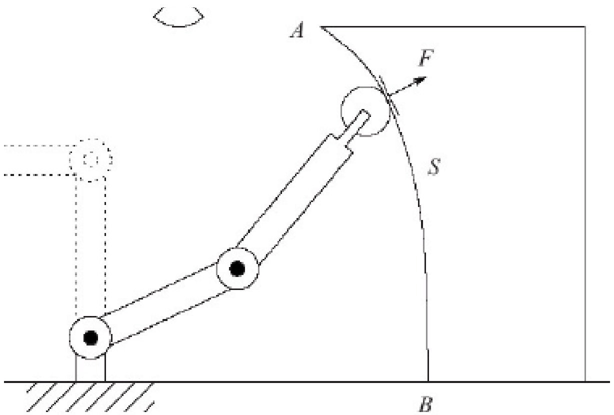
FRTF20 Applied Robotics
Lecture 3

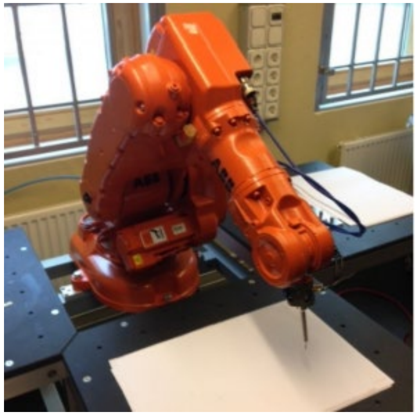
GIORGOS NIKOLERIS / ANDERS ROBERTSSON

Robotics in this course

- Hands-on-experience and project
- Applied **robot programming** (3d-simulation/CAD: RobotStudio)
- The **following conceptual problems** must be resolved to make a robot succeed in performing a typical task:
  - Forward Kinematics
  - Inverse Kinematics
  - Velocity Kinematics/Jacobians
  - Dynamics
  - Path Planning and Trajectory Generation
  - Motion Control
  - (Force Control)
  - Sequence programming (and task description)

From task description to performed task

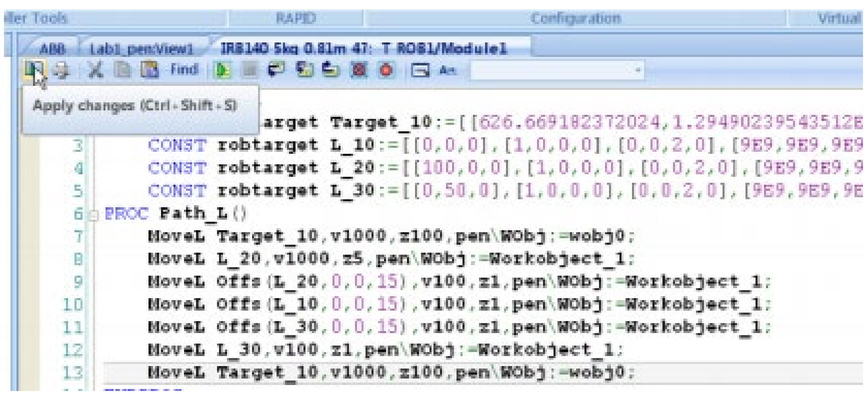




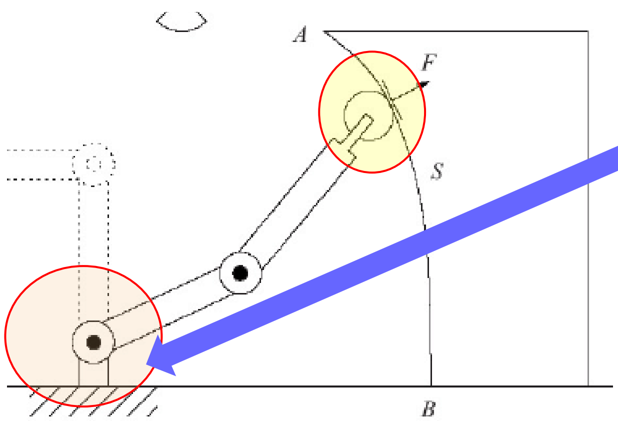
Applied **robot programming**: Robot + CAD of workcell

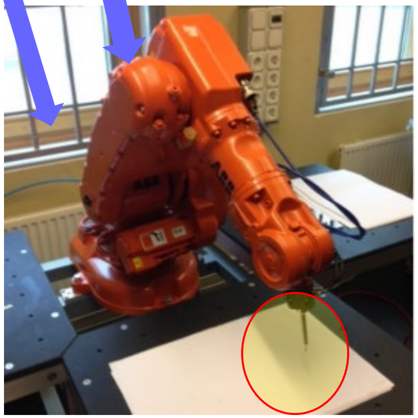
RobotStudio Exercises

(1) Paths, frames and configurations (2) I/O (3) collision avoidance



So how is it working behind the scenes?
From program instruction to motor angles





```
    CONST robtarget L_20:=[[100,0,0],[1,0,0,0],[0,0,2,0],[9E9,9E9,9
    CONST robtarget L_30:=[[0,50,0],[1,0,0,0],[0,0,2,0],[9E9,9E9,9E
PROC Path_L()
    MoveL Target_10,v1000,z100,pen\WObj:=wobj0;
    MoveL L_20,v1000,z5,pen\WObj:=Workobject_1;
    MoveL Offs(L_20,0,0,15),v100,z1,pen\WObj:=Workobject_1;
    MoveL Offs(L_10,0,0,15),v100,z1,pen\WObj:=Workobject_1;
```

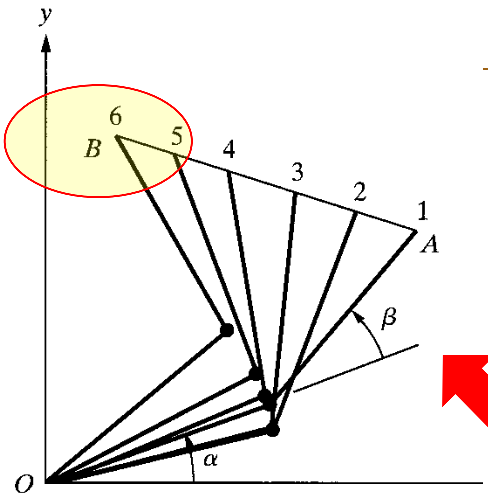
From MoveL L_20 to joint values [q1..q6]

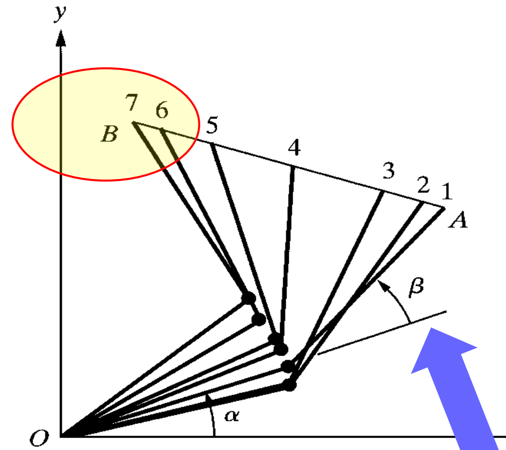
(Lec 2,3) Frames, Forward/Inverse kinematics (joint angles ←→ pose)

(Lec 3) Relation between velocities (Jacobian)

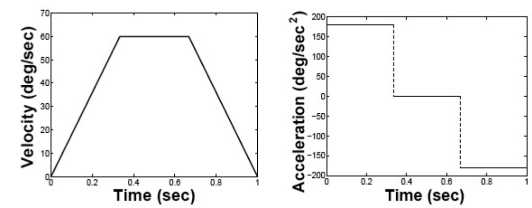
Computer exercises [matlab]: frames; DH-modelling, ...

From program instruction to paths and trajectories of motor angles





Velocity and acceleration constraints in joint space or in Cartesian motion (or combo)



(b) (c)

```
CONST robtarget L_20:=[[ 0,0,0],[1,0,0,0],[0,0,2,0],[9  ,9E9,9
CONST robtarget L_30:=[[0  ,0],[1,0,0,0],[0,0,2,0],[9E9  E9,9E
PROC Path_L()
    MoveL Target_10,v1000,z100,pen\WObj:=wobj0;
    MoveL L_20,v1000,z5,pen\WObj:=Workobject_1;
    MoveL Offs(L_20,0,0,15),v100,z1,pen\WObj:=Workobject_1;
    MoveL Offs(L_10,0,0,15),v100,z1,pen\WObj:=Workobject_1;
```

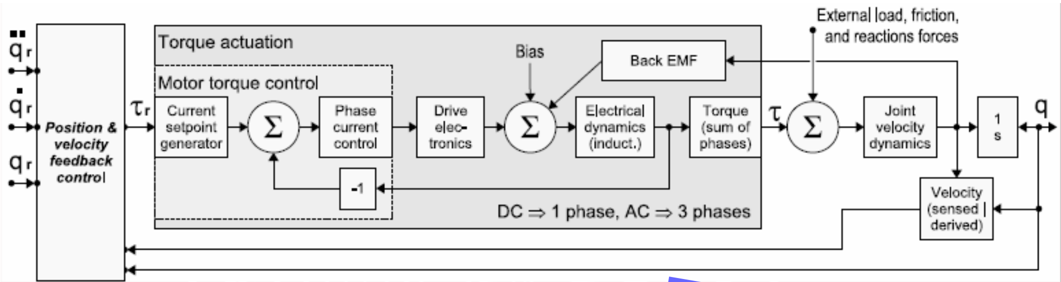
From MoveL L_20 to joint values [q1..q6]

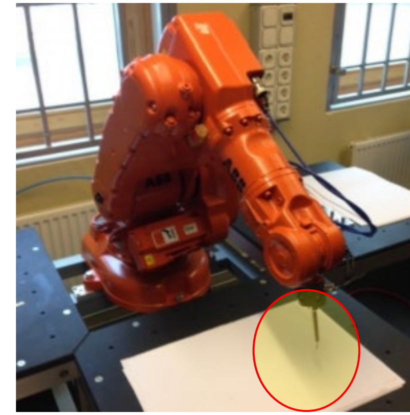
(Lec 2,3) Frames, Forward/Inverse kinematics (joint angles ←→ pose)

(Lec 3) Relation between velocities (Jacobian)

(Lec 3) Paths and trajectories (geometric vs dynamic due to v_max, tau_max)

From program instruction to motor angles





```
CONST robtarget L_20:=[[100, 0],[1,0,0,0],[0,0,2,0],[9E9, E9,9
CONST robtarget L_30:=[[0,50, ],[1,0,0,0],[0,0,2,0],[9E9, 9,9E
PROC Path_L()
    MoveL Target_10,v1000,z100,pen\W bj:=wobj0;
    MoveL L_20,v1000,z5,pen\WObj:=Wor bject_1;
    MoveL Offs(L_20,0,0,15),v100,z1,pen WObj:=Workobject_1;
    MoveL Offs(L_10,0,0,15),v100,z1,pen\ bj:=Workobject_1;
```

From MoveL L_20 to joint values [q1..q6]

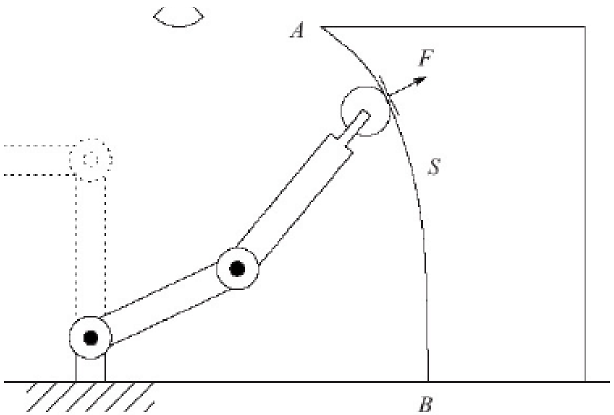
(Lec 2,3) Frames, Forward/Inverse kinematics (joint angles ←-

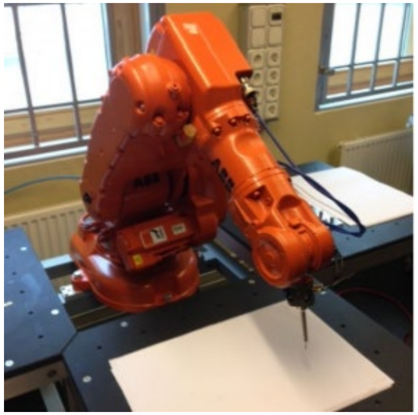
(Lec 3) Relation between velocities (Jacobian)

(Lec 4) ”From q_ref to q ” , servo control: PID + FeedForward

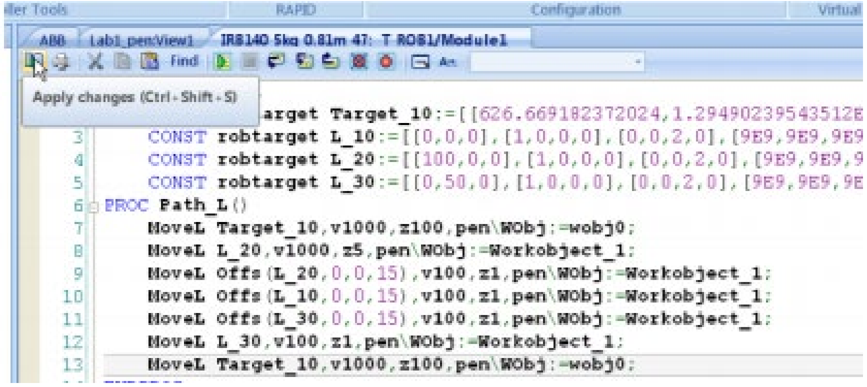
(Lec 5) Robot modeling to find dynamic process model:

From task description to performed task

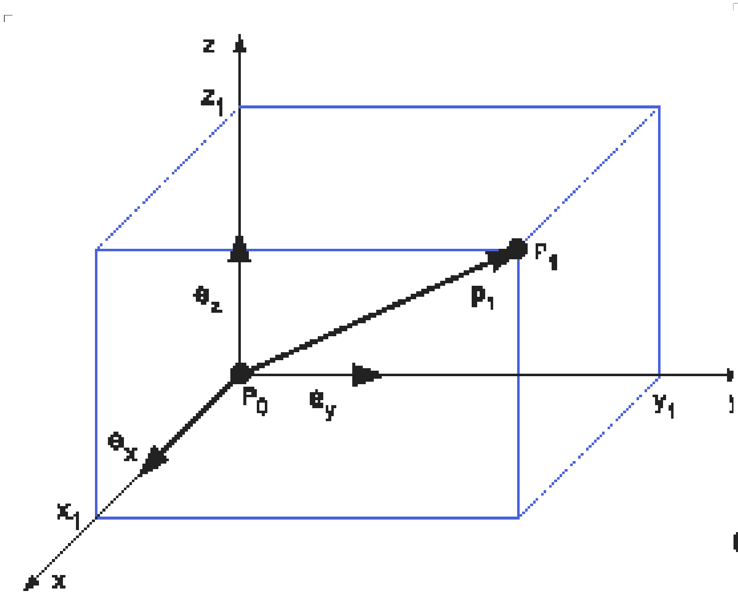


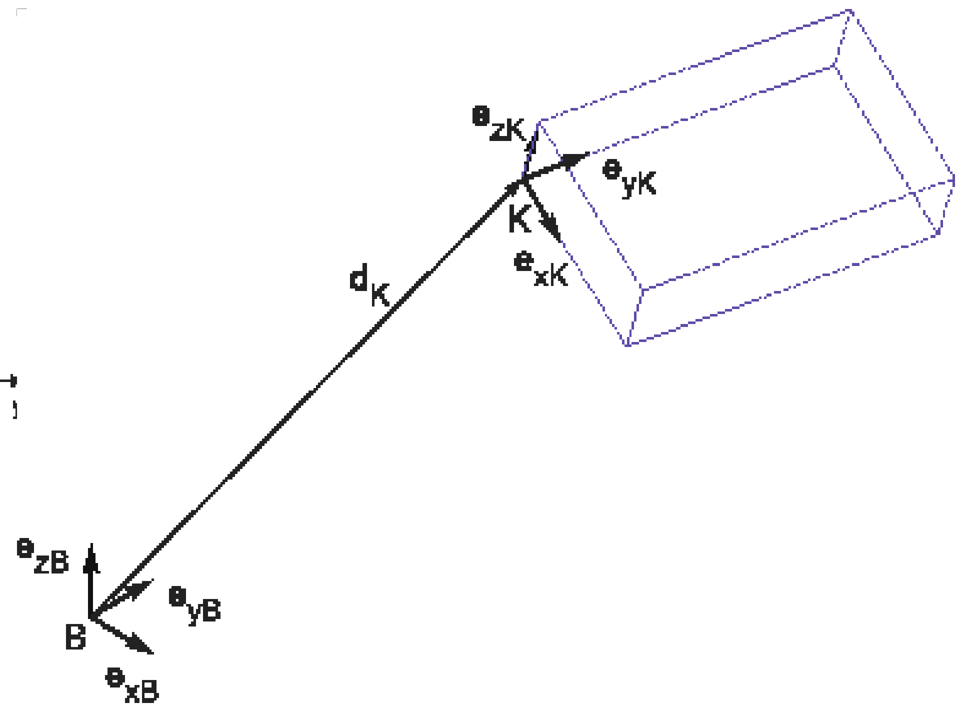


Modeling and robot programming: Robot + CAD of workcell
RobotStudio Exercises 1,2, and 3:
Paths and configurations , I/O , collision avoidance



Representing Positions & Orientations





$$ {}^{B}\boldsymbol{R}_{K} = \begin{pmatrix} {}^{B}\boldsymbol{e}_{xK} & {}^{B}\boldsymbol{e}_{yK} & {}^{B}\boldsymbol{e}_{zK} \end{pmatrix} = \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix} $$

Conversions between rotational representations

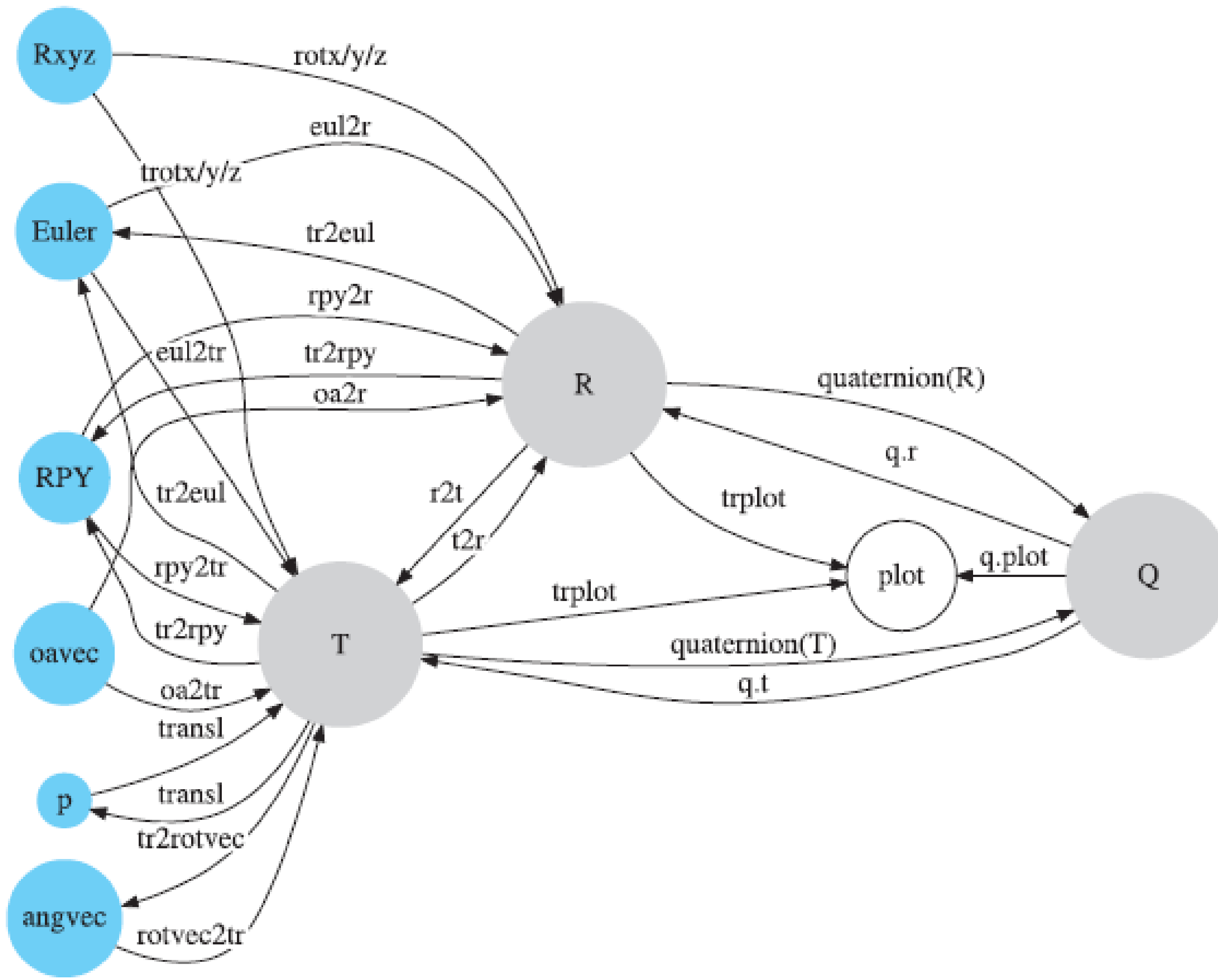
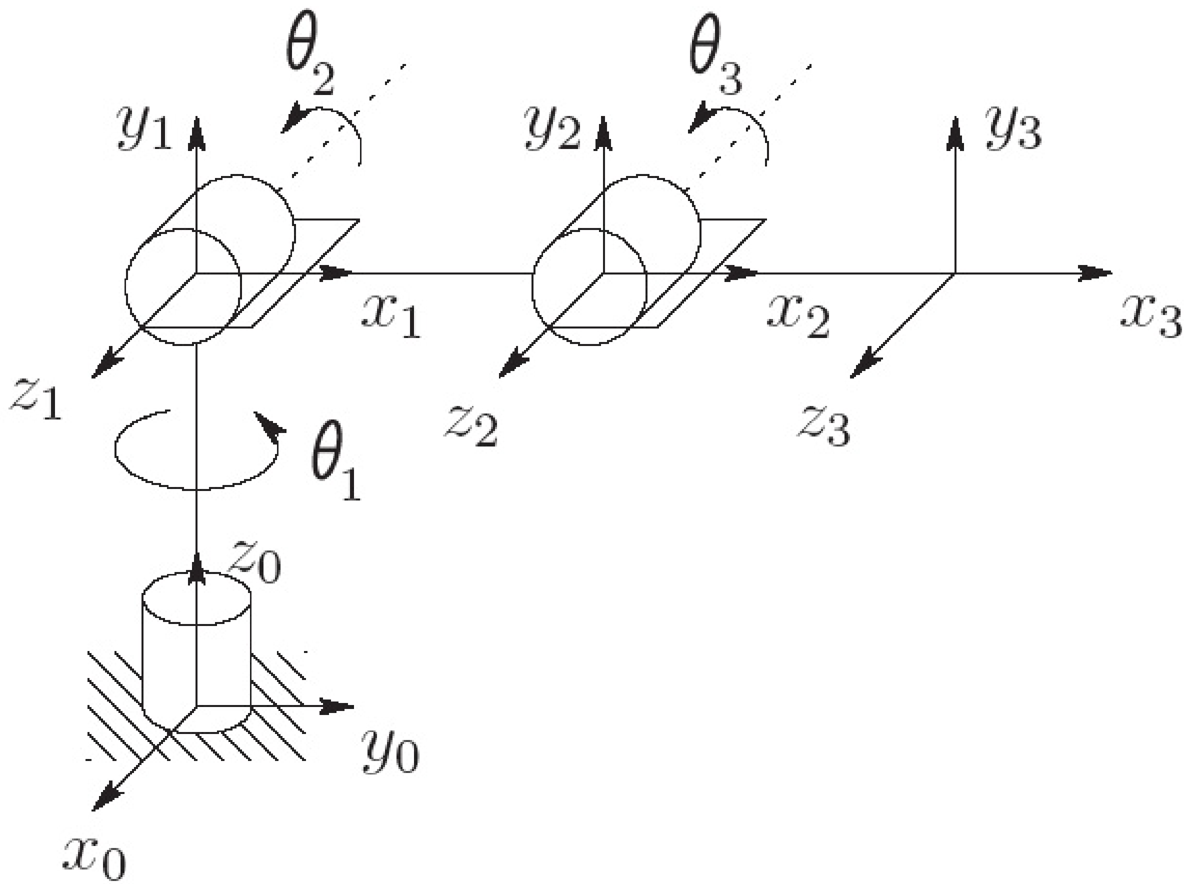


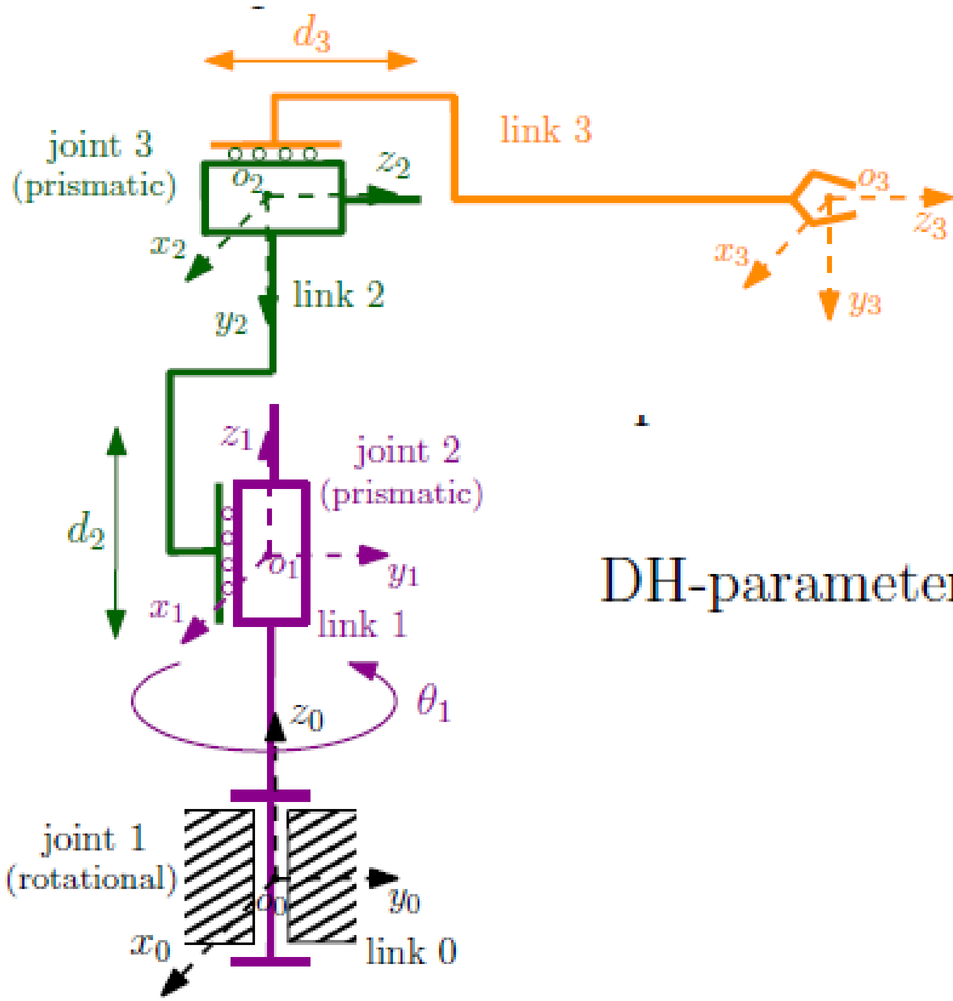
Fig. 2.15.
Conversion between rotational representations

Forward Kinematics

Find the position and orientation of the end effector given the values for the joint variables



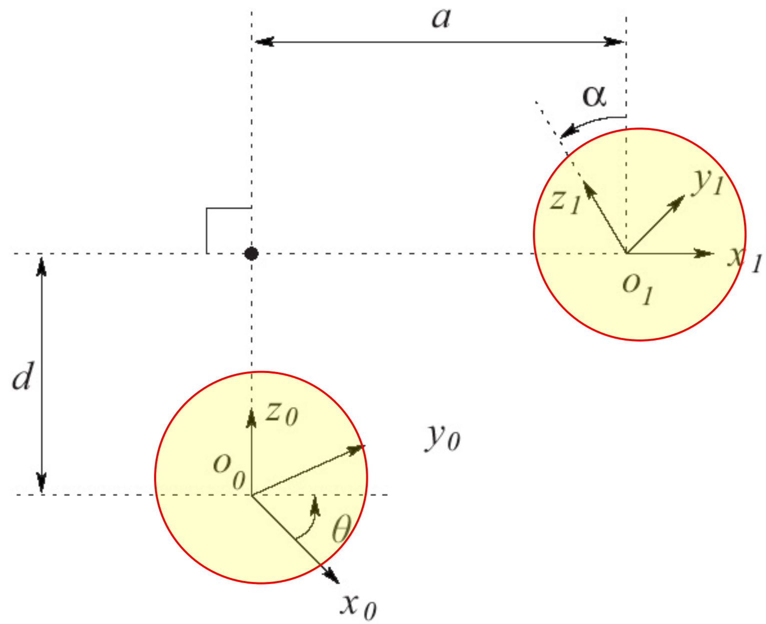
Representation from link-to-link (outwards)



DH-parameters:

Link	θ_i	d_i	a_i	α_i
1	θ_1	d_1	0	0
2	0	d_2	0	$-\frac{\pi}{2}$
3	0	d_3	0	0

Denavit-Hartenberg convention



The axis x_i intersects and is perpendicular to axis z_{i-1}

$\boldsymbol{\theta_i}$.= joint angle: the angle between x_{i-1} and x_i measured about z_{i-1}.
θ_i is variable if joint i is revolute.

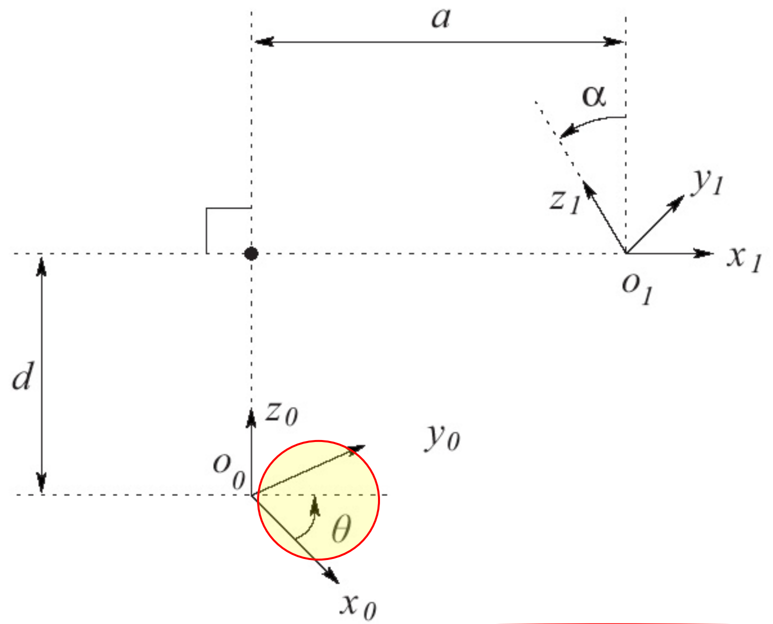
$\boldsymbol{d_i}$ =link offset: distance along z_{i-1} from O_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic

$\boldsymbol{a_i}$= link length: distance along x_i from O_i to the intersection of the x_i and z_{i-1} axes.

$\boldsymbol{\alpha_i}$ = link twist: the angle between z_{i-1} and z_i measured about x_i.

LUND UNIVERSITY

Denavit-Hartenberg convention



Rot_{z,θ_i}

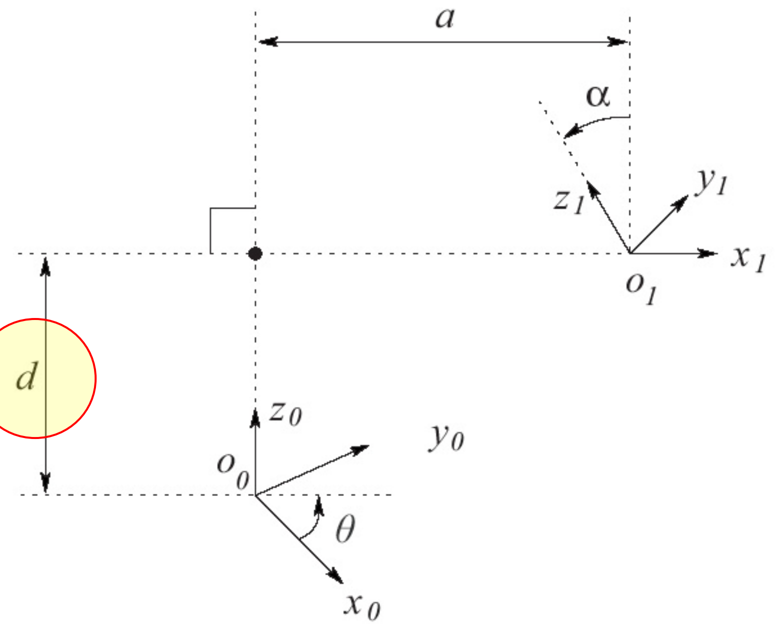
$\boldsymbol{\theta_i}$.= joint angle: the angle between x_{i-1} and x_i measured about z_{i-1}.
θ_i is variable if joint i is revolute.

$\boldsymbol{d_i}$ =link offset: distance along z_{i-1} from O_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic

$\boldsymbol{a_i}$= link length: distance along x_i from O_i to the intersection of the x_i and z_{i-1} axes.

$\boldsymbol{\alpha_i}$ = link twist: the angle between z_{i-1} and z_i measured about x_i.

Denavit-Hartenberg convention



$Trans_{z,d_i}$

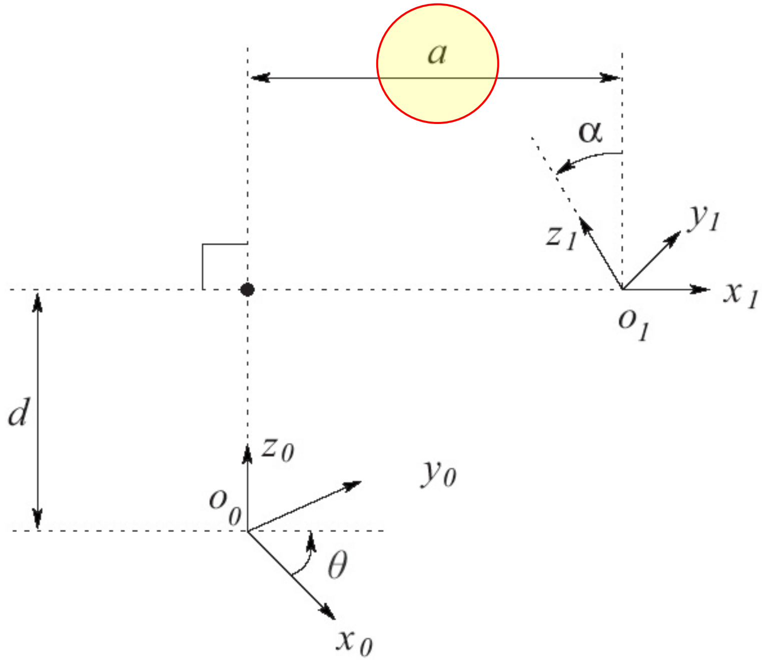
$\boldsymbol{\theta_i}$.= joint angle: the angle between x_{i-1} and x_i measured about z_{i-1}.
θ_i is variable if joint i is revolute.

$\boldsymbol{d_i}$ =link offset: distance along z_{i-1} from O_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic

$\boldsymbol{a_i}$= link length: distance along x_i from O_i to the intersection of the x_i and z_{i-1} axes.

$\boldsymbol{\alpha_i}$ = link twist: the angle between z_{i-1} and z_i measured about x_i.

Denavit-Hartenberg convention



$Trans_{x,a_i}$

$\boldsymbol{\theta_i}$.= joint angle: the angle between x_{i-1} and x_i measured about z_{i-1}.
θ_i is variable if joint i is revolute.

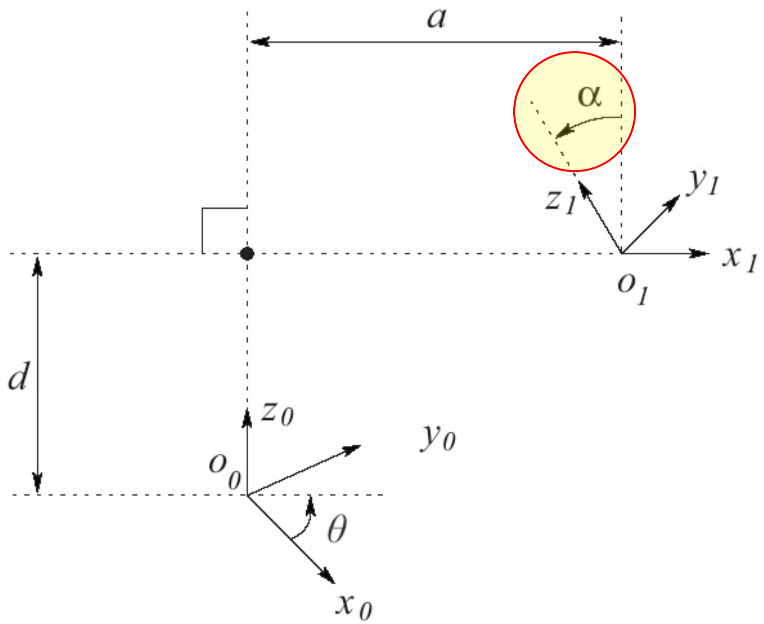
$\boldsymbol{d_i}$ =link offset: distance along z_{i-1} from O_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic

$\boldsymbol{a_i}$= link length: distance along x_i from O_i to the intersection of the x_i and z_{i-1} axes.

$\boldsymbol{\alpha_i}$ = link twist: the angle between z_{i-1} and z_i measured about x_i.

LUND UNIVERSITY

Denavit-Hartenberg convention



$$Rot_{x,\alpha_i}$$

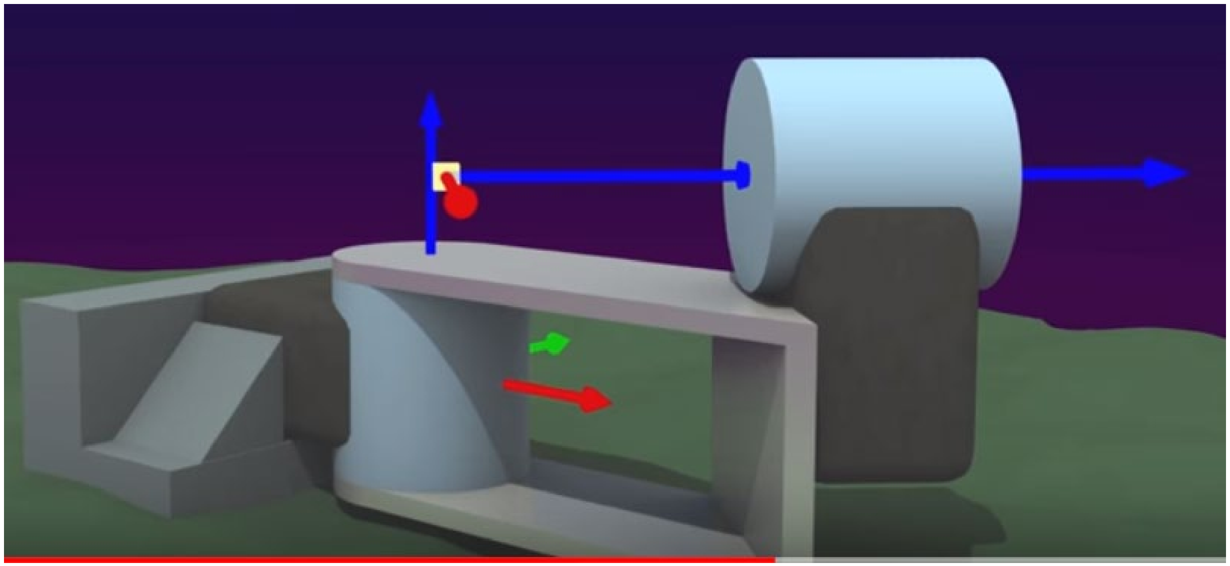
$\boldsymbol{\theta_i}$.= joint angle: the angle between x_{i-1} and x_i measured about z_{i-1}.
θ_i is variable if joint i is revolute.

$\boldsymbol{d_i}$ =link offset: distance along z_{i-1} from O_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic

$\boldsymbol{a_i}$= link length: distance along x_i from O_i to the intersection of the x_i and z_{i-1} axes.

$\boldsymbol{\alpha_i}$ = link twist: the angle between z_{i-1} and z_i measured about x_i.

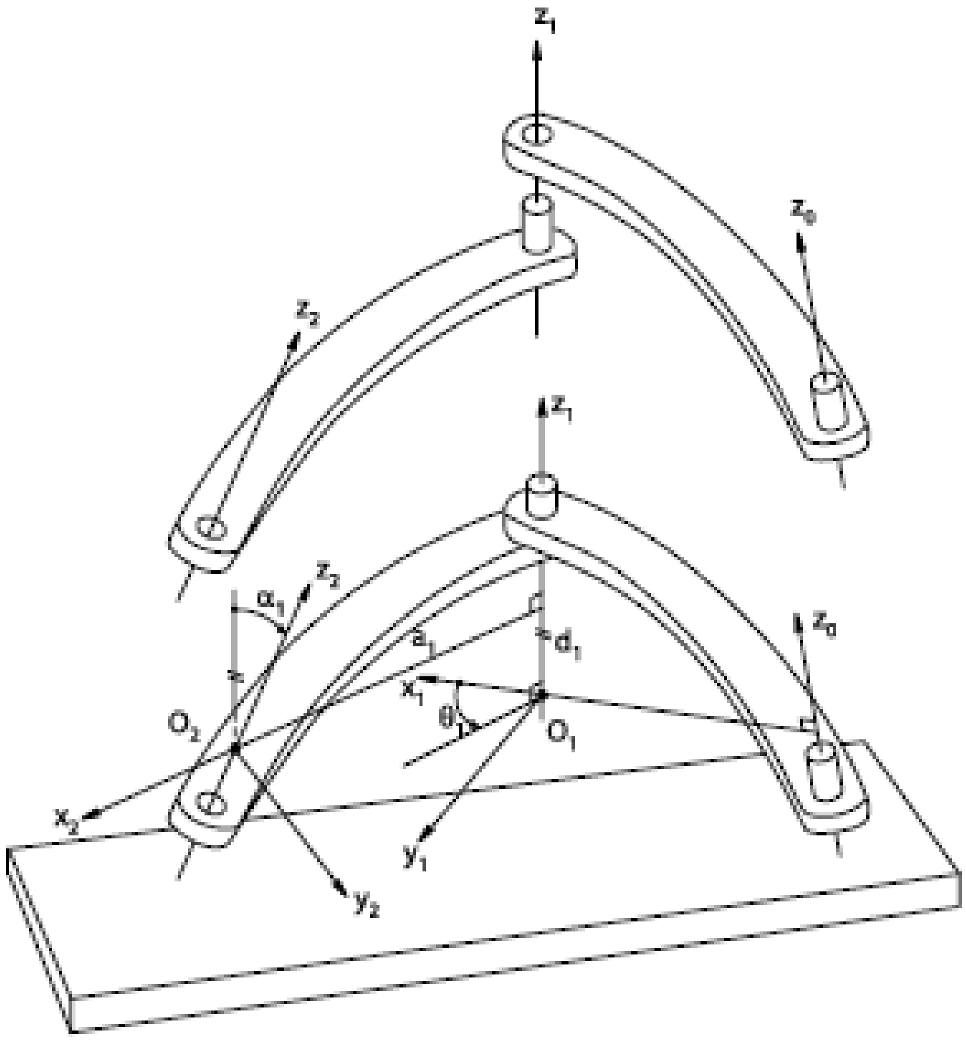
Denavit-Hartenberg



https://www.youtube.com/watch?v=rA9tm0gTIn8

(Good illustration but ’wrong direction/sign’ of theta!)

A General Configuration



LUND UNIVERSITY

D-H Representation: A-matrices

$$\mathrm{A_i} = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

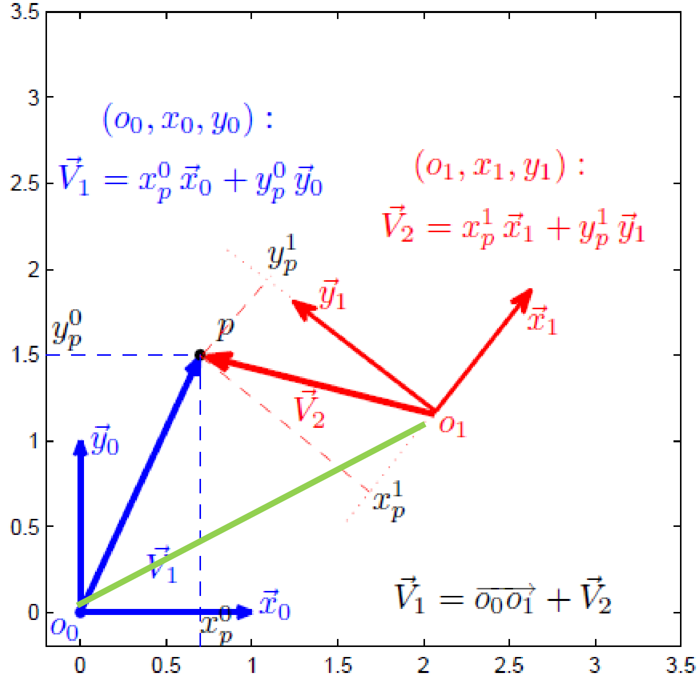
$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

LUND UNIVERSITY

Recap: Homogenous Transformation

Combine both rotation and translation in one matrix



$$\underbrace{\begin{bmatrix} p^0 \\ 1 \end{bmatrix}}_{P^0} = \boxed{\begin{bmatrix} x_p^0 \\ y_p^0 \\ z_p^0 \\ 1 \end{bmatrix} = \begin{bmatrix} (x_1^0)_x & (y_1^0)_x & (z_1^0)_x & (o_1^0)_x \\ (x_1^0)_y & (y_1^0)_y & (z_1^0)_y & (o_1^0)_y \\ (x_1^0)_z & (y_1^0)_z & (z_1^0)_z & (o_1^0)_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p^1 \\ y_p^1 \\ z_p^1 \\ 1 \end{bmatrix}} = \underbrace{\begin{bmatrix} R_1^0 & o_1^0 \\ 0_{1\times 3} & 1 \end{bmatrix}}_{H_1^0} \underbrace{\begin{bmatrix} p^1 \\ 1 \end{bmatrix}}_{P^1}$$

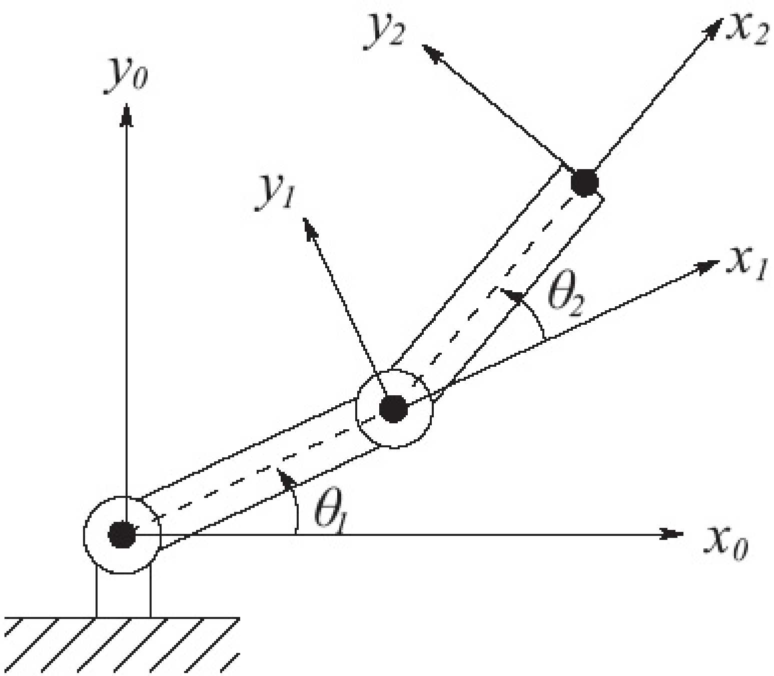
D-H Representation: ORDER MATTERS!

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

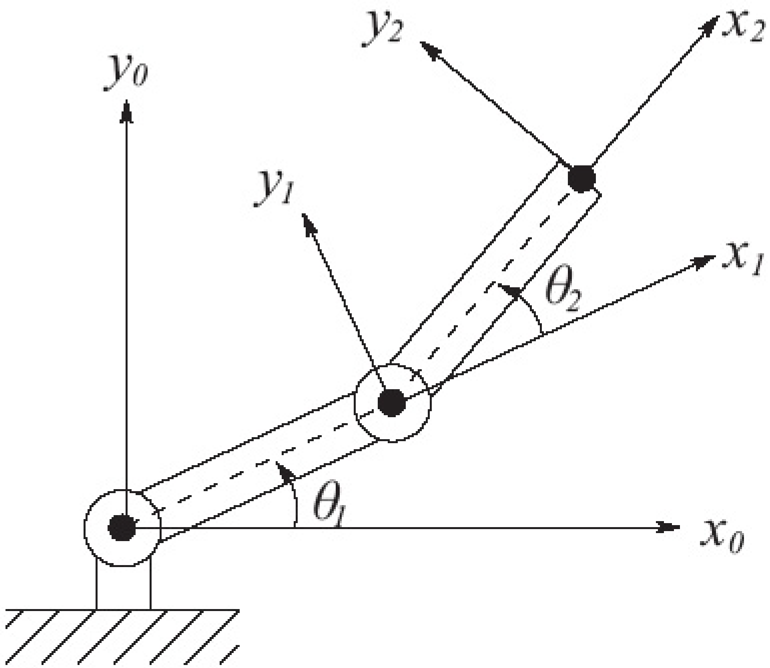
Two-link planar manipulator revisited



Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1^*
2	a_2	0	0	θ_2^*

* variable

Forward kinematics

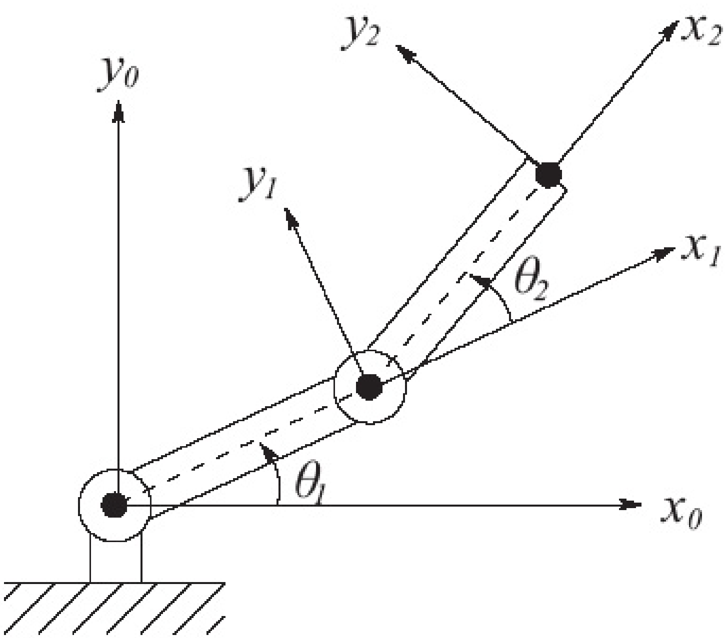


$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1 c_1 \\ s_1 & c_1 & 0 & a_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2 c_2 \\ s_2 & c_2 & 0 & a_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

LUND UNIVERSITY

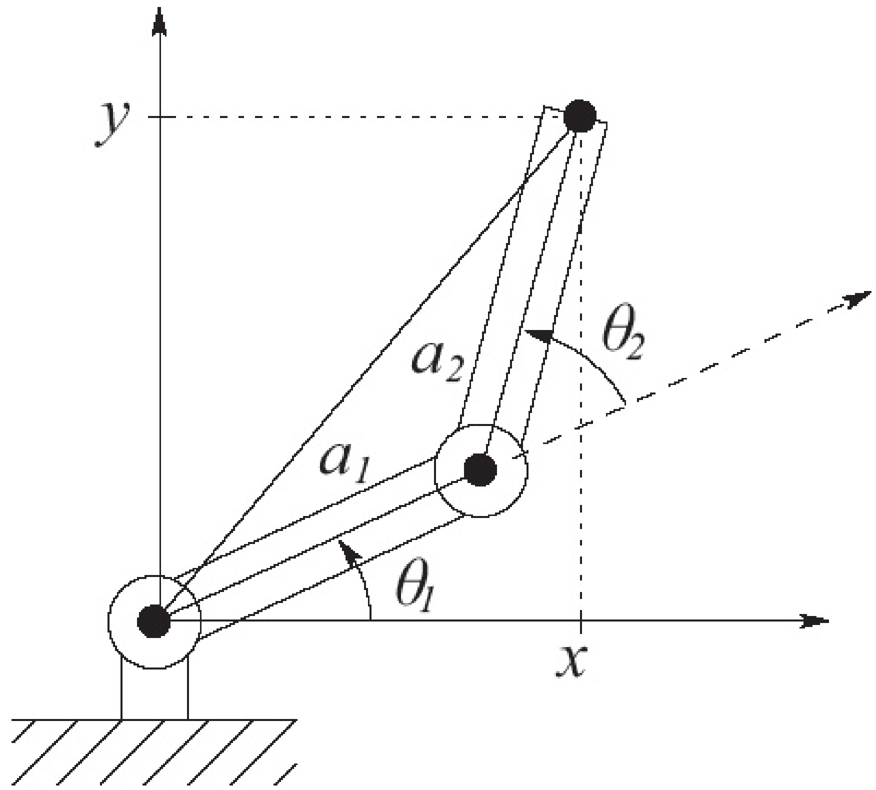
Forward kinematics continued



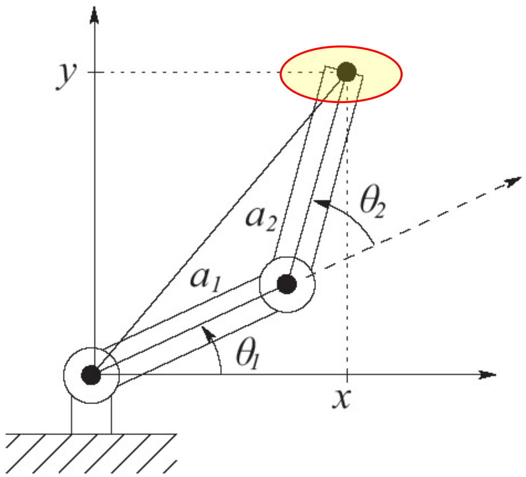
$$T_2^0 = A_1 A_2 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1 c_1 + a_2 c_{12} \\ s_{12} & c_{12} & 0 & a_1 s_1 + a_2 s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Kinematics

Find the values for the joint variables given the position and orientation of the end effector



Inverse kinematics : Analytical Solution



$$T_2^0 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1c_1 + a_2c_{12} \\ s_{12} & c_{12} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_1c_1 + a_2c_{12} \\ a_1s_1 + a_2s_{12} \end{bmatrix}$$

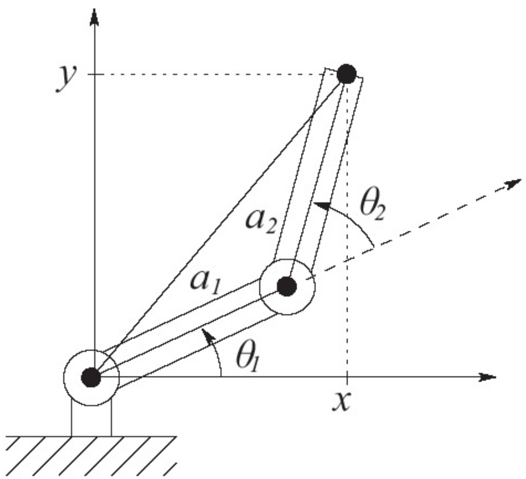
$$x^2 + y^2 = a_1^2 + a_2^2 + 2a_1a_2(\cos\theta_1 \cos\theta_{12} + \sin\theta_1 \sin\theta_{12})$$

$$x^2 + y^2 = a_1^2 + a_2^2 + 2a_1a_2 \cos\theta_2$$

$$\cos\theta_2 = \frac{x^2 + y^2 - a_1^2 + a_2^2}{2a_1a_2}$$

LUND UNIVERSITY

Inverse kinematics: Analytical Solution cont'd



$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \end{bmatrix}$$

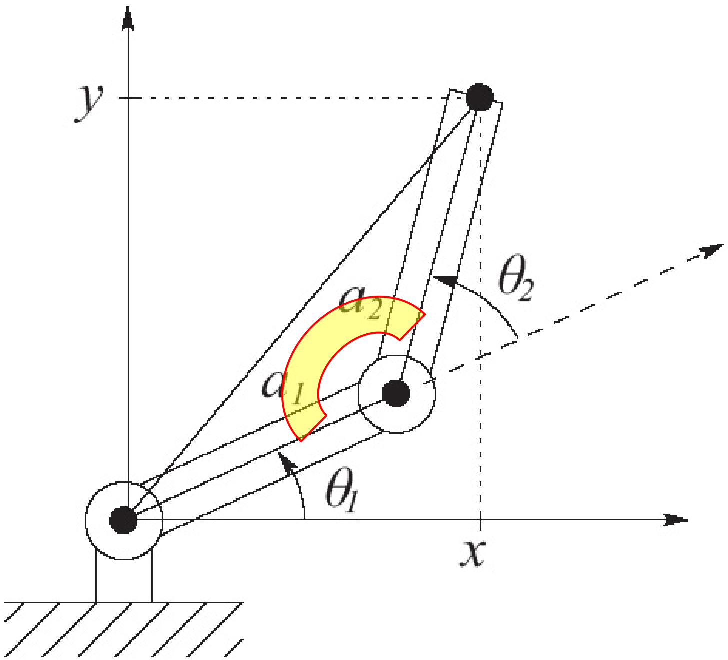
$$\frac{s_{12}}{c_{12}} = \frac{y - a_1 s_1}{x - a_1 c_1}$$

$$\theta_{12} = \arctan\left(\frac{y - a_1 s_1}{x - a_1 c_1}\right)$$

$$\theta_1 = \theta_{12} - \theta_2$$

LUND UNIVERSITY

Inverse kinematics : Geometric Solution



$$x^2 + y^2 = a_1^2 + a_2^2 - 2a_1a_2\cos(\pi - \theta_2)$$

$$\cos\theta_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2}$$

for greater accuracy

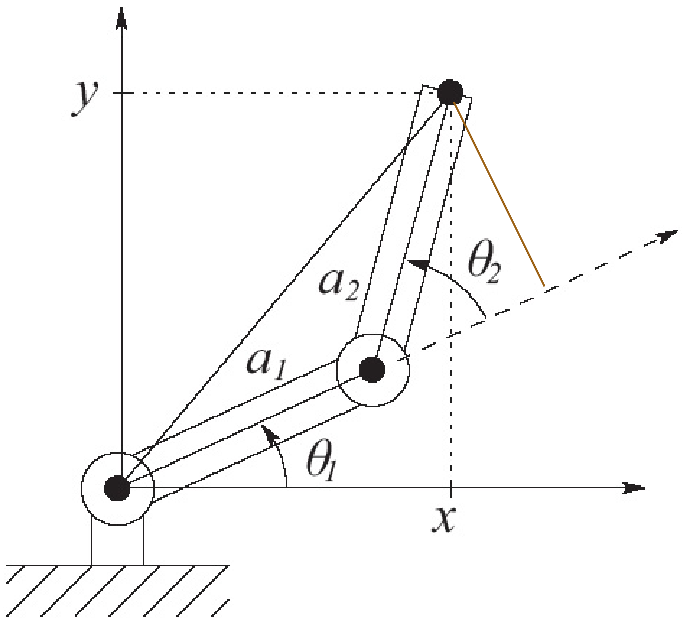
$$\tan^2\frac{\theta_2}{2} = \frac{1-\cos\theta}{1+\cos\theta} = \frac{2a_1a_2 - x^2 - y^2 + a_1^2 + a_2^2}{2a_1a_2 + x^2 + y^2 - a_1^2 - a_2^2}$$

$$= \frac{\left(a_1^2 + a_2^2\right)^2 - \left(x^2 + y^2\right)}{\left(x^2 + y^2\right) - \left(a_1^2 - a_2^2\right)^2}$$

$$\theta_2 = \pm 2\tan^{-1}\sqrt{\frac{\left(a_1^2 + a_2^2\right)^2 - \left(x^2 + y^2\right)}{\left(x^2 + y^2\right) - \left(a_1^2 - a_2^2\right)^2}}$$

LUND UNIVERSITY

Inverse kinematics: Geometric Solution cont'd



Now that if θ_2 is known, we can make use of two triangles to find:

Use **atan2()** instead of **atan()** for having more precision

$$\phi = \text{atan2}(y, x)$$

$$\psi = \text{atan2}(a_2 \sin\theta_2, a_1 + a_2 \cos\theta_2)$$

$$\theta_1 = \phi - \psi$$

LUND UNIVERSITY

Inverse kinematics : Numerical Solution

Closed form (analytical and geometrical) solutions, exist only for certain types of manipulators with simplified geometry.

To solve general cases of manipulators numerical methods can be applied to the inverse kinematics problem

Multiple solutions

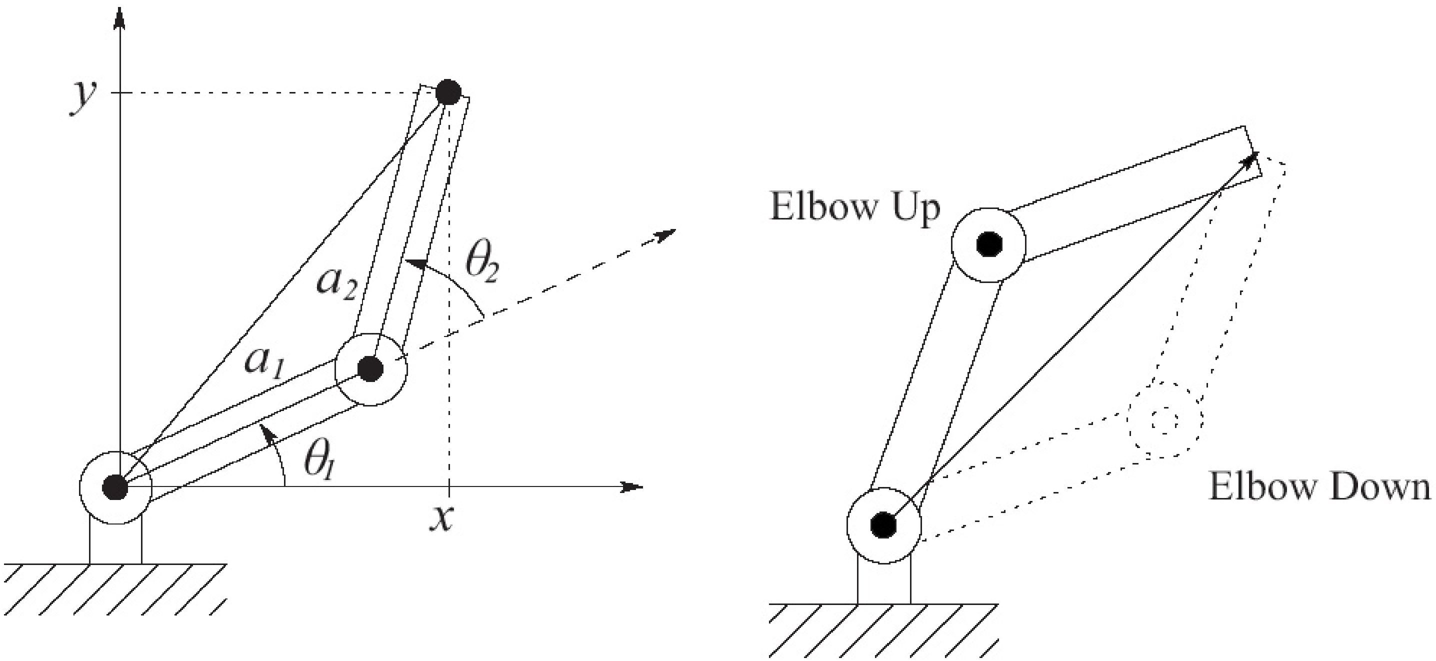


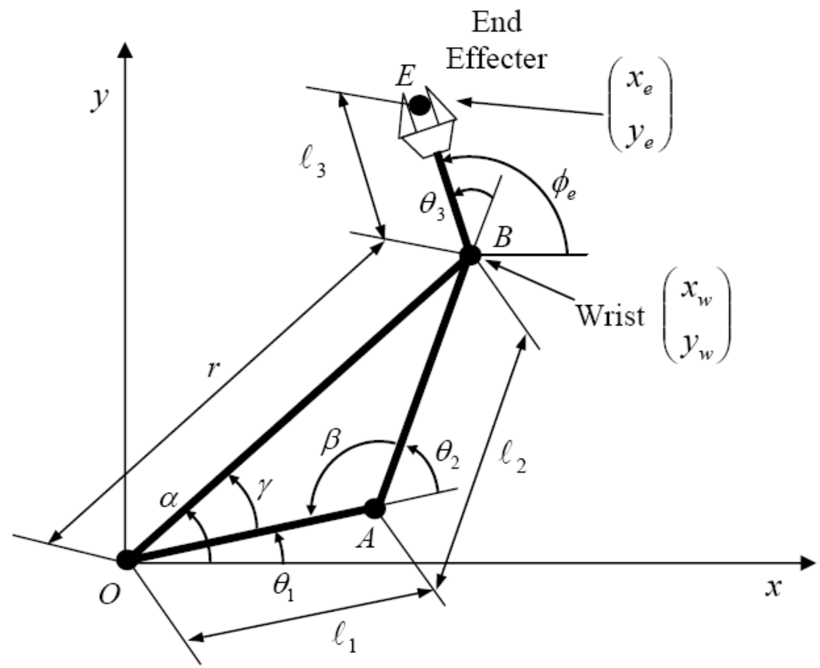
Figure 1.21: The two-link elbow robot has two solutions to the inverse kinematics except at singular configurations, the elbow up solution and the elbow down solution.

LUND UNIVERSITY

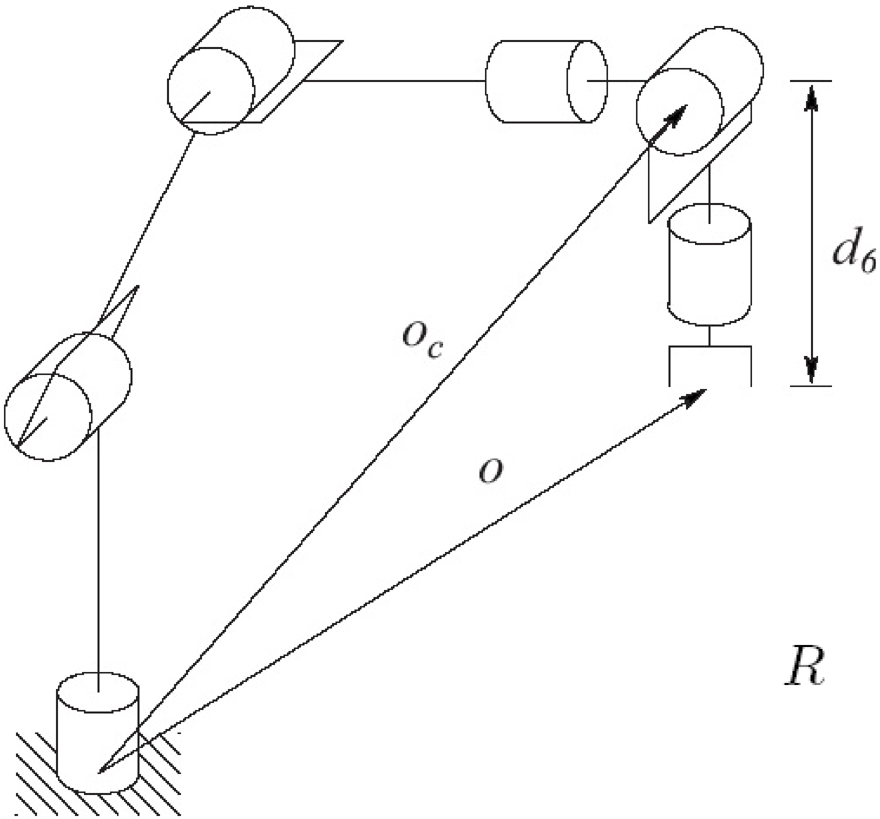
Kinematic decoupling

For manipulators with six joints with the last three joints intersecting at a point it is possible to decouple the inverse kinematics problem into two simpler problems:

- Inverse position kinematics
- Inverse orientation kinematics



Kinematic decoupling



$$o_c^0 = o - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}$$

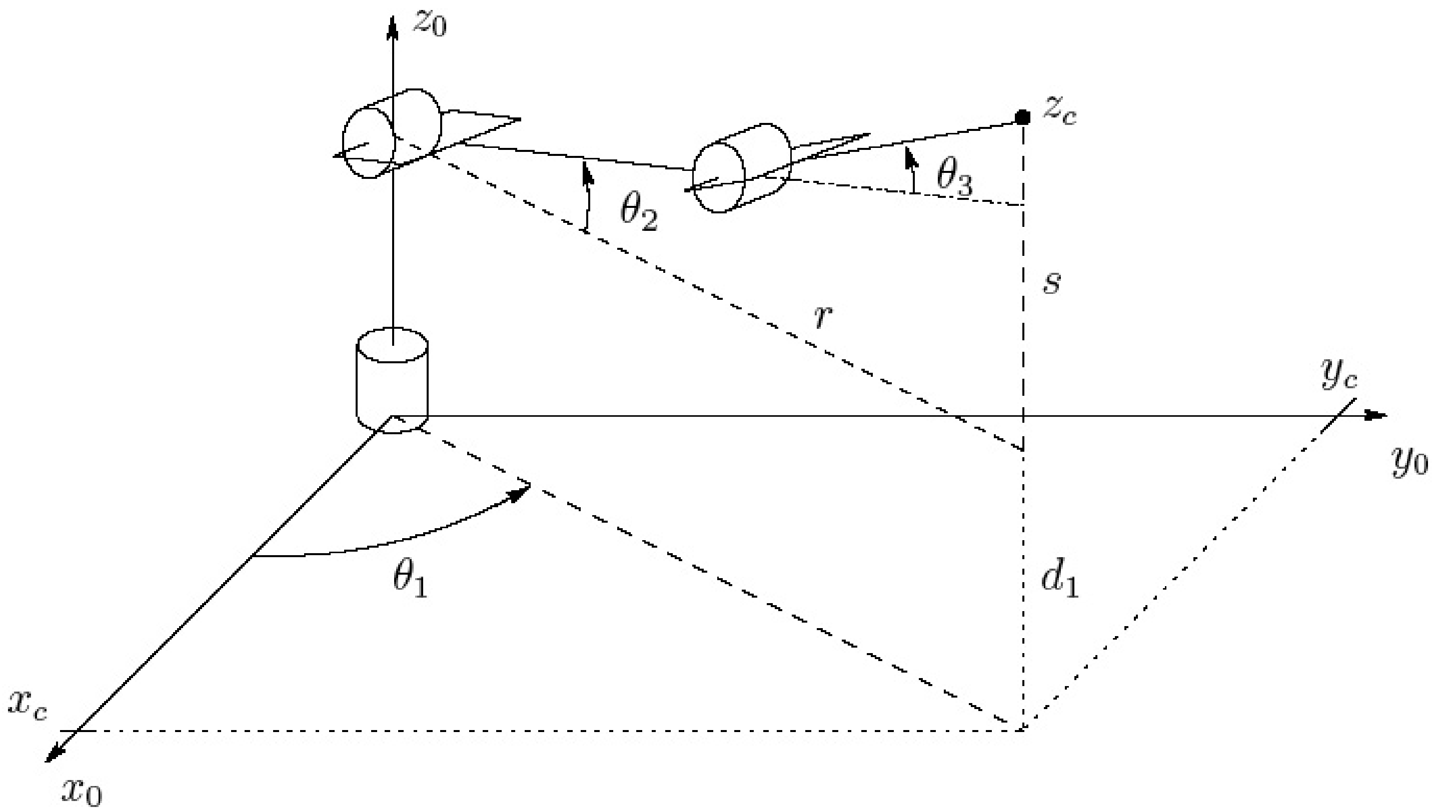
wrist center: with the last three joints intersecting

$$R = R_3^0 R_6^3$$

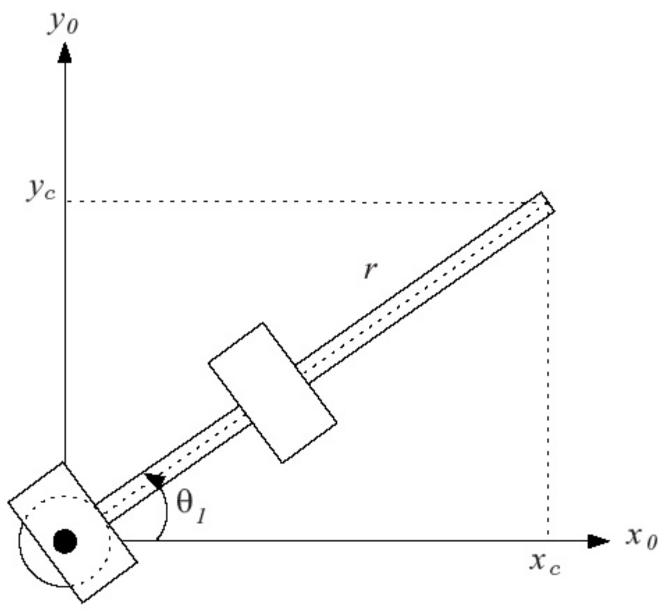
$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R.$$

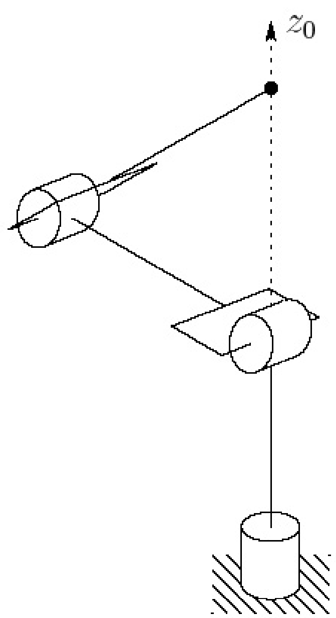
LUND UNIVERSITY

Inverse kinematics

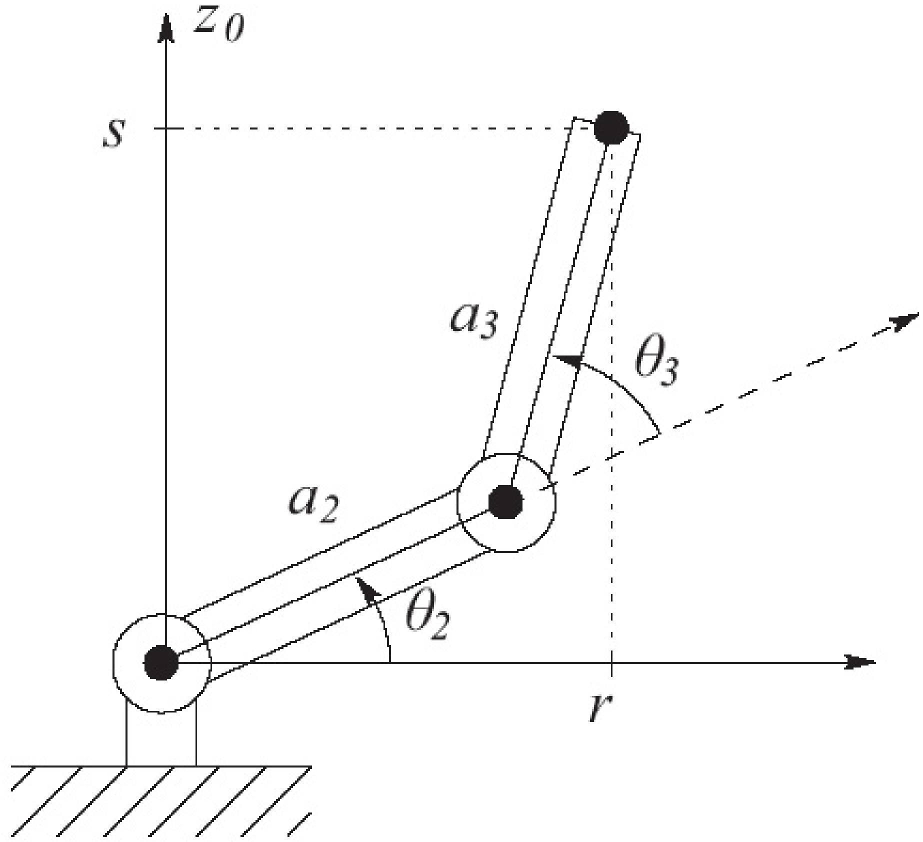


Inverse position: First Joint Angle



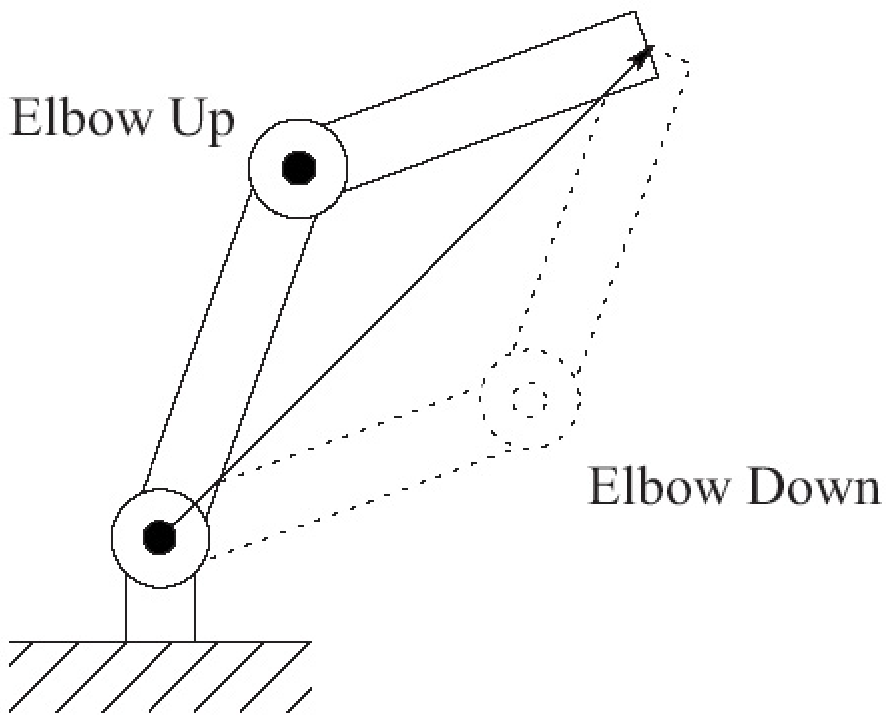


Inverse Position: Joint Angles 2 & 3



Projecting onto the plane formed by links 2 and 3

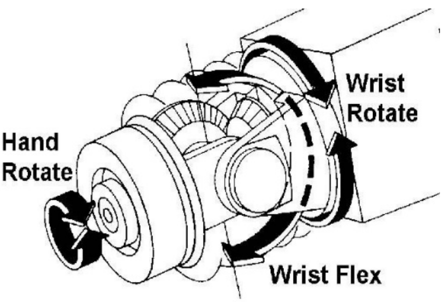
Inverse kinematics



a_i

Inverse orientation: spherical wrist

$$R_6^3 = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 \\ -s_5c_6 & s_5s_6 & c_5 \end{bmatrix}$$



$$\begin{bmatrix} c\varphi c\theta c\psi - s\varphi s\psi & -c\varphi c\theta s\psi - s\varphi c\psi & c\varphi s\theta \\ s\varphi c\theta c\psi + c\varphi s\psi & -s\varphi c\theta s\psi + c\varphi c\psi & s\varphi s\theta \\ -s\theta c\psi & s\theta s\psi & c\theta \end{bmatrix}$$

Velocity kinematics

Velocities can be expressed in either the Cartesian or the Joint space

$$D = \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} \text{ Cartesian Velocities where } \upsilon = \begin{bmatrix} \upsilon_x \\ \upsilon_y \\ \upsilon_z \end{bmatrix} \text{ and } \omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

$$\dot{Q} = \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} \quad \text{Joint Velocities}$$

- Velocities can only be added if they are defined in the same space
- Motion of the end effector (n frame) is taken with respect to the base space (0 frame)
- Linear Velocity effects are physically separable from angular velocity effects

The Jacobian of the manipulator

We seek expressions of the form $$\xi = J\dot{q}$$

where vector ξ is the ***body velocity,***

J is the ***manipulator Jacobian*** or ***Jacobian*** for short

and $\dot{q}$ is the vector of ***joint velocities***

The Derivative of a Rotation Matrix

$$R(\theta)R(\theta)^T = I$$

R is orthogonal

$$\left[\frac{d}{d\theta}R\right]R(\theta)^T + R(\theta)\left[\frac{d}{d\theta}R^T\right] = 0$$

Differentiate

$$S = \left[\frac{d}{d\theta}R\right]R(\theta)^T$$

Define *S* as:

$$S^T = \left(\left[\frac{d}{d\theta}R\right]R(\theta)^T\right)^T = R(\theta)\left[\frac{d}{d\theta}R\right]^T$$

$$(AB)^T = B^T A^T$$

$$S + S^T = 0$$

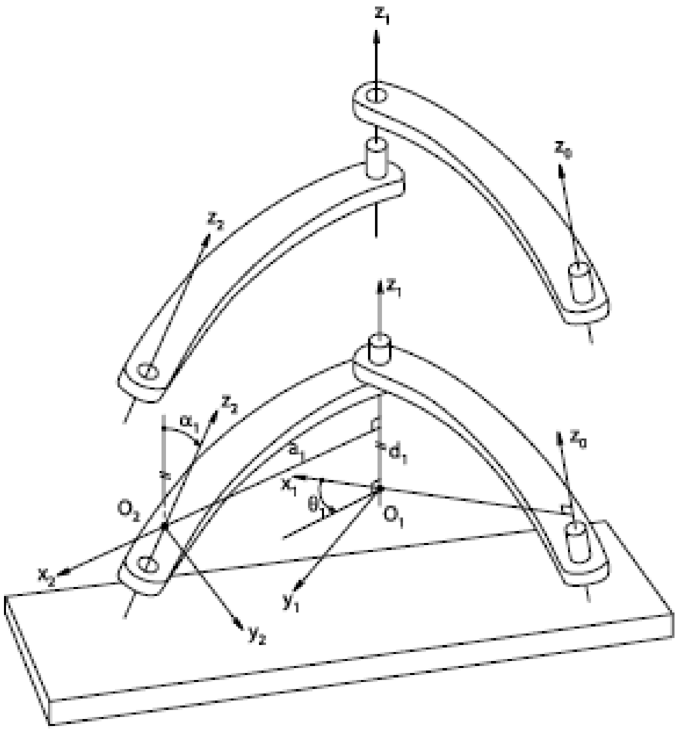
Substitute in (2)-> S is skew symmetric

$$\frac{d}{d\theta}R = SR(\theta)$$

The derivative of a Rotation Matrix is derived by matrix multiplication by a skew symmetric matrix S

Addition of Angular Velocities

$$\omega_{0,2}^{0} = \omega_{0,1}^{0} + R_1^0 \omega_{1,2}^{1}$$



Linear Velocity

Linear Velocity of a Point Attached to a Moving Frame

$$\dot{p}^0 = \omega \times r + \upsilon$$

Derivation of the Jacobian

$$T_n^0 = \begin{bmatrix} R_n^0(q) & o_n^0(q) \\ 0 & 1 \end{bmatrix} \text{where } q = [q_1, \ldots, q_n] \text{ is the vector of joint variables}$$

$$S(\omega_n^0) = \dot{R}_n^0 (R_n^0)^T \text{ defines the angular velocity vector } \omega_n^0$$

$$\upsilon_n^0 = \dot{o}_n^0 \text{ the linear velocity}$$

We seek expressions for:

$$\upsilon_n^0 = J_\upsilon \dot{q}$$

$$\omega_n^0 = J_\omega \dot{q}$$

LUND UNIVERSITY

Angular velocities

Prismatic Joint: $\omega_i^{i-1} = 0$

Revolute Joint: $\omega_i^{i-1} = \dot{q}_i z_{i-1}^{i-1} = \dot{q}_i \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$

$$J_\omega = \left[\rho_1 z_0 \ldots \rho_n z_{n-1}\right] \text{where} \begin{array}{l} \rho_i = 1 \text{ if joint } i \text{ is revolute} \\ \rho_i = 0 \text{ if joint } i \text{ is prismatic} \end{array}$$

Velocities: Prismatic Joints

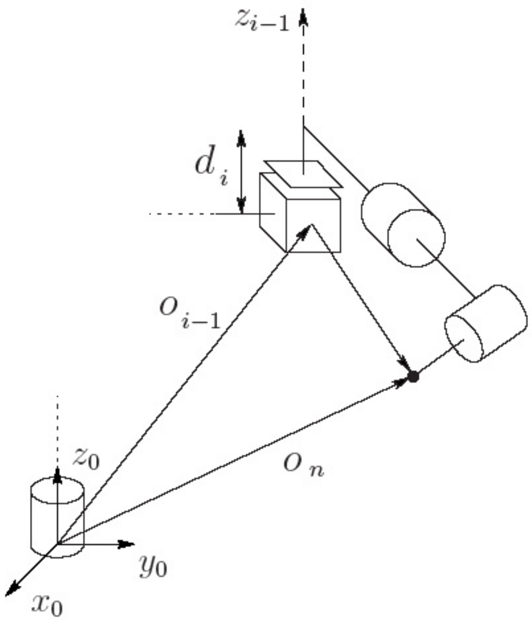


Figure 4.1: Motion of the end effector due to primsmatic joint i.

$$\omega_i^{i-1} = 0$$

$$\dot{o}_i^{i-1} = \dot{d}_i \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_{\upsilon_i} = z_{i-1}$$

Velocities: Revolute Joints

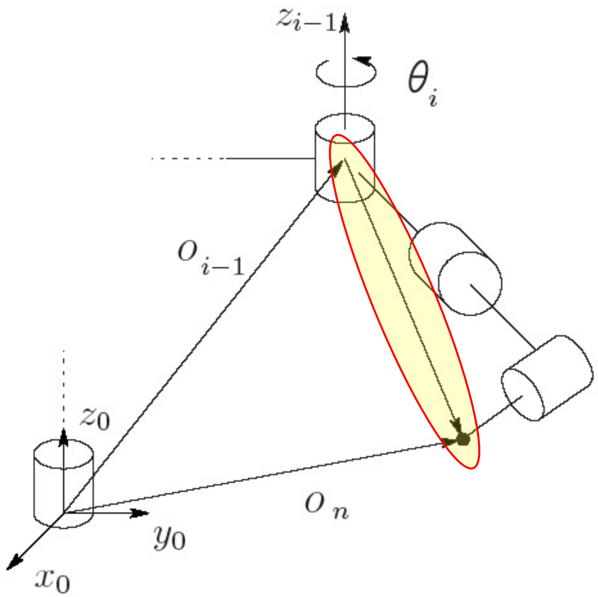


Figure 4.2: Motion of the end effector due to revolute joint i.

$$\omega_i^{i-1} = \dot{\theta}_i z_{i-1} = \dot{\theta}_i \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\dot{o}_i^{i-1} = \omega \times r \quad \text{where} \quad r = o_i - o_{i-1}$$

$$J_{\upsilon_i} = z_{i-1} \times (o_n - o_{i-1})$$

LUND UNIVERSITY

Transforming velocities

The two frames O_0 and O_1 are related by the homogeneous transformation

$$H = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If a particle has velocity $v_1(t)$ relative to O_1 what is the velocity relative to O_0 ?

$$v_1(t) = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$$

$$v_0(t) = ?$$

Transforming velocities

$$
\begin{aligned}
p_0 &= Rp_1 + d \\
\dot{p}_0 &= R\dot{p}_1 \\
&= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ 0 \end{bmatrix}
\end{aligned}
$$

$Note: If\ using\ homoeneous\ transforms\ v_1(t) = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$

Singularities

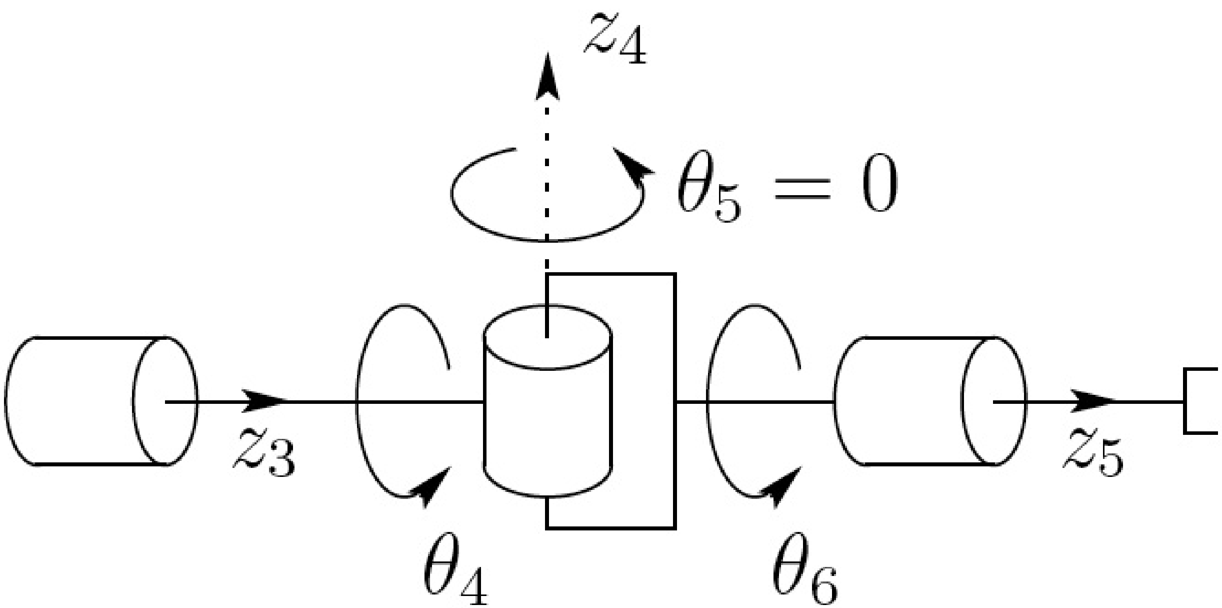


Figure 4.4: Spherical wrist singularity.

- Beware of the ”singularity”!

Singularities

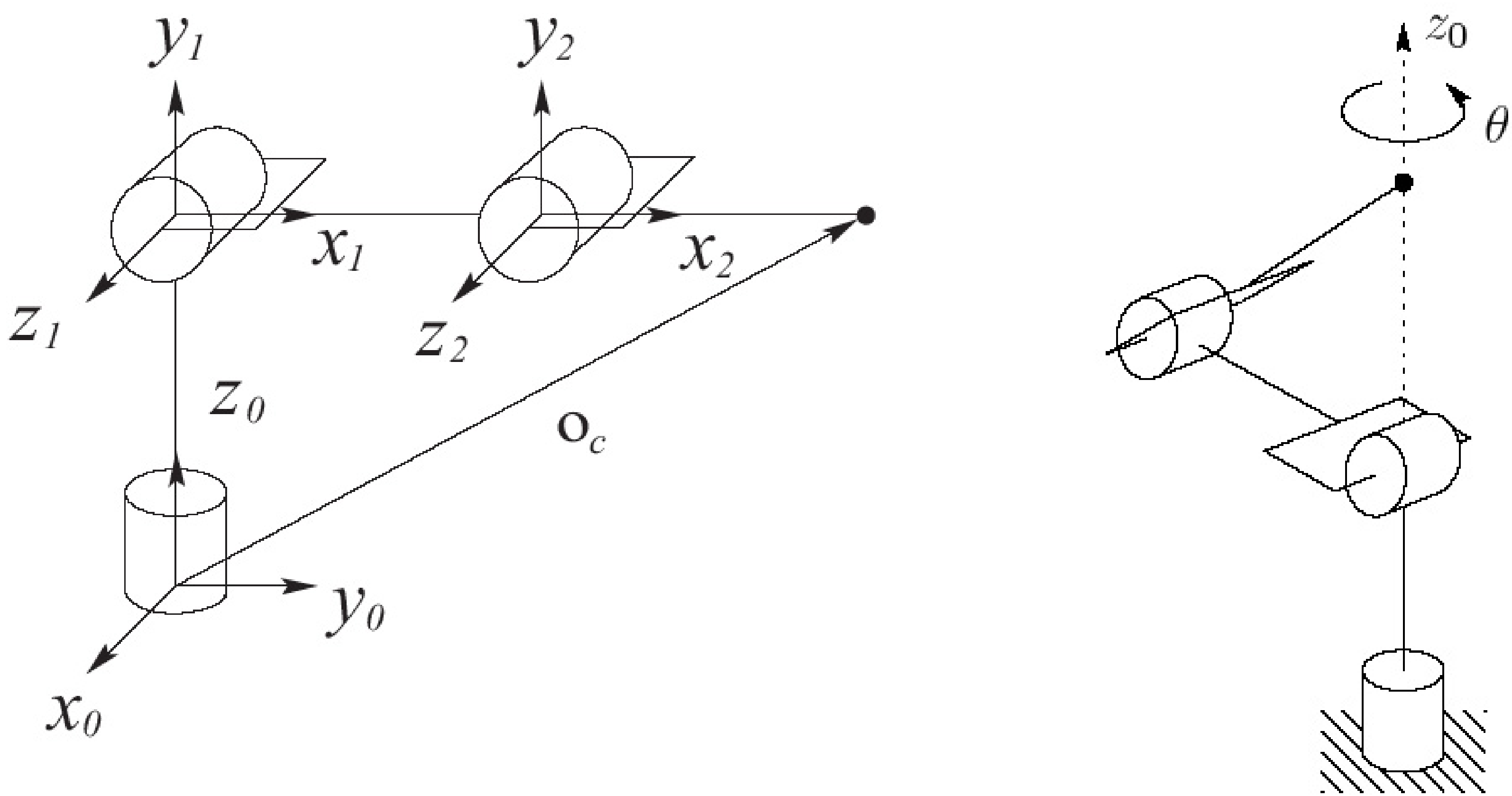


Figure 4.5: Elbow manipulator.

Singularities

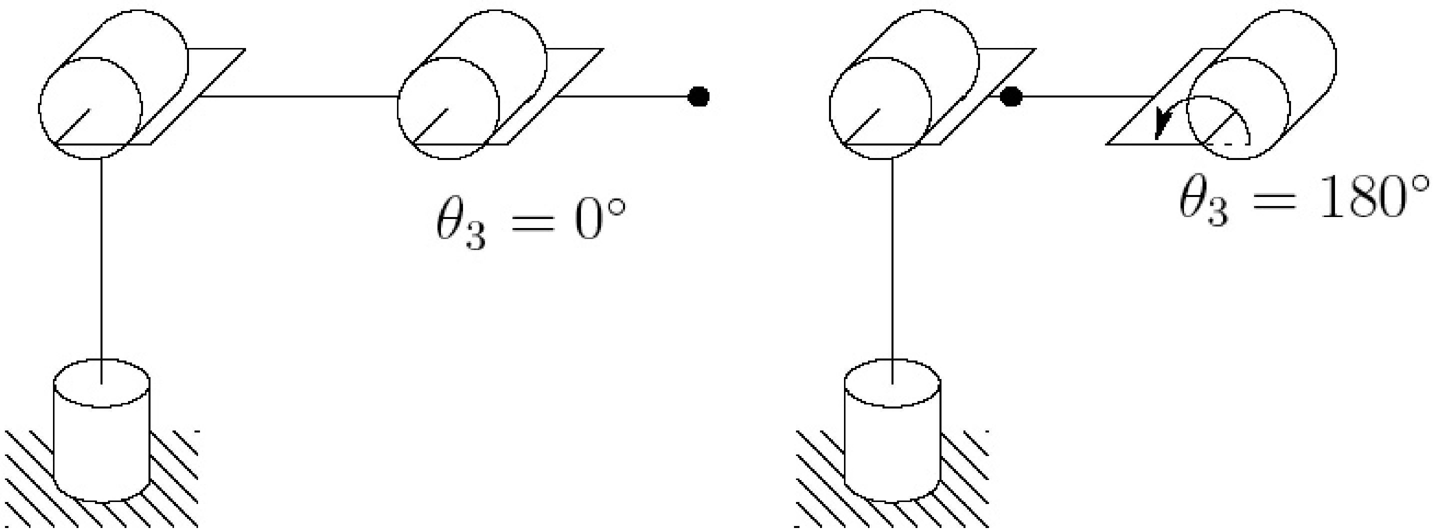
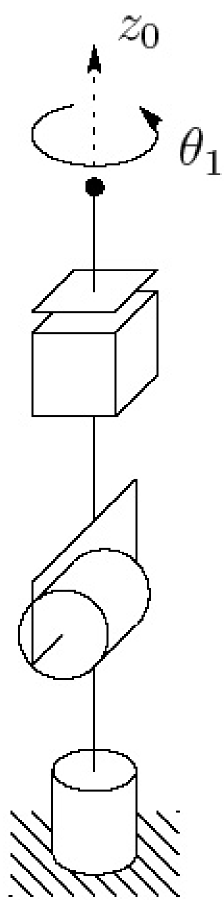


Figure 4.6: Elbow singularities of the elbow manipulator.

Singularities



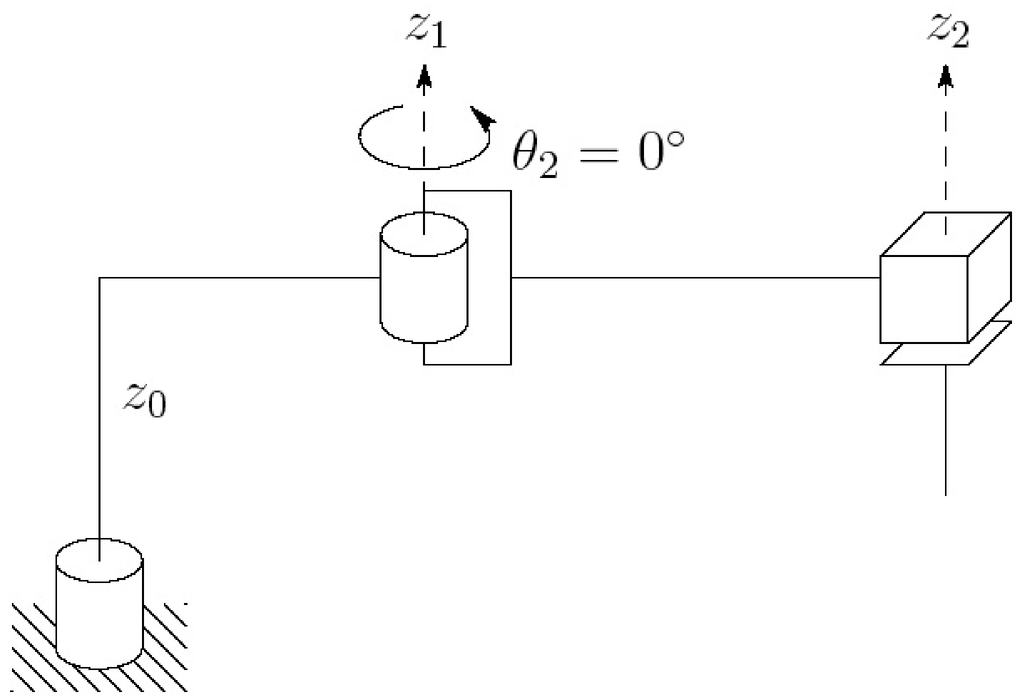


Figure 4.10: SCARA manipulator singularity.

Inverse velocity and acceleration

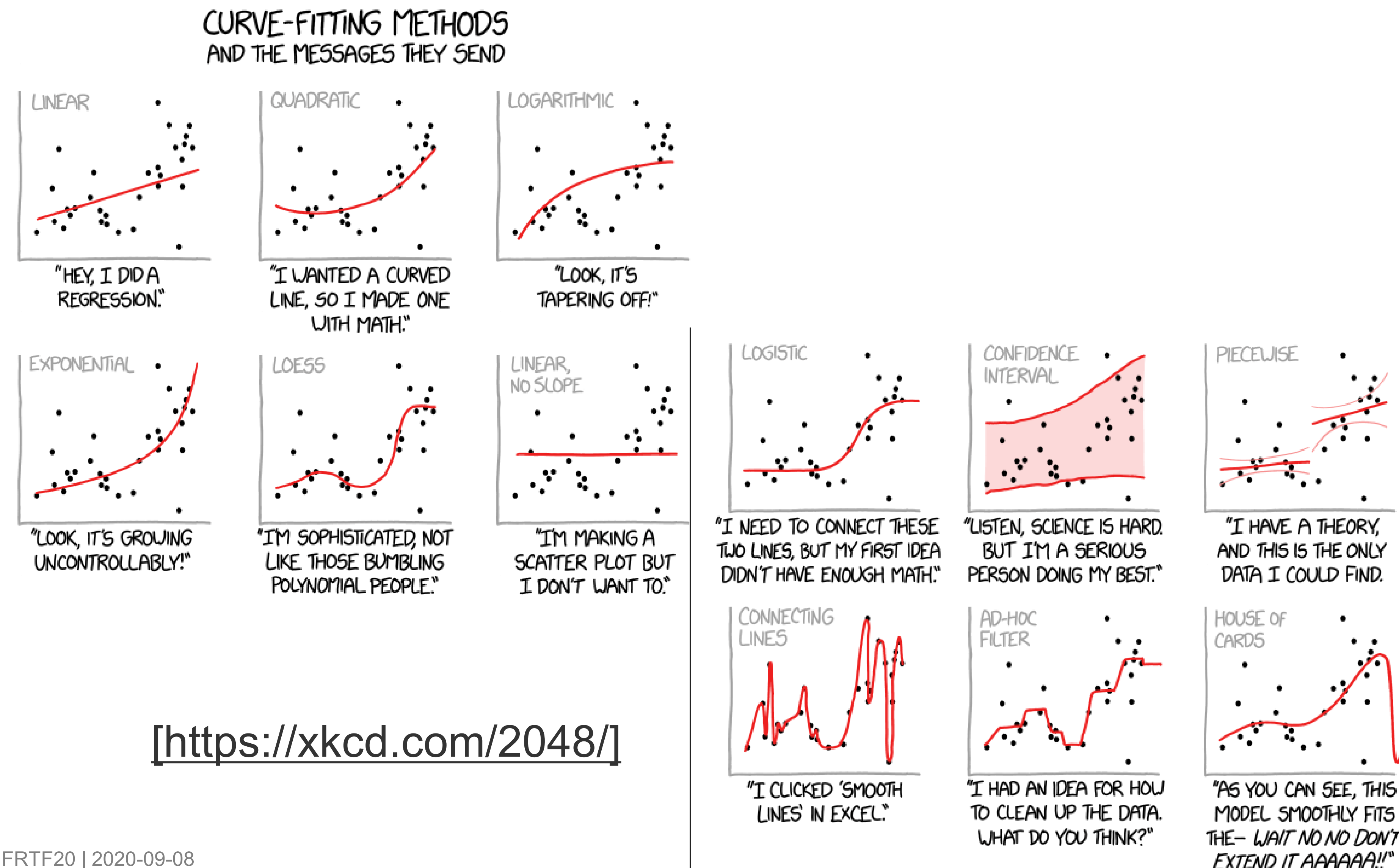
When the Jacobian is square and nonsingular (manipulators with 6 joints):

$$\dot{q} = J_\alpha(q)^{-1}\dot{X}$$

$$\ddot{q} = J_\alpha(q)^{-1}\left[\ddot{X} - \left(\frac{d}{dt}J_\alpha(q)\right)\dot{q}\right]$$

LUND UNIVERSITY

Curves, paths and trajectories



[https://xkcd.com/2048/]

Path and Trajectory Planning

A ***path*** from q_s to q_f in configuration space is defined as a continuous map

$$\gamma : [0,1] \rightarrow Q, \text{ with } \gamma(0) = q_s \text{ and } \gamma(1) = q_f$$

A ***path*** is a geometric description of motion
(positions and orientations)

A ***trajectory*** is a function of time from $q(t)$ such that

$$q(t_0) = q_s \text{ and } q(t_f) = q_f$$

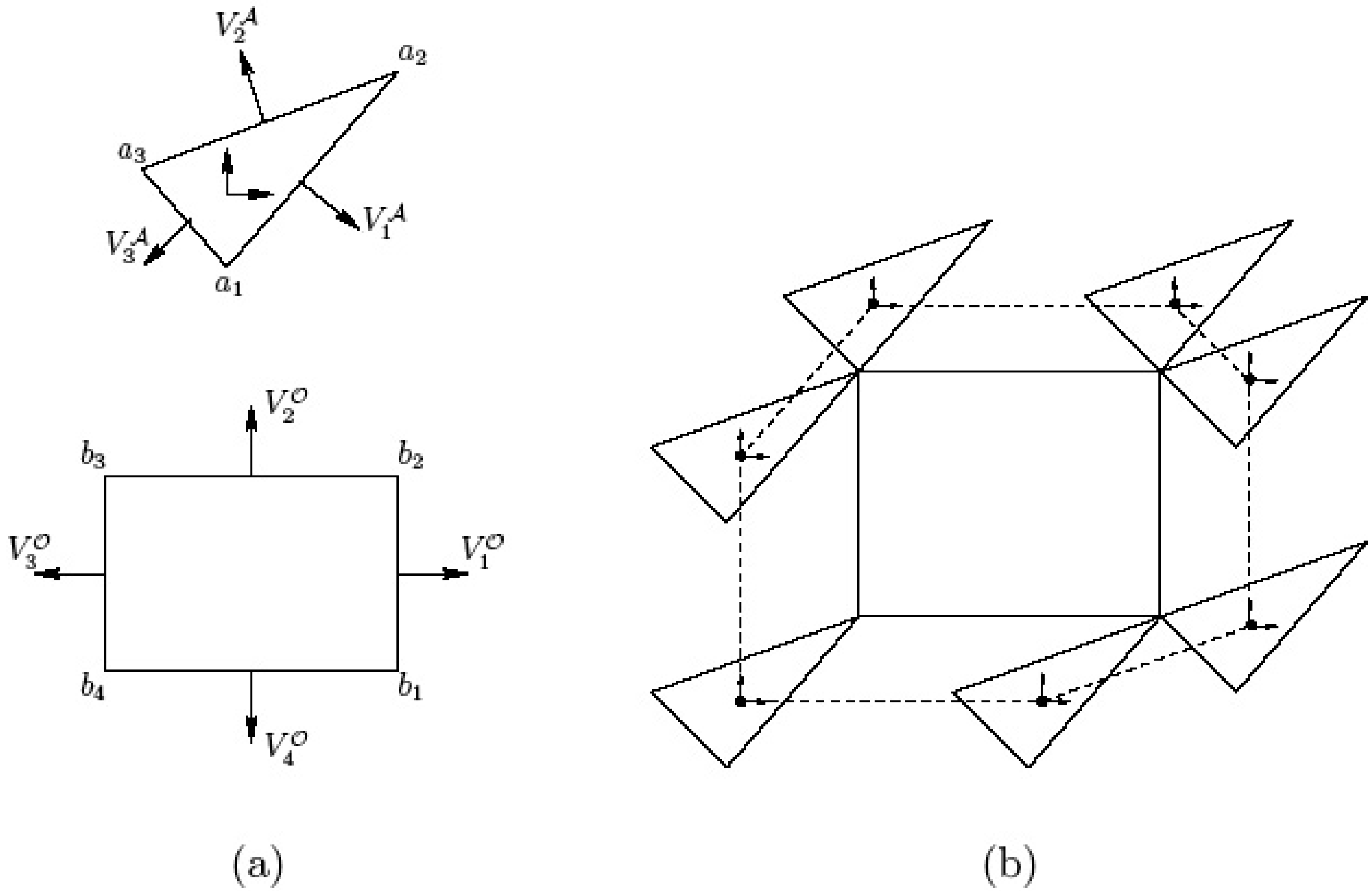
A ***trajectory*** is a dynamic description of motion
(velocities and accelerations)

Path Planning

Assuming that the initial and final configurations of the robot are known, find a collision free path connecting these configurations

1. Configuration Space
2. Artificial Potential Fields
3. Probabilistic Roadmap

A Rigid Body that Translates in Plane



LUND UNIVERSITY

Two-Link Planar Arm

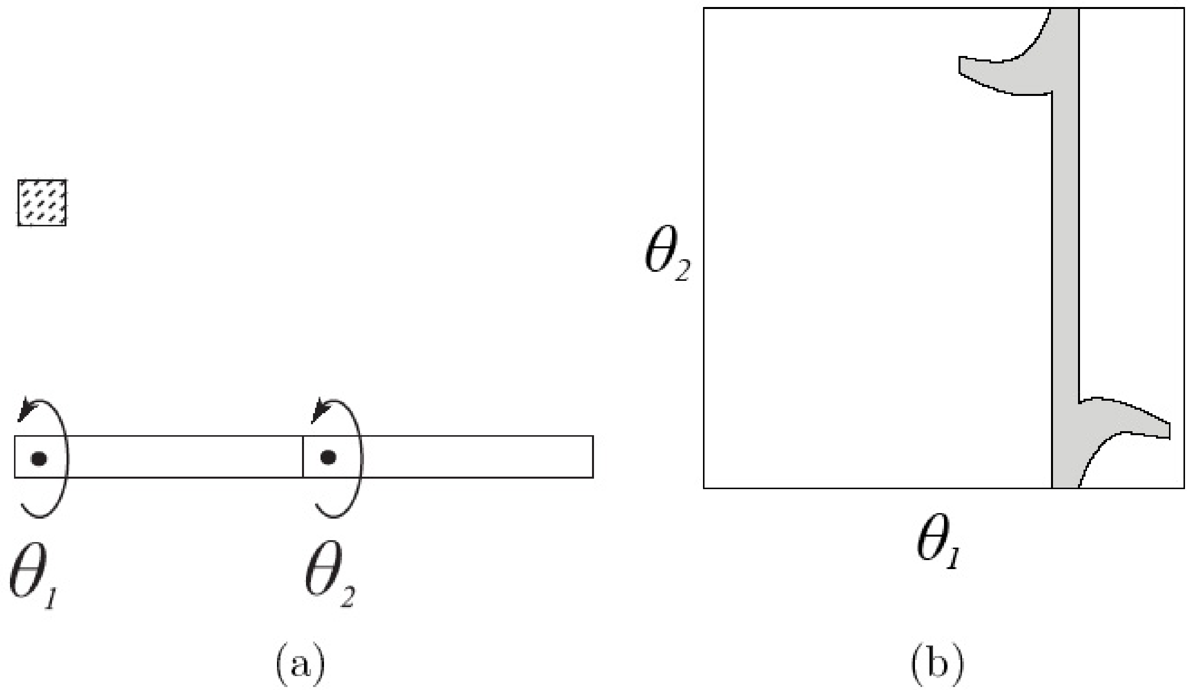
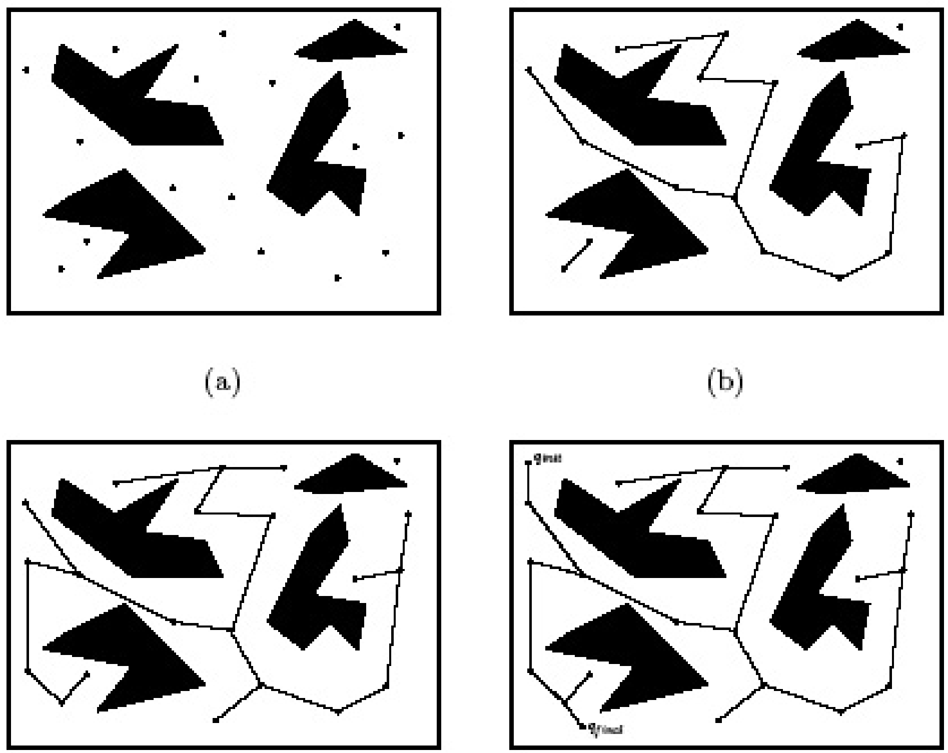


Figure 5.2: (a) The robot is a two-link planar arm and the workspace contains a single, small polygonal obstacle. (b) The corresponding configuration space obstacle region contains all configurations $q = (\theta_1, \theta_2)$ such that the arm at configuration q intersects the obstacle.

Potential Fields

- Treat the robot as a point particle
- Define an attractive potential field based on the goal field
- Define repulsive potential fields on all obstacles
- Use a gradient descent algorithm to find the path

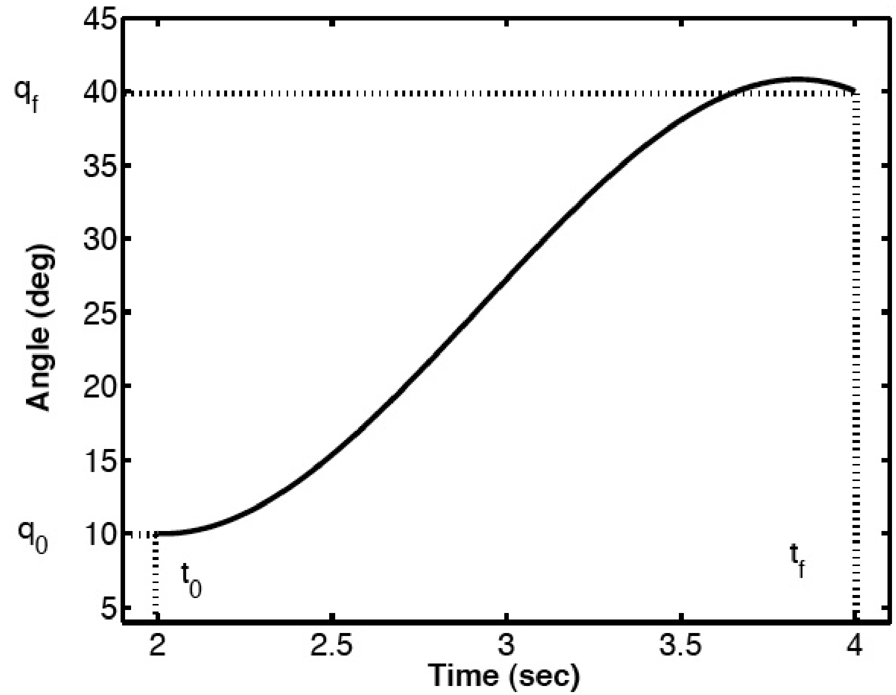
Probabilistic Roadmap methods



(a) (b) (c) (d)

Trajectory Planning

A trajectory is a function of time $q(t)$ such that $q(t_0) = q_s$ and $q(t_f) = q_f$



LUND UNIVERSITY

Trajectory Planning

5.2 PATH VS. TRAJECTORY

- Path: A sequence of robot configurations in a particular order without regard to the timing of these configurations.
- Trajectory: It concerned about when each part of the path must be attained, thus specifying timing.

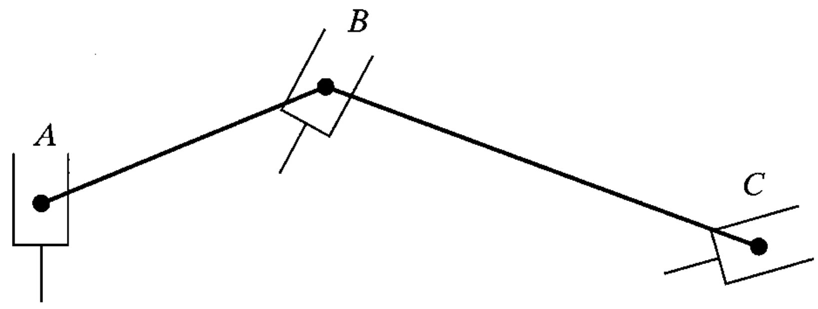
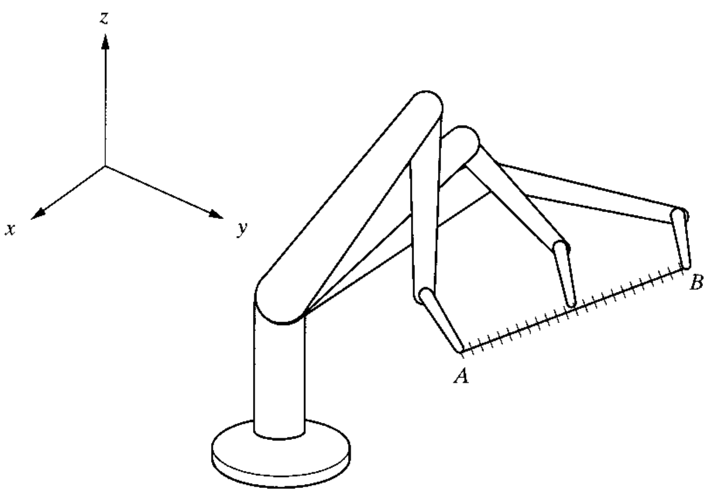


Fig. 5.1 Sequential robot movements in a path.

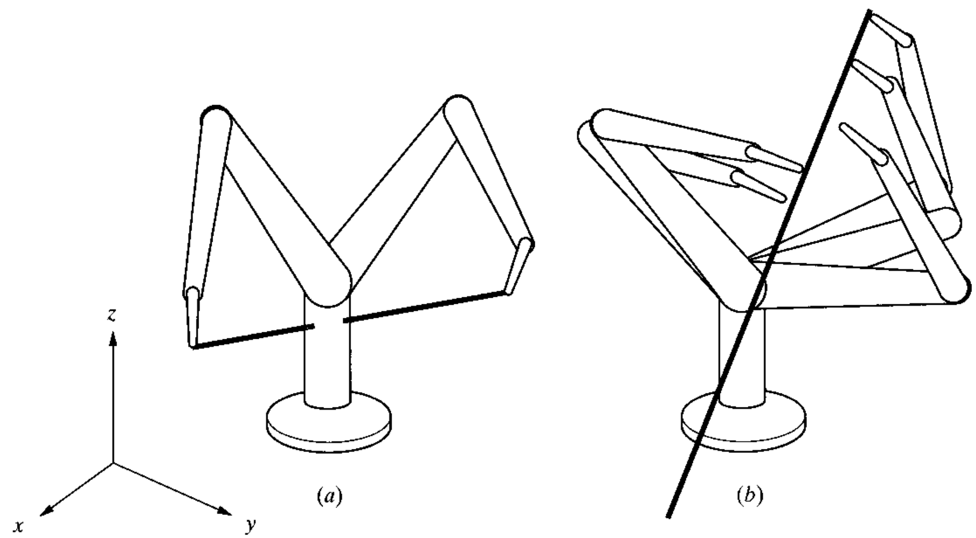
LUND UNIVERSITY

JOINT-SPACE VS. CARTESIAN-SPACE DESCRIPTIONS

- Joint-space description:
  - The description of the motion to be made by the robot by its joint values.
  - The motion between the two points is unpredictable.
- Cartesian space description:
  - The motion between the two points is known at all times and controllable.
  - It is easy to visualize the trajectory, but it is difficult to ensure that it is singularity free.



Sequential motions of a robot to follow a straight line.

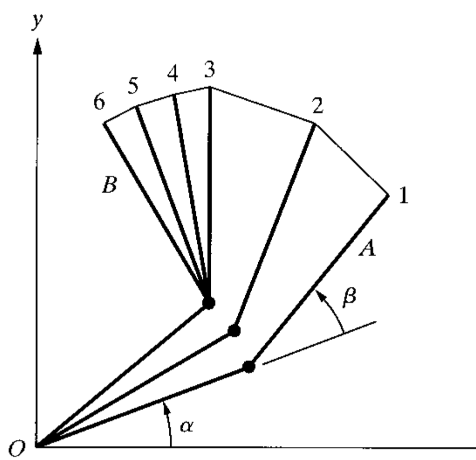


Cartesian-space trajectory (a) The trajectory specified in Cartesian coordinates may force the robot to run into itself, and (b) the trajectory may requires a sudden change in the joint angles.

LUND UNIVERSITY

BASICS OF TRAJECTORY PLANNING

- Let's consider a simple 2 degree of freedom robot.
- We desire to move the robot from Point A to Point B.
- Let's assume that both joints of the robot can move at the maximum rate of 10 degree/sec.
- Let's assume that both joints of the robot can move at the maximum rate of 10 degree/sec.

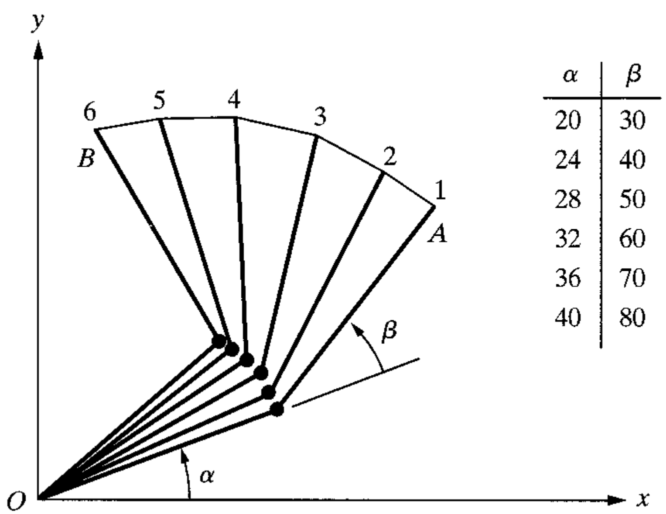


α	β
20	30
30	40
40	50
40	60
40	70
40	80

Joint-space nonnormalized movements of a robot with two degrees of freedom.

- **Move the robot from A to B, to run both joints at their maximum angular velocities.**
- **After 2 [sec], the lower link will have finished its motion, while the upper link continues for another 3 [sec].**
- **The path is irregular and the distances traveled by the robot's end are not uniform.**

- Let's assume that the motions of both joints are normalized by a <u>common factor</u> such that the joint with smaller motion will move proportionally slower and the both joints will start and stop their motion simultaneously.



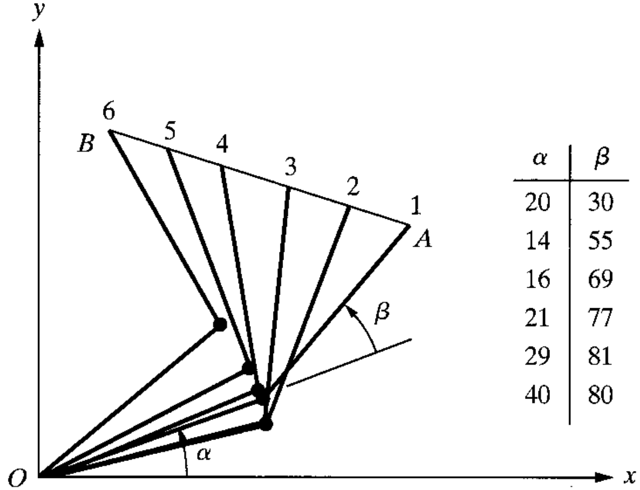
α	β
20	30
24	40
28	50
32	60
36	70
40	80

Fig. 5.5 Joint-space, normalized movements of a robot with two degrees of freedom.

- **Both joints move at different speeds, but move continuously together.**
- **The resulting trajectory will be different.**

BASICS OF TRAJECTORY PLANNING

- The simplest solution would be to draw a line between points A and B, so called interpolation.

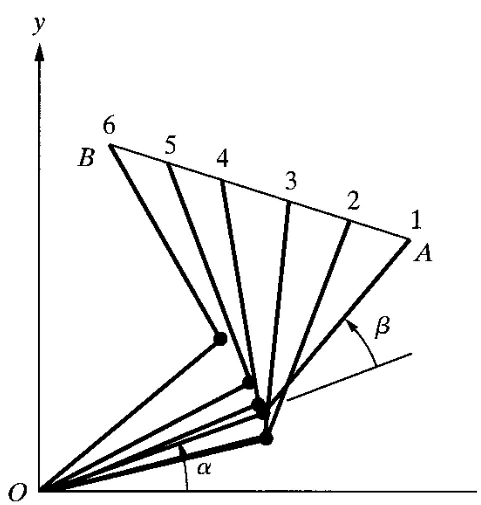


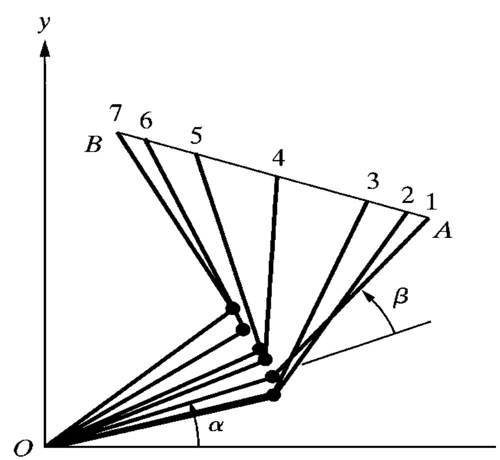
α	β
20	30
14	55
16	69
21	77
29	81
40	80

Fig. 5.6 Cartesian-space movements of a two-degree-of-freedom robot.

- **Divide the line into five segments and solve for necessary angles α and β at each point.**
- **The joint angles are not uniformly changing.**

Overview





BASICS OF TRAJECTORY PLANNING

- Let's assume that the robot's hand follow a known path between point A to B with straight line.
- The simplest solution would be to draw a line between points A and B, so called interpolation.

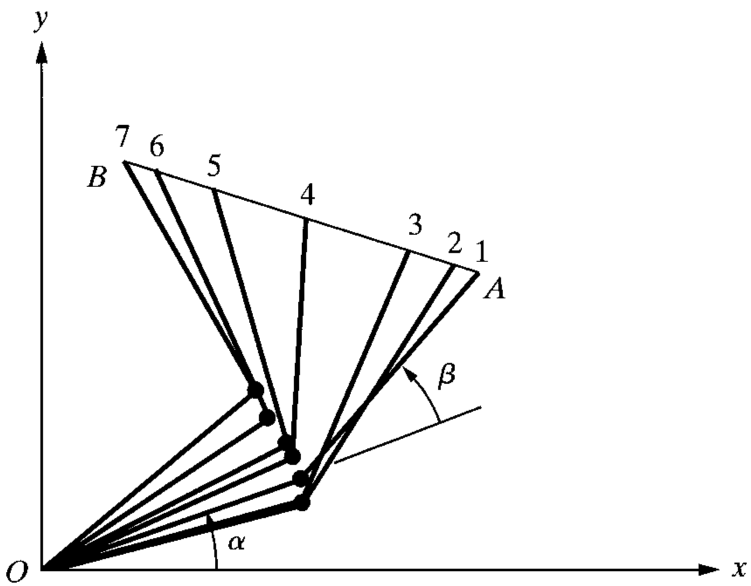


Fig. 5.7 Trajectory planning with an acceleration-deceleration regiment.

- **It is assumed that the robot's actuators are strong enough to provide large forces necessary to accelerate and decelerate the joints as needed.**
- **Divide the segments differently.**
  - **The arm move at smaller segments as we speed up at the beginning.**
  - Go at a constant cruising rate.
  - Decelerate with smaller segments as approaching point B.

BASICS OF TRAJECTORY PLANNING

- Next level of trajectory planning is between multiple points for continuous movements.
- Stop-and-go motion create jerky motions with unnecessary stops.
- **Blend the two portions of the motion at point B.**
- **Specify two via point D and E before and after point B**

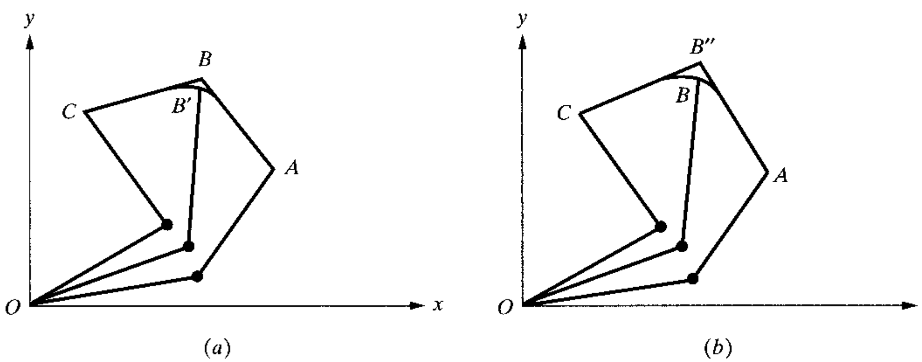


Fig. 5.8 Blending of different motion segments in a path.

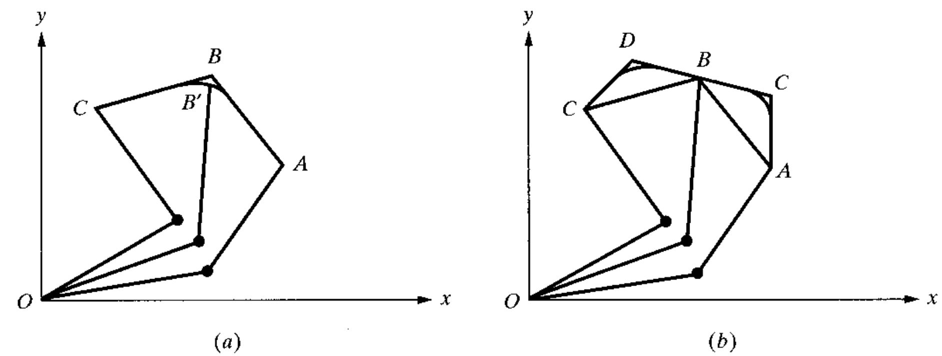


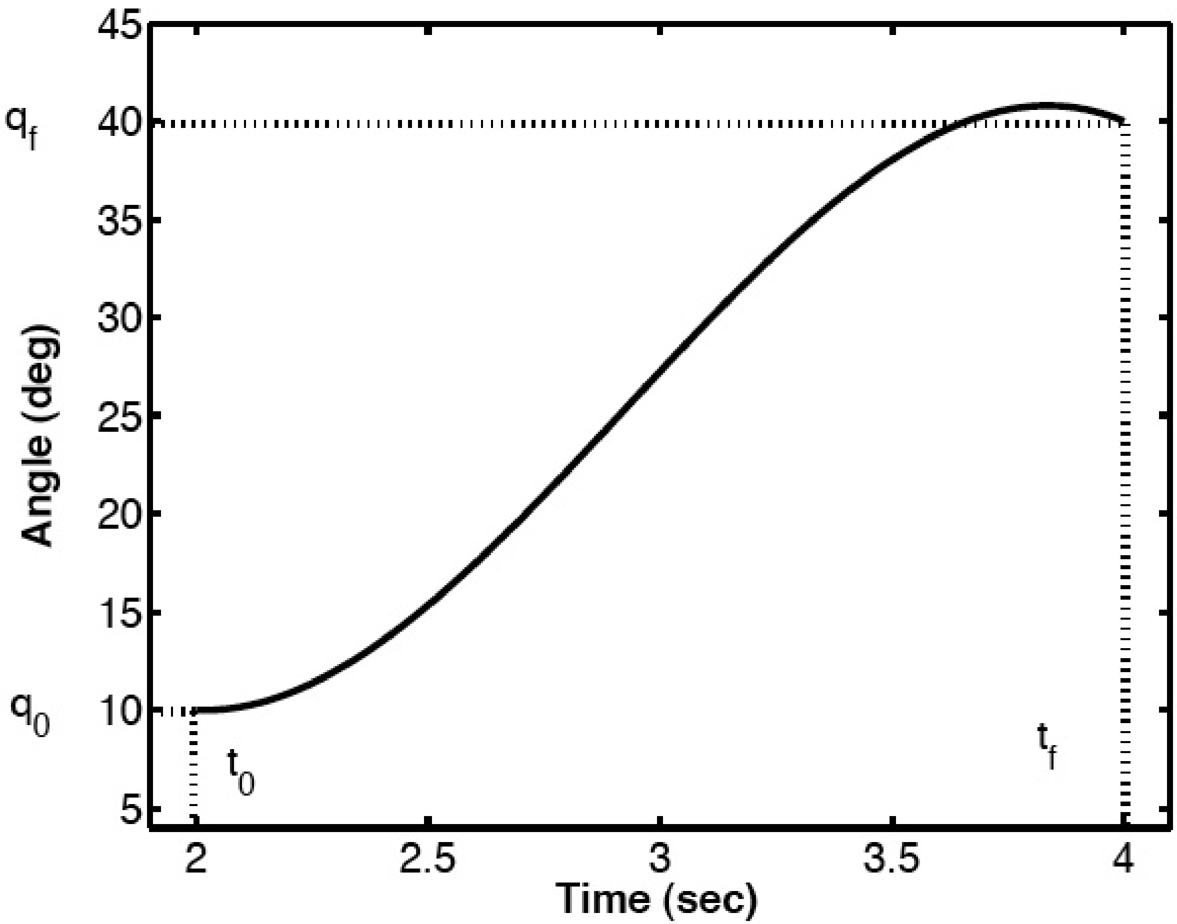
Fig. 5.9 An alternative scheme for ensuring that the robot will go through a specified point during blending of motion segments. Two via points D and E are picked such that point B will fall on the straight-line section of the segment ensuring that the robot will pass through point B.

LUND UNIVERSITY

Trajectory Types

1. Trajectories for Point to Point Motion
2. Cubic Polynomial Trajectories *(smooth motion)*
3. Quintic Polynomial Trajectories *(no jerk)*
4. Linear Segments with Parabolic Blends
5. Minimum Time Trajectories
6. Trajectories for Paths Specified by Via Points

A Typical Joint Space Trajectory



$$q(t_0) = q_0$$

$$\dot{q}(t_0) = \upsilon_0$$

$$q(t_f) = q_f$$

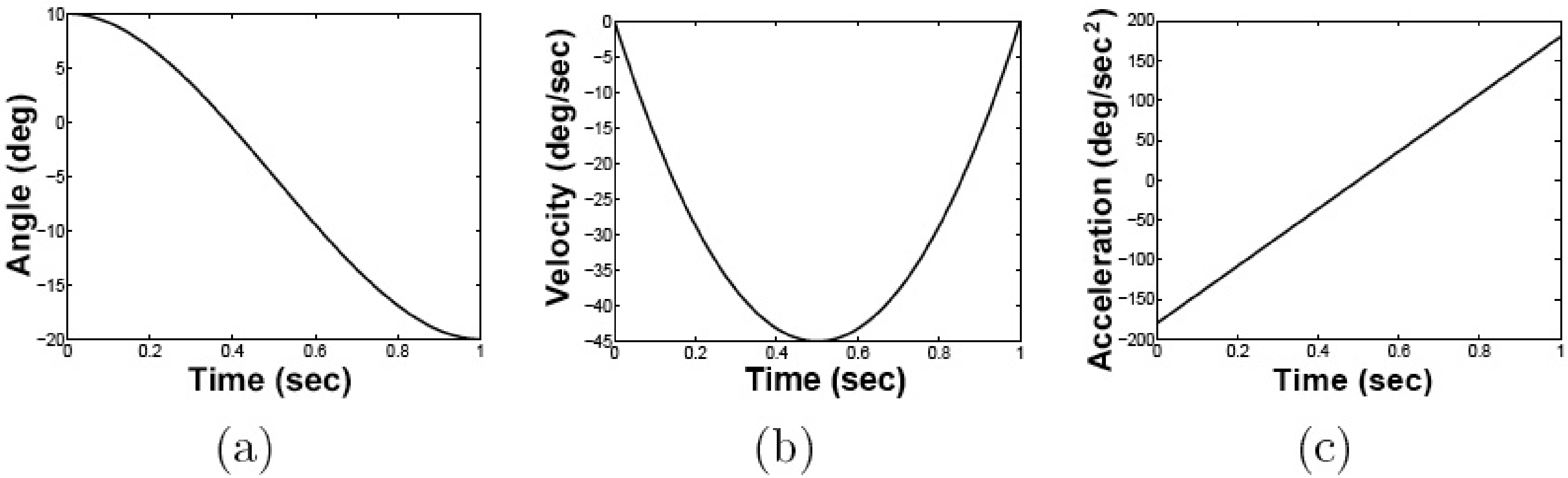
$$\dot{q}(t_f) = \upsilon_f$$

$$\ddot{q}(t_0) = \alpha_0$$

$$\ddot{q}(t_f) = \alpha_f$$

LUND UNIVERSITY

Cubic Polynomial trajectory



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

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

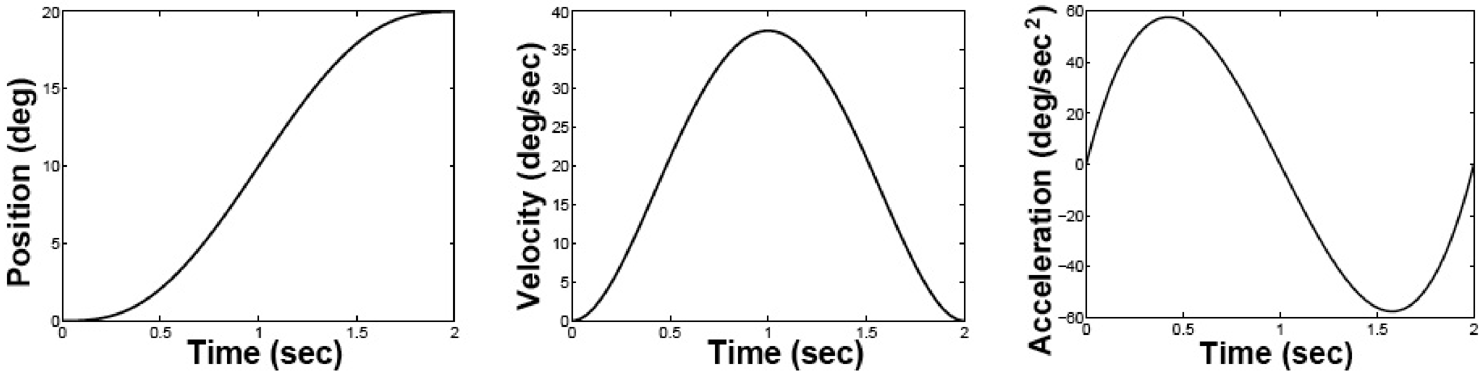
q_0

υ_0

q_f

υ_f

Quintic Polynomial Trajectory



(a) (b) (c)

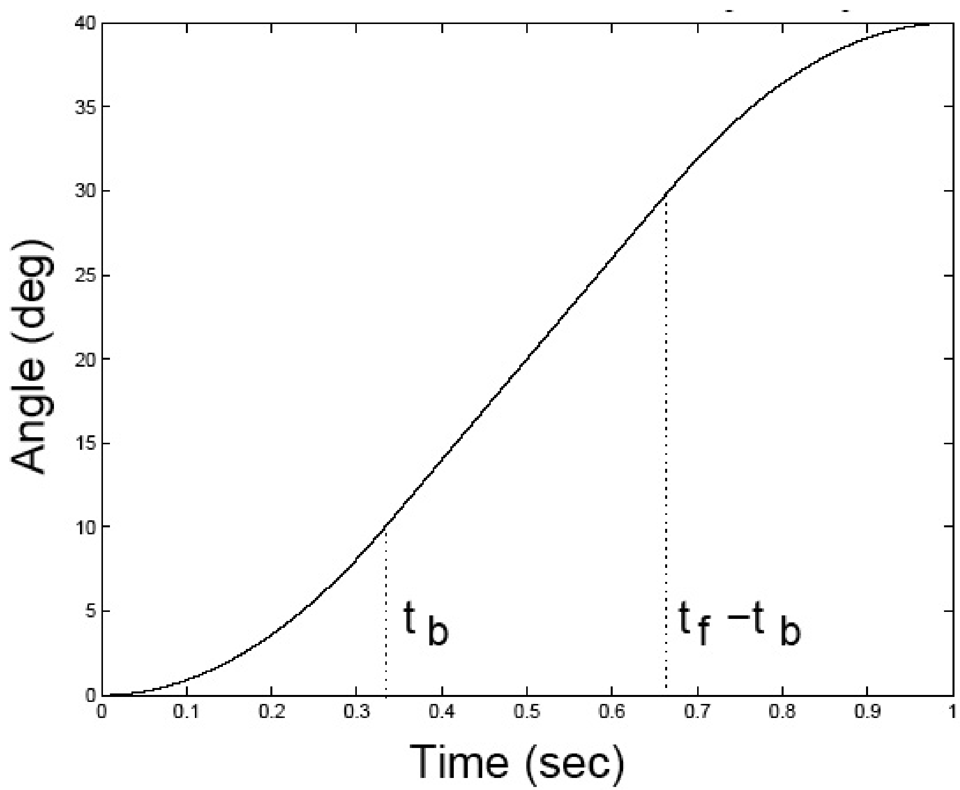
$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \quad q_0 \quad q_f$$

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \quad \upsilon_0 \quad \upsilon_f$$

$$\ddot{q}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \quad \alpha_0 \quad \alpha_f$$

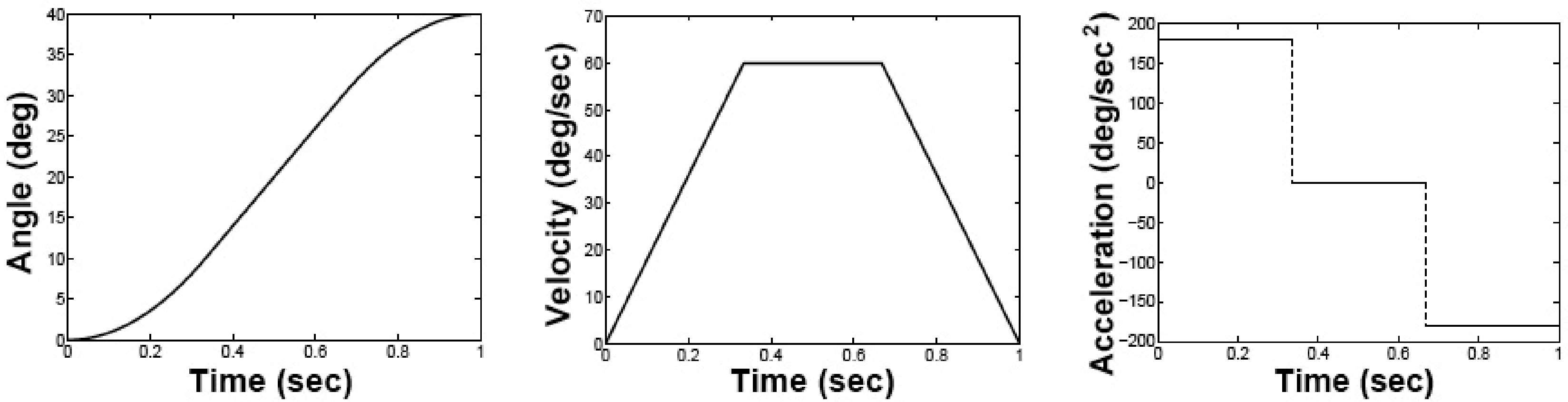
Blend Times for LSPB Trajectory

Linear Segments with Parabolic Blends



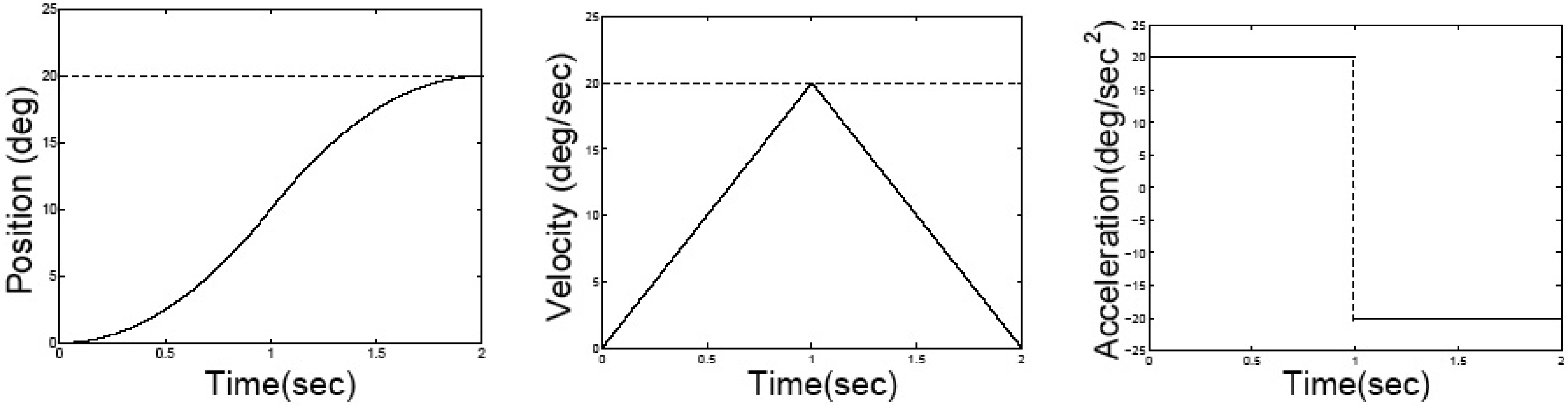
LUND UNIVERSITY

LSPB Trajectory



(a) (b) (c)

Minimum-Time Trajectory



(a) (b) (c)

Cubic Spline Trajectory with Blending Constraints

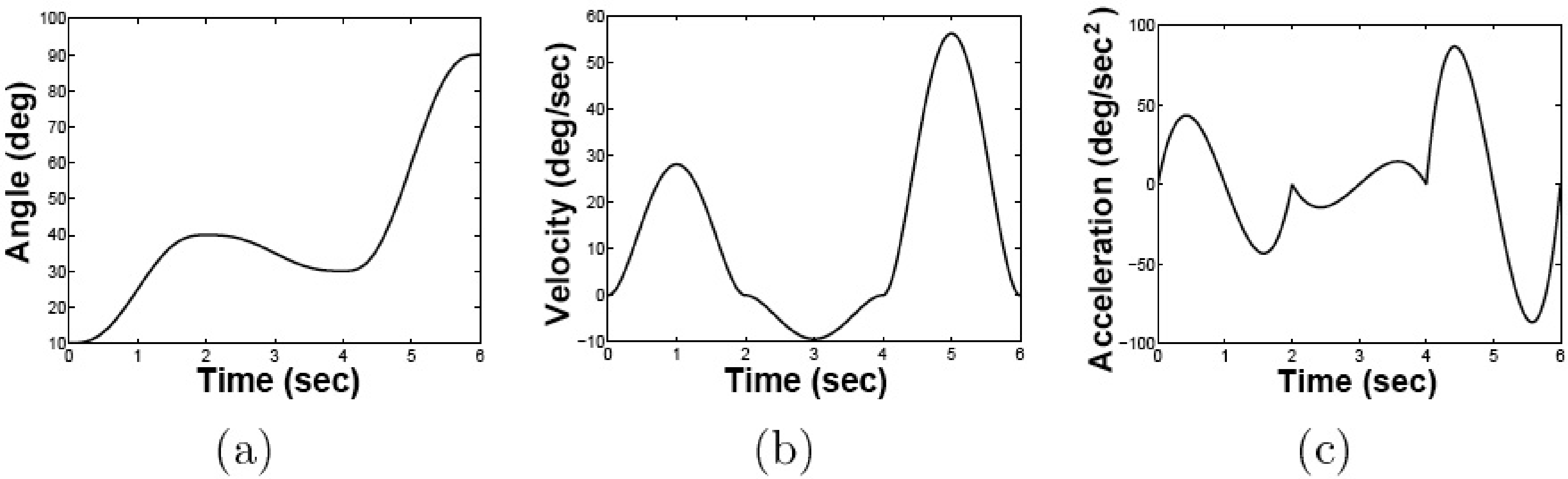
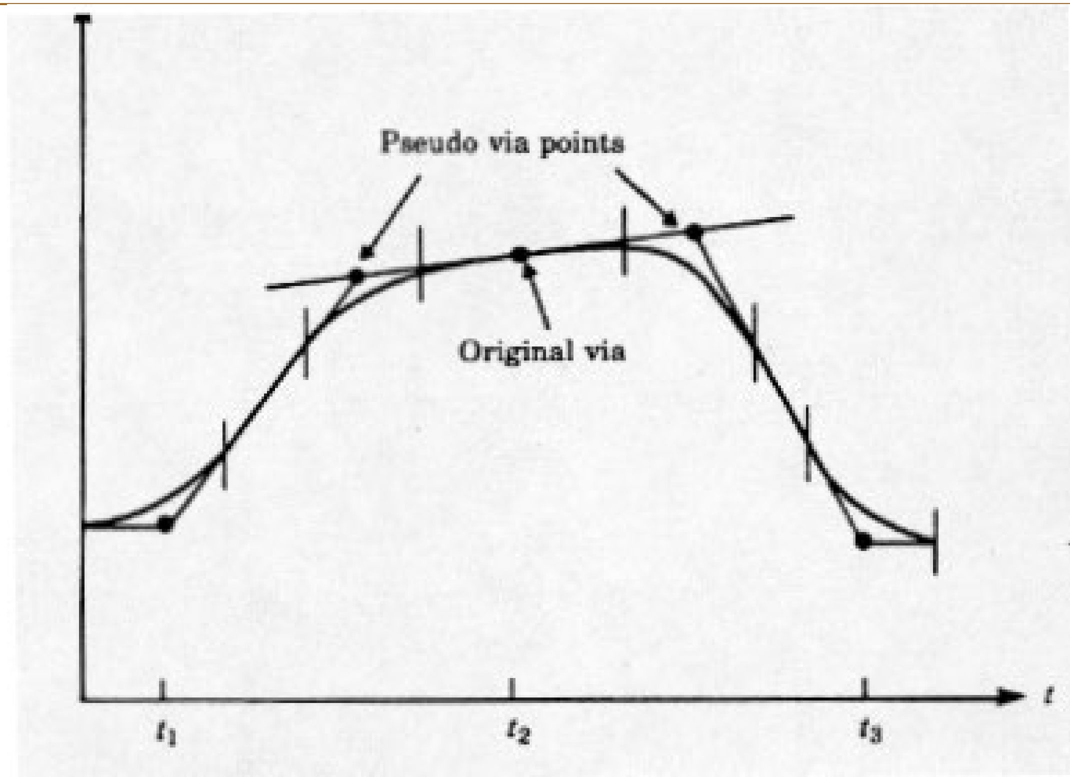


Figure 5.19: (a) Trajectory with multiple quintic segments. (b) Velocity profile for multiple quintic segments. (c) Acceleration profile for multiple quintic segments.

Forced via-points

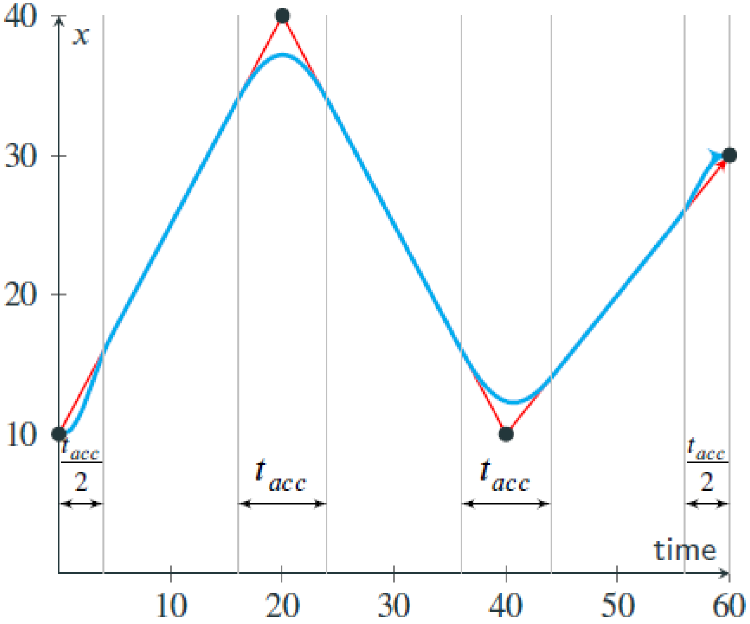


Example:

When having e.g., consecutive Move-instructions in RAPID

MoveL p1, v1000, z30, tool2
MoveL p2, v1000, z30, tool2
MoveL p3, v1000, z30, tool2

Via-points



- Introduce smooth curves (blends)
  - position, velocity and acceleration are continuous
  - blend (acceleration) time is t_{acc}
- But we don't get to the via points…
- If t_{acc} small
  - go close to the via points, but acceleration is high
- If t_{acc} large
  - acceleration is low, but further from the points

Interpolating rotations

For the **orientation planning** we might interpolate (e.g., linearly) the components of the unit vectors **n**(t), **s(t)**, and **a(t)**

...but it does not guarantee the orthonormality of unit vectors at instant of time

Compare with linear transition between two points

$$\mathbf{p}(s) = \mathbf{p}_i + \frac{s}{\|\mathbf{p}_f - \mathbf{p}_i\|}(\mathbf{p}_f - \mathbf{p}_i)$$

where $s : 0 \rightarrow 1$

(not able to define frame uniquely)

Interpolating Rotations – Euler angles

- An alternative way is to interpolate three **Euler angles** $\boldsymbol{\varphi} = (\phi, \theta, \psi)$
  - Connect $\boldsymbol{\varphi}_i$ to $\boldsymbol{\varphi}_f$
  - It is convenient to choose a cubic polynomial or a linear segment with parabolic blends timing law
  - $\boldsymbol{\omega}_e$ of the frame is related to $\dot{\boldsymbol{\varphi}}$ and has continuous magnitude
- The profiles for position, velocity and acceleration are

$$\boldsymbol{\varphi}(s) = \boldsymbol{\varphi}_i + \frac{s}{\|\boldsymbol{\varphi}_f - \boldsymbol{\varphi}_i\|}(\boldsymbol{\varphi}_f - \boldsymbol{\varphi}_i),$$

Using timing law $s(t)$ on the natural parameter

$$\dot{\boldsymbol{\varphi}}(s) = \frac{\dot{s}}{\|\boldsymbol{\varphi}_f - \boldsymbol{\varphi}_i\|}(\boldsymbol{\varphi}_f - \boldsymbol{\varphi}_i),$$

The angular velocity $\boldsymbol{\omega}$ is linearly related to $\dot{\boldsymbol{\varphi}}$

$$\ddot{\boldsymbol{\varphi}}(s) = \frac{\ddot{s}}{\|\boldsymbol{\varphi}_f - \boldsymbol{\varphi}_i\|}(\boldsymbol{\varphi}_f - \boldsymbol{\varphi}_i),$$

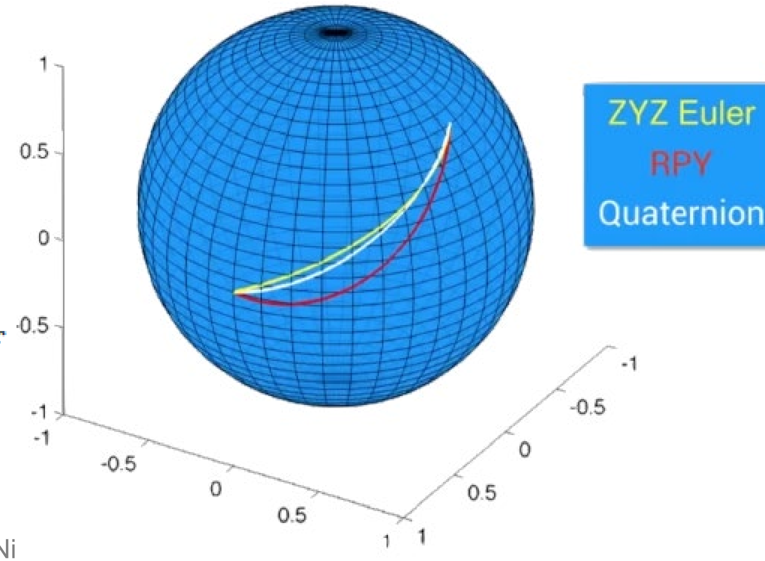
Poor predictability of the intermediate orientation

Rotations: Quaternion interpolation

- Spherical linear interpolation (slerp)
- Constant angular velocity about a fixed axis in space
- **Shortest and most direct path** between two orientations
- It is the standard

$$\mathring{q}(s) = \frac{\sin((1-s)\theta)\mathring{q}_i + \sin(s\theta)\mathring{q}_f}{\sin(\theta)}$$

$$\cos(\theta) = s_i s_f + v_{x,i}v_{x,f} + v_{y,i}v_{y,f} + v_{z,i}v_{z,f}$$



Interpolation in Peter Corke's Matlab toolbox

There are two main functions for interpolation in Peter Corke's Robotics toolbox

jtraj - Compute a joint space trajectory

[Q,QD,QDD] = jtraj(Q0, QF, M)

is a joint space interpolation from Q0 to QF with M intermediate points

ctraj - Cartesian trajectory between two poses

TC = ctraj(T0, T1, N)

is a Cartesian interpolation from T0 to T1 with N intermediate points

These corresponds to MoveJ/MoveAbsJ and MoveL in RAPID

LUND UNIVERSITY