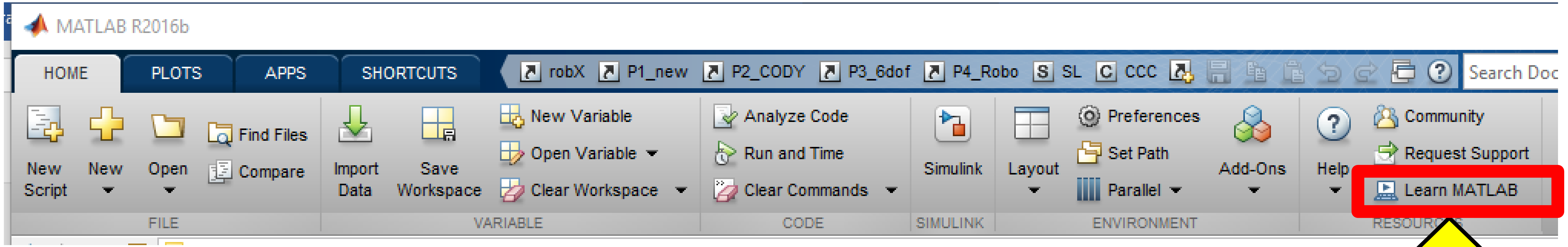




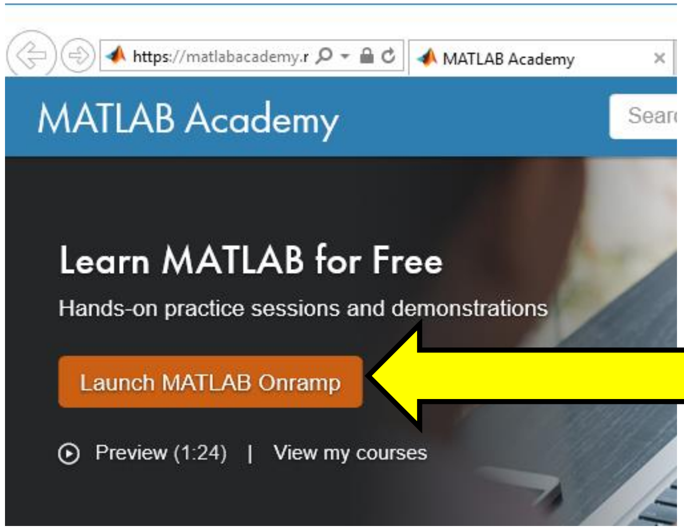
For Students

The 1st Stop: …. For students

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



1.)



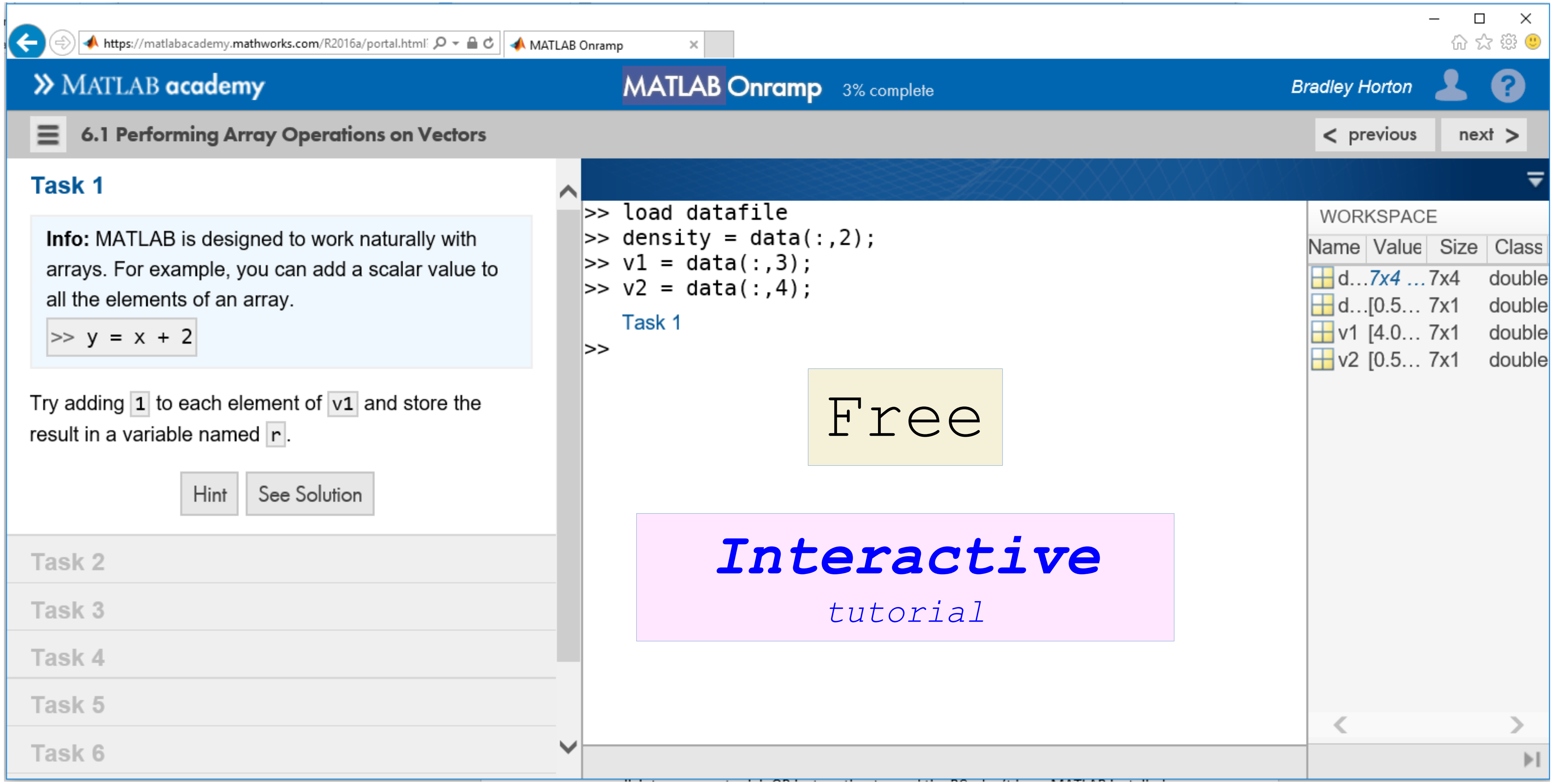
2.)

Launch the FREE course called **MATLAB OnRamp**

For Students

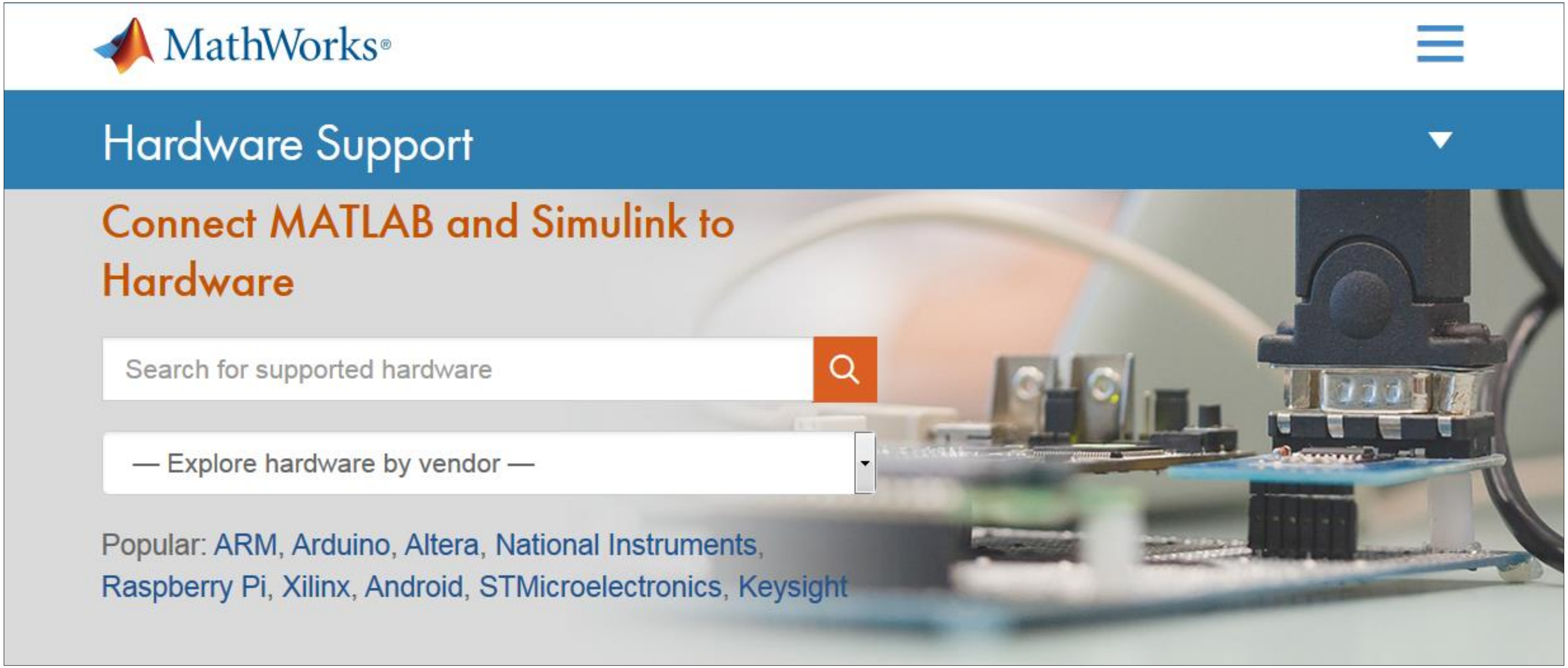
The 1st Stop: …. For students

- **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



Wrap up

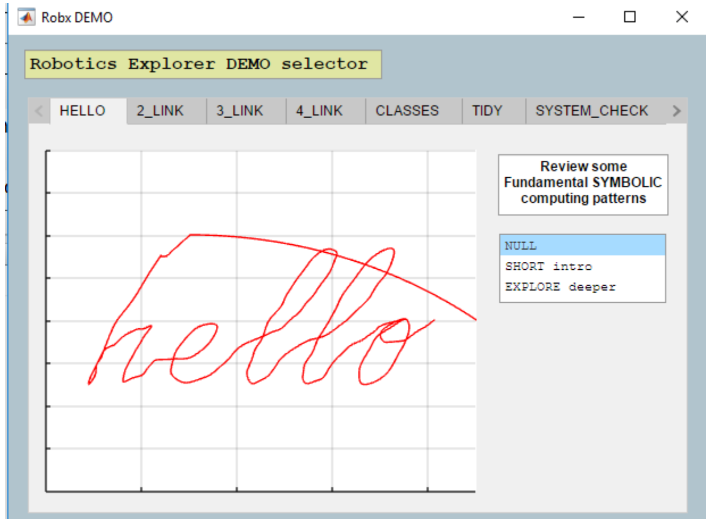
Q/A:

- 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*

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

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

R2016b




```
>> bh_robx_startup
```