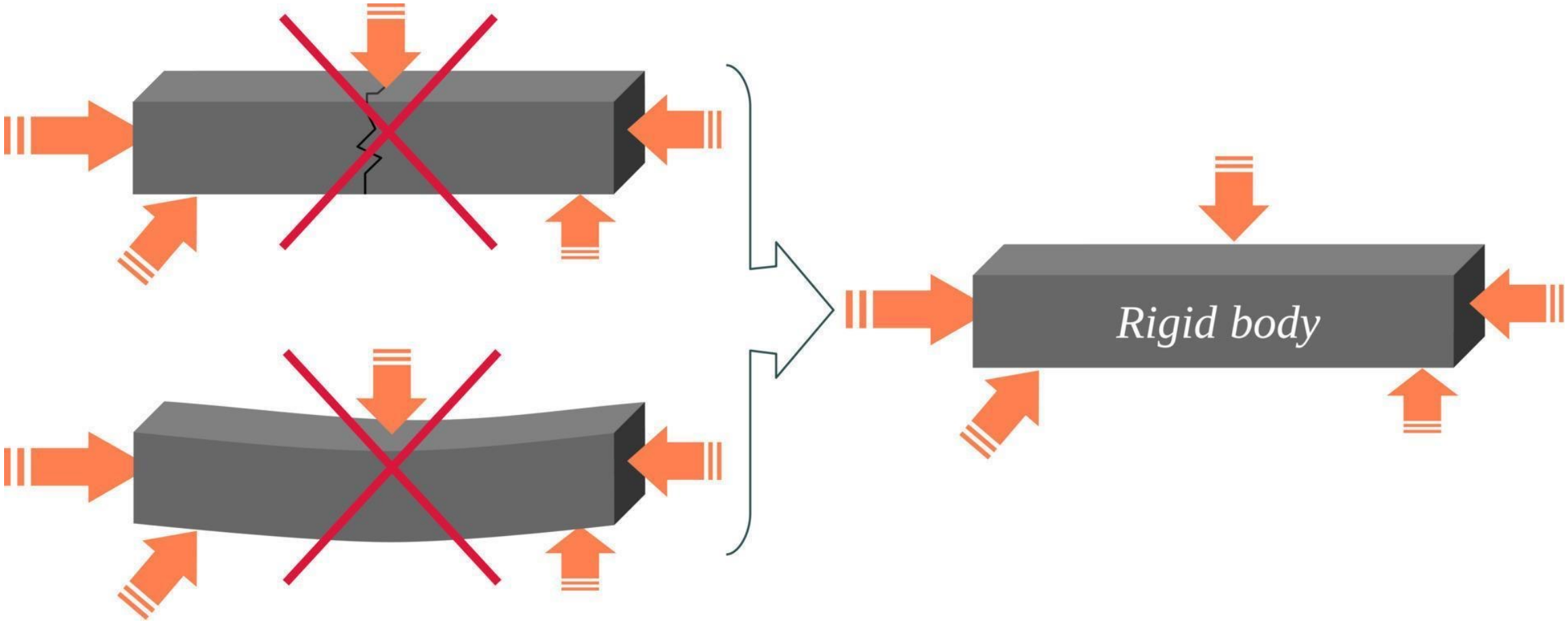
WPI

Lecture 14

Intro to Manipulator Dynamics and Ethics in Robotics

Alexandros Lioulemes, PhD

Rigid Body



Acceleration of a Rigid Body

- The linear and angular accelerations are the time derivatives of the linear and angular velocity

$$ {}^{B}\dot{V}_{Q} = \frac{d}{dt}\,{}^{B}V_{Q} = \lim_{\Delta t \to o} \frac{{}^{B}V_{Q}(t+\Delta t) - {}^{B}V_{Q}(t)}{\Delta t}, $$

$$ {}^{A}\dot{\Omega}_{B} = \frac{d}{dt}\,{}^{A}\Omega_{B} = \lim_{\Delta t \to o} \frac{{}^{A}\Omega_{B}(t+\Delta t) - {}^{A}\Omega_{B}(t)}{\Delta t}. $$

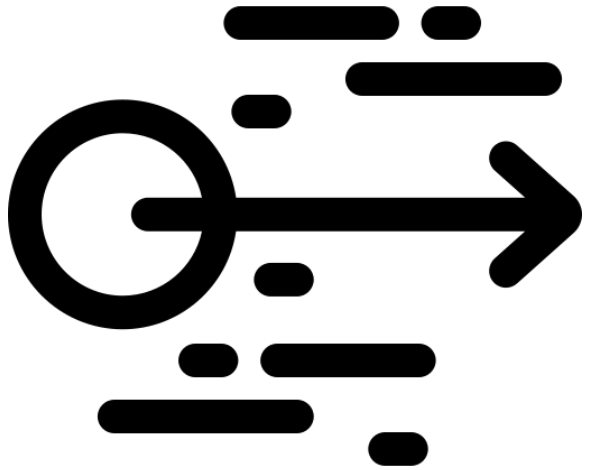
$$ \dot{v}_{A} = {}^{U}\dot{V}_{AORG} $$

$$ \dot{\omega}_{A} = {}^{U}\dot{\Omega}_{A}. $$

Linear acceleration

- The rate of change of linear velocity with respect to time

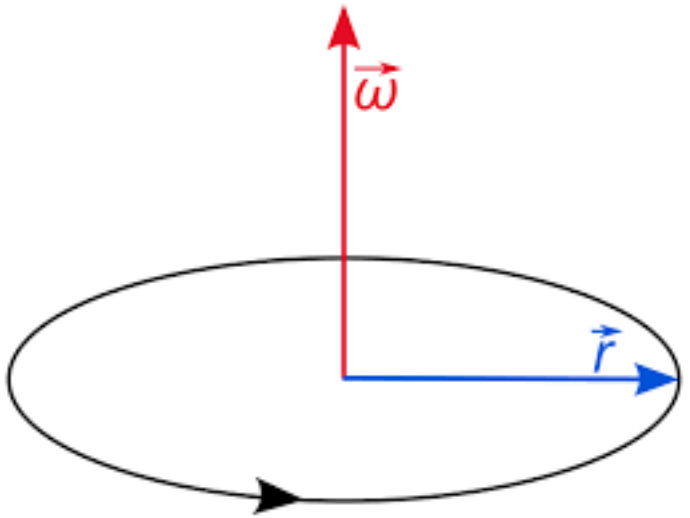
$$ {}^{A}\dot{V}_{Q} = {}^{A}\dot{V}_{BORG} + {}^{A}\Omega_{B} \times ({}^{A}\Omega_{B} \times {}^{A}_{B}R\,{}^{B}Q) + {}^{A}\dot{\Omega}_{B} \times {}^{A}_{B}R\,{}^{B}Q. $$



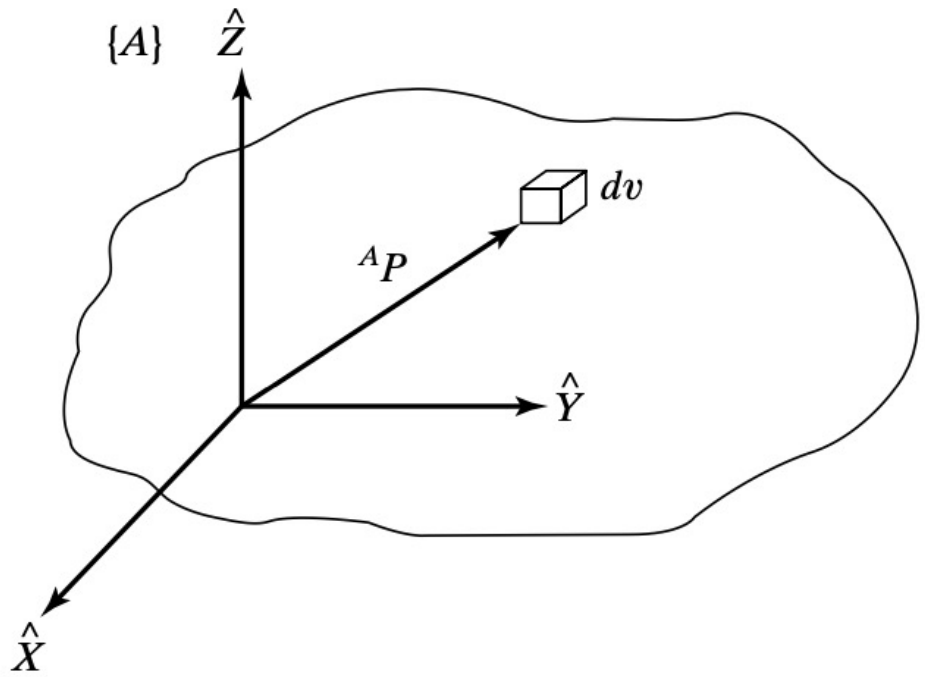
Angular acceleration

- The rate at which the angular velocity of a rigid body changes with respect to time

$$ {}^{A}\dot{\Omega}_{C} = {}^{A}\dot{\Omega}_{B} + {}^{A}_{B}R\,{}^{B}\dot{\Omega}_{C} + {}^{A}\Omega_{B} \times {}^{A}_{B}R\,{}^{B}\Omega_{C}. $$



Mass Distribution



$$^{A}I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix},$$

$$I_{xx} = \iiint_V (y^2 + z^2)\rho dv,$$

$$I_{yy} = \iiint_V (x^2 + z^2)\rho dv,$$

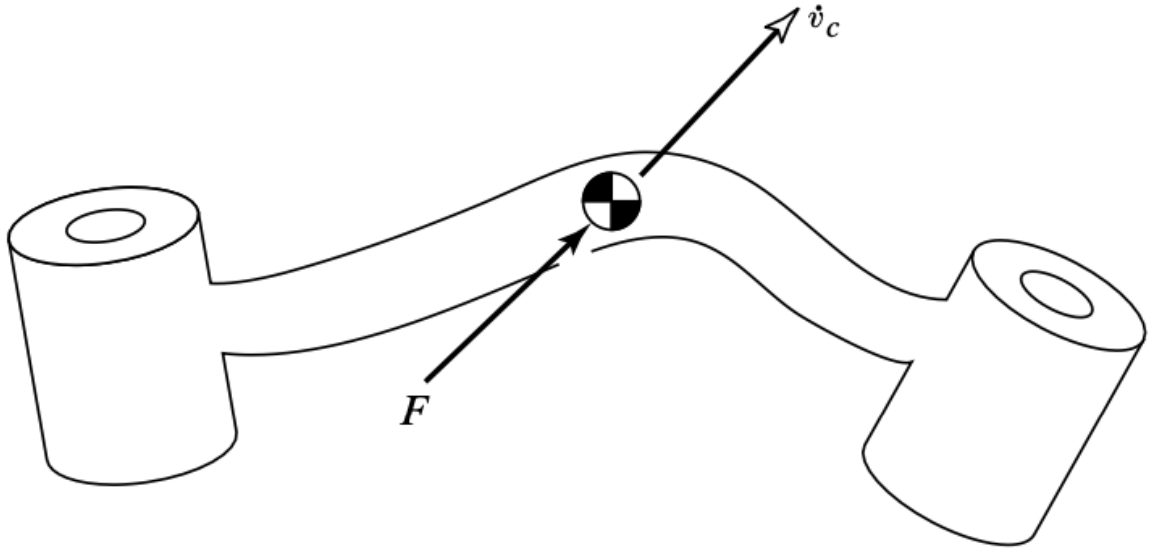
$$I_{zz} = \iiint_V (x^2 + y^2)\rho dv,$$

$$I_{xy} = \iiint_V xy\rho dv,$$

$$I_{xz} = \iiint_V xz\rho dv,$$

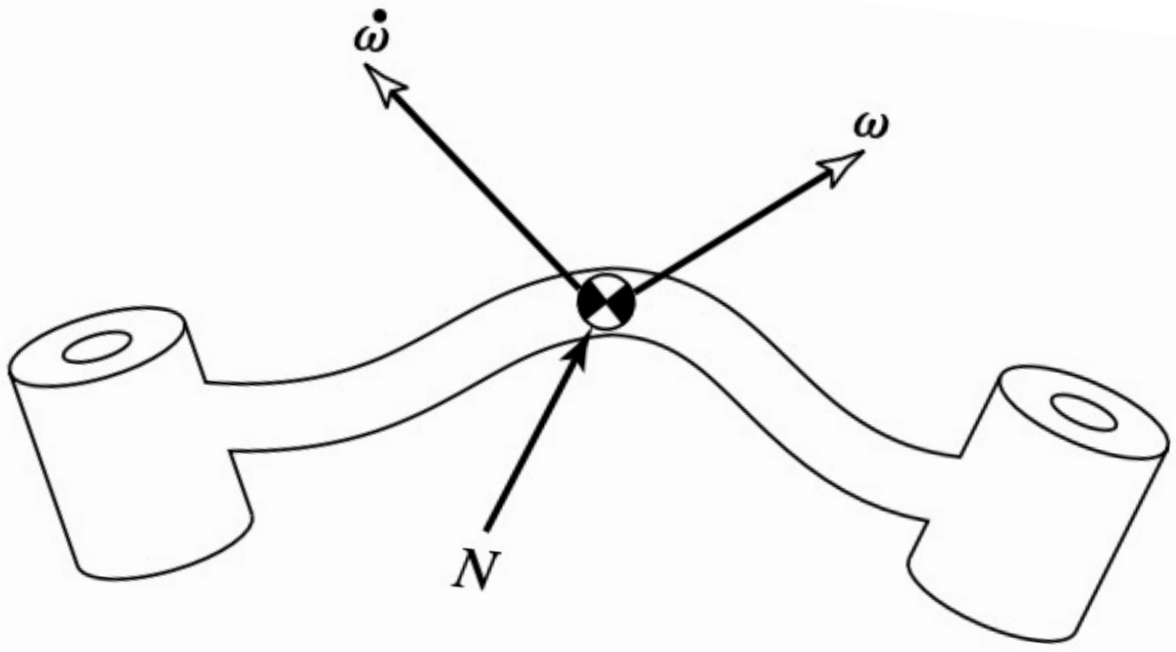
$$I_{yz} = \iiint_V yz\rho dv,$$

Newton's Equation



$$F = m\dot{v}_C$$

Euler's Equation



$$N = {}^{C}I\dot{\omega} + \omega \times {}^{C}I\omega$$

Iterative Newton-Euler Dynamic Formulation

- *Determine the joint torques* required to follow a specified manipulator trajectory, assuming that the *joint positions, velocities, and accelerations are known*
- Given this information, along with the robot's kinematic structure and mass distribution, we can compute the torques necessary to generate the desired motion
- Recursive Newton-Euler (RNE)

The iterative Newton-Euler dynamics algorithm

- Outward iterations to compute velocities and accelerations

Outward iterations: $i : 0 \rightarrow 5$

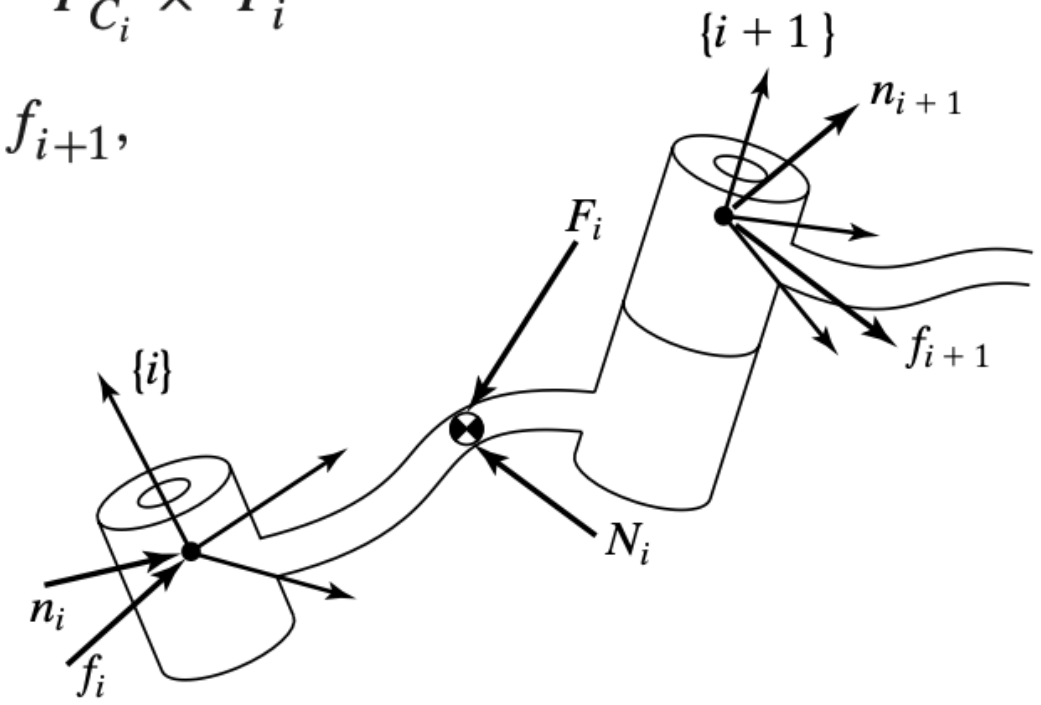
$$
\begin{aligned}
{}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1},\\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i + {}^{i+1}_{i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1},\\
{}^{i+1}\dot{v}_{i+1} &= {}^{i+1}_{i}R({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i),\\
{}^{i+1}\dot{v}_{C_{i+1}} &= {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{C_{i+1}}\\
&\quad + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{C_{i+1}}) + {}^{i+1}\dot{v}_{i+1},\\
{}^{i+1}F_{i+1} &= m_{i+1}\,{}^{i+1}\dot{v}_{C_{i+1}},\\
{}^{i+1}N_{i+1} &= {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\omega_{i+1}.
\end{aligned}
$$

Iterative Newton-Euler Dynamic Formulation

- Inward iterations to compute forces and torques

Inward iterations: $i : 6 \rightarrow 1$

$$
\begin{aligned}
{}^{i}f_i &= {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} + {}^{i}F_i, \\
{}^{i}n_i &= {}^{i}N_i + {}^{i}_{i+1}R\,{}^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_i \\
&\quad + {}^{i}P_{i+1} \times {}^{i}_{i+1}R\,{}^{i+1}f_{i+1}, \\
\tau_i &= {}^{i}n_i^T\,{}^{i}\hat{Z}_i.
\end{aligned}
$$



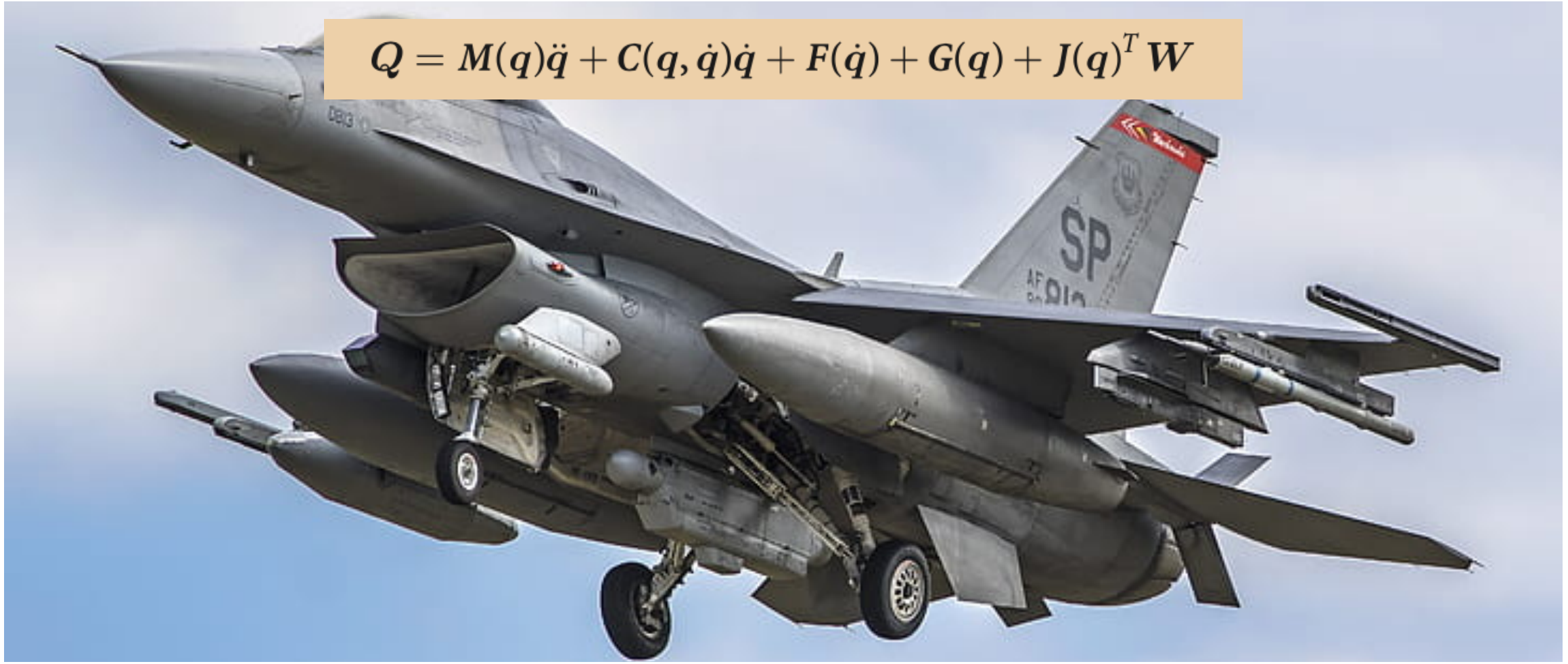
Langangian Formulation of Manipulator Dynamics

- Lagrangian method is rooted in the principles of energy
- L = kinetic - potential energy $\mathcal{L}(\Theta, \dot{\Theta}) = k(\Theta, \dot{\Theta}) - u(\Theta)$
- Euler-Lagrange equation: $\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\Theta}} - \frac{\partial \mathcal{L}}{\partial \Theta} = \tau$


SP

Rigid-Body Equations of Motions

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$



Rigid-Body Equations of Motions - RNE

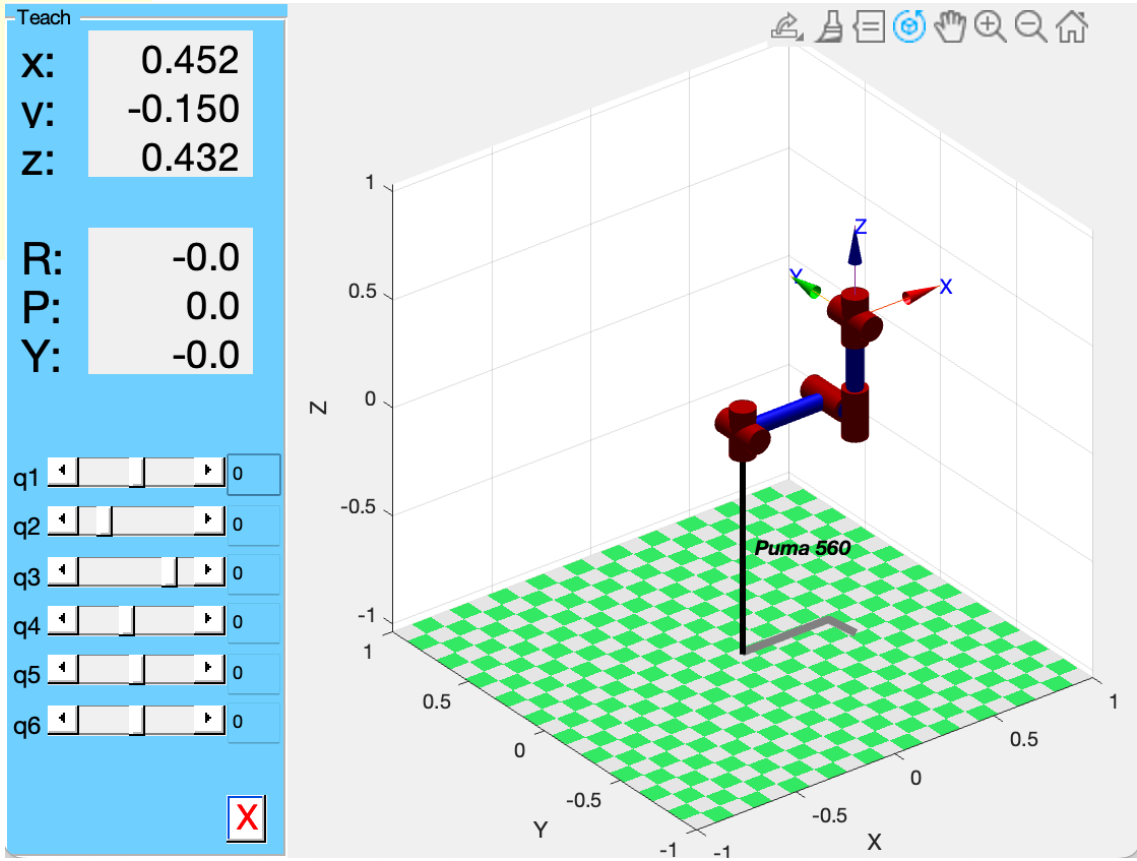
```
>> help rne
--- help for SerialLink/rne ---

 SerialLink/rne Inverse dynamics

  TAU = R.rne(Q, QD, QDD, OPTIONS) is the joint torque required for the
  robot R to achieve the specified joint position Q (1xN), velocity QD
  (1xN) and acceleration QDD (1xN), where N is the number of robot joints.

  TAU = R.rne(X, OPTIONS) as above where X=[Q,QD,QDD] (1x3N).

  [TAU,WBASE] = R.rne(X, GRAV, FEXT) as above but the extra output is the
  wrench on the base.

  Options::
   'gravity',G    specify gravity acceleration (default [0,0,9.81])
   'fext',W       specify wrench acting on the end-effector W=[Fx Fy Fz Mx My Mz]
   'slow'         do not use MEX file

  Trajectory operation::

  If Q,QD and QDD (MxN), or X (Mx3N) are matrices with M rows representing a
  trajectory then TAU (MxN) is a matrix with rows corresponding to each trajectory
  step.
```

Rigid-Body Equations of Motions - RNE

```
>> mdl_puma560
>> Q = p560.rne(qn, qz, qz)

Q =

   -0.0000   31.6399    6.0351    0.0000    0.0283
```

$$\dot{q} = \ddot{q} = 0$$

```
>> mdl_puma560
>> Q = p560.rne(qn, qz, qz, 'gravity', [0 0 0])

Q =

     0     0     0     0     0     0
```



Rigid-Body Equations of Motions - RNE

```
>> q = jtraj(qz, qr, 10)

q =

         0         0         0         0         0         0
         0    0.0181   -0.0181         0         0         0
         0    0.1200   -0.1200         0         0         0
         0    0.3297   -0.3297         0         0         0
         0    0.6231   -0.6231         0         0         0
         0    0.9477   -0.9477         0         0         0
         0    1.2411   -1.2411         0         0         0
         0    1.4508   -1.4508         0         0         0
         0    1.5527   -1.5527         0         0         0
         0    1.5708   -1.5708         0         0         0
```

Rigid-Body Equations of Motions - RNE

```
>> q = jtraj(qz, qr, 10)

q =

         0         0         0         0         0         0
         0    0.0181   -0.0181         0         0         0
         0    0.1200   -0.1200         0         0         0
         0    0.3297   -0.3297         0         0         0
         0    0.6231   -0.6231         0         0         0
         0    0.9477   -0.9477         0         0         0
         0    1.2411   -1.2411         0         0         0
         0    1.4508   -1.4508         0         0         0
         0    1.5527   -1.5527         0         0         0
         0    1.5708   -1.5708         0         0         0
```

```
>> Q = p560.rne(q, 0*q, 0*q)

Q =

         0   37.4837    0.2489         0         0         0
    0.0000   37.4590    0.2489         0         0         0
   -0.0000   37.0931    0.2489    0.0000    0.0000         0
    0.0000   35.1470    0.2489   -0.0000   -0.0000         0
    0.0000   29.8883    0.2489         0         0         0
    0.0000   21.1462    0.2489         0         0         0
   -0.0000   11.3341    0.2489    0.0000    0.0000         0
    0.0000    3.6906    0.2489   -0.0000   -0.0000         0
    0.0000   -0.1006    0.2489         0         0         0
   -0.0000   -0.7752    0.2489         0         0         0
```

```
>> about(Q)
Q [double] : 10x6 (480 bytes)
>> Q(5,:)

ans =

    0.0000   29.8883    0.2489         0         0         0
```

Rigid-Body Equations of Motions - RNE

```
>> q = jtraj(qz, qr, 10)

q =

         0         0         0         0         0         0
         0    0.0181   -0.0181         0         0         0
         0    0.1200   -0.1200         0         0         0
         0    0.3297   -0.3297         0         0         0
         0    0.6231   -0.6231         0         0         0
         0    0.9477   -0.9477         0         0         0
         0    1.2411   -1.2411         0         0         0
         0    1.4508   -1.4508         0         0         0
         0    1.5527   -1.5527         0         0         0
         0    1.5708   -1.5708         0         0         0
```

```
>> Q = p560.rne(q, 0*q, 0*q)

Q =

         0   37.4837    0.2489         0         0         0
    0.0000   37.4590    0.2489         0         0         0
   -0.0000   37.0931    0.2489    0.0000    0.0000         0
    0.0000   35.1470    0.2489   -0.0000   -0.0000         0
    0.0000   29.8883    0.2489         0         0         0
    0.0000   21.1462    0.2489         0         0         0
   -0.0000   11.3341    0.2489    0.0000    0.0000         0
    0.0000    3.6906    0.2489   -0.0000   -0.0000         0
    0.0000   -0.1006    0.2489         0         0         0
   -0.0000   -0.7752    0.2489         0         0         0
```

```
>> about(Q)
Q [double] : 10x6 (480 bytes)
>> Q(5,:)

ans =

    0.0000   29.8883    0.2489         0         0         0

>> p560.rne(qn, [1 0 0 0 0 0], qz, 'gravity', [0 0 0])

ans =

   30.5332    0.6280   -0.3607   -0.0003   -0.0000         0
```

Rigid-Body Equations of Motions - Gravity term

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$

```
>> gravload = p560.gravload(qn)

gravload =

   -0.0000   31.6399    6.0351    0.0000    0.0283         0

>> p560.gravity'

ans =

         0         0    9.8100
```

Rigid-Body Equations of Motions - Inertia matrix

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$

```
>> M = p560.inertia(qn)

M =

    3.6594   -0.4044    0.1006   -0.0025    0.0000   -0.0000
   -0.4044    4.4137    0.3509    0.0000    0.0024    0.0000
    0.1006    0.3509    0.9378    0.0000    0.0015    0.0000
   -0.0025    0.0000    0.0000    0.1925    0.0000    0.0000
    0.0000    0.0024    0.0015    0.0000    0.1713    0.0000
   -0.0000    0.0000    0.0000    0.0000    0.0000    0.1941
```

Rigid-Body Equations of Motions - Coriolis matrix

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$

```
>> qd = [0 0 1 0 0 0];
>> C = p560.coriolis(qn, qd)

C =

    0.3607   -0.0957   -0.0957    0.0005   -0.0004    0.0000
   -0.0000    0.3858    0.3858         0   -0.0000         0
         0         0   -0.0000    0.0000   -0.0009         0
   -0.0000    0.0000    0.0000   -0.0000    0.0000   -0.0000
    0.0000    0.0009    0.0009         0         0         0
    0.0000         0         0    0.0000         0         0
```

Rigid-Body Equations of Motions - Friction

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$

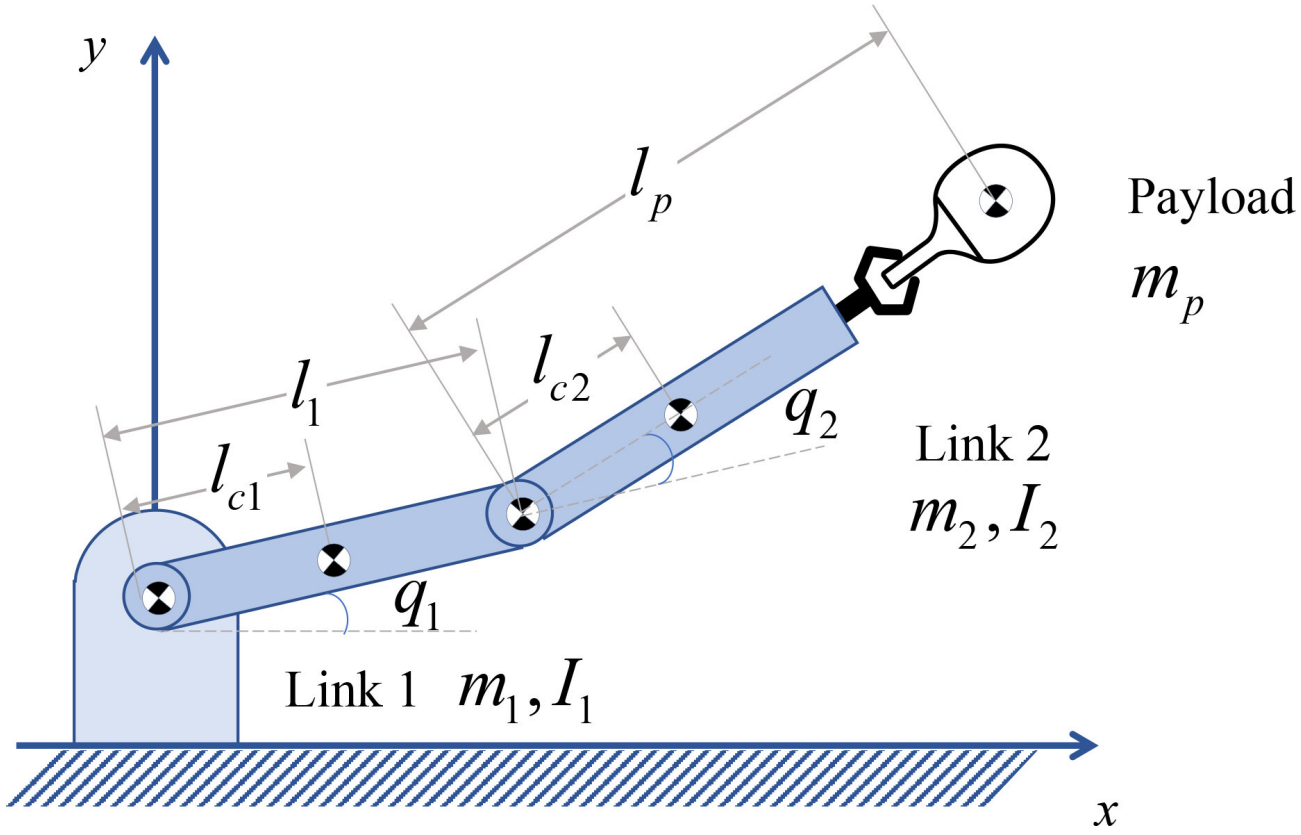
```
>> p560.links(1).dyn
Revolute(std): theta=q, d=0, a=0, alpha=1.5708, offset=0
  m     = 0
  r     = 0            0            0
  I     = | 0            0            0            |
          | 0            0.35         0            |
          | 0            0            0            |
  Jm    = 0.0002
  Bm    = 0.00148
  Tc    = 0.395      (+) -0.435     (-)
  G     = -62.61
  qlim = -2.792527 to 2.792527
```

Rigid-Body Equations of Motions - Friction

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$

```
>> p560.links(1).dyn
Revolute(std): theta=q, d=0, a=0, alpha=1.5708, offset=0
  m     = 0
  r     = 0            0            0
  I     = | 0            0            0           |
          | 0            0.35         0           |
          | 0            0            0           |
  Jm    = 0.0002
  Bm    = 0.00148
  Tc    = 0.395      (+) -0.435     (-)
  G     = -62.61
  qlim = -2.792527 to 2.792527
```

Rigid-Body Equations of Motions - Payload effect



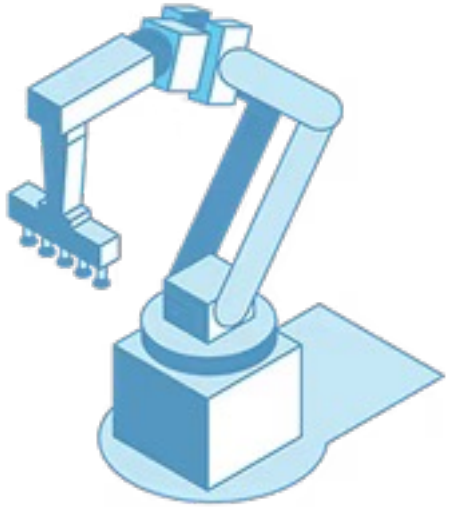
Rigid-Body Equations of Motions - Payload effect

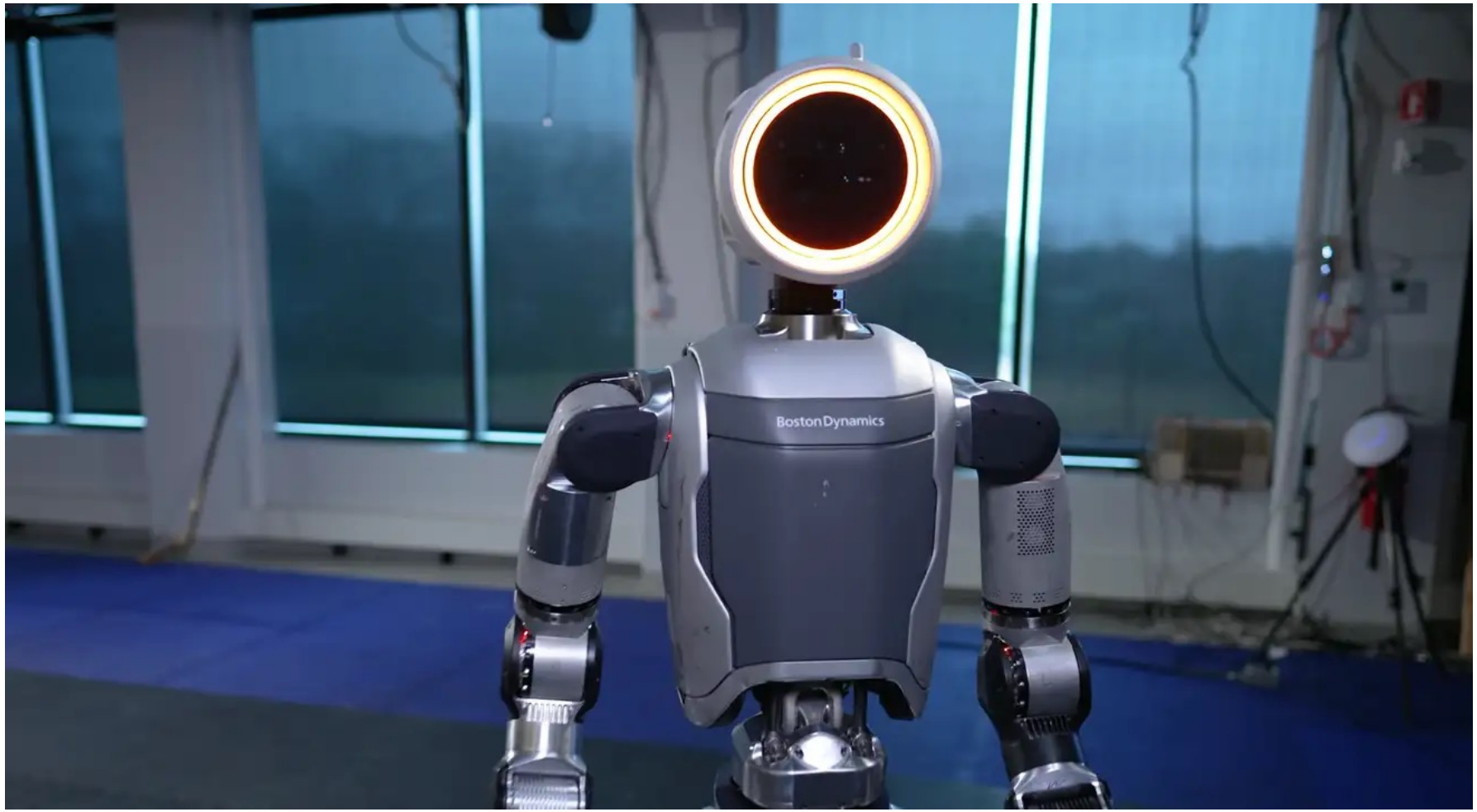
```
>> p560.payload(2.5, [0 0 0.1]);
>> M_loaded = p560.inertia(qn)

M_loaded =

    4.8902   -0.3992    0.2162   -0.1243    0.0000   -0.0000
   -0.3992    5.5908    1.0243    0.0000    0.1746    0.0000
    0.2162    1.0243    1.5569   -0.0000    0.0983    0.0000
   -0.1243    0.0000   -0.0000    0.2050    0.0000    0.0000
    0.0000    0.1746    0.0983    0.0000    0.1963    0.0000
   -0.0000    0.0000    0.0000    0.0000    0.0000    0.1941
```

```
>> M = p560.inertia(qn)

M =

    3.6594   -0.4044    0.1006   -0.0025    0.0000   -0.0000
   -0.4044    4.4137    0.3509    0.0000    0.0024    0.0000
    0.1006    0.3509    0.9378    0.0000    0.0015    0.0000
   -0.0025    0.0000    0.0000    0.1925    0.0000    0.0000
    0.0000    0.0024    0.0015    0.0000    0.1713    0.0000
   -0.0000    0.0000    0.0000    0.0000    0.0000    0.1941
```

Rigid-Body Equations of Motions - Base Force

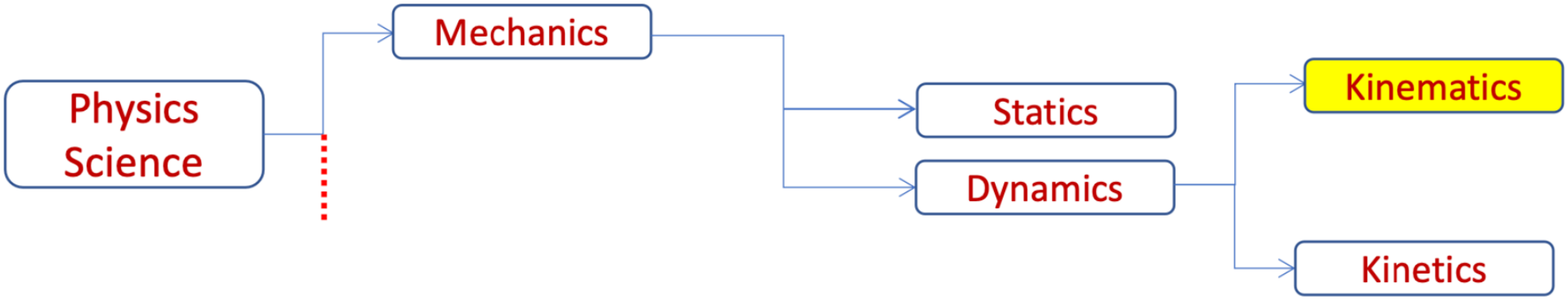
```
>> [Q,Wb] = p560.rne(qn, qz, qz);
>> Wb'

ans =

   -0.0000    0.0000  253.6866  -51.9499  -48.1620   -0.0000

>> sum([p560.links.m])

ans =

   25.8600
```




Boston Dynamics

Manipulator Kinematics and Dynamics

- **Kinematics**: Study of the motion of the body including position, velocity, and acceleration without considering the force causing the motion.
- **Kinetics**: Study of forces in the system. Usually known as **Dynamics**!



Statics

- The reason why we study Statics is that we want to figure out the **joint torques** required to counteract the **external forces** exerted on the robot's end-effector

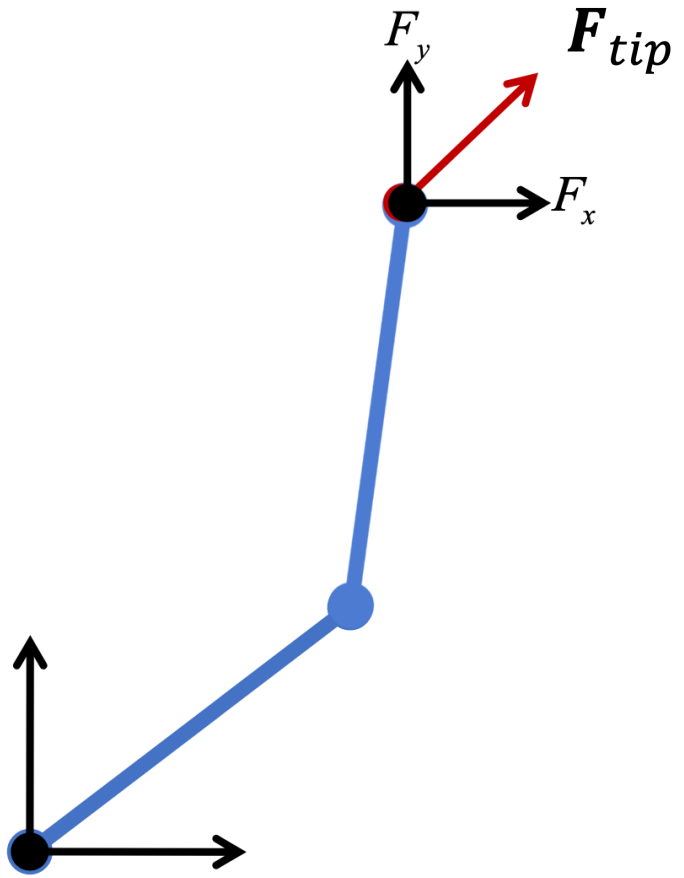
Dynamics

- The reason why we study Dynamics is that we want to know how to generate **control signals (torques)** for our robot to stick to a **predefined trajectory**.

Static Forces

- We seek to model the relation between joint torques and task-space forces
- $\boldsymbol{F}_{tip}$: External force applied to the end-effector

$$\boldsymbol{\tau} \Leftrightarrow \boldsymbol{F}_{tip}$$



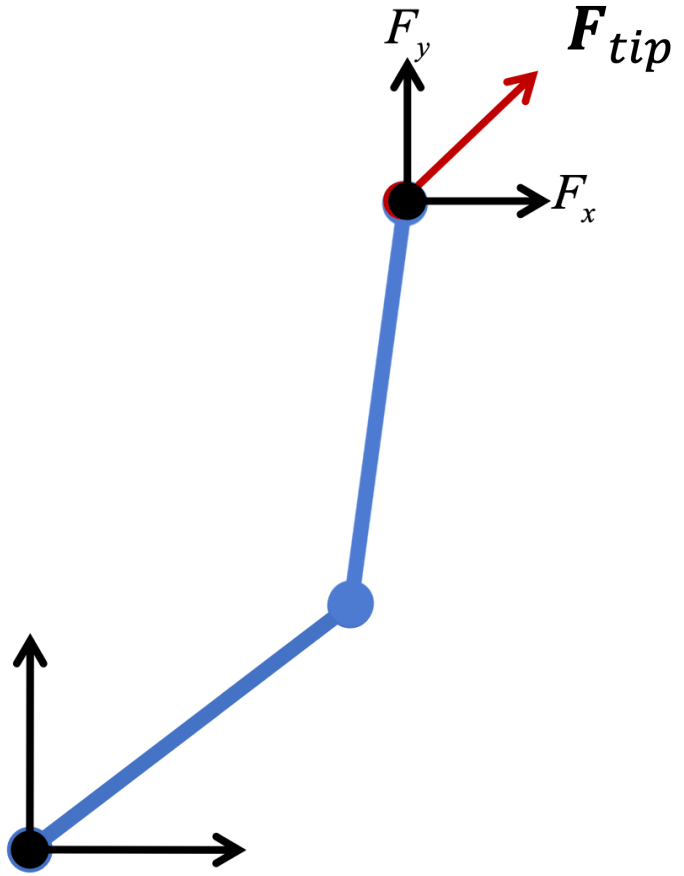
Static Forces

- We seek to model the relation between joint torques and task-space forces
- $\boldsymbol{F}_{tip}$: External force applied to the end-effector

$$\boldsymbol{\tau} \Leftrightarrow \boldsymbol{F}_{tip}$$

$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{F}_{tip}$$

Manipulator Jacobian

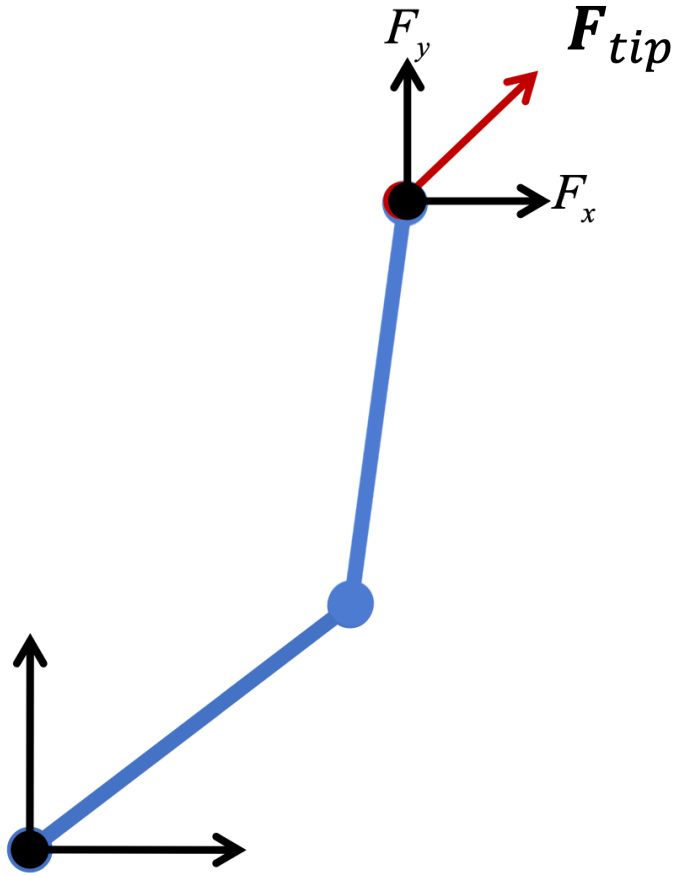


Static Forces

What is the dimension of $\boldsymbol{\tau}$, $\boldsymbol{J}$ and $\boldsymbol{F}$?

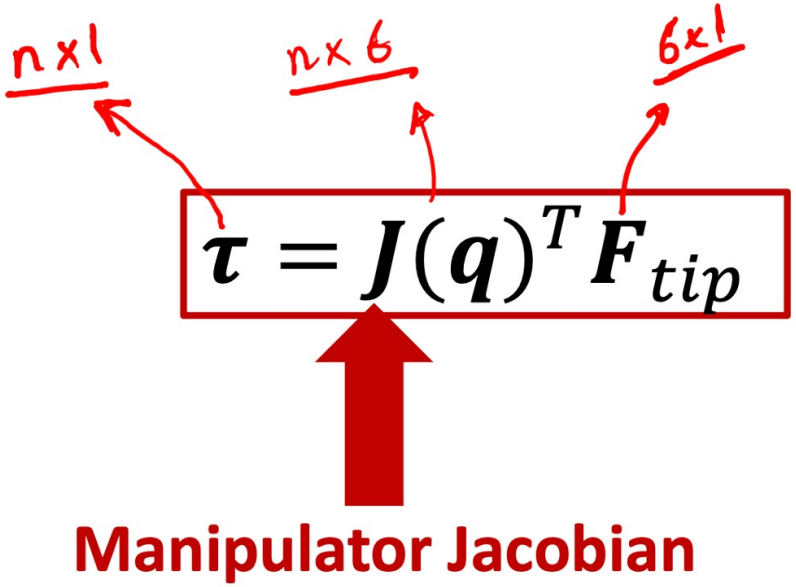
$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{F}_{tip}$$

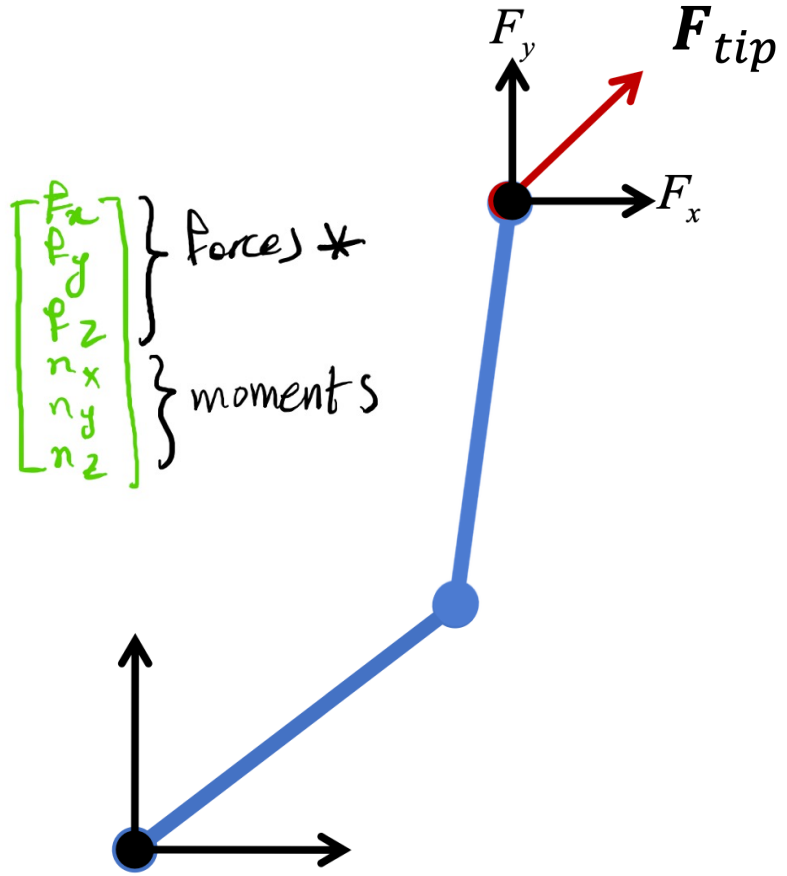
Manipulator Jacobian



Static Forces

What is the dimension of $\boldsymbol{\tau}$, $\boldsymbol{J}$ and $\boldsymbol{F}$?



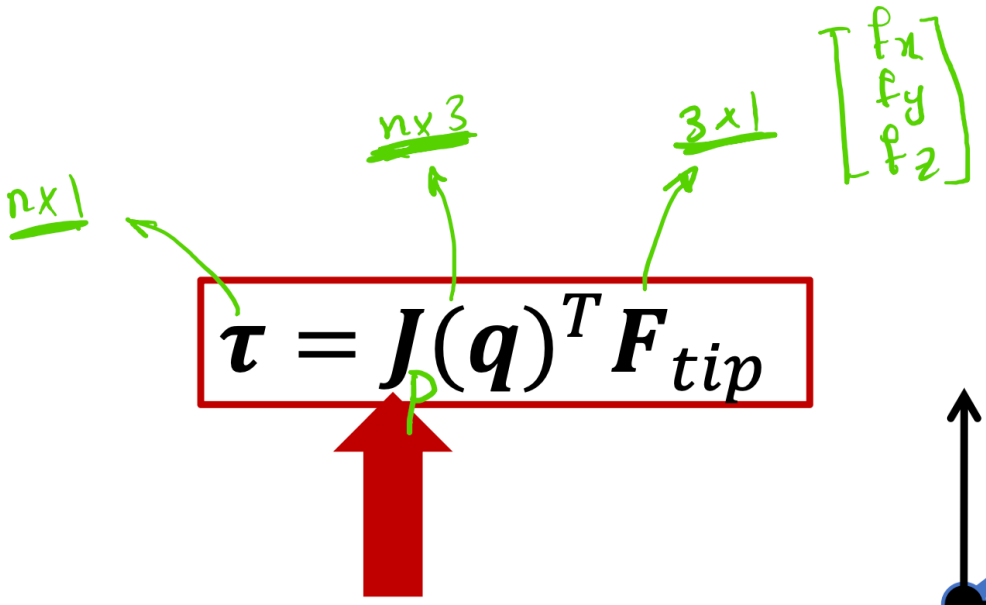


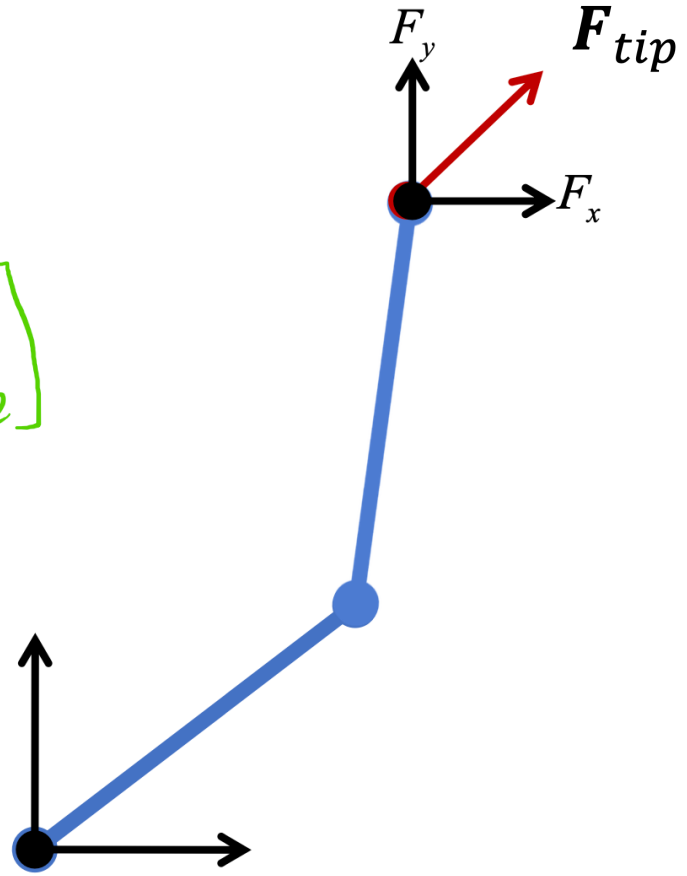
Static Forces

Use the upper half of Jacobian (J_p) if dealing only with forces (Fx, Fy, Fz)

Static Forces

What is the dimension of $\boldsymbol{\tau}$, $\boldsymbol{J}$ and $\boldsymbol{F}$?



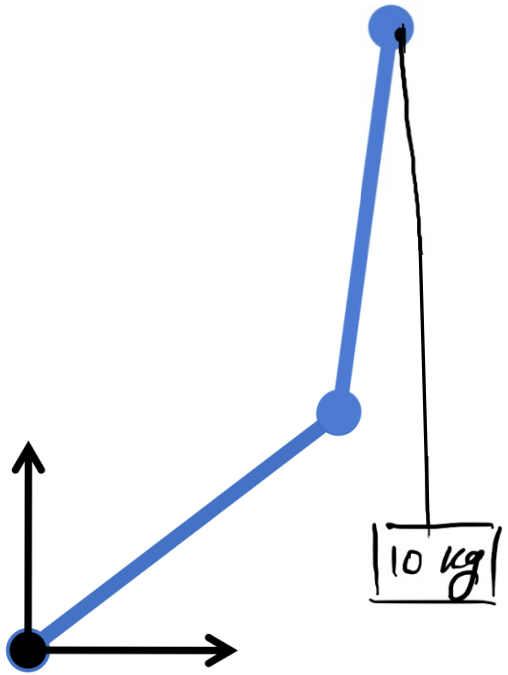


Static Forces - Example

What vector of joint toques do I need to counteract the tip force if the robot is to hold a 10 kg weight on its tip?

Static Forces - Example

What vector of joint toques do I need to counteract the tip force if the robot is to hold a 10 kg weight on its tip?

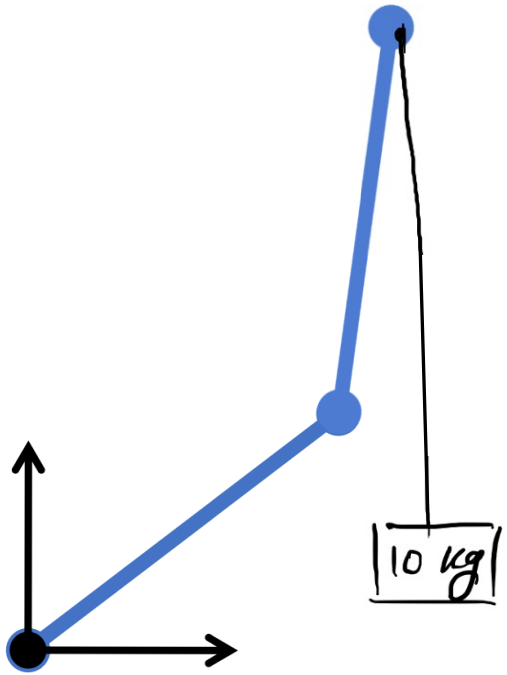


Static Forces - Example

What vector of joint toques do I need to counteract the tip force if the robot is to hold a 10 kg weight on its tip?

$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{F}_{tip}$$

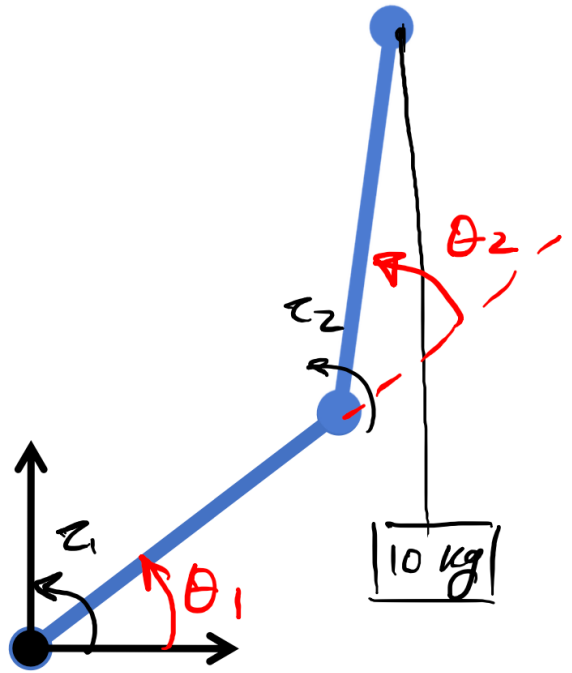
$$J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$



Static Forces - Example - Solution

What vector of joint toques do I need to counteract the tip force if the robot is to hold a 10 kg weight on its tip?

$$\vec{\tau} = J^T \vec{F}_{tip}$$

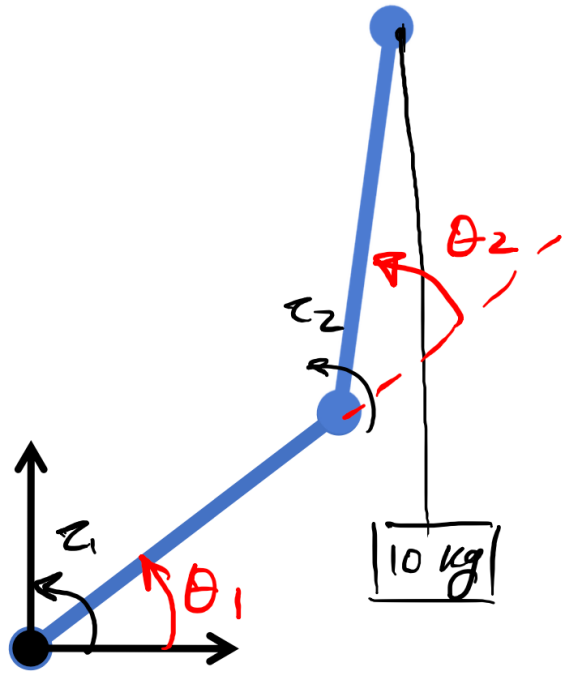


Static Forces - Example - Solution

What vector of joint toques do I need to counteract the tip force if the robot is to hold a 10 kg weight on its tip?

$$\vec{\tau} = J^T \vec{F}_{tip}$$

$$\vec{F}_{tip} = \begin{bmatrix} 0 \\ -98 \end{bmatrix}_{2\times 1}$$



Static Forces - Example - Solution

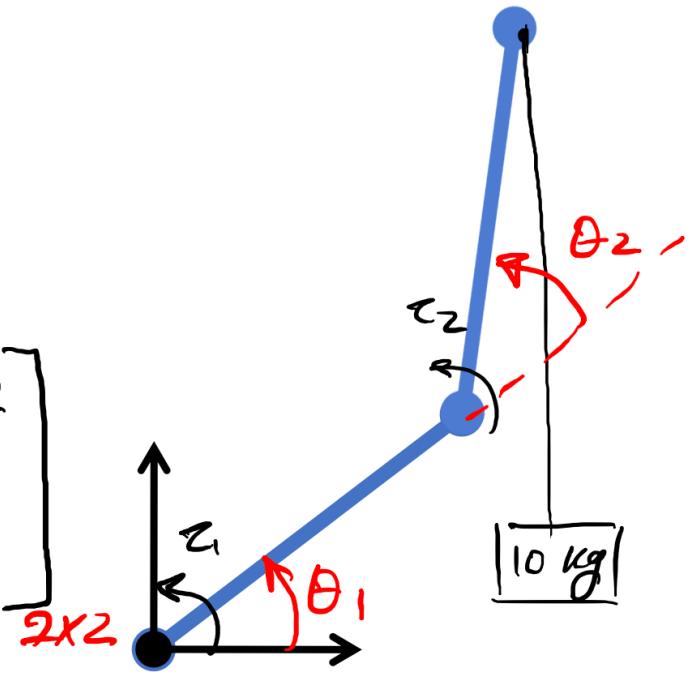
What vector of joint toques do I need to counteract the tip force if the robot is to hold a 10 kg weight on its tip?

$$\vec{\tau} = J^T \vec{F}_{tip}$$

$$\vec{F}_{tip} = \begin{bmatrix} 0 \\ -98 \end{bmatrix}_{2\times 1}$$

$$* \; J^T = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & l_1 c_1 + l_2 c_{12} \\ -l_2 s_{12} & l_2 c_{12} \end{bmatrix}_{2\times 2}$$

$$\vec{\tau}_{2\times 1} = J^T_{2\times 2} \; \vec{F}_{tip\;2\times 1}$$

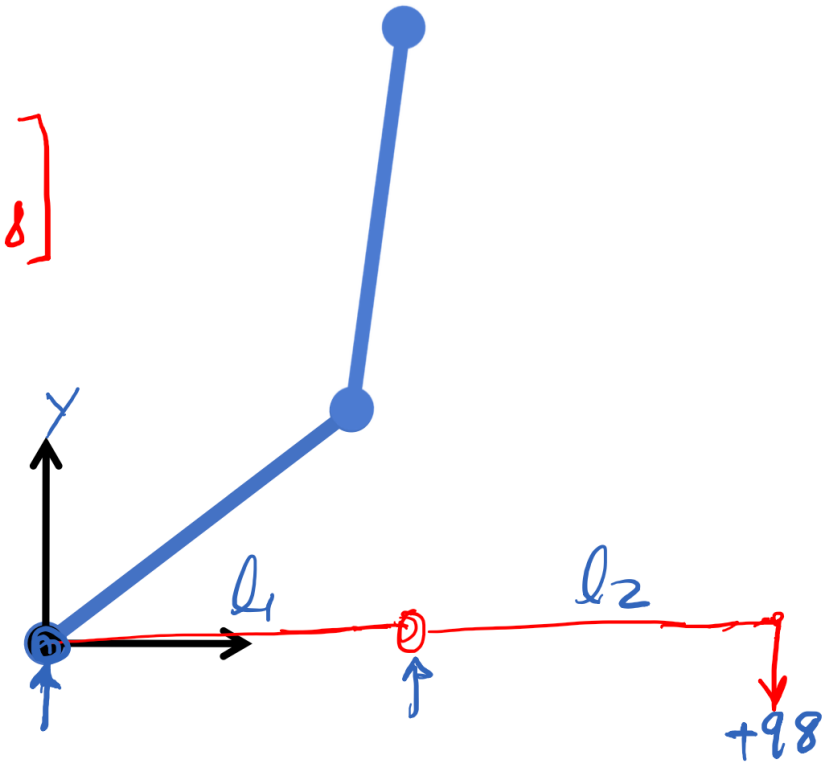


Static Forces - Example - Solution

$$\vec{\tau} = J^T(q) \begin{bmatrix} 0 \\ -98 \end{bmatrix}$$

$$\theta_1 = 0,\ \theta_2 = 0 \Rightarrow \vec{\tau} = \begin{bmatrix} 0 & l_1 + l_2 \\ 0 & l_2 \end{bmatrix} \begin{bmatrix} 0 \\ -98 \end{bmatrix}$$

$$\Rightarrow \vec{\tau} = \begin{bmatrix} -98(l_1 + l_2) \\ -98 l_2 \end{bmatrix} \begin{matrix} \leftarrow \tau_1 \\ \leftarrow \tau_2 \end{matrix}$$




BostonDynamics

Boston Dynamics Ethical Principles

Our mission is to imagine and create exceptional robots that enrich people's lives. By building robots that have the mobility, dexterity and perception required to contribute to our modern society, we strive to create a future in which humans and machines work together to improve everyone's safety, productivity and quality of life.

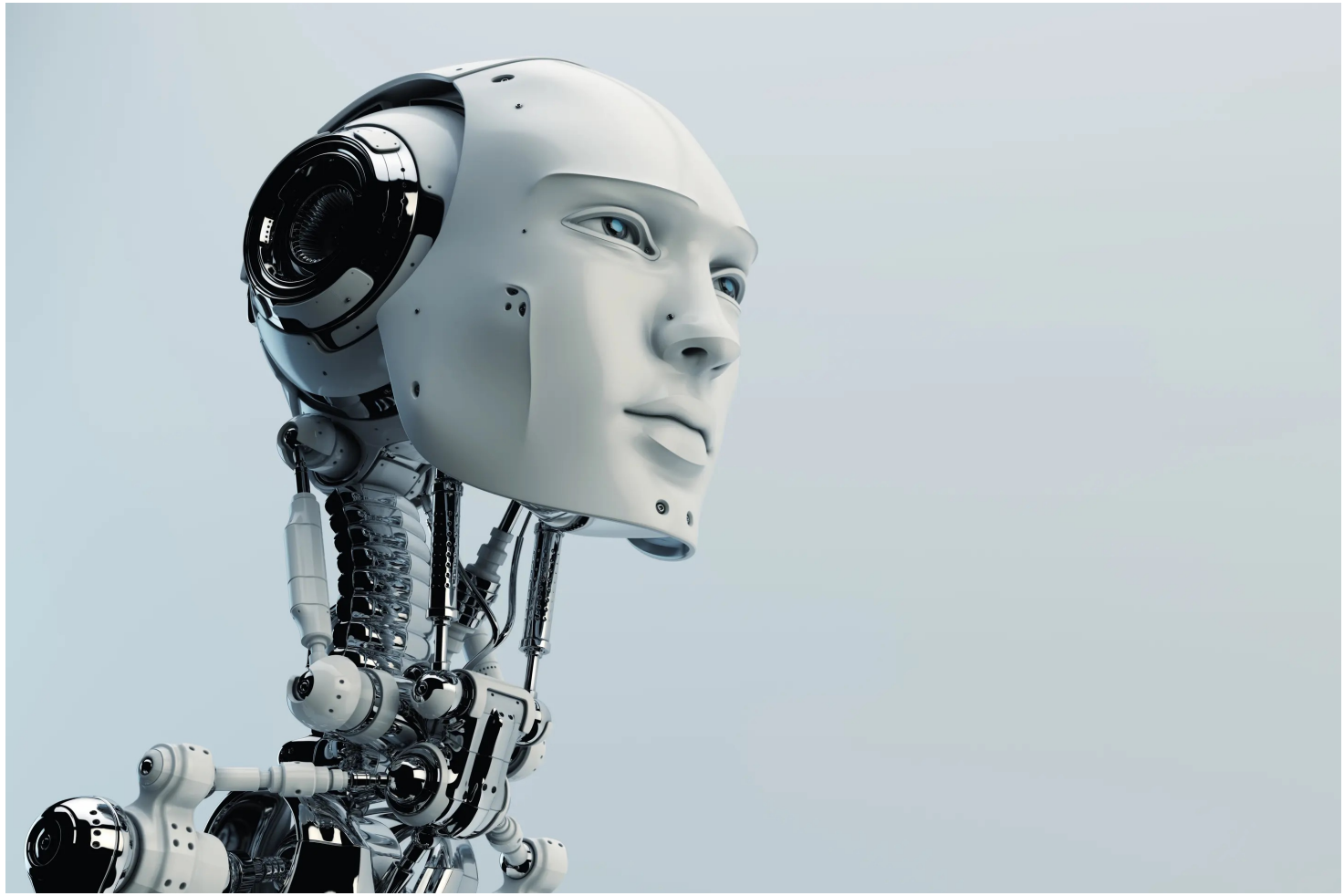
We are motivated by curiosity and respect for humans and animals

Curiosity and respect for the natural world are at the heart of our work on robots. Building a legged machine that can approximate the mobility of people and animals is a grand challenge. After decades of work, we have just begun to scratch the surface of the potential capabilities of robots. Nevertheless, the general mobility of our robots allows us to turn our focus to how this new type of automation can be used to improve our lives. We see this as the next step in the human history of building machines to reduce the danger, repetition and physically difficult aspects of work.

We prioritize the human element in human-robot partnerships

The next generation of robots will work amongst us and so the working partnership becomes important. We believe that robots will play a complementary role to human ingenuity and creativity in making work safer and more enjoyable. While artificial intelligence is rapidly advancing, we believe that only people have the intelligence to manage the full complexity and richness of real-world conditions in our diverse places of work. We build robots designed from the ground up to leverage human intelligence, keep people safe and relieve the most burdensome work.

https://bostondynamics.com/ethics/



Ethics in Robotics

- Explore the ethical considerations surrounding the development and use of robotics technology

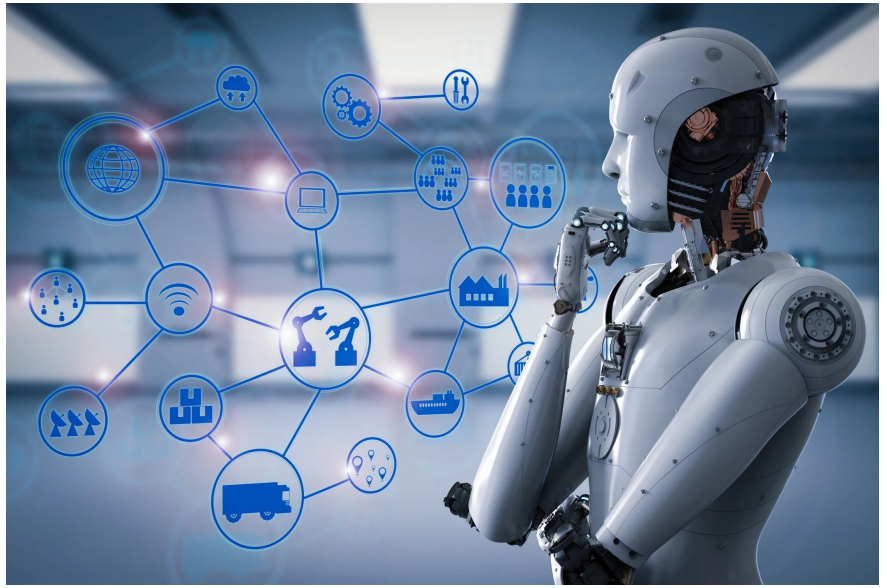
Understanding the Impact

Defining Ethics in Robotics

- Ethics in robotics refers to the moral principles that govern the design, development, deployment, and use of robots and autonomous systems
- It involves considering the societal, environmental, and individual implications of these technologies

Impact Areas:

- Safety
- Privacy
- Employment
- Autonomy
- Bias



Ethical Dilemmas in Robotics

Human-Robot Interaction

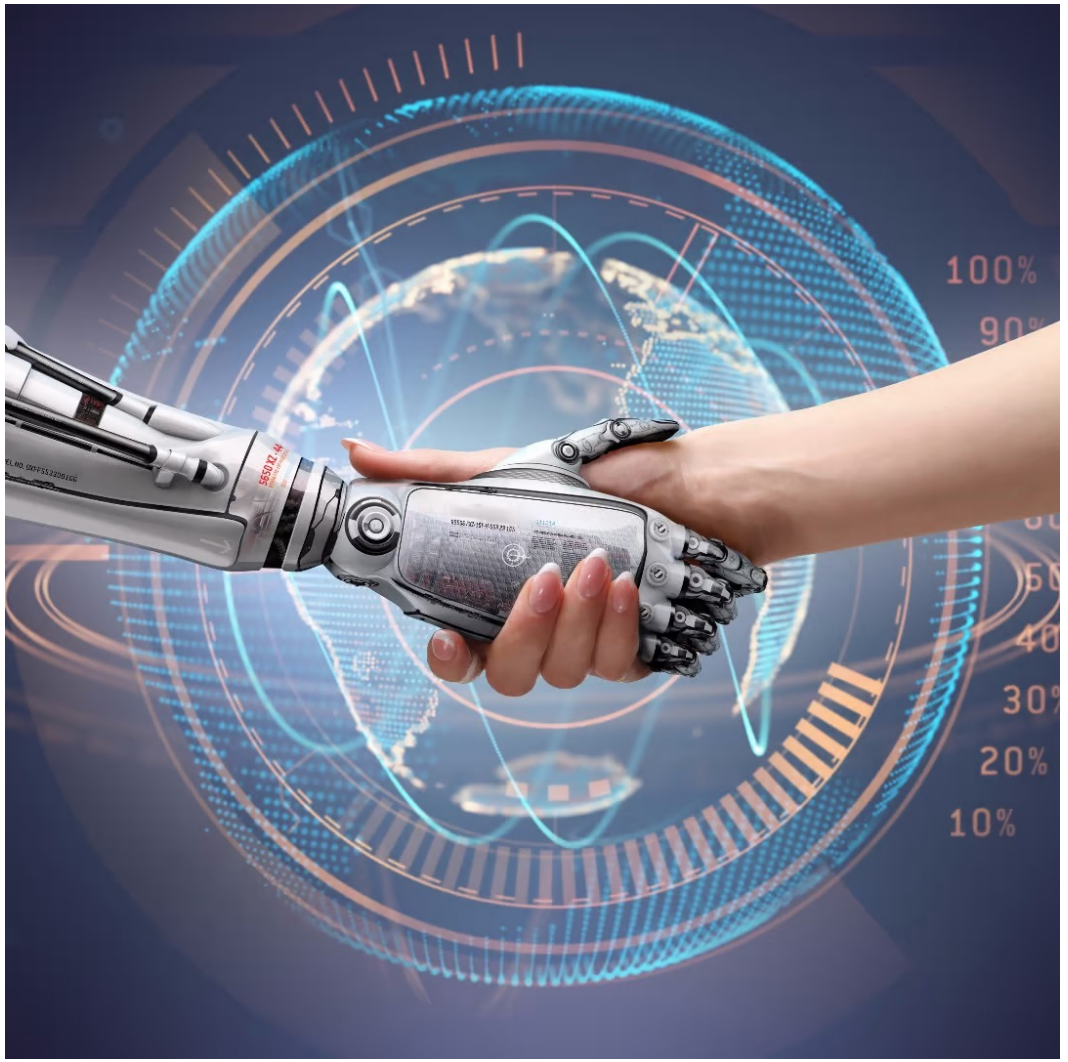
- As robots become more integrated into society, ethical dilemmas arise in how humans interact with them
- Questions of robot rights, responsibilities, and moral agency challenge traditional ethical frameworks

Autonomous Weapons

- The development of autonomous weapons raises concerns about the ethical implications of delegating life-and-death decisions to machines
- The need for international regulations to prevent the misuse of such technology is paramount

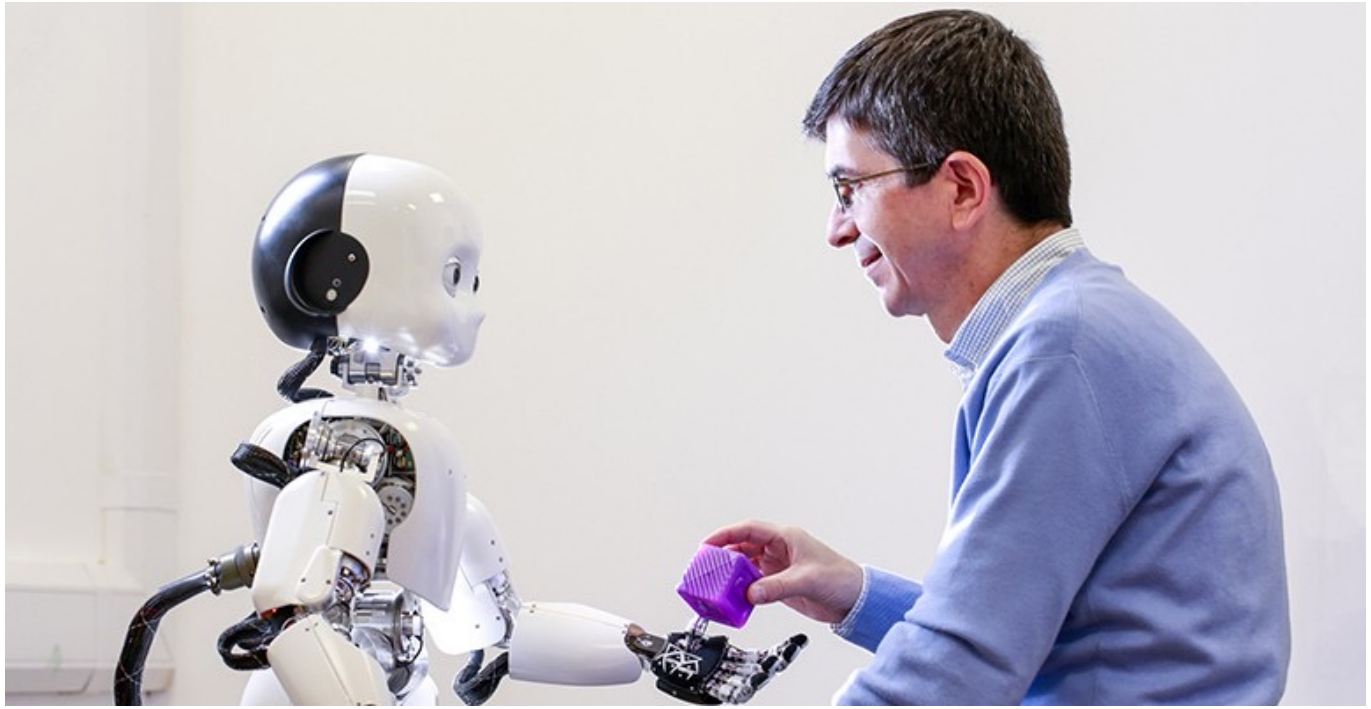
Principles for Ethical Robotics

- Transparency
- Accountability
- Privacy
- Fairness
- Beneficence



Moving Forward

- Collaborative Approach
- Education and Awareness
- Continues Evaluation



… end of Lecture 14