Robot Dynamics & Control

METR 4202: Advanced Control & **Robotics**

Dr Surya Singh -- Lecture # 4

August 20, 2014

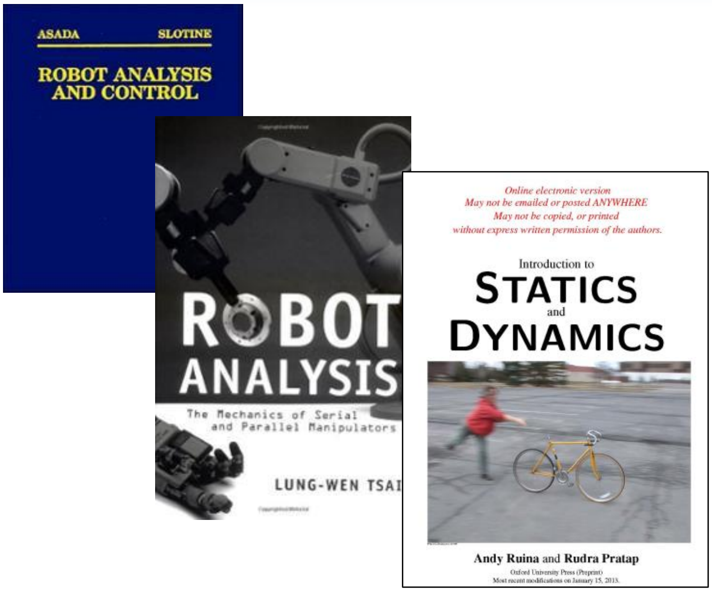
metr4202@itee.uq.edu.au

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

Schedule

Week	Date	Lecture (W: 11:10-12:40, 24-402)
1	30-Jul	Introduction
2	6-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	13-Aug	Robot Kinematics (& Ekka Day)
4	20-Aug	Robot Dynamics & Control
5	27-Aug	Robot Trajectories & Motion
6	3-Sep	Sensors & Measurement
7	10-Sep	Perception (Computer Vision)
8	17-Sep	Navigation & Localization (+ Prof. M. Srinivasan)
9	24-Sep	Motion Planning + Control
	1-Oct	*Study break*
10	8-Oct	State-Space Modelling
11	15-Oct	Shaping the Dynamic Response
12	22-Oct	Linear Observers & LQR
13	29-Oct	Applications in Industry & Course Review

Reference Material



Outline

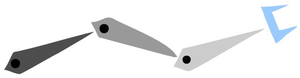
- Denavit Hartenberg Notation
- Parallel Robots

- Jacobians & Differential Motion
- Multibody Dynamics Refresher

- Newton-Euler Formulation
- Lagrange Formulation

Inverse Kinematics

- Forward: angles → position
  x = *f* (**θ**)
- Inverse: position → angles
  θ = *f*$^{-1}$(**x**)
- Analytic Approach



- Numerical Approaches:
  - Jacobian: $J = \frac{\delta x}{\delta q} \rightarrow \delta q \approx J^{-1} \delta x$
  - J^T Approximation: $\tau = J^T \cdot \mathbf{F} \rightarrow \Delta q \approx J^T \Delta x$
    - Slotine & Sheridan method
  - Cyclical Coordinate Descent

Inverse Kinematics

- Inverse Kinematics is the problem of finding the joint parameters given only the values of the homogeneous transforms which model the mechanism
  (i.e., the pose of the end effector)
- Solves the problem of where to drive the joints in order to get the hand of an arm or the foot of a leg in the right place
- In general, this involves the solution of a set of simultaneous, non-linear equations
- Hard for serial mechanisms, easy for parallel

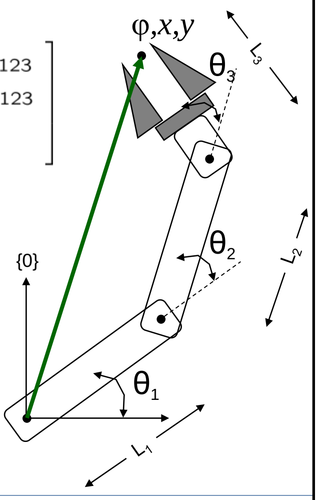
Solution Methods

- Unlike with systems of linear equations, there are no general algorithms that may be employed to solve a set of nonlinear equation
- **Closed-form** and **numerical** methods exist
- We will concentrate on analytical, closed-form methods
- These can be characterized by two methods of obtaining a solution: **algebraic** and **geometric**

Inverse Kinematics: Algebraic Approach

- We have a series of equations which define this system
- Recall, from Forward Kinematics:

$$^0T_3 = \begin{bmatrix} c_{\theta_{123}} & -s_{\theta_{123}} & 0 & L_1 c_{\theta_1} + L_2 c_{\theta_{12}} + L_3 c_{\theta_{123}} \\ s_{\theta_{123}} & c_{\theta_{123}} & 0 & L_1 s_{\theta_1} + L_2 s_{\theta_{12}} + L_3 s_{\theta_{123}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- The end-effector pose is given by

$$^0T_3 = \begin{bmatrix} c_\phi & -s_\phi & 0 & x \\ s_\phi & c_\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Equating terms gives us a set of algebraic relationships

Denavit Hartenberg [DH] Notation

- J. Denavit and R. S. Hartenberg first proposed the use of homogeneous transforms for articulated mechanisms

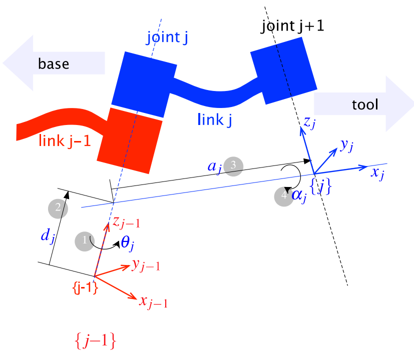
  (But B. Roth, introduced it to robotics)

- A kinematics "short-cut" that reduced the number of parameters by adding a structure to frame selection

- For two frames positioned in space, the first can be moved into coincidence with the second by a sequence of 4 operations:
  - rotate around the x_{i-1} axis by an angle α_i
  - translate along the x_{i-1} axis by a distance a_i
  - translate along the new z axis by a distance d_i
  - rotate around the new z axis by an angle θ_i

Denavit-Hartenberg Convention

- link length a_i the offset distance between the z_{i-1} and z_i axes along the x_i axis;
- link twist α_i the angle from the z_{i-1} axis to the z_i axis about the x_i axis;



Art c/o P. Corke

- link offset d_i the distance from the origin of frame i-1 to the x_i axis along the z_{i-1} axis;
- joint angle θ_i the angle between the x_{i-1} and x_i axes about the z_{i-1} axis.

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. θ (rot *z*) → d (trans *z*) → a (trans *x*) → α (rot *x*)

Denavit-Hartenberg → Rotation Matrix

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

$$
\begin{aligned}
{}^{i-1}A_i &= Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdots \\
&\quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

DH Example [1]: RRR Link Manipulator

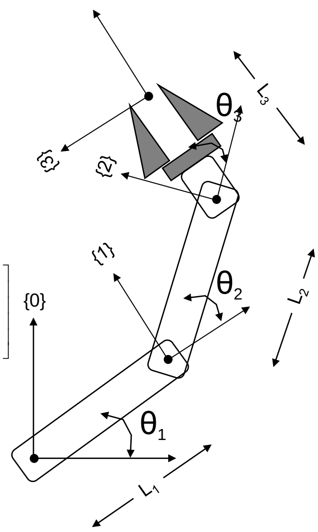
1. Assign the frames at the joints …
2. Fill DH Table …

Link	a_i	α_i	d_i	θ_i
1	L_1	0	0	θ_1
2	L_2	0	0	θ_2
3	L_3	0	0	θ_3

$$^0A_1 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & L_1c_{\theta_1} \\ s_{\theta_1} & c_{\theta_1} & 0 & L_1s_{\theta_1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \; ^1A_2 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & L_2c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & L_2s_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \; ^2A_3 = \begin{bmatrix} c_{\theta_3} & -s_{\theta_3} & 0 & L_3c_{\theta_3} \\ s_{\theta_3} & c_{\theta_3} & 0 & L_3s_{\theta_3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^0T_3 = {}^0A_1\,{}^1A_2\,{}^2A_3$$

$$= \begin{bmatrix} c_{\theta_{123}} & -s_{\theta_{123}} & 0 & L_1c_{\theta_1} + L_2c_{\theta_{12}} + L_3c_{\theta_{123}} \\ s_{\theta_{123}} & c_{\theta_{123}} & 0 & L_1s_{\theta_1} + L_2s_{\theta_{12}} + L_3s_{\theta_{123}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



DH Example [2]: RRP Link Manipulator

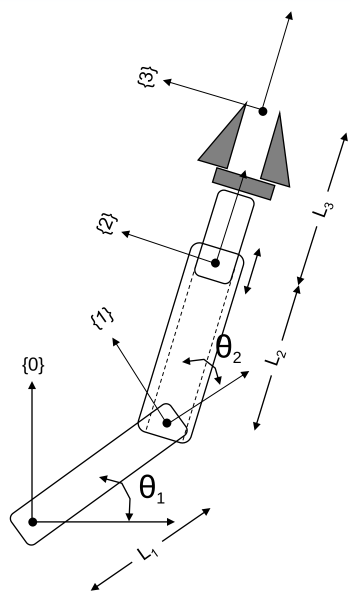
1. Assign the frames at the joints …
2. Fill DH Table …

Link	a_i	α_i	d_i	θ_i
1	L_1	0	0	θ_1
2	L_2	0	0	θ_2
3	L_3	0	0	0

$$^0A_1 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & L_1c_{\theta_1} \\ s_{\theta_1} & c_{\theta_1} & 0 & L_1s_{\theta_1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \; ^1A_2 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & L_2c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & L_2s_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \; ^2A_3 = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

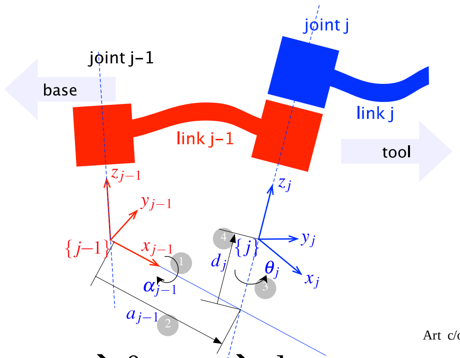
$$^0T_3 = {}^0A_1\,{}^1A_2\,{}^2A_3$$

$$= \begin{bmatrix} c_{\theta_{12}} & -s_{\theta_{12}} & 0 & L_1c_{\theta_1} + (L_2 + L_3)c_{\theta_{12}} \\ s_{\theta_{12}} & c_{\theta_{12}} & 0 & L_1s_{\theta_1} + (L_2 + L_3)s_{\theta_{12}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Modified DH

- Made "popular" by Craig's *Intro. to Robotics* book
- Link coordinates attached to the near by joint

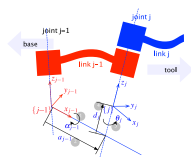


Art c/o P. Corke

- a $_{\text{(trans } x\text{-}1)}$ → α $_{\text{(rot } x\text{-}1)}$ → θ $_{\text{(rot } z)}$ → d $_{\text{(trans } z)}$

Modified DH [2]

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



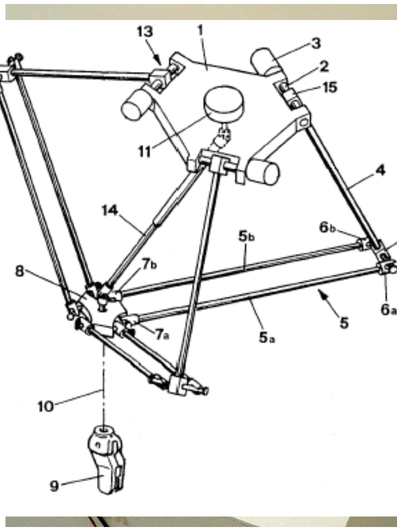
$$\Rightarrow {}^{i-1}A_i = R_x(\alpha_{i-1})\, T_x(a_{i-1})\, R_z(\theta_i)\, T_x(d_i)$$

- Refactoring Standard → to Modified

$$\underbrace{\{R_z(\theta_1)\,T_z(d_1)\,T_x(a_1)\,R_x(\alpha_1)\}}_{\text{DH}_1}\cdot\underbrace{\{R_z(\theta_2)\,T_z(d_2)\,T_x(a_2)\,R_x(\alpha_2)\}}_{\text{DH}_2}\cdot\underbrace{\{R_z(\theta_3)\,T_z(d_3)\}}_{\text{End Effector}}$$

$$=\underbrace{\{R_z(\theta_1)\,T_z(d_1)\}}_{\text{Base}}\cdot\underbrace{\{T_x(a_1)\,R_x(\alpha_1)\,R_z(\theta_2)\,T_z(d_2)\}}_{\text{MDH}_1}\cdot\underbrace{\{T_x(a_2)\,R_x(\alpha_2)\,R_z(\theta_3)\,T_z(d_3)\}}_{\text{MDH}_2}$$

Parallel Manipulators



Sources: Wikipedia, "Delta Robot", ParallelMic.Org, "Delta Parallel Robot", and US Patent 4,976,582

- 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:

- # Limbs (***m***) = # DOF (***F***)
- The joints are arranged in an identical pattern
- The # and location of actuated joints are the same

Thus:

- Number of Loops (L): One less than # of limbs

$$L = m - 1 = F - 1$$

- Connectivity (C_k)

$$\sum_{k=1}^{m} C_k = (\lambda + 1)\,F - \lambda$$

Where: λ: The DOF of the space that the system is in (e.g., λ=6 for 3D space).

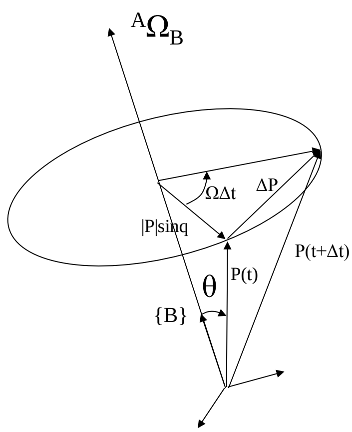
Robot Dynamics

Angular Velocity

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

$$\begin{aligned}
|\Delta \mathbf{P}| &= \left(\left|{}^{A}\mathbf{P}\right| \sin\theta\right)\left(\left|{}^{A}\Omega_B\right| \Delta t\right) \\
\frac{|\Delta \mathbf{P}|}{\Delta t} &= \left(\left|{}^{A}\mathbf{P}\right| \sin\theta\right)\left(\left|{}^{A}\Omega_B\right|\right) \\
&= {}^{A}\Omega_B \times {}^{A}\mathbf{P}
\end{aligned}$$

$${}^{A}\mathbf{V}_P = {}^{A}\Omega_B \times {}^{A}R_B \, {}^{B}\mathbf{P}$$



Velocity

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

$$^{A}\mathbf{P} = {}^{A}\mathbf{P}_B + {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P}$$

- Differentiating w/r/t to **t** we find

$$^{A}\mathbf{V}_P = \frac{d}{dt}{}^{A}\mathbf{P} = \lim_{\Delta t \to 0} \frac{^{A}\mathbf{P}(t+\Delta t) - {}^{A}\mathbf{P}(t)}{\Delta t}$$
$$= {}^{A}\dot{\mathbf{P}}_B + {}^{A}_{B}\mathbf{R}\,{}^{B}\dot{\mathbf{P}} + {}^{A}_{B}\dot{\mathbf{R}}\,{}^{B}\mathbf{P}$$

- 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

$$\mathbf{V} = \omega \times \mathbf{r}$$

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

$$\rightarrow \mathbf{V} = \Omega\mathbf{r}$$

Velocity Representations

- Euler Angles
  - For Z-Y-X (α,β,γ):

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

- 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

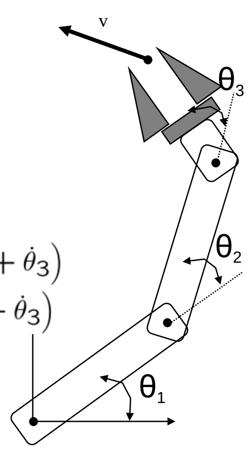
- Consider again the schematic of the planar manipulator shown. We found that the end effector position is given by

$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1+\theta_2) + L_3 \cos(\theta_1+\theta_2+\theta_3)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1+\theta_2) + L_3 \sin(\theta_1+\theta_2+\theta_3)$$

- Differentiating w/r/t to t

$$\dot{x} = -L_1 \mathsf{s}_1 \dot{\theta}_1 - L_2 \mathsf{s}_{12} \left(\dot{\theta}_1 + \dot{\theta}_2\right) - L_3 \mathsf{s}_{123} \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right)$$
$$\dot{y} = L_1 \mathsf{c}_1 \dot{\theta}_1 + L_2 \mathsf{c}_{12} \left(\dot{\theta}_1 + \dot{\theta}_2\right) + L_3 \mathsf{c}_{123} \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right)$$

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



Manipulator Velocities [2]

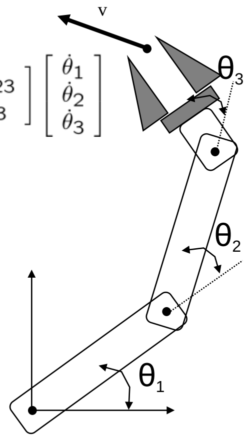
- Rearranging, we can recast this relation in matrix form

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

- Or

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \frac{\partial x}{\partial \theta_3} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial y}{\partial \theta_3} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

- 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 → Jacobian

$$d\mathbf{x} = J d\theta$$

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

$$J = \begin{bmatrix} [J_1] & [J_2] \end{bmatrix}$$

$$v = J_1 \dot{\theta}_1 + J_2 \dot{\theta}_2$$

Infinitesimal Rotations

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

$$\mathbf{R}_x(d\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cd\phi & -sd\phi \\ 0 & sd\phi & cd\phi \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -d\phi_x \\ 0 & d\phi_x & 1 \end{bmatrix}$$

$$\mathbf{R}_y(d\phi) = \begin{bmatrix} cd\phi & 0 & sd\phi \\ 0 & 1 & 0 \\ -sd\phi & 0 & cd\phi \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & d\phi_y \\ 0 & 1 & 0 \\ -d\phi_y & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_z(d\phi) = \begin{bmatrix} cd\phi & -sd\phi & 0 \\ sd\phi & cd\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & -d\phi_z & 0 \\ d\phi_z & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Note that:

$$R_x(d\varphi) R_y(d\varphi) = R_y(d\varphi) R_x(d\varphi)$$

→ Therefore … they **commute**

The Jacobian

- In general, the Jacobian takes the form
  (for example, **j joints** and in **i 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

$$\dot{\mathbf{X}} = \mathbf{J}(\theta)\dot{\theta}$$

Jacobian [2]

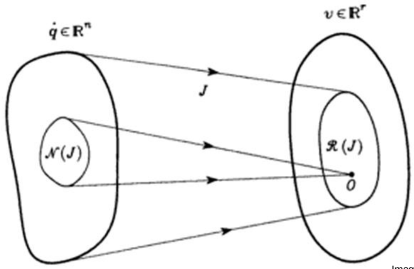


Image:. Sciavicco and Siciliano, *Modelling and Control of Robot Manipulators*, 2nd ed, 2000

- Jacobian can be viewed as a mapping from
  Joint velocity space ($\dot{q}$) to
  Operational velocity space (*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** (**ϵ=1**), then $\mathbf{v}_i = \frac{dz}{dt}$
- If the joint is **revolute** (**ϵ=0**), then $\omega = \frac{d\theta}{dt}$ (in the $\hat{k}$ direction)

$$\therefore v = \sum_{i=1}^{N} \left(\varepsilon_i v_i + \bar{\varepsilon}_i \left(\omega_i \times \mathbf{p}_{i-1}^{i} \right) \right) \qquad \omega = \sum_{i=1}^{N} \left(\bar{\varepsilon}_i \left(\dot{\boldsymbol{\theta}}_i \right) \right) = \sum_{i=1}^{N} \left(\bar{\varepsilon}_i \mathbf{z}_i \left(\dot{\theta}_i \right) \right)$$

$$\rightarrow v = J_v \dot{\mathbf{q}} \qquad \boldsymbol{\omega} = J_\omega \dot{\mathbf{q}}$$

- Combining them (with **v**=(Δx, Δθ))

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}$$

Jacobian: **Explicit Form [2]**

- The overall Jacobian takes the form

$$J = \begin{bmatrix} \frac{\partial x_P}{\partial q_1} & \cdots & \frac{\partial x_P}{\partial q_n} \\ \bar{\varepsilon}_1 z_1 & \cdots & \bar{\varepsilon}_1 z_n \end{bmatrix}$$

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

$$^{F}J = \begin{bmatrix} ^{F}J_v \\ ^{F}J_\omega \end{bmatrix} = \begin{bmatrix} \frac{\partial\, ^{F}x_P}{\partial q_1} & \cdots & \frac{\partial\, ^{F}x_P}{\partial q_n} \\ \bar{\varepsilon}_1\, ^{F}z_1 & \cdots & \bar{\varepsilon}_1\, ^{F}z_n \end{bmatrix}$$

Where: $^{F}\mathbf{z}_i = {}^{F}_{i}R\, ^{i}\mathbf{z}_i$ & $^{i}\mathbf{z}_i = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$

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
  - Langrangian 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

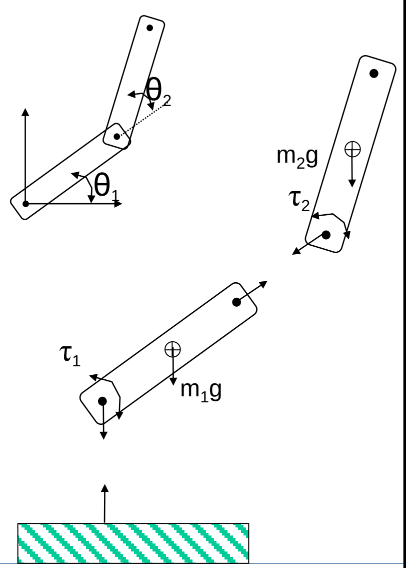
- Inverse Dynamics:
  - Find the (actuator) torques/forces required to generate a desired trajectory of the manipulator

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



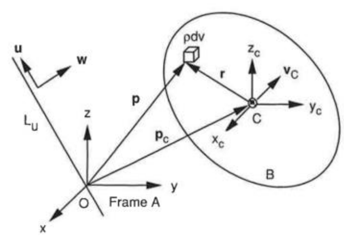
Dynamics

- For Manipulators, the general form is

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

where
  - τ is a vector of joint torques
  - Θ 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 Θ so the dynamics varies as the manipulator move

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:

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

- The scalar product of I^o_u with a second axis (**w**) is called the product of inertia

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

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

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

Where: $\mathbf{r_g}$: *radius of gyration* of B w/r/t to L

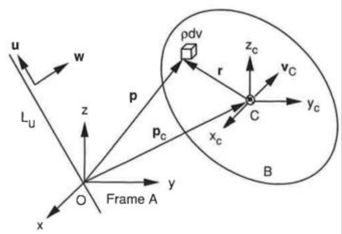
$$r_g = p^2 - \left(p^T u\right)^2 = (u \times p)^2$$

Dynamics: Mass Matrix & Inertia Matrix

- This can be written in a Matrix form as:

$$\mathbf{I}_u^O = I_B^O \mathbf{u}$$

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



$$I_B^O = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yz} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

- Where to get I_{xx}, etc? ➔ Parallel Axis Theorem

If CM is the center of mass, then:

$$I_{xx}^O = I_{xx}^{CM} + m\left(y_c^2 + z_c^2\right) \qquad I_{xy}^O = I_{xx}^{CM} + m x_c y_c$$
$$I_{yy}^O = I_{yy}^{CM} + m\left(x_c^2 + z_c^2\right) \qquad I_{yz}^O = I_{xx}^{CM} + m y_c z_c$$
$$I_{zz}^O = I_{zz}^{CM} + m\left(x_c^2 + y_c^2\right) \qquad I_{zx}^O = I_{xx}^{CM} + m z_c x_c$$

Dynamics: Mass Matrix

- The Mass Matrix: Determining via the Jacobian!

$$K = \sum_{i=1}^{N} K_i$$

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

$$v_{C_i} = J_{v_i}\dot{\theta} \qquad J_{v_i} = \left[\begin{array}{cccccc} \frac{\partial \mathbf{p}_{C_1}}{\partial \theta_1} & \cdots & \frac{\partial \mathbf{p}_{C_i}}{\partial \theta_i} & \underbrace{0}_{i+1} & \cdots & \underbrace{0}_{n} \end{array}\right]$$

$$\omega_i = J_{\omega_i}\dot{\theta} \qquad J_{\omega_i} = \left[\begin{array}{cccccc} \bar{\varepsilon}_1 Z_1 & \cdots & \bar{\varepsilon}_i Z_i & \underbrace{0}_{i+1} & \cdots & \underbrace{0}_{n} \end{array}\right]$$

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

! M is symmetric, positive definite $\therefore m_{ij} = m_{ji}, \dot{\boldsymbol{\theta}}^T M \dot{\boldsymbol{\theta}} > 0$

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
- Using this formulation and the concept of virtual work we can find the forces and torques acting on the system.
- This may seem more involved but is often easier to formulate for complex systems

$$L = K - P$$

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

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

Dynamics – Langrangian Mechanics [2]

$L = K - P, \dot{\boldsymbol{\theta}}$: Generalized Velocities, M : Mass Matrix

$$\boldsymbol{\tau} = \sum_{i=1}^{N} \tau_i = \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\boldsymbol{\theta}}}\right) - \frac{\partial K}{\partial \boldsymbol{\theta}} + \frac{\partial P}{\partial \boldsymbol{\theta}}$$

$$K = \tfrac{1}{2}\dot{\theta}^T M(\theta)\,\dot{\theta}$$

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

$$\rightarrow \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\theta}}\right) - \frac{\partial K}{\partial \theta} = \left[M\ddot{\theta} + \dot{M}\dot{\theta}\right] - \left[\tfrac{1}{2}\dot{\theta}^T M(\theta)\,\dot{\theta}\right] = M\ddot{\theta} + \underbrace{\left\{\dot{M}\dot{\theta} - \tfrac{1}{2}\begin{bmatrix} \dot{\theta}^T \frac{\partial M}{\partial \theta_1}\dot{\theta} \\ \vdots \\ \dot{\theta}^T \frac{\partial M}{\partial \theta_n}\dot{\theta} \end{bmatrix}\right\}}_{\mathbf{v}(\theta, \dot{\theta})}$$

$$\mathbf{v}\left(\theta, \dot{\theta}\right) = \underbrace{C(\theta)\left[\dot{\theta}^2\right]}_{\text{Centrifugal}} + \underbrace{B(\theta)\left[\dot{\theta}\dot{\theta}\right]}_{\text{Coriolis}}$$

$$\Rightarrow \boldsymbol{\tau} = M(\theta)\ddot{\boldsymbol{\theta}} + \mathbf{v}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right) + \mathbf{g}(\boldsymbol{\theta})$$

Dynamics – Langrangian Mechanics [3]

- The Mass Matrix: Determining via the Jacobian!

$$K = \sum_{i=1}^{N} K_i$$

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

$$v_{C_i} = J_{v_i}\dot{\theta} \qquad J_{v_i} = \left[\begin{array}{cccccc} \frac{\partial \mathbf{p}_{C_1}}{\partial \theta_1} & \cdots & \frac{\partial \mathbf{p}_{C_i}}{\partial \theta_i} & \underbrace{0}_{i+1} & \cdots & \underbrace{0}_{n} \end{array}\right]$$

$$\omega_i = J_{\omega_i}\dot{\theta} \qquad J_{\omega_i} = \left[\begin{array}{cccccc} \bar{\varepsilon}_1 Z_1 & \cdots & \bar{\varepsilon}_i Z_i & \underbrace{0}_{i+1} & \cdots & \underbrace{0}_{n} \end{array}\right]$$

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

! M is symmetric, positive definite $\therefore m_{ij} = m_{ji}, \dot{\boldsymbol{\theta}}^T M \dot{\boldsymbol{\theta}} > 0$

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 **p**: $d\mathbf{p} = \mathbf{J}d\mathbf{q}$

$$\mathbf{q} = \left[q_1 \quad \dots \quad q_n\right] \rightarrow \mathbf{p} = \left[p_1 \quad \dots \quad p_n\right]$$

→Thus: the kinetic energy and gravity terms become

$$KE = \tfrac{1}{2}\dot{\mathbf{p}}^T \mathbf{H}^* \dot{\mathbf{p}} \qquad \mathbf{G}^* = \left(\mathbf{J}^{-1}\right)^T \mathbf{G}$$

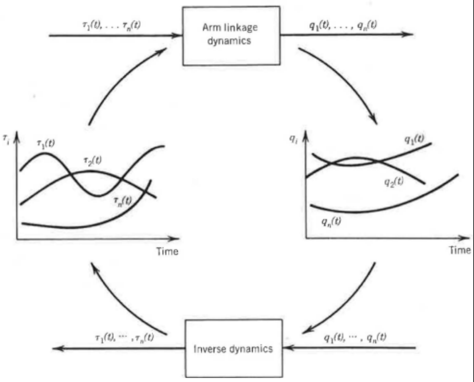
where: $\mathbf{H}^* = \left(\mathbf{J}^{-1}\right)^T \mathbf{H}\mathbf{J}^{-1}$

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

    $\mathbf{q} = [q_1 \ \cdots \ q_n] \rightarrow [\theta_1(t) \ \ \theta_2(t) \ \ \theta_3(t)]$
  - Outputs are joint torques to be applied at each instance

    $\boldsymbol{\tau} = [\tau_1 \ \cdots \ \tau_n]$



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

Graphic from Asada & Slotine p. 119

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

$$\dot{\theta} = \mathbf{J}(\theta)^{-1}\dot{\mathbf{X}}$$

- 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:

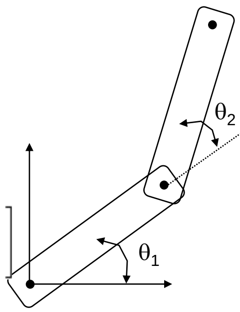
$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

- The Jacobian for this is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

- 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 θ_2 approaches 0 or π 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

$$F \cdot \delta\mathbf{X} = \tau \cdot \delta\theta$$

- Or

$$F^T \delta\mathbf{X} = \tau^T \delta\theta$$

- Earlier we saw that

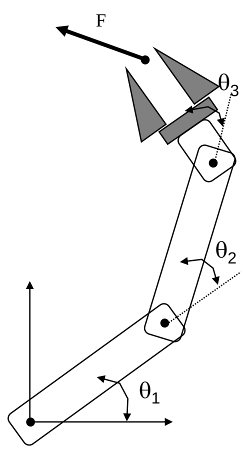
$$\delta\mathbf{X} = \mathbf{J}\delta\theta$$

- So that

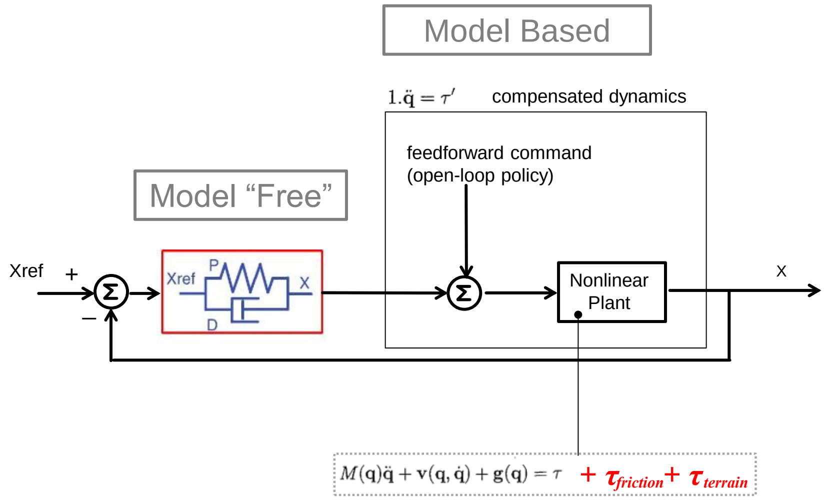
$$F^T \mathbf{J} = \tau^T$$

- Or

$$\tau = \mathbf{J}^T F$$

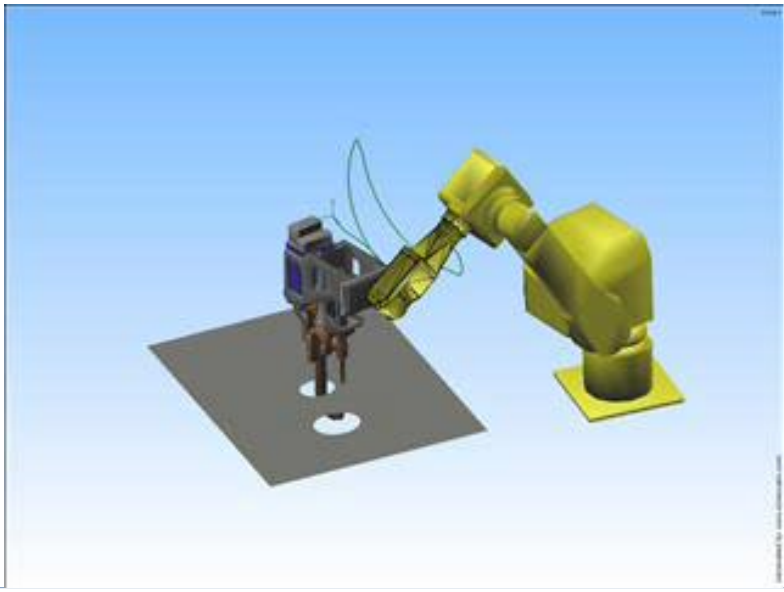


Operation Space (Computed Torque)



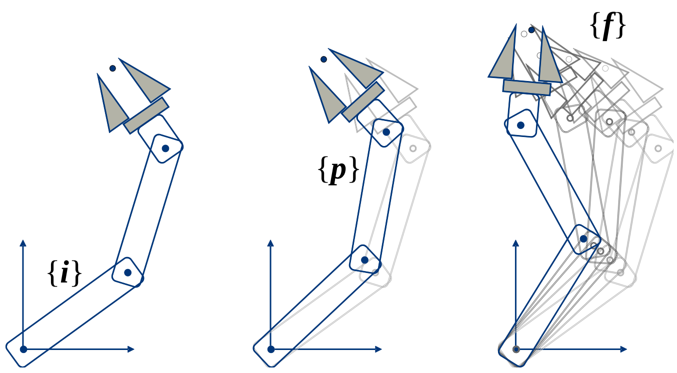
Compensated Manipulation

Trajectory Generation & Planning



Trajectory Generation

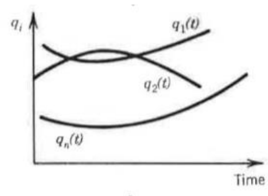
- The goal is to get from an initial position $\{\boldsymbol{i}\}$ to a final position $\{\boldsymbol{f}\}$ via a path points $\{\boldsymbol{p}\}$



Joint Space

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

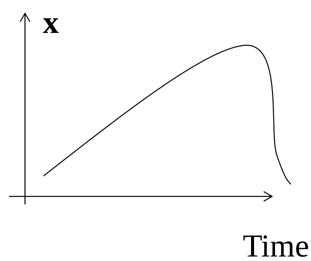
- + Since we control the joints, this is more direct
- -- If we want to follow a particular trajectory, not easy
  - at best lots of intermediate points
  - No guarantee that you can solve the Inverse Kinematics for all path points



Cartesian Workspace

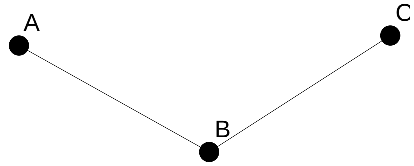
Consider the **Cartesian positions** as a function of time

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



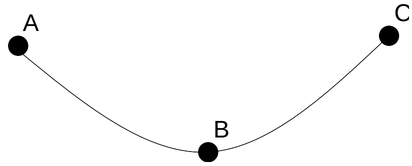
Polynomial Trajectories

- Straight line Trajectories



- Simpler

- Polynomial Trajectories



$$u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

- 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

Dynamic Simulation Software

- v-rep
  virtual robot experimentation platform
- Reflexxes

http://www.coppeliarobotics.com/ http://www.reflexxes.com/