Robot Dynamics
(& Control)

METR 4202: Advanced Control & Robotics

Dr Surya Singh — Lecture # 4

metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/

Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Lecture (W: 12:05-1:50, 50-N201)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>29-Jul</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>5-Aug</td>
<td>Representing Position &amp; Orientation &amp; State (Frames, Transformation Matrices &amp; Affine Transformations)</td>
</tr>
<tr>
<td>3</td>
<td>12-Aug</td>
<td>Robot Kinematics Review (&amp; Ekka Day)</td>
</tr>
<tr>
<td>4</td>
<td>19-Aug</td>
<td><strong>Robot Dynamics</strong></td>
</tr>
<tr>
<td>5</td>
<td>26-Aug</td>
<td>Robot Sensing: Perception</td>
</tr>
<tr>
<td>6</td>
<td>2-Sep</td>
<td>Robot Sensing: Multiple View Geometry</td>
</tr>
<tr>
<td>7</td>
<td>9-Sep</td>
<td>Robot Sensing: Feature Detection (as Linear Observers)</td>
</tr>
<tr>
<td>8</td>
<td>16-Sep</td>
<td>Probabilistic Robotics: Localization</td>
</tr>
<tr>
<td>9</td>
<td>23-Sep</td>
<td>Quiz &amp; Guest Lecture (SLAM?)</td>
</tr>
<tr>
<td>10</td>
<td>30-Sep</td>
<td>Study break</td>
</tr>
<tr>
<td>11</td>
<td>7-Oct</td>
<td>Motion Planning</td>
</tr>
<tr>
<td>12</td>
<td>14-Oct</td>
<td>State-Space Modelling</td>
</tr>
<tr>
<td>13</td>
<td>21-Oct</td>
<td>Shaping the Dynamic Response</td>
</tr>
<tr>
<td>13</td>
<td>28-Oct</td>
<td>LQR + Course Review</td>
</tr>
</tbody>
</table>
Announcements!

- **Lab 1:**
  - Demos Tomorrow!

- **Lab 2:**
  - Hand Tracking

- **Reading for Next Week:**
  - Corke: §10.2 + Ch. 11 + § 12.1-12.2
You will get 6 random cards.

For each of the two trials you can pick 4 cards. Both rounds can be the same or different card combinations. The "C program" on each card gives a two-character "code": Ex: "D2", "C3", "H4", "S5" These strings are then concatenated together for the points generator. Ex: "D2C3H4S5".

Outline

Review:
- Denavit-Hartenberg Notation
- Parallel Robots

Jacobians & Differential Motion
- Multibody Dynamics Refresher

Newton-Euler Formulation
- Lagrange Formulation
DH: Where to place frame?

1. Align an axis along principal motion
   1. **Rotary (R):** align rotation axis along the $z$ axis
   2. **Prismatic (P):** align slider travel along $x$ axis

2. Orient so as to position $x$ axis towards next frame

3. $\theta_{(\text{rot } z)} \rightarrow d_{(\text{trans } z)} \rightarrow a_{(\text{trans } x)} \rightarrow \alpha_{(\text{rot } x)}$

---

### Denavit-Hartenberg $\rightarrow$ Rotation Matrix

- Each transformation is a product of 4 “basic” transformations (instead of 6)

$$i^{-1}A_i = \text{Rot}_{z, \theta_i} \text{Trans}_{z, d_i} \text{Trans}_{x, a_i} \text{Rot}_{x, \alpha_i}$$

$$= \begin{bmatrix}
    c\theta_i & -s\theta_i & 0 & 0 \\
    s\theta_i & c\theta_i & 0 & 0 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & d_i \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    1 & 0 & 0 & a_i \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}$$

$$= \begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & c\alpha_i & -s\alpha_i & 0 \\
    0 & s\alpha_i & c\alpha_i & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    c\theta_i & s\theta_i & c\alpha_i & a_i \\
    -s\theta_i & c\theta_i & s\alpha_i & a_i \\
    -c\theta_i & c\theta_i & c\alpha_i & a_i \\
    0 & s\alpha_i & c\alpha_i & d_i \\
    0 & 0 & 0 & 1
\end{bmatrix}$$
DH Example [1]: RRR Link Manipulator

1. Assign the frames at the joints …
2. Fill DH Table …

<table>
<thead>
<tr>
<th>Link</th>
<th>a_i</th>
<th>α_i</th>
<th>d_i</th>
<th>θ_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>l_1</td>
<td>0</td>
<td>0</td>
<td>θ_1</td>
</tr>
<tr>
<td>2</td>
<td>l_2</td>
<td>0</td>
<td>0</td>
<td>θ_2</td>
</tr>
<tr>
<td>3</td>
<td>l_3</td>
<td>0</td>
<td>0</td>
<td>θ_3</td>
</tr>
</tbody>
</table>

\[ a_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & L_{c_{\theta_i}} \\ s_{\theta_i} & c_{\theta_i} & 0 & L_{s_{\theta_i}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \]

\[ a_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & L_{c_{\theta_i}} \\ s_{\theta_i} & c_{\theta_i} & 0 & L_{s_{\theta_i}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \]

\[ a_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & L_{c_{\theta_i}} \\ s_{\theta_i} & c_{\theta_i} & 0 & L_{s_{\theta_i}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \]

\[ \alpha = a_i a_2 \alpha_2 a_3 \alpha_3 \]

\[ T_i = a_i a_2 \alpha_2 a_3 \alpha_3 \]

DH Example [2]: RRP Link Manipulator

1. Assign the frames at the joints …
2. Fill DH Table …

<table>
<thead>
<tr>
<th>Link</th>
<th>a_i</th>
<th>α_i</th>
<th>d_i</th>
<th>θ_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>l_1</td>
<td>0</td>
<td>0</td>
<td>θ_1</td>
</tr>
<tr>
<td>2</td>
<td>l_2</td>
<td>0</td>
<td>0</td>
<td>θ_2</td>
</tr>
<tr>
<td>3</td>
<td>l_3</td>
<td>0</td>
<td>0</td>
<td>θ_3</td>
</tr>
</tbody>
</table>

\[ a_i = \begin{bmatrix} c_{\theta_i} & 0 & -s_{\theta_i} & 0 \\ s_{\theta_i} & 0 & c_{\theta_i} & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \]

\[ a_i = \begin{bmatrix} c_{\theta_i} & 0 & -s_{\theta_i} & 0 \\ s_{\theta_i} & 0 & c_{\theta_i} & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \]

\[ a_i = \begin{bmatrix} c_{\theta_i} & 0 & -s_{\theta_i} & 0 \\ s_{\theta_i} & 0 & c_{\theta_i} & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \]

\[ \alpha = a_i a_2 \alpha_2 a_3 \alpha_3 \]

\[ T_i = a_i a_2 \alpha_2 a_3 \alpha_3 \]

\[ T_i = a_i a_2 \alpha_2 a_3 \alpha_3 \]
Modified DH

- Made “popular” by Craig’s *Intro. to Robotics* book
- Link coordinates attached to the near by joint

\[ \begin{align*}
&\text{a (trans } x-1) \Rightarrow \alpha \ (\text{rot } x-1) \Rightarrow \theta \ (\text{rot } z) \Rightarrow d \ (\text{trans } z) \\
\end{align*} \]

Modified DH \[\star\]

- Gives a similar result
  (but it’s not commutative)

\[ A_i^{i-1} = R_x (\alpha_{i-1}) T_x (a_{i-1}) R_z (\theta_i) T_x (d_i) \]

- Refactoring Standard \(\Rightarrow\) to Modified

\[
\begin{array}{c}
\{ R_z (\theta_1) T_z (d_1) T_x (a_1) R_x (\alpha_1) \} \cdot \{ R_z (\theta_2) T_z (d_2) T_x (a_2) R_x (\alpha_2) \} \cdot \{ R_z (\theta_3) T_z (d_3) \}
\end{array}
\]

\[
\begin{array}{c}
\text{DH}_1 \quad \text{DH}_2 \quad \text{End Effector}
\end{array}
\]

\[
\begin{array}{c}
\{ R_z (\theta_1) T_z (d_1) \} \cdot \{ T_x (a_1) R_x (\alpha_1) R_z (\theta_2) T_z (d_2) \} \cdot \{ T_x (a_2) R_x (\alpha_2) R_z (\theta_3) T_z (d_3) \}
\end{array}
\]

\[
\begin{array}{c}
\text{Base} \quad \text{MDH}_1 \quad \text{MDH}_2
\end{array}
\]
Parallel Manipulators

- The “central” Kinematic structure is made up of closed-loop chain(s)

Compared to Serial Mechanisms:
  + Higher Stiffness
  + Higher Payload
  + Less Inertia
  - Smaller Workspace
  - Coordinated Drive System
  - More Complex & $$$


Symmetrical Parallel Manipulator

A sub-class of Parallel Manipulator:
  o # Limbs ($m$) = # DOF ($F$)
  o The joints are arranged in an identical pattern
  o The # and location of actuated joints are the same

Thus:
  o Number of Loops ($L$): One less than # of limbs
    $$ L = m - 1 = F - 1 $$
  o Connectivity ($C_k$)
    $$ \sum_{k=1}^{m} C_k = (\lambda + 1) F - \lambda $$
    Where: $\lambda$: The DOF of the space that the system is in (e.g., $\lambda=6$ for 3D space).
Robot Dynamics

Robot Dynamics

Regrasping
Angular Velocity

• If we look at a small timeslice as a frame rotates with a moving point, we find

\[
|\Delta P| = \left( |\Delta P| \sin \theta \right) (|A\Omega_B| \Delta t)
\]

\[
\frac{|\Delta P|}{\Delta t} = \left( |A\Omega_B| \sin \theta \right) (|A\Omega_B|)
\]

\[
= A\Omega_B \times A\mathbf{P}
\]

\[
A\mathbf{V}_P = A\Omega_B \times A\mathbf{R}_B^B \mathbf{P}
\]

Velocity

• Recall that we can specify a point in one frame relative to another as

\[
A\mathbf{P} = A\mathbf{P}_B + A\mathbf{R}_B^B \mathbf{P}
\]

• Differentiating w/r/t to \(t\) we find

\[
A\mathbf{V}_P = \frac{d}{dt} A\mathbf{P} = \lim_{\Delta t \to 0} \frac{A\mathbf{P}(t + \Delta t) - A\mathbf{P}(t)}{\Delta t}
\]

\[
= A\dot{\mathbf{P}}_B + A\mathbf{R}_B^B \dot{\mathbf{P}} + A\mathbf{R}_B^B \mathbf{P}
\]

• This can be rewritten as

\[
A\mathbf{V}_P = A\mathbf{V}_{BORG} + A\mathbf{R}_B^B \mathbf{V}_P + A\Omega_B \times A\mathbf{R}_B^B \mathbf{P}
\]
Skew – Symmetric Matrix

\[ V = \omega \times r \]

\[ \Omega = \begin{bmatrix}
0 & -\omega_z & \omega_y \\
\omega_z & 0 & -\omega_x \\
-\omega_y & \omega_x & 0
\end{bmatrix} \]

\[ \rightarrow V = \Omega r \]

Velocity Representations

- Euler Angles
  - For Z-Y-X \((\alpha, \beta, \gamma)\):

\[
\begin{pmatrix}
\dot{\alpha} \\
\dot{\beta} \\
\dot{\gamma}
\end{pmatrix} =
\begin{pmatrix}
-S\beta & 0 & 1 \\
C\beta S\gamma & C\gamma & 0 \\
C\beta C\gamma & -S\beta & 0
\end{pmatrix}
\begin{pmatrix}
\omega_x \\
\omega_y \\
\omega_z
\end{pmatrix}
\]

- Quaternions

\[
\begin{pmatrix}
\dot{\varepsilon}_0 \\
\dot{\varepsilon}_1 \\
\dot{\varepsilon}_2 \\
\dot{\varepsilon}_3
\end{pmatrix} =
\frac{1}{2}
\begin{pmatrix}
\varepsilon_1 & -\varepsilon_2 & -\varepsilon_3 \\
\varepsilon_0 & \varepsilon_3 & -\varepsilon_2 \\
-\varepsilon_3 & \varepsilon_0 & \varepsilon_1 \\
\varepsilon_2 & -\varepsilon_1 & \varepsilon_0
\end{pmatrix}
\begin{pmatrix}
\omega_x \\
\omega_y \\
\omega_z
\end{pmatrix}
\]
Manipulator Velocities

- Consider again the schematic of the planar manipulator shown. We found that the end effector position is given by

\[
\begin{align*}
x &= L_1 \cos \theta_1 + L_2 \cos (\theta_1 + \theta_2) + L_3 \cos (\theta_1 + \theta_2 + \theta_3) \\
y &= L_1 \sin \theta_1 + L_2 \sin (\theta_1 + \theta_2) + L_3 \sin (\theta_1 + \theta_2 + \theta_3)
\end{align*}
\]

- Differentiating w/r/t to \( t \)

\[
\begin{align*}
\dot{x} &= -L_1 s_1 \dot{\theta}_1 - L_2 s_{12} (\dot{\theta}_1 + \dot{\theta}_2) - L_3 s_{123} (\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3) \\
\dot{y} &= L_1 c_1 \dot{\theta}_1 + L_2 c_{12} (\dot{\theta}_1 + \dot{\theta}_2) + L_3 c_{123} (\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3)
\end{align*}
\]

- This gives the end effector velocity as a function of pose and joint velocities.

Manipulator Velocities [2]

- Rearranging, we can recast this relation in matrix form

\[
\begin{bmatrix}
\dot{x} \\
\dot{y}
\end{bmatrix} =
\begin{bmatrix}
-L_1 s_1 & -L_2 s_{12} & -L_3 s_{123} \\
L_1 c_1 + L_2 c_{12} + L_3 c_{123} & L_2 c_{12} + L_3 c_{123} & L_3 c_{123}
\end{bmatrix}
\begin{bmatrix}
\dot{\theta}_1 \\
\dot{\theta}_2 \\
\dot{\theta}_3
\end{bmatrix}
\]

- Or

\[
\begin{bmatrix}
\dot{x} \\
\dot{y}
\end{bmatrix} =
\frac{\partial x}{\partial \theta_1} \dot{\theta}_1 + \frac{\partial x}{\partial \theta_2} \dot{\theta}_2 + \frac{\partial x}{\partial \theta_3} \dot{\theta}_3
\]

- The resulting matrix is called the Jacobian and provides us with a mapping from Joint Space to Cartesian Space.
Moving On…Differential Motion

- Transformations also encode differential relationships
- Consider a manipulator (say 2DOF, RR)

\[ x(\theta_1, \theta_2) = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \]
\[ y(\theta_1, \theta_2) = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \]

- Differentiating with respect to the angles gives:

\[
dx = \frac{\partial x(\theta_1, \theta_2)}{\partial \theta_1}d\theta_1 + \frac{\partial x(\theta_1, \theta_2)}{\partial \theta_2}d\theta_2
\]
\[
dy = \frac{\partial y(\theta_1, \theta_2)}{\partial \theta_1}d\theta_1 + \frac{\partial y(\theta_1, \theta_2)}{\partial \theta_2}d\theta_2
\]

Differential Motion [2]

- Viewing this as a matrix \( \rightarrow \) Jacobian

\[
dx = Jd\theta
\]

\[
J = \begin{bmatrix}
-l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\
l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2)
\end{bmatrix}
\]

\[
\dot{v} = J_1 \ddot{\theta}_1 + J_2 \ddot{\theta}_2
\]
Infinitesimal Rotations

- \( \cos (d\phi) = 1, \sin (d\phi) = d\phi \)

\[
R_x (d\phi) = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos d\phi & -\sin d\phi \\
0 & \sin d\phi & \cos d\phi
\end{bmatrix} \approx \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & -d\phi \\
0 & d\phi & 1
\end{bmatrix}
\]

\[
R_y (d\phi) = \begin{bmatrix}
\cos d\phi & 0 & \sin d\phi \\
0 & 1 & 0 \\
-\sin d\phi & 0 & \cos d\phi
\end{bmatrix} \approx \begin{bmatrix}
1 & 0 & d\phi \\
0 & 1 & 0 \\
-d\phi & 0 & 1
\end{bmatrix}
\]

\[
R_z (d\phi) = \begin{bmatrix}
\cos d\phi & -\sin d\phi & 0 \\
\sin d\phi & \cos d\phi & 0 \\
0 & 0 & 1
\end{bmatrix} \approx \begin{bmatrix}
1 & -d\phi & 0 \\
1 & 0 & d\phi \\
0 & 0 & 1
\end{bmatrix}
\]

- Note that:

\[ R_x (d\phi) R_y (d\phi) = R_y (d\phi) R_x (d\phi) \]

\( \rightarrow \) Therefore … they **commute**

---

The Jacobian

- In general, the Jacobian takes the form
  (for example, **joints** and in **operational space**)

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\vdots \\
\dot{x}_i
\end{bmatrix} = 
\begin{bmatrix}
\frac{\partial x_1}{\partial \theta_1} & \frac{\partial x_1}{\partial \theta_2} & \cdots & \frac{\partial x_1}{\partial \theta_j} \\
\frac{\partial x_2}{\partial \theta_1} & \frac{\partial x_2}{\partial \theta_2} & \cdots & \frac{\partial x_2}{\partial \theta_j} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial x_i}{\partial \theta_1} & \frac{\partial x_i}{\partial \theta_2} & \cdots & \frac{\partial x_i}{\partial \theta_j}
\end{bmatrix} 
\begin{bmatrix}
\dot{\theta}_1 \\
\dot{\theta}_2 \\
\vdots \\
\dot{\theta}_j
\end{bmatrix}
\]

- Or more succinctly

\[
\dot{X} = J(\theta)\dot{\theta}
\]
Jacobian [2]

- Jacobian can be viewed as a mapping from Joint velocity space ($\dot{\mathbf{q}}$) to Operational velocity space ($\mathbf{v}$)

Revisiting The Jacobian

- I told you:

$$
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\vdots \\
\dot{x}_i \\
\end{bmatrix} =
\begin{bmatrix}
\frac{\partial x_1}{\partial \theta_1} & \frac{\partial x_1}{\partial \theta_2} & \cdots & \frac{\partial x_1}{\partial \theta_j} \\
\frac{\partial x_2}{\partial \theta_1} & \frac{\partial x_2}{\partial \theta_2} & \cdots & \frac{\partial x_2}{\partial \theta_j} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial x_i}{\partial \theta_1} & \frac{\partial x_i}{\partial \theta_2} & \cdots & \frac{\partial x_i}{\partial \theta_j} \\
\end{bmatrix}
\begin{bmatrix}
\dot{\theta}_1 \\
\dot{\theta}_2 \\
\vdots \\
\dot{\theta}_j \\
\end{bmatrix}
$$

- True, but we can be more “explicit”
Jacobian: **Explicit Form**

- For a serial chain (robot): The velocity of a link with respect to the proceeding link is dependent on the type of link that connects them.
  - If the joint is **prismatic** ($\epsilon=1$), then $v_i = \frac{dz_i}{dt}$
  - If the joint is **revolute** ($\epsilon=0$), then $o = \frac{d\theta_i}{dt}$ (in the $k$ direction)

  $\therefore v = \sum_{i=1}^{N} \left( \epsilon_i v_i + \bar{e}_i (\omega_i \times p_{i-1,i}) \right)$

  $\omega = \sum_{i=1}^{N} \left( \bar{e}_i (\dot{\theta}_i) \right) = \sum_{i=1}^{N} \left( \bar{e}_i z_i (\dot{\theta}_i) \right)$

  $\rightarrow v = J_v \dot{q}$

  $o = J_o \dot{q}$

- Combining them (with $v=(\Delta x, \Delta \theta)$)

  $J = \begin{bmatrix} J_v \\ J_o \end{bmatrix}$

---

Jacobian: **Explicit Form** [2]

- The overall Jacobian takes the form

  $J = \begin{bmatrix} \frac{\partial x_p}{\partial q_1} & \ldots & \frac{\partial x_p}{\partial q_n} \\ \frac{\partial \bar{e}_1 z_i}{\partial q_1} & \ldots & \frac{\partial \bar{e}_1 z_i}{\partial q_n} \end{bmatrix}$

- The Jacobian for a particular frame (F) can be expressed:

  $^F J = \begin{bmatrix} ^F J_v \\ ^F J_o \end{bmatrix} = \begin{bmatrix} \frac{\partial ^F x_p}{\partial q_1} & \ldots & \frac{\partial ^F x_p}{\partial q_n} \\ \frac{\partial \bar{e}_1 z_i}{\partial q_1} & \ldots & \frac{\partial \bar{e}_1 z_i}{\partial q_n} \end{bmatrix}$

Where: $^F z_i = ^i R^i z_i$ & $^i z_i = (0 \ 0 \ 1)$
Motivating Example:
Remotely Driven 2DOF Manipulator

• Introduce $Q_1 = \tau_1 + \tau_2$ and $Q_2 = \tau_2$
• Derivation posted to website

Dynamics

• We can also consider the forces that are required to achieve a particular motion of a manipulator or other body

• Understanding the way in which motion arises from torques applied by the actuators or from external forces allows us to control these motions

• There are a number of methods for formulating these equations, including
  – Newton-Euler Dynamics
  – Lagrangian Mechanics
Dynamics of Serial Manipulators

- Systems that keep on manipulating (the system)

- Direct Dynamics:
  - Find the response of a robot arm with torques/forces applied

- Inverse Dynamics:
  - Find the (actuator) torques/forces required to generate a desired trajectory of the manipulator

Dynamics – Newton-Euler

- In general, we could analyse the dynamics of robotic systems using classical Newtonian mechanics
  \[
  \sum F = m \ddot{x} \\
  \sum T = J \ddot{\theta}
  \]

- This can entail iteratively calculating velocities and accelerations for each link and then computing force and moment balances in the system

- Alternatively, closed form solutions may exist for simple configurations
Dynamics

• For Manipulators, the general form is

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$$

where

• $\tau$ is a vector of joint torques
• $\Theta$ is the nx1 vector of joint angles
• $M(\Theta)$ is the nxn mass matrix
• $V(\Theta, \dot{\Theta})$ is the nx1 vector of centrifugal and Coriolis terms
• $G(\Theta)$ is an nx1 vector of gravity terms

• Notice that all of these terms depend on $\Theta$ so the dynamics varies as the manipulator moves.

Dynamics: Inertia

• The moment of inertia (second moment) of a rigid body B relative to a line L that passes through a reference point O and is parallel to a unit vector $u$ is given by:

$$I_u^O = \int_V p \times (u \times p) \rho dV$$

$$= \int_V \left[ p^2 u - (p^T u) p \right] \rho dV$$

• The scalar product of $I_u^O$ with a second axis ($w$) is called the product of inertia:

$$I_{uw}^O = I_u^O \cdot w = \int_V \left[ (u^T w) p^2 - (p^T u) (p^T w) \right] \rho dV$$

• If $u = w$, then we get the moment of inertia:

$$I_{uu} = \int_V \left[ p^2 - (p^T u)^2 \right] \rho dV = mr_g^2$$

Where: $r_g$: radius of gyration of B w/r/t to L

$$r_g = p^2 - (p^T u)^2 = (u \times p)^2$$
Dynamics: Mass Matrix & Inertia Matrix

- This can be written in a Matrix form as:

\[ I^O_B = I^O_B u \]

- Where \( I^O_B \) is the inertial matrix or inertial tensor of the body B about a reference point O

\[
I^O_B = \begin{bmatrix}
I_{xx} & I_{xy} & I_{xz} \\
I_{yx} & I_{yy} & I_{yz} \\
I_{zx} & I_{zy} & I_{zz}
\end{bmatrix}
\]

- Where to get \( I_{xx}, \) etc? \( \Rightarrow \) Parallel Axis Theorem

If CM is the center of mass, then:

\[
\begin{align*}
I^O_{xx} &= I^{CM}_{xx} + m\left(y_c^2 + z_c^2\right) \\
I^O_{yy} &= I^{CM}_{yy} + m\left(x_c^2 + z_c^2\right) \\
I^O_{zz} &= I^{CM}_{zz} + m\left(x_c^2 + y_c^2\right) \\
I^O_{xy} &= I^{CM}_{xy} + m x_c y_c \\
I^O_{yz} &= I^{CM}_{yz} + m y_c z_c \\
I^O_{zx} &= I^{CM}_{zx} + m z_c x_c
\end{align*}
\]

Dynamics: Mass Matrix

- The Mass Matrix: Determining via the Jacobian!

\[
K = \sum_{i=1}^{N} K_i
\]

\[
K_i = \frac{1}{2} \left( m_i v_{C_i}^T v_{C_i} + \boldsymbol{\omega}_i^T I_{C_i} \boldsymbol{\omega}_i \right)
\]

\[
v_{C_i} = J_{v_i} \dot{\boldsymbol{\theta}} \\
\boldsymbol{\omega}_i = J_{\omega_i} \dot{\boldsymbol{\theta}}
\]

\[
J_{v_i} = \begin{bmatrix}
\frac{\partial p_{C_1}}{\partial \theta_1} & \cdots & \frac{\partial p_{C_i}}{\partial \theta_i} & 0 & \cdots & 0 \\
& \ddots & & \ddots & \ddots & \ddots \\
& & \frac{\partial p_{C_1}}{\partial \theta_i} & \cdots & \frac{\partial p_{C_i}}{\partial \theta_i} & 0 & \cdots & 0
\end{bmatrix}
\]

\[
J_{\omega_i} = \begin{bmatrix}
\bar{e}_1 Z_1 & \cdots & \bar{e}_i Z_i & 0 & \cdots & 0 \\
& \ddots & & \ddots & \ddots & \ddots \\
& & \bar{e}_1 Z_1 & \cdots & \bar{e}_i Z_i & 0 & \cdots & 0
\end{bmatrix}
\]

\[
M = \sum_{i=1}^{N} \left( m_i J_{v_i}^T J_{v_i} + J_{\omega_i}^T I_{C_i} J_{\omega_i} \right)
\]

\( M \) is symmetric, positive definite \( \therefore m_{ij} = m_{ji}, \dot{\boldsymbol{\theta}}^T M \dot{\boldsymbol{\theta}} > 0 \)
Dynamics – Langrangian Mechanics

• Alternatively, we can use Langrangian Mechanics to compute the dynamics of a manipulator (or other robotic system)

• The Langrangian is defined as the difference between the Kinetic and Potential energy in the system

\[ L = K - P \]

• Using this formulation and the concept of virtual work we can find the forces and torques acting on the system.

\[ \mathbf{F} = \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\mathbf{x}}} \right) - \frac{\partial L}{\partial \mathbf{x}} \]

\[ \mathbf{\tau} = \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} \]

• This may seem more involved but is often easier to formulate for complex systems

Dynamics – Langrangian Mechanics [2]

\[ L = K - P, \dot{\theta} : \text{Generalized Velocities}, M : \text{Mass Matrix} \]

\[ \mathbf{\tau} = \sum_{i=1}^{N} \mathbf{r}_i - \frac{d}{dt} \left( \frac{\partial K}{\partial \dot{\theta}} \right) - \frac{\partial K}{\partial \theta} + \frac{\partial P}{\partial \theta} \]

\[ K = \frac{1}{2} \dot{\theta}^T M (\theta) \dot{\theta} \]

\[ \frac{d}{dt} \left( \frac{\partial K}{\partial \dot{\theta}} \right) = \frac{d}{dt} \left( \frac{\partial}{\partial \dot{\theta}} \left( \frac{1}{2} \dot{\theta}^T M (\theta) \dot{\theta} \right) \right) = \frac{d}{dt} \left( M \ddot{\theta} \right) = M \ddot{\theta} + \ddot{M} \dot{\theta} \]

\[ \frac{d}{dt} \left( \frac{\partial K}{\partial \theta} \right) - \frac{\partial K}{\partial \theta} = \left[ \ddot{\theta}^T \dot{M} \dot{\theta} \right] - \frac{1}{2} \dot{\theta}^T \dot{M} \dot{\theta} \]

\[ \mathbf{v} (\theta, \dot{\theta}) = \mathbf{C} (\theta) \ddot{\theta}^2 + \mathbf{B} (\theta) \dot{\theta} \]

\[ \mathbf{\tau} = M (\theta) \ddot{\theta} + \mathbf{v} (\theta, \dot{\theta}) + \mathbf{g} (\theta) \]
Dynamics – Langrangian Mechanics [3]

- The Mass Matrix: Determining via the Jacobian!

\[ K = \sum_{i=1}^{N} K_i \]

\[ K_i = \frac{1}{2} \left( m_i v_{C_i}^T v_{C_i} + \omega_i^T I_{C_i} \omega_i \right) \]

\[ v_{C_i} = J_{v_i} \dot{\theta} \quad J_{v_i} = \begin{bmatrix} \frac{\partial p_{C_1}}{\partial \theta_1} & \cdots & \left. \frac{\partial p_{C_i}}{\partial \theta_i} \right|_{i+1} & \cdots & \left. \frac{\partial p_{C_n}}{\partial \theta_i} \right|_n \end{bmatrix} \]

\[ \omega_i = J_{\omega_i} \dot{\theta} \quad J_{\omega_i} = \begin{bmatrix} \bar{\varepsilon}_1 Z_1 & \cdots & \bar{\varepsilon}_i Z_i \end{bmatrix} \]

\[ \therefore M = \sum_{i=1}^{N} \left( m_i J_{v_i}^T J_{v_i} + J_{\omega_i}^T I_{C_i} J_{\omega_i} \right) \]

\( M \) is symmetric, positive definite. \( \therefore m_{ij} = m_{ji}, \quad \dot{\theta}^T M \dot{\theta} > 0 \)

Generalized Coordinates

- A significant feature of the Lagrangian Formulation is that any convenient coordinates can be used to derive the system.

- Go from Joint → Generalized
  - Define \( \mathbf{p} \): \( d\mathbf{p} = \mathbf{J} d\mathbf{q} \)
    \[ \mathbf{q} = \begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix} \rightarrow \mathbf{p} = \begin{bmatrix} p_1 & \cdots & p_n \end{bmatrix} \]

  \( \dot{\mathbf{p}} = \mathbf{H} \dot{\mathbf{q}} \) \( \dot{\mathbf{q}} = \left( \mathbf{J}^{-1} \right)^T G \)

  Thus: the kinetic energy and gravity terms become

  \[ KE = \frac{1}{2} \mathbf{p}^T \mathbf{H} \dot{\mathbf{p}} \quad G^* = \left( \mathbf{J}^{-1} \right)^T \mathbf{G} \]

  where: \( \mathbf{H}^* = \left( \mathbf{J}^{-1} \right)^T \mathbf{H} \mathbf{J}^{-1} \)
Inverse Dynamics

- Forward dynamics governs the dynamic responses of a manipulator arm to the input torques generated by the actuators.

- The inverse problem:
  - Going from joint angles to torques
  - Inputs are desired trajectories described as functions of time
    \[ q = [q_1, \ldots, q_n] \rightarrow [\theta_1(t), \theta_2(t), \theta_n(t)] \]
  - Outputs are joint torques to be applied at each instance
    \[ \tau = [\tau_1, \ldots, \tau_n] \]

- Computation “big” (6DOF arm: 66,271 multiplications), but not scary (4.5 ms on PDP11/45)

Also: Inverse Jacobian

- In many instances, we are also interested in computing the set of joint velocities that will yield a particular velocity at the end effector
  \[ \dot{\theta} = J(\theta)^{-1} \dot{X} \]

- We must be aware, however, that the inverse of the Jacobian may be undefined or singular. The points in the workspace at which the Jacobian is undefined are the **singularities** of the mechanism.

- Singularities typically occur at the workspace boundaries or at interior points where degrees of freedom are lost
Inverse Jacobian Example

- For a simple two link RR manipulator:
  \[ x = L_1 \cos \theta_1 + L_2 \cos (\theta_1 + \theta_2) \]
  \[ y = L_1 \sin \theta_1 + L_2 \sin (\theta_1 + \theta_2) \]

- The Jacobian for this is
  \[
  \begin{bmatrix}
  \dot{x} \\
  \dot{y}
  \end{bmatrix} =
  \begin{bmatrix}
  -L_1 s_1 -L_2 s_{12} \\
  L_1 c_1 + L_2 c_{12}
  \end{bmatrix}
  \begin{bmatrix}
  \dot{\theta}_1 \\
  \dot{\theta}_2
  \end{bmatrix}
  \]

- Taking the inverse of the Jacobian yields
  \[
  \begin{bmatrix}
  \dot{\theta}_1 \\
  \dot{\theta}_2
  \end{bmatrix} = \frac{1}{L_1 L_2 s_2}
  \begin{bmatrix}
  L_2 c_{12} & L_2 s_{12} \\
  -L_1 c_1 -L_2 c_{12} & -L_1 s_1 -L_2 s_{12}
  \end{bmatrix}
  \begin{bmatrix}
  \dot{x} \\
  \dot{y}
  \end{bmatrix}
  \]

- Clearly, as \( \theta_2 \) approaches 0 or \( \pi \) this manipulator becomes singular.

Static Forces

- We can also use the Jacobian to compute the joint torques required to maintain a particular force at the end effector.
- Consider the concept of virtual work
  \[ F \cdot \delta X = \tau \cdot \delta \theta \]

- Or
  \[ F^T \delta X = \tau^T \delta \theta \]

- Earlier we saw that
  \[ \delta X = J \delta \theta \]

- So that
  \[ F^T J = \tau^T \]

- Or
  \[ \tau = J^T F \]
Nonlinear Plant

\[ \mathbf{\dot{q}} = \tau' \]

compensated dynamics

\[ M(q)\dot{q} + v(q, \dot{q}) + g(q) = \tau + \tau_{\text{friction}} + \tau_{\text{terrain}} \]

feedforward command
(open-loop policy)

Model Based

Model "Free"

Compensated Manipulation

[Image of a person operating a robotic arm]
Trajectory Generation

- The goal is to get from an initial position \( \{i\} \) to a final position \( \{f\} \) via a path points \( \{p\} \)
Joint Space

Consider only the **joint positions** as a function of time

- + Since we control the joints, this is more direct
- -- If we want to follow a particular trajectory, not easy
  - at best lots of intermediate points
  - No guarantee that you can solve the Inverse Kinematics for all path points

Cartesian Workspace

Consider the **Cartesian positions** as a function of time

- + Can track shapes exactly
- -- We need to solve the inverse kinematics and dynamics
**Polynomial Trajectories**

- Straight line Trajectories

- Polynomial Trajectories

- Simpler

  \[ u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \]

- Parabolic blends are smoother

- Use “pseudo via points”

---

**Summary**

- Kinematics is the study of motion without regard to the forces that create it

- Kinematics is important in many instances in Robotics

- The study of dynamics allows us to understand the forces and torques which act on a system and result in motion

- Understanding these motions, and the required forces, is essential for designing these systems
Cool Robotics Share

Dynamic Simulation Software

• v-rep
  virtual robot experimentation platform

  http://www.coppeliarobotics.com/

• Reflexxes

  http://www.reflexxes.com/
More Cool Robotics Share