Vision and Control of Industrial Robots 01211433

Lecture 9 : Robot Dynamics

Dr.Varodom Toochinda

Dept. of Mechanical Engineering

Kasetsart University

Outline

- Robot dynamics formulation
  - Euler-Lagrange method
  - Newton-Euler method
- RTSX demo
- Robot dynamics properties

Euler-Lagrange Derivation

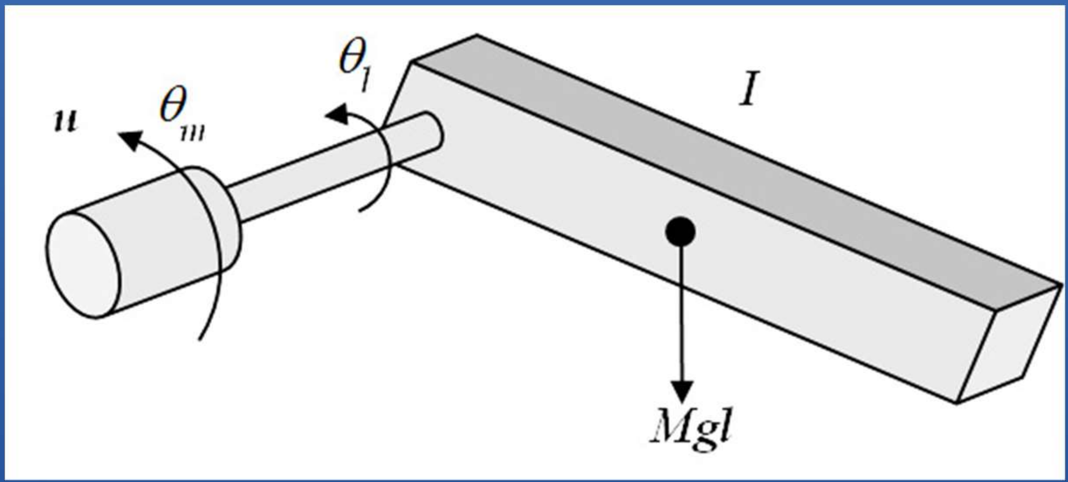
- Form Lagrangian L = K – P
  - K = kinetic energy
  - P = potential energy
- Euler-Lagrange equation

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i, i = 1, \dots n$$

$(q_1, \dots q_n)$ Generalized coordinates

τ_i Generalized forces

Example: 1-link manipulator

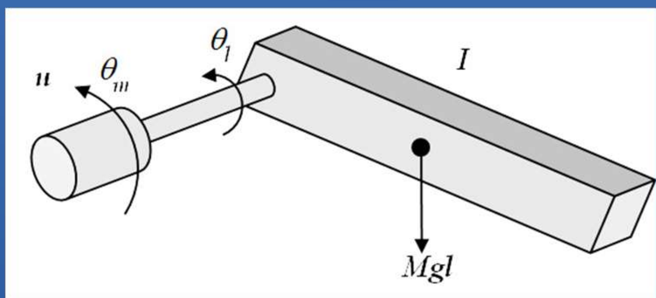


Choose θ_l as generalized coordinate

Kinetic energy : $$K = \frac{1}{2} J_m \dot{\theta}_m^2 + \frac{1}{2} J_l \dot{\theta}_l^2 = \frac{1}{2} (r^2 J_m + J_l) \dot{\theta}_l^2$$

Potential energy : $$P = Mgl(1 - cos\theta_l)$$

Example : 1-link manipulator



Define $J = r^2 J_m + J_l$

Lagrangian $L = \frac{1}{2} J \dot{\theta}_l^2 + Mgl(1 - cos\theta_l)$

Apply to Euler-Lagrange equation

$$J\ddot{\theta}_l + Mglsin\theta_l = \tau_l$$

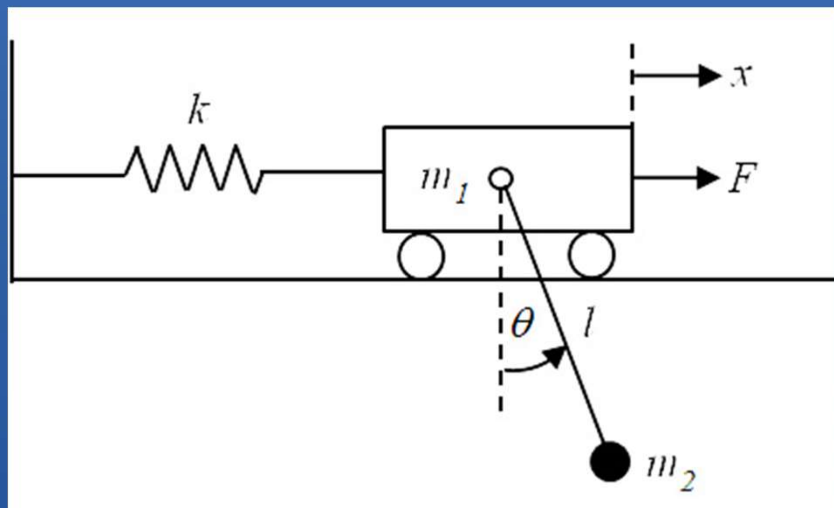
With damping + friction

$$\tau_l = u - B\dot{\theta}_l$$

$$B = rB_m + B_l$$

$$J\ddot{\theta}_l + B\dot{\theta}_l + Mglsin\theta_l = u$$

Example : cart+pendulum



See details in textbook

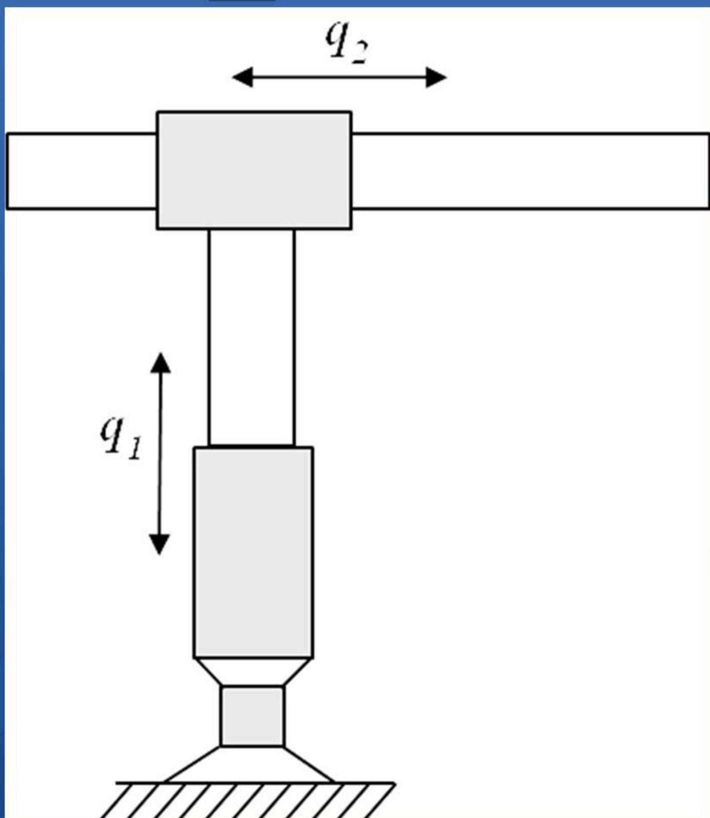
$$\begin{bmatrix} m_1 + m_2 & m_2 l cos\theta \\ m_2 l cos\theta & m_2 l^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & -m_2 l sin\theta \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}^2 \\ \dot{\theta}^2 \end{bmatrix} + \begin{bmatrix} kx \\ m_2 g l sin\theta \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix}$$

Robot Equation of Motion

Applying Euler-Lagrange method to a general robot can be daunting.

Textbook provides some examples of 2-link manipulators.

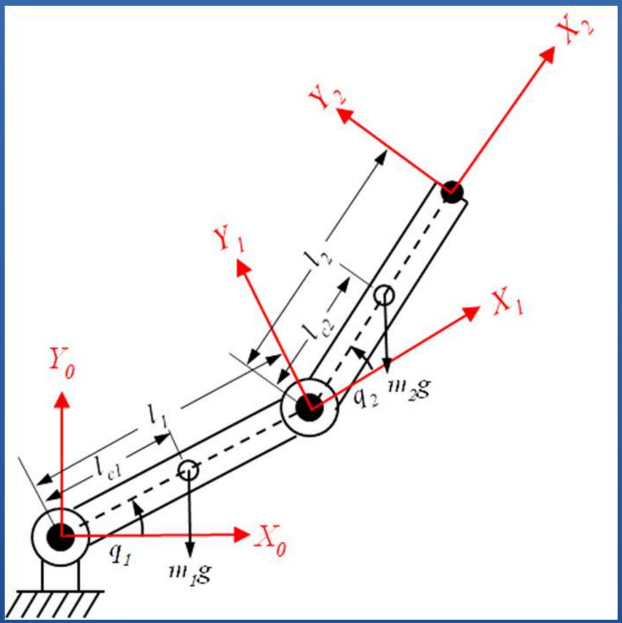
2-link prismatic robot



$$\left(m_1 + m_2\right)\ddot{q}_1 + g\left(m_1 + m_2\right) = f_1$$

$$m_2\ddot{q}_2 = f_2$$

2-link revolute robot



$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + c_{121}\dot{q}_1\dot{q}_2 + c_{211}\dot{q}_2\dot{q}_1 + c_{221}\dot{q}_2^2 +$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + c_{112}\dot{q}_1^2 + g_2 = \tau_2$$

$$c_{111} = \frac{1}{2}\frac{\partial d_{11}}{\partial q_1} = 0$$

$$c_{121} = c_{211} = \frac{1}{2}\frac{\partial d_{11}}{\partial q_2} = -m_2 l_1 l_{c2} sinq_2 = \xi$$

$$c_{221} = \frac{\partial d_{12}}{\partial q_2} - \frac{1}{2}\frac{\partial d_{22}}{\partial q_1} = \xi$$

$$c_{112} = \frac{\partial d_{21}}{\partial q_1} - \frac{1}{2}\frac{\partial d_{11}}{\partial q_2} = -\xi$$

$$c_{122} = c_{212} = \frac{1}{2}\frac{\partial d_{22}}{\partial q_1} = 0$$

$$c_{222} = \frac{1}{2}\frac{\partial d_{22}}{\partial q_2} = 0$$

$$d_{11} = m_1 l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + 2 l_1 l_{c2} cosq_2) + I_1 + I_2$$

$$d_{12} = d_{21} = m_2(l_{c2}^2 + l_1 l_{c2} cosq_2) + I_2$$

$$d_{22} = m_2 l_{c2}^2 + I_2$$

$$g_2 = \frac{\partial P}{\partial q_2} = m_2 l_{c2} g\, cos(q_1 + q_2)$$

$$g_1 = \frac{\partial P}{\partial q_1} = (m_1 l_{c1} + m_2 l_1) g\, cosq_1 + m_2 l_{c2} g\, cos(q_1 + q_2)$$

Newton-Euler method

- Recursive approach
- Suitable for computer programming
- See example in textbook

General form of robot EOM

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + g(q) + J(q)^T\vartheta = \Gamma$$

Inertia matrix

Coriolis and Centrifugal force

Friction

Gravity

Wrench

Generalized force/torque

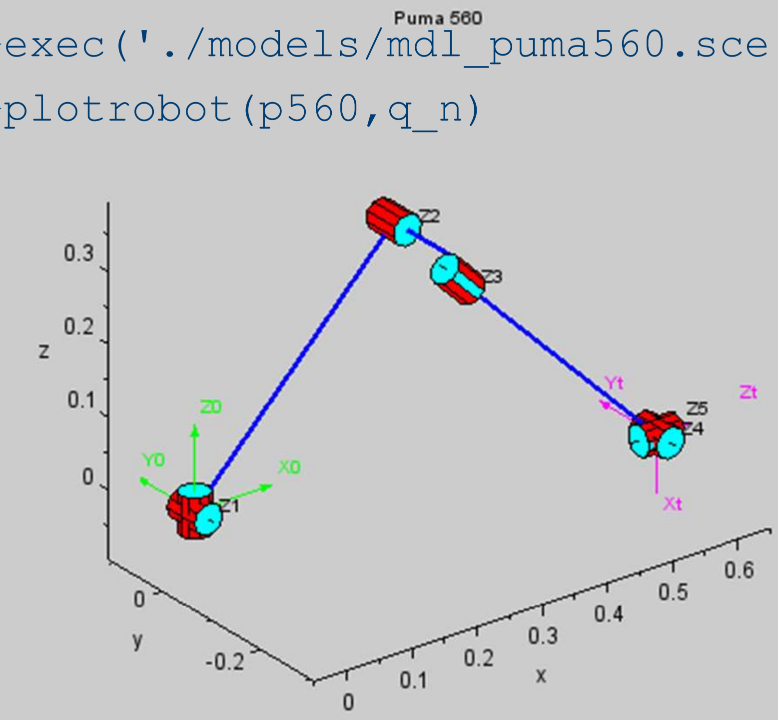
Called “inverse dynamics”

Can compute force/torque given joint position/velocity/acceleration

RTSX rne() function

Create PUMA560

```
-->exec('./models/mdl_puma560.sce',-1);
-->plotrobot(p560,q_n)
```



RTSX rne() function

RTSX

Nominal pose,
Zero velocity/
acceleration

```
-->Tq = rne(p560, q_n, zeros(1,6), zeros(1,6))
 Tq  =
    0.    31.63988    6.035138    0.    0.0282528    0.
```

Without gravity

```
-->Tq = rne(p560, q_n, zeros(1,6), zeros(1,6),[0 0 0]')
 Tq  =
    0.    0.    0.    0.    0.    0.
```

Joint 1 moves at
Velocity 1 rad/s
No gravity

```
-->rne(p560,q_n,[1 0 0 0 0 0],zeros(1,6),[0 0 0]')
 ans  =
- 30.533206    0.6280224  - 0.3607472  - 0.0003056    0.    0.
```

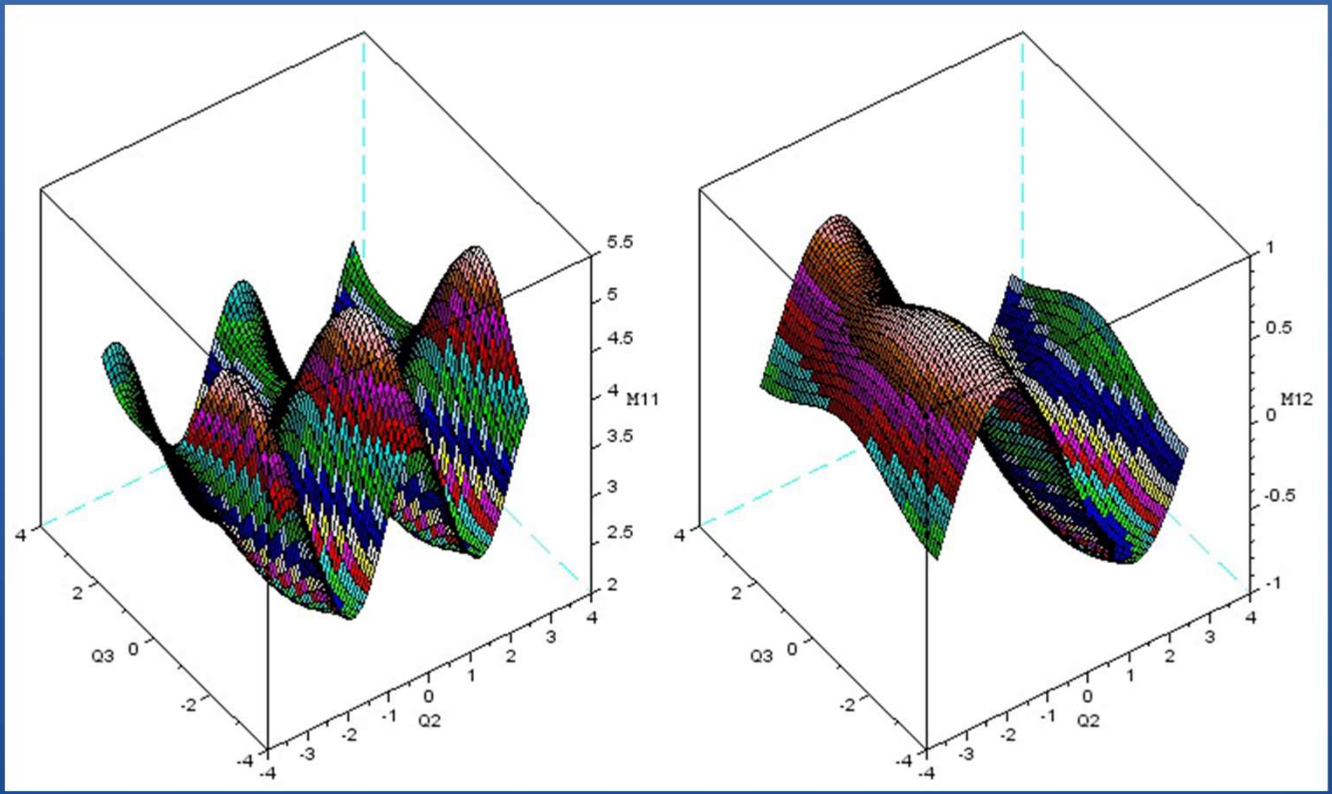
RTSX inertia() function

- Compute inertia matrix

```
-->M=inertia(p560,q_n)
M  =
    3.6593754  - 0.4043612    0.1006136  - 0.0025170    0.           0.
  - 0.4043612    4.4137419    0.3508907    0.           0.0023595    0.
    0.1006136    0.3508907    0.9378416    0.           0.0014802    0.
  - 0.0025170    0.           0.           0.1925317    0.           0.0000283
    0.           0.0023595    0.0014802    0.           0.1713485    0.
    0.           0.           0.           0.0000283    0.           0.1941045
```

Inertia matrix when q2, q3 move

Significant Change!



m_{11} m_{12}

RTSX coriolis() function

- Compute coriolis matrix
- Ex. all joints move at velocity 0.5 rad/s

```
-->q_d=0.5*[1 1 1 1 1 1]; C=coriolis(p560,q_n,q_d)
 C  =
 
    0.          - 0.9115459    0.2172555    0.0012865  - 0.0025932    0.00006
    0.3140112   - 5.551D-17    0.5786335  - 0.0010762  - 0.0001034  - 0.0000059
  - 0.1803736   - 0.1928778    0.         - 0.0004544  - 0.0023017  - 0.0000059
  - 0.0001528     0.0005860  - 0.0000358    0.            0.0002566  - 0.0000424
    0.            0.0000207    0.0013810  - 0.0002061    0.         - 0.0000059
    0.            0.00002      0.00002      0.0000283    0.0000059    0.
```

RTSX gravload() function

- Equivalent to rne() with no joint movement

```
-->gravload(p560,q_n)
 ans  =
    0.    31.63988    6.035138    0.    0.0282528    0.
```

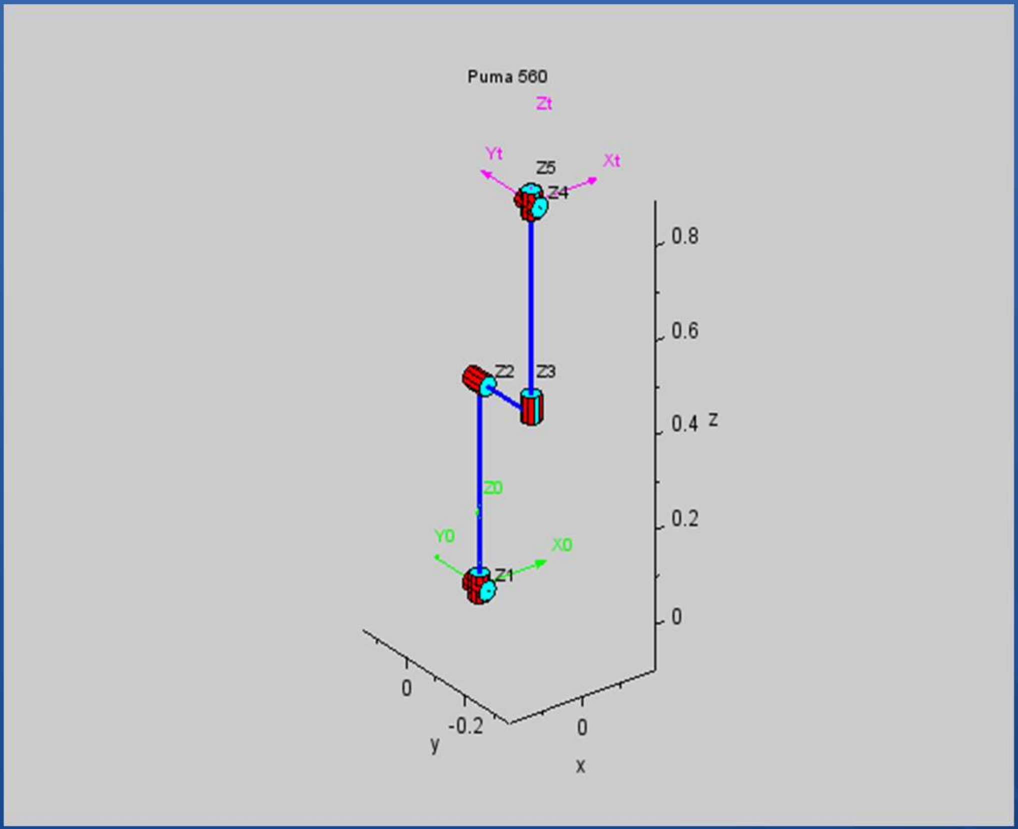
Suppose the robot is attached to the ceiling

```
-->p560a = AttachBase(p560,trotx(pi));
-->gravload(p560a, q_n)
 ans  =
    0.  - 31.63988  - 6.035138    0.  - 0.0282528    0.
```

RTSX gravload() function

- Torques at q_r pose

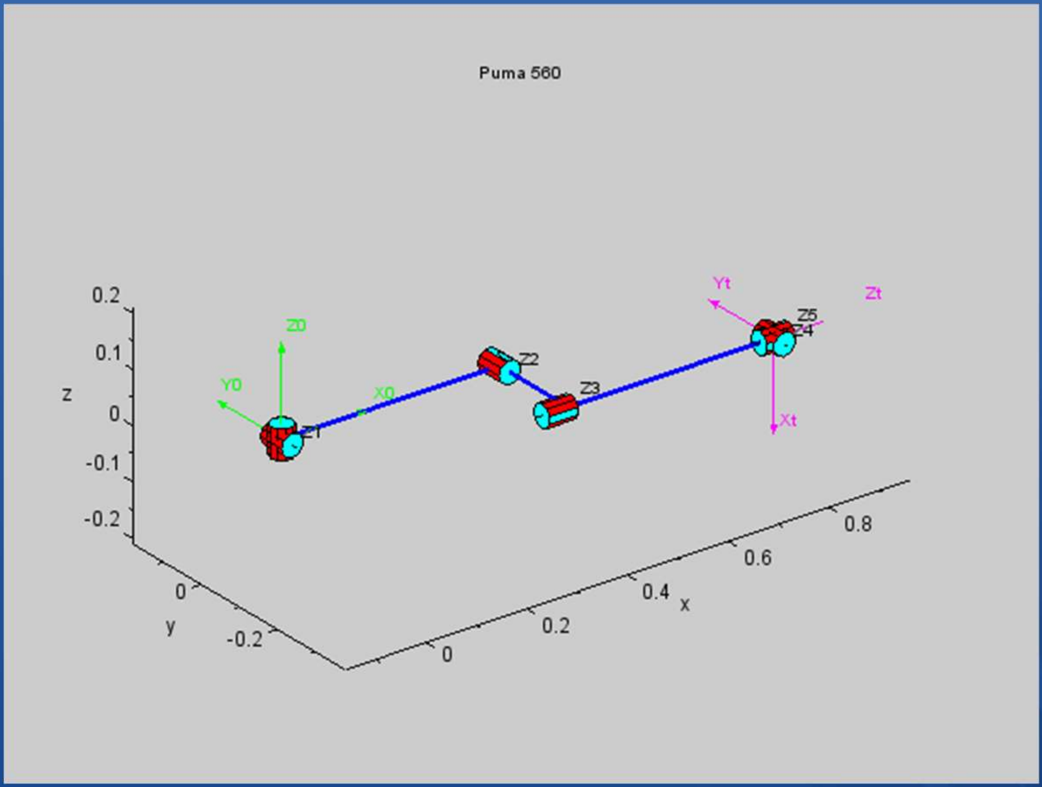
```
-->Tq = gravload(p560,q_r)
 Tq  =
    0.   - 0.7752352    0.2489287
```



RTSX gravload() function

- Torque at q_s pose

```
-->Tq = gravload(p560,q_s)
 Tq  =
    0.     46.006938    8.7722001
```



RTSX accel() function

$$\ddot{q} = M^{-1}(q)(\Gamma - C(q,\dot{q})\dot{q} - F(\dot{q}) - g(q))$$

Called “forward dynamics”

Can compute acceleration given torque/force

```
-->qdd = accel(p560,q_n, [1 0 0 0 0 0], [1 0 0 0 0 0])'
 qdd  =
 8.0575111  - 6.2074899  - 4.5923293    0.1069225  - 0.0397362  - 0.0000156
```