FRTF20 Applied Robotics
Lecture 7

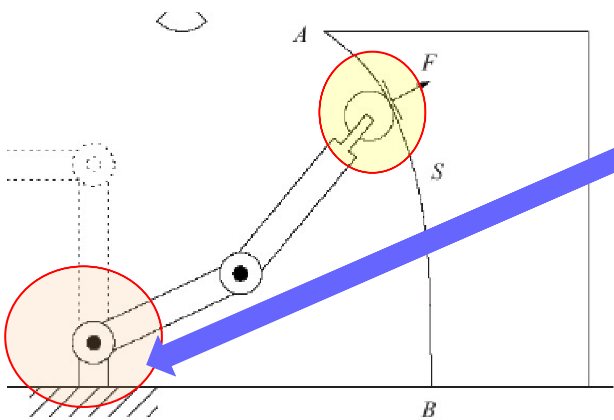
ANDERS ROBERTSSON

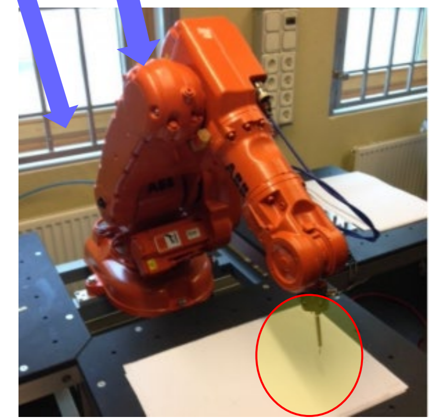
Robotics in this course

- 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
  - Path Planning and Trajectory Generation
  - Dynamics
  - Motion Control
  - (Force Control)
  - Sequence programming (and task description)

LUND UNIVERSITY

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;
```

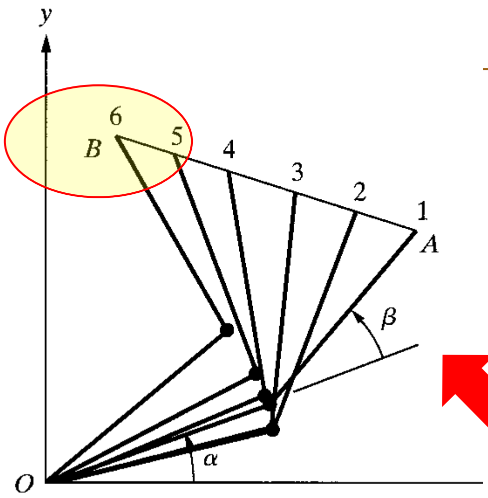
From MoveL L_20 to joint values [q1..q6]

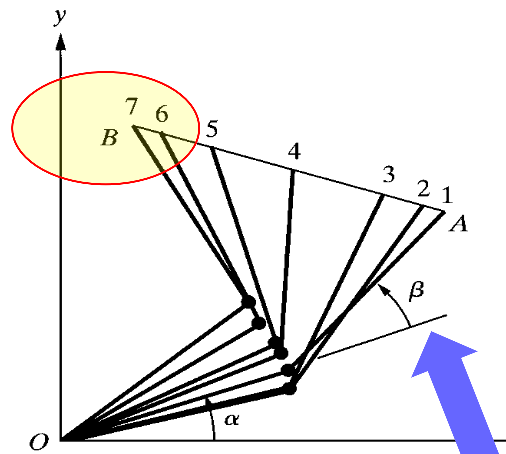
(Lec 2,3) Frames, Forward/Inverse kinematics (joint angles ←→ pose)

(Lec 3) Relation between velocities (Jacobian)

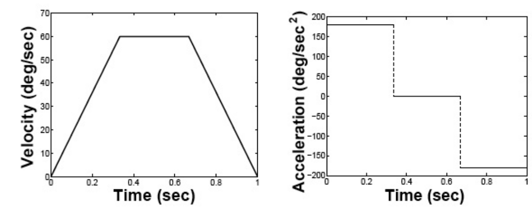
Computer exercises [matlab]: frames; DH-modelling, ...

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;
```

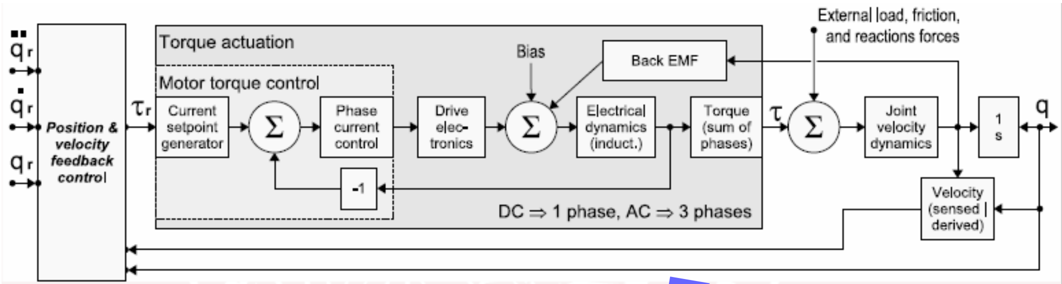
From MoveL L_20 to joint values [q1..q6]

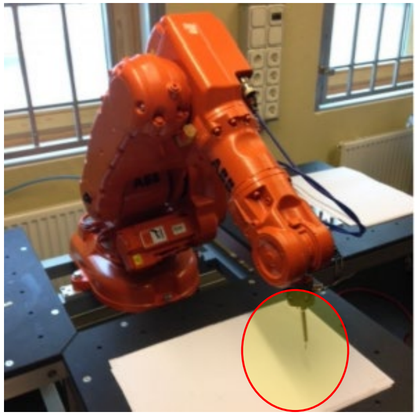
(Lec 2,3) Frames, Forward/Inverse kinematics (joint angles <-> pose)

(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:=[[100,0,0],[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]

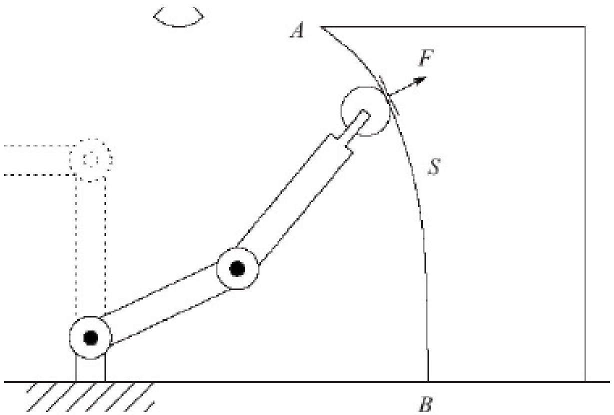
(Lec 2,3) Frames, Forward/Inverse kinematics (joint angles ←-

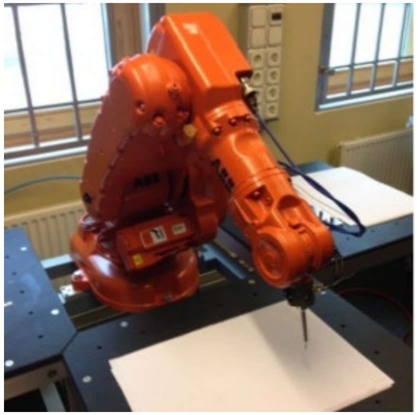
(Lec 3) Relation between velocities (Jacobian)

(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

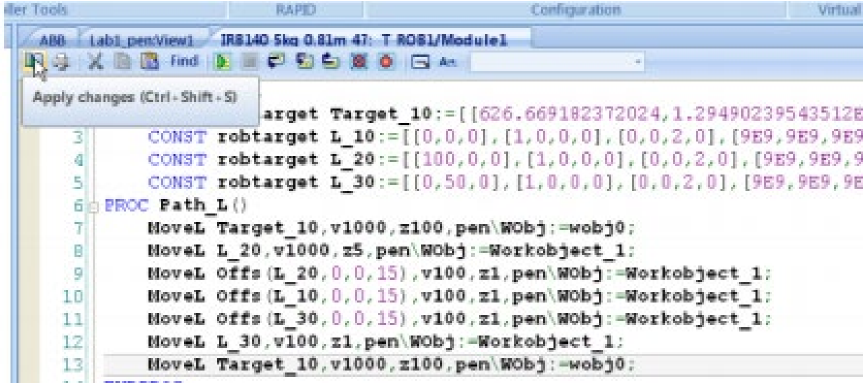




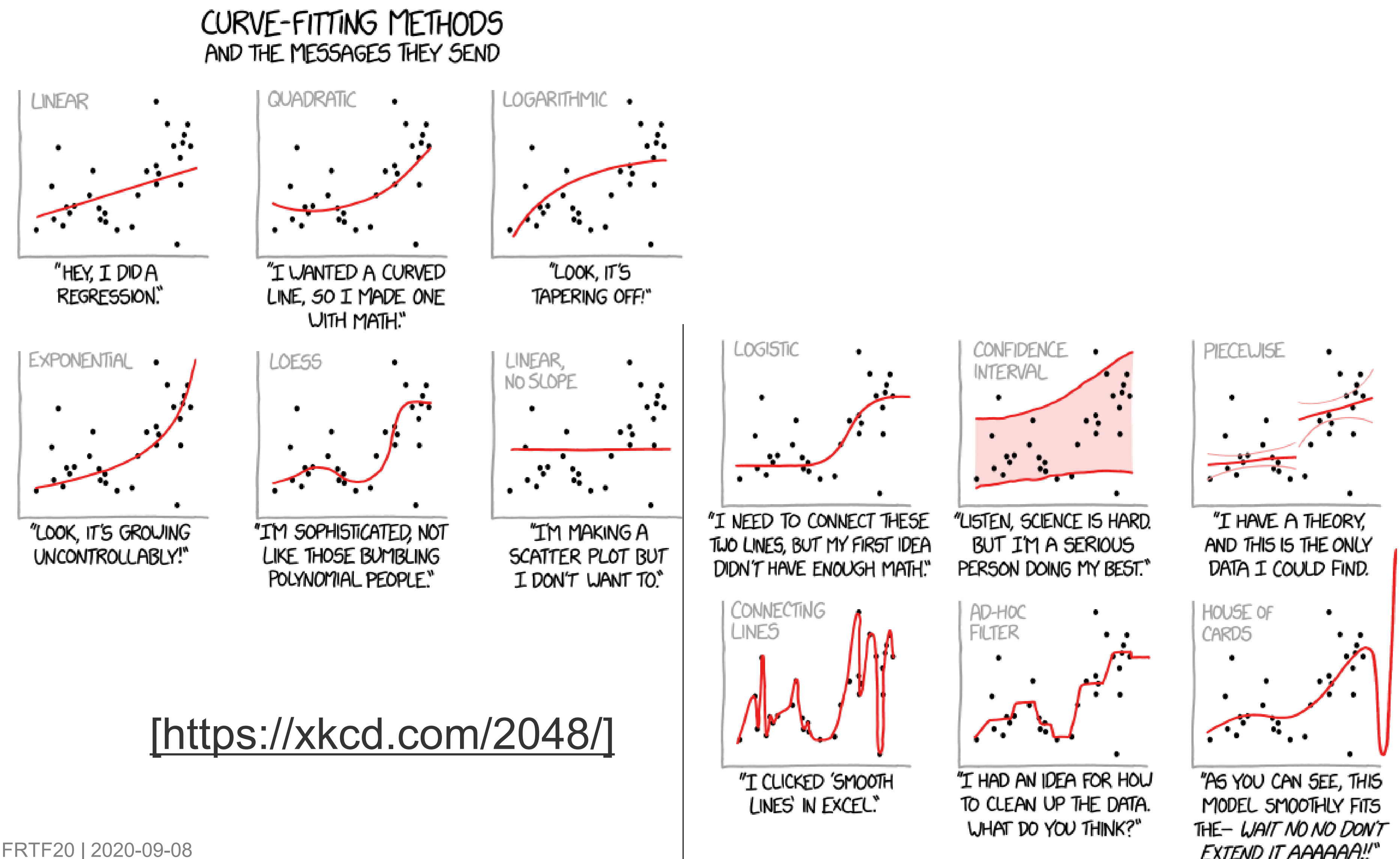
Modeling and robot programming: Robot + CAD of workcell

RobotStudio Exercises 1,2, and 3:

Paths and configurations , I/O , collision avoidance



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)

LUND UNIVERSITY

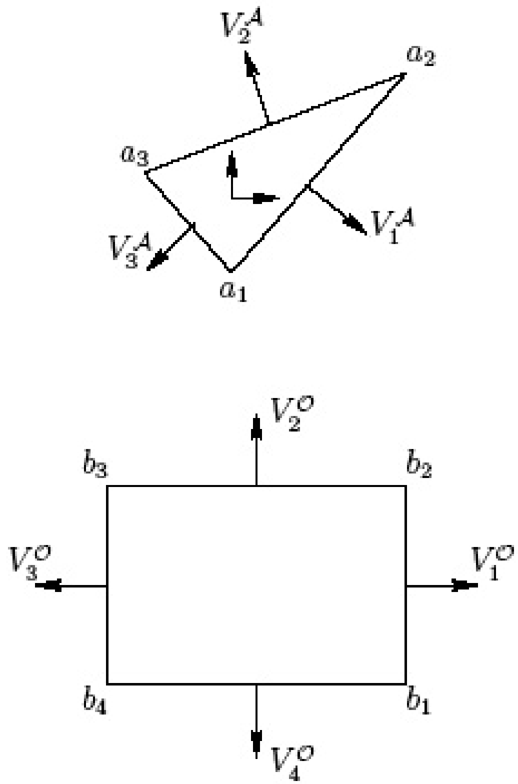
Path Planning

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

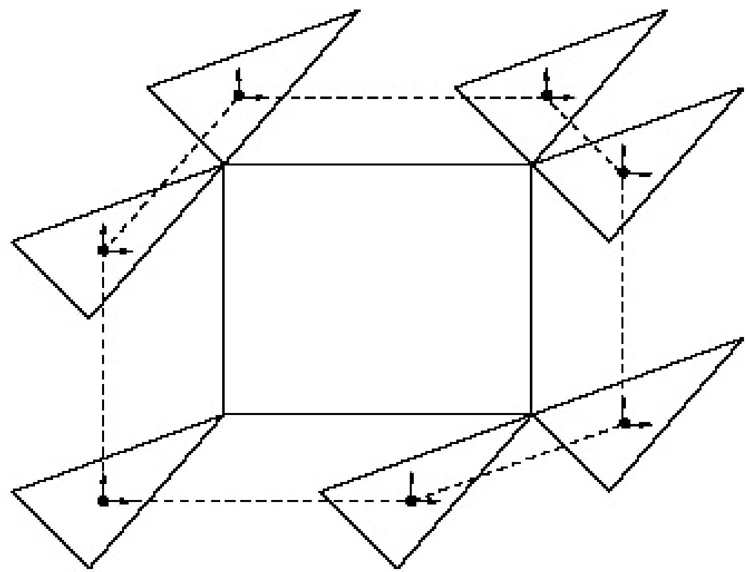
1. Configuration Space
2. Artificial Potential Fields
3. Probabilistic Roadmap

LUND UNIVERSITY

A Rigid Body that Translates in Plane



(a)



(b)

Two-Link Planar Arm

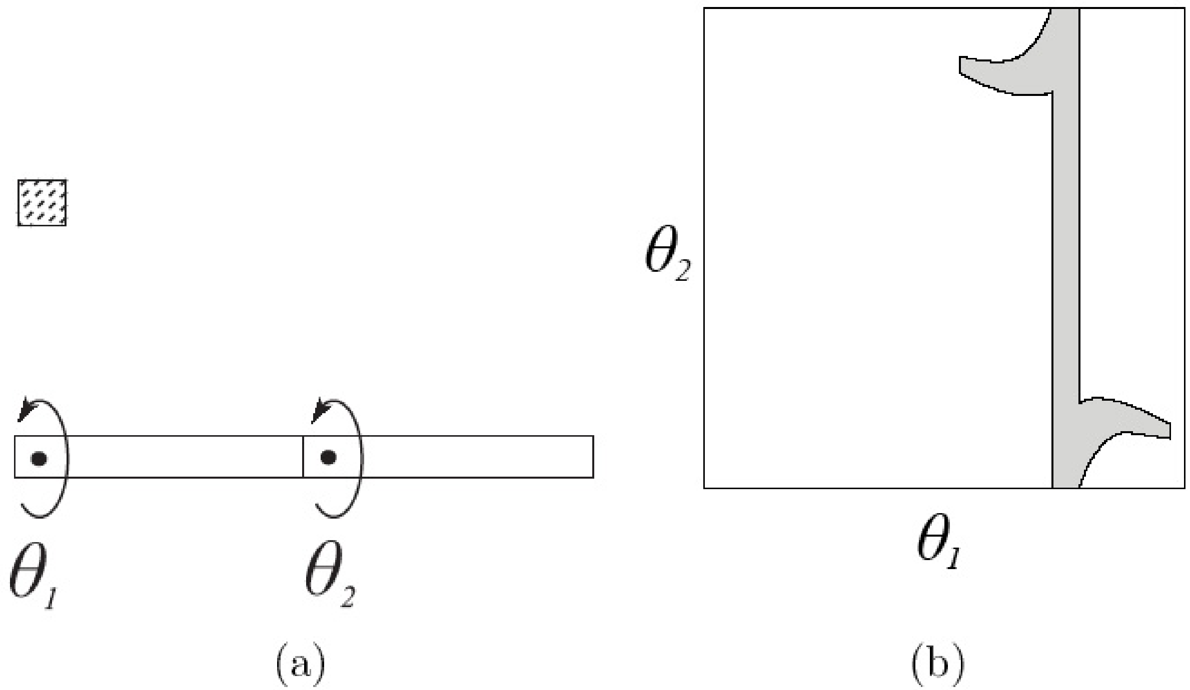
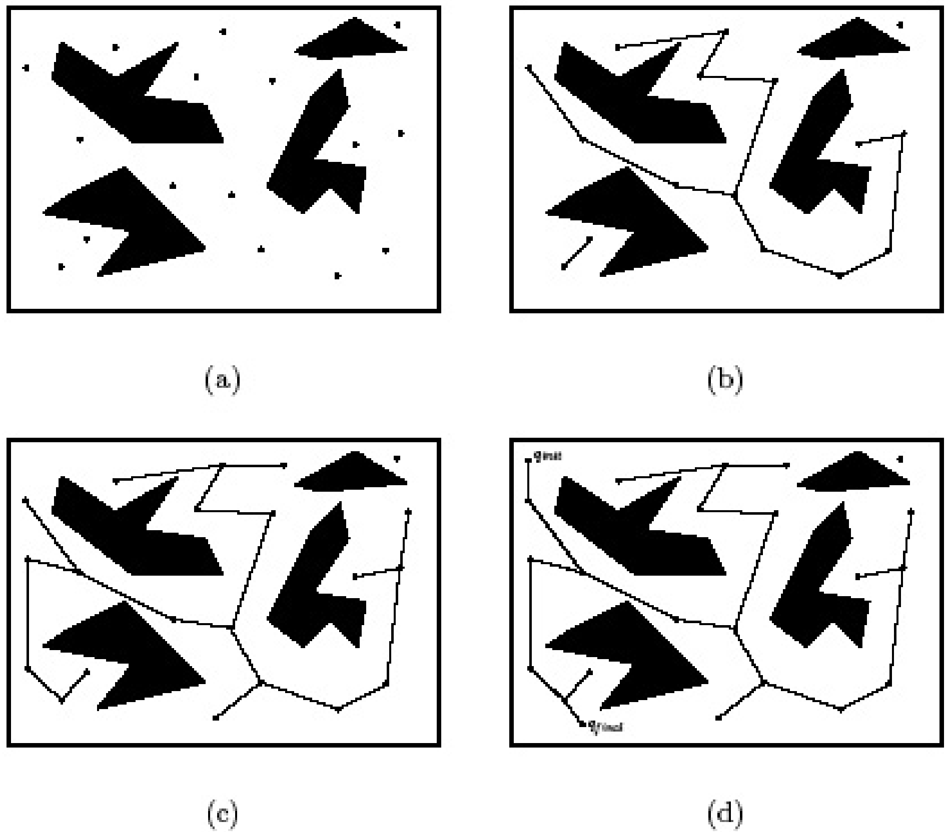


Figure 5.2: (a) The robot is a two-link planar arm and the workspace contains a single, small polygonal obstacle. (b) The corresponding configuration space obstacle region contains all configurations $q = (\theta_1, \theta_2)$ such that the arm at configuration q intersects the obstacle.

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

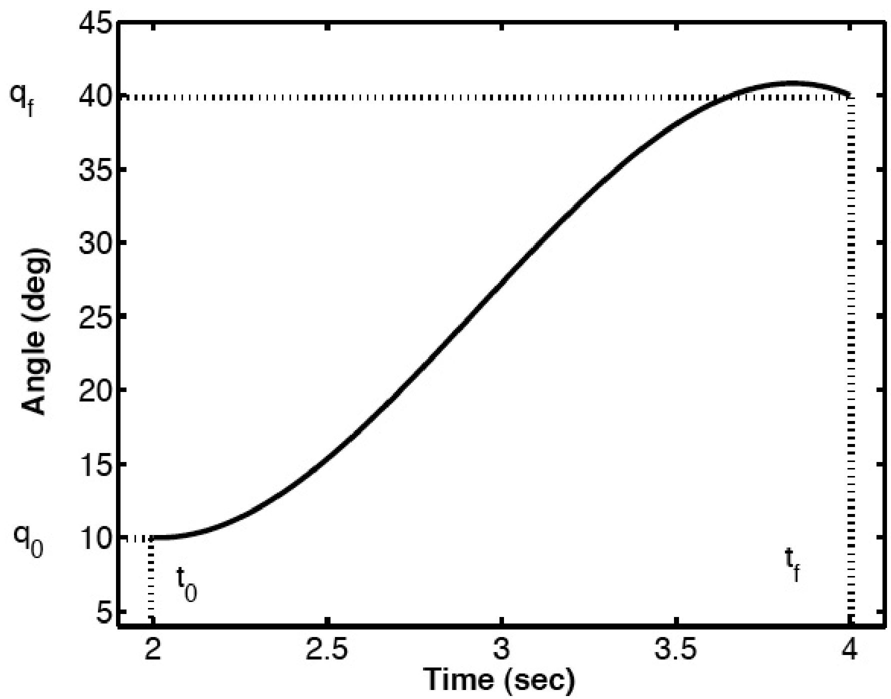
Probabilistic Roadmap methods



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

Trajectory Planning

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



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.

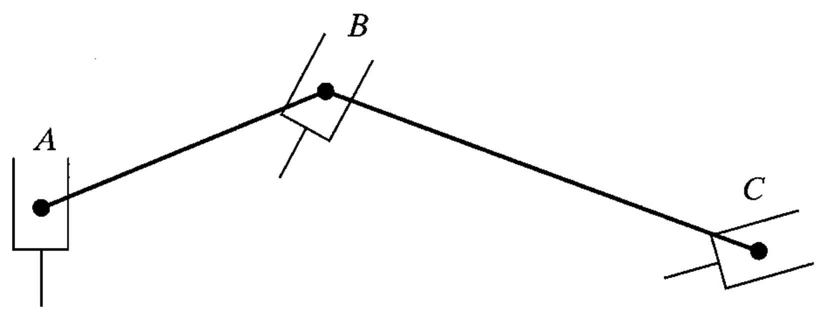
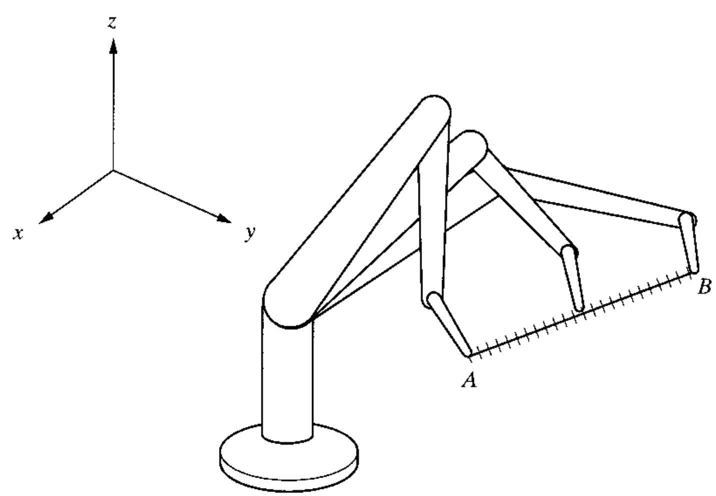
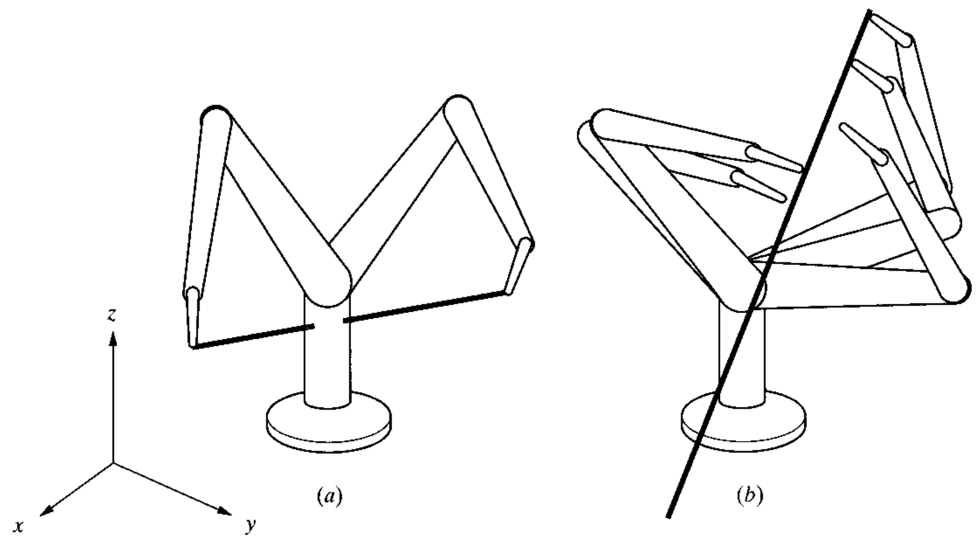


Fig. 5.1 Sequential robot movements in a path.

- 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.



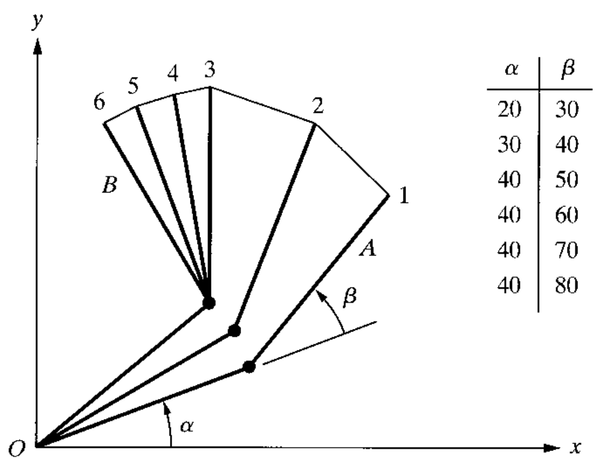
Sequential motions of a robot to follow a straight line.



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

- 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.

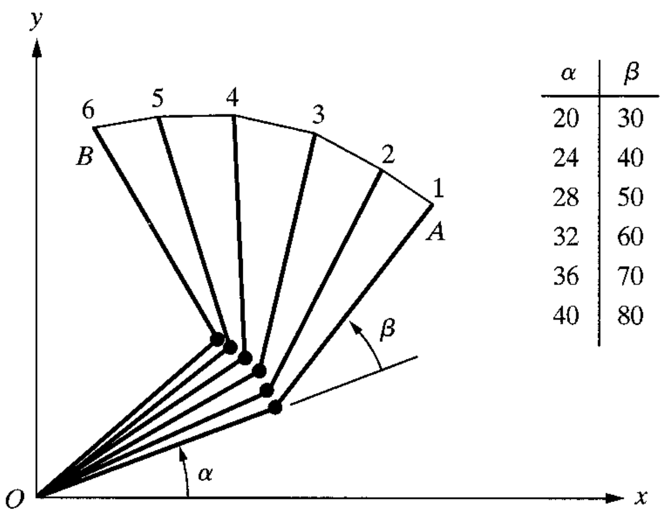


α	β
20	30
30	40
40	50
40	60
40	70
40	80

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 common factor such that the joint with smaller motion will move proportionally slower and the both joints will start and stop their motion simultaneously.

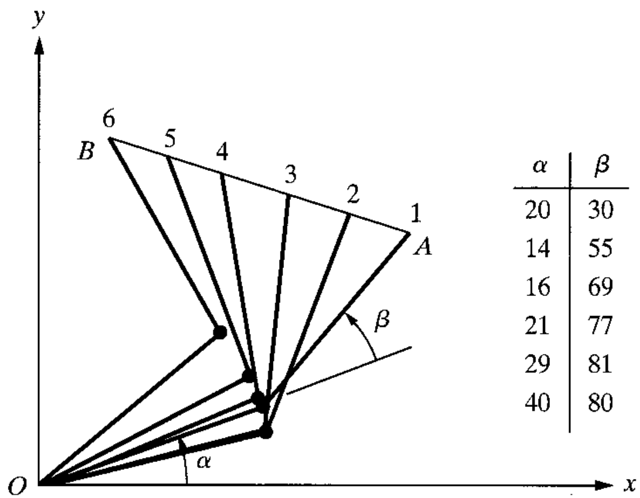


α	β
20	30
24	40
28	50
32	60
36	70
40	80

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.**

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

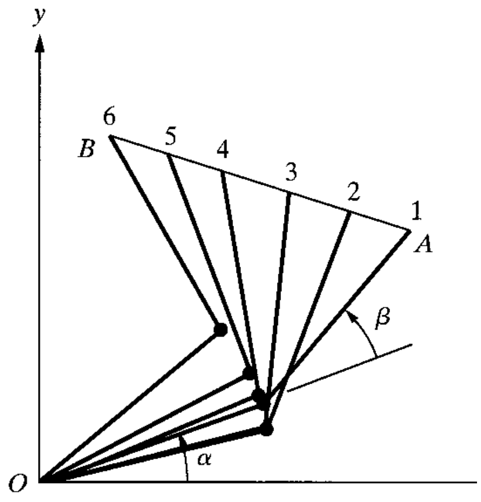


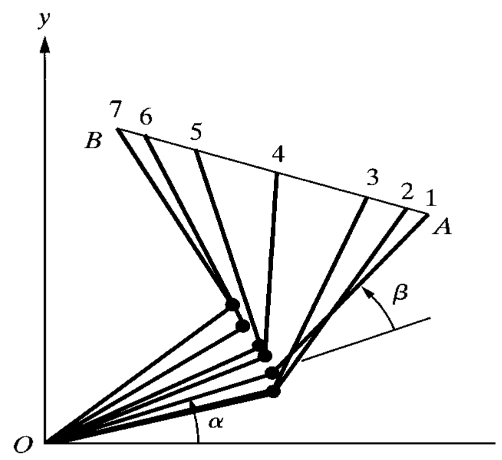
α	β
20	30
14	55
16	69
21	77
29	81
40	80

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.

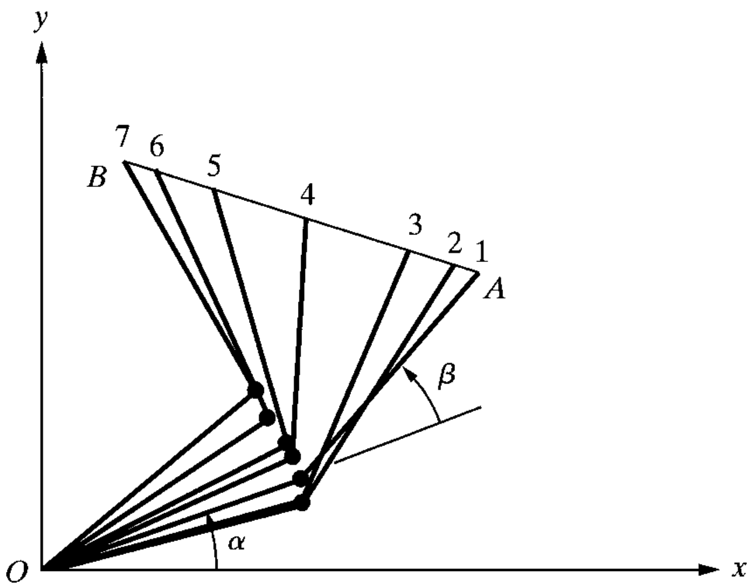


Fig. 5.7 Trajectory planning with an acceleration-deceleration regiment.

- **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**

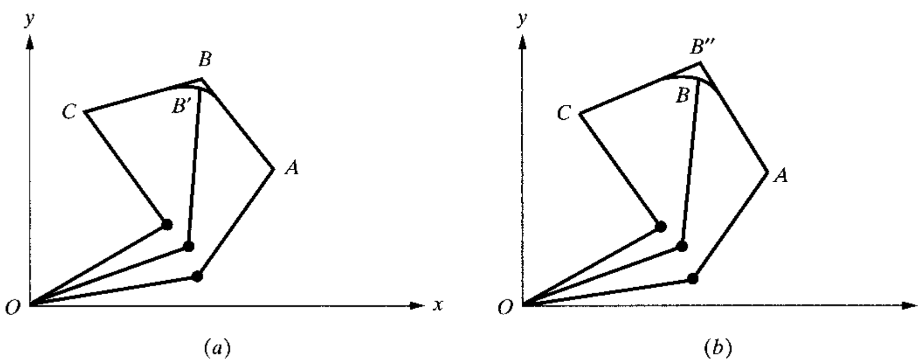


Fig. 5.8 Blending of different motion segments in a path.

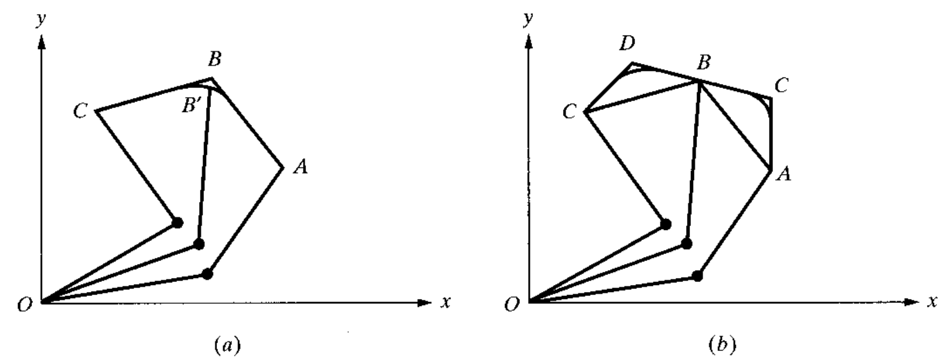


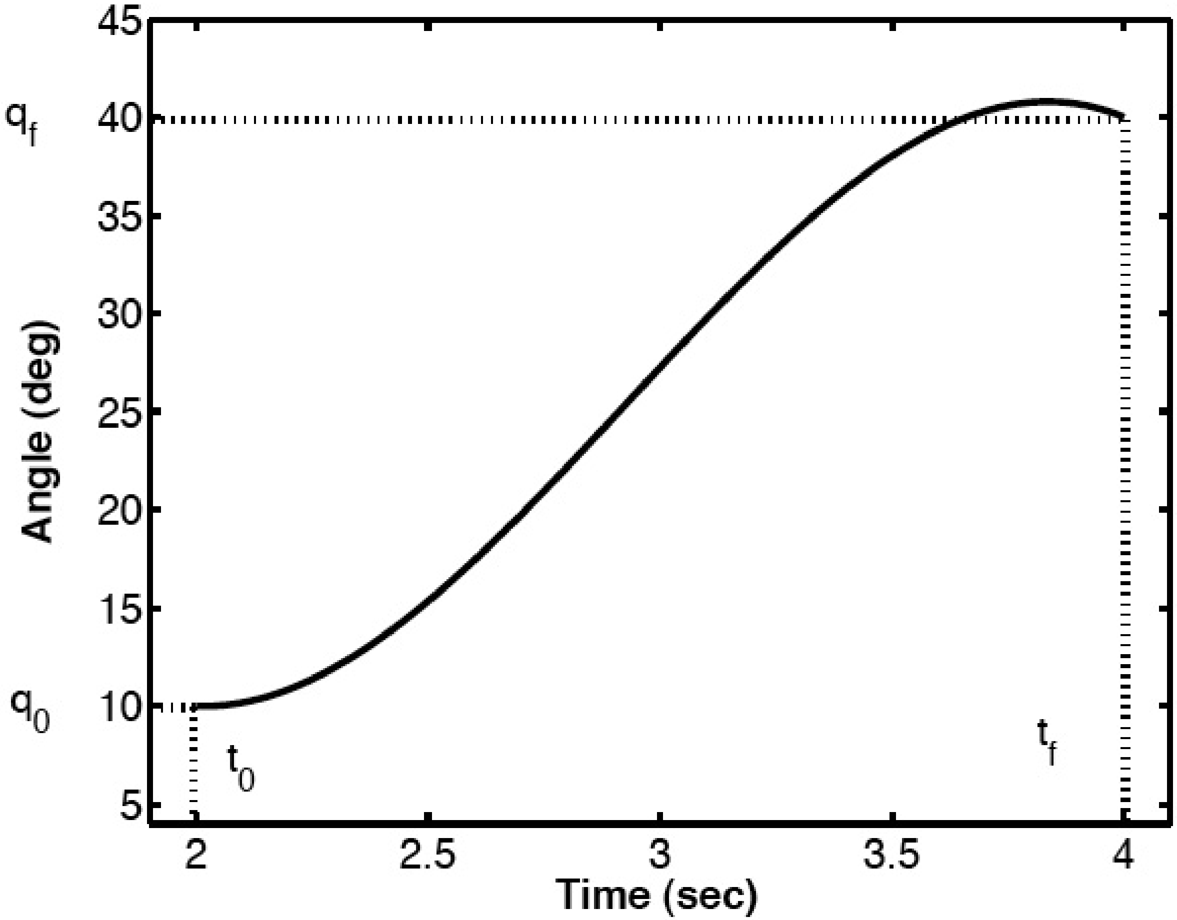
Fig. 5.9 An alternative scheme for ensuring that the robot will go through a specified point during blending of motion segments. Two via points D and E are picked such that point B will fall on the straight-line section of the segment ensuring that the robot will pass through point B.

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



$$q(t_0) = q_0$$

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

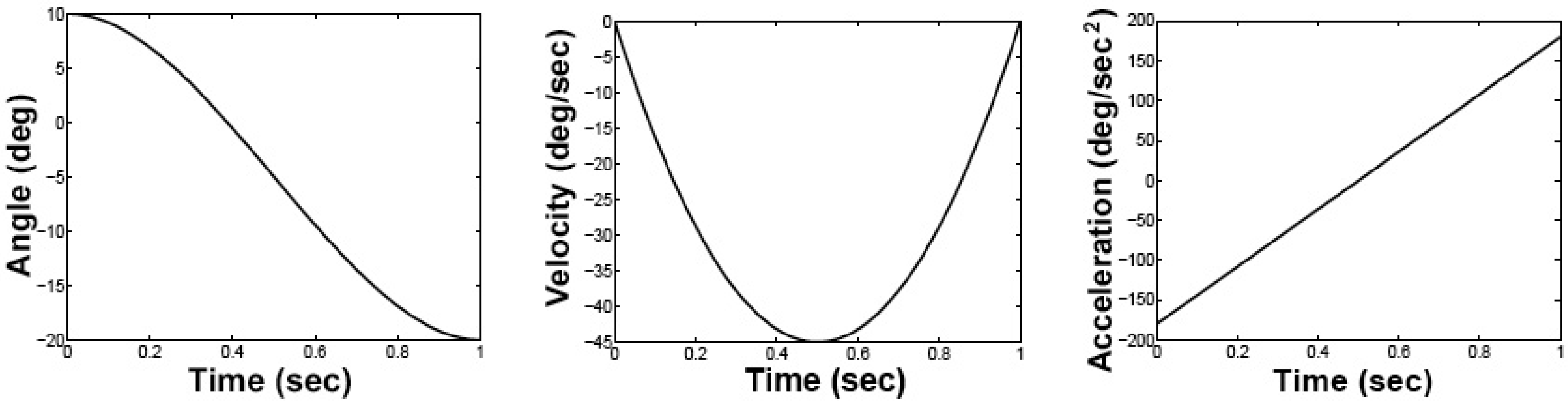
$$q(t_f) = q_f$$

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

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

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

Cubic Polynomial trajectory



(a) (b) (c)

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

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

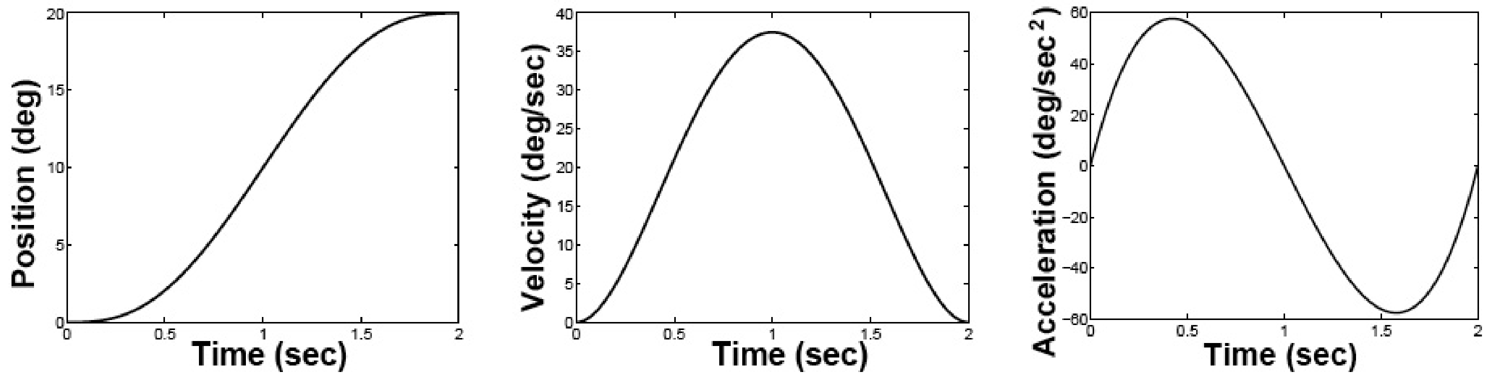
q_0

υ_0

q_f

υ_f

Quintic Polynomial Trajectory



(a) (b) (c)

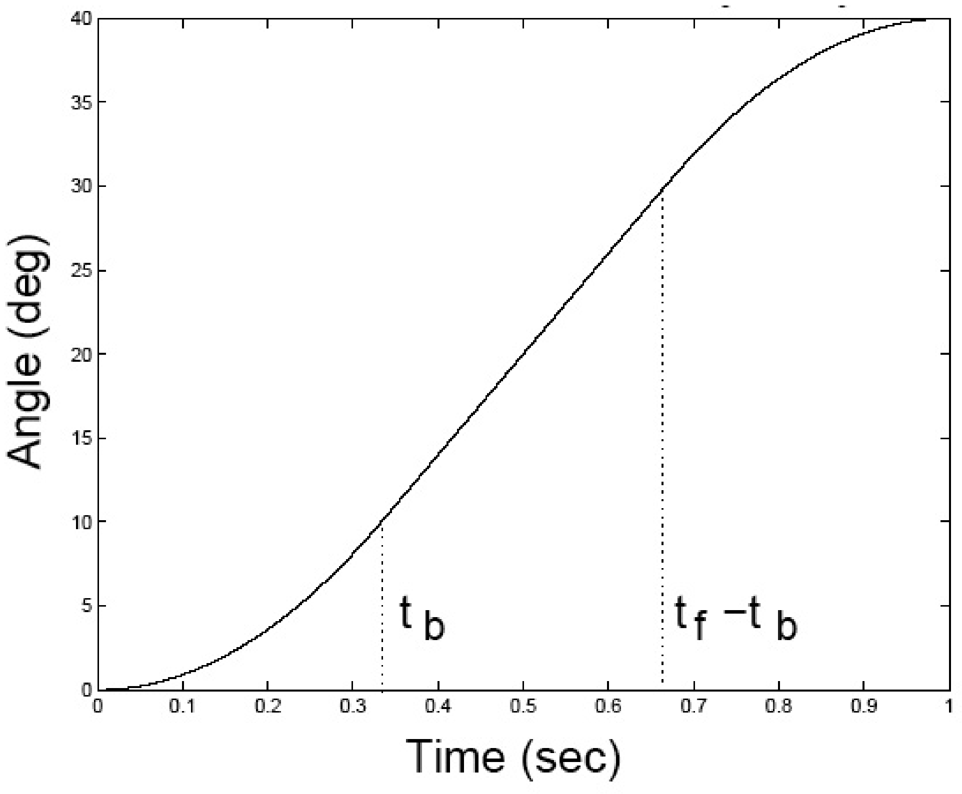
$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \qquad q_0 \quad q_f$$

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

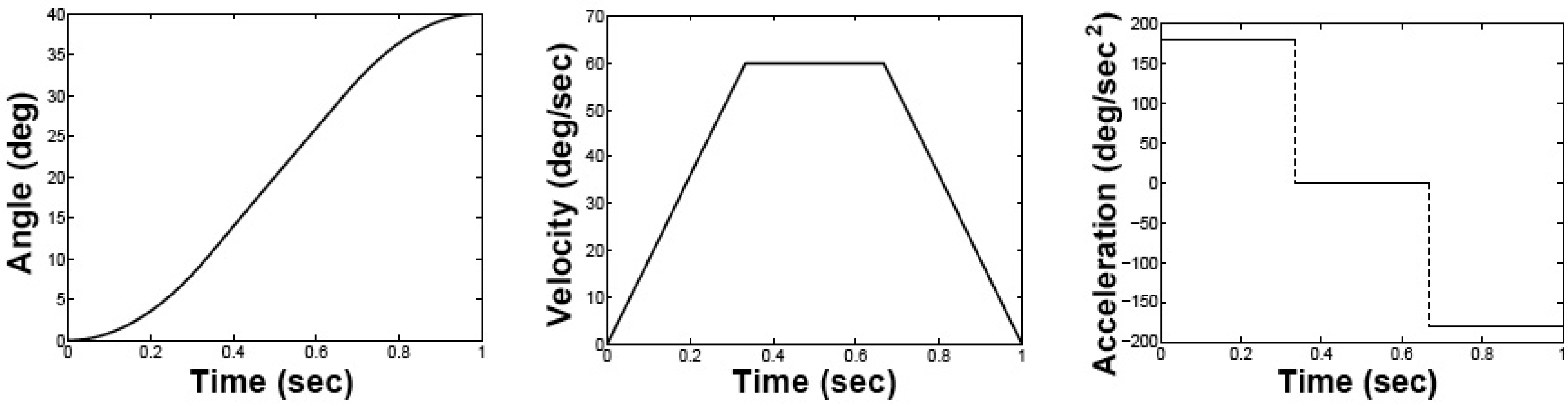
$$\ddot{q}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \qquad \alpha_0 \quad \alpha_f$$

Blend Times for LSPB Trajectory

Linear Segments with Parabolic Blends

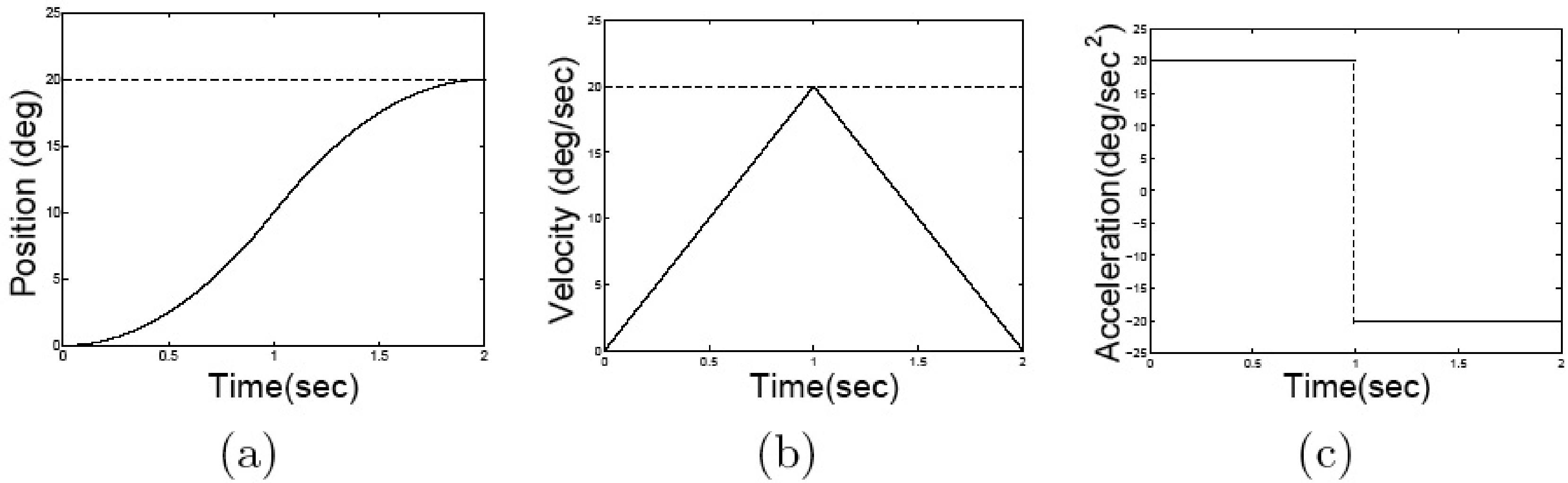


LSPB Trajectory



(a) (b) (c)

Minimum-Time Trajectory



Cubic Spline Trajectory with Blending Constraints

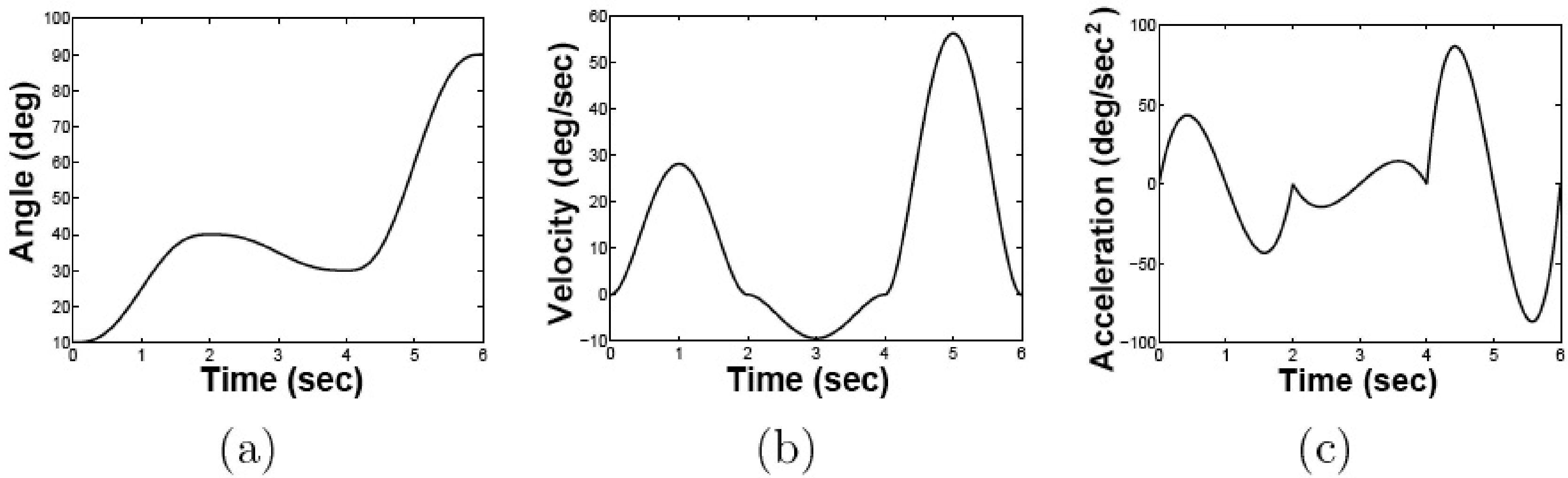
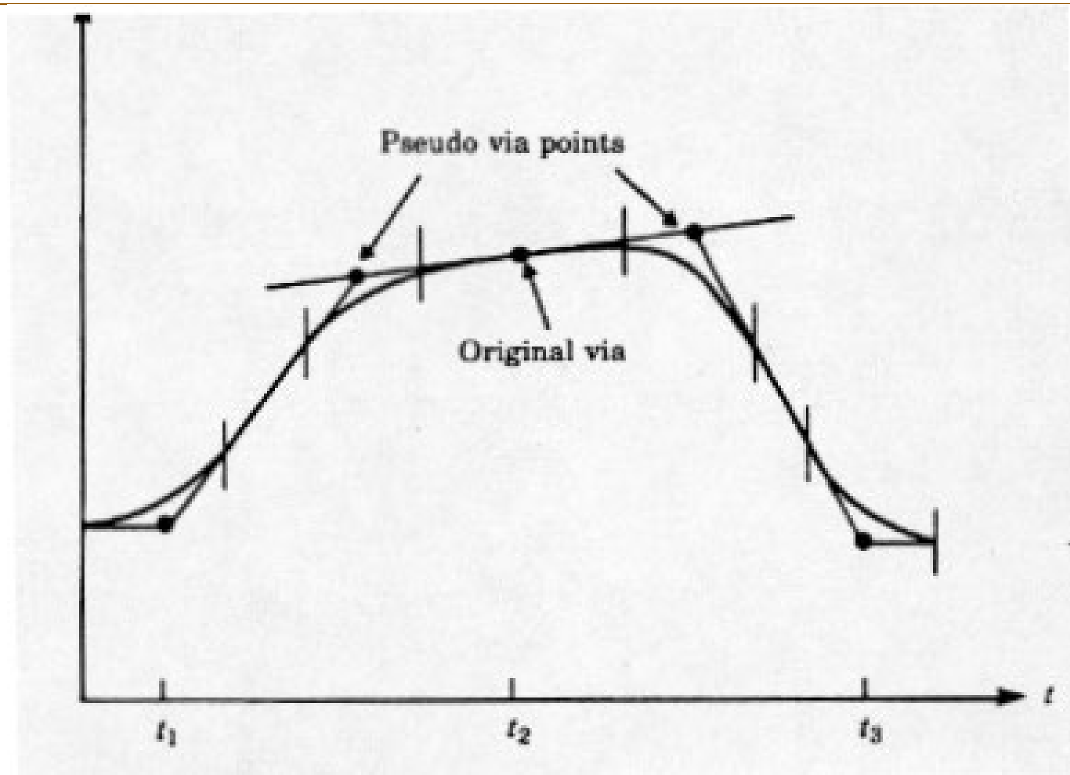


Figure 5.19: (a) Trajectory with multiple quintic segments. (b) Velocity profile for multiple quintic segments. (c) Acceleration profile for multiple quintic segments.

Forced via-points



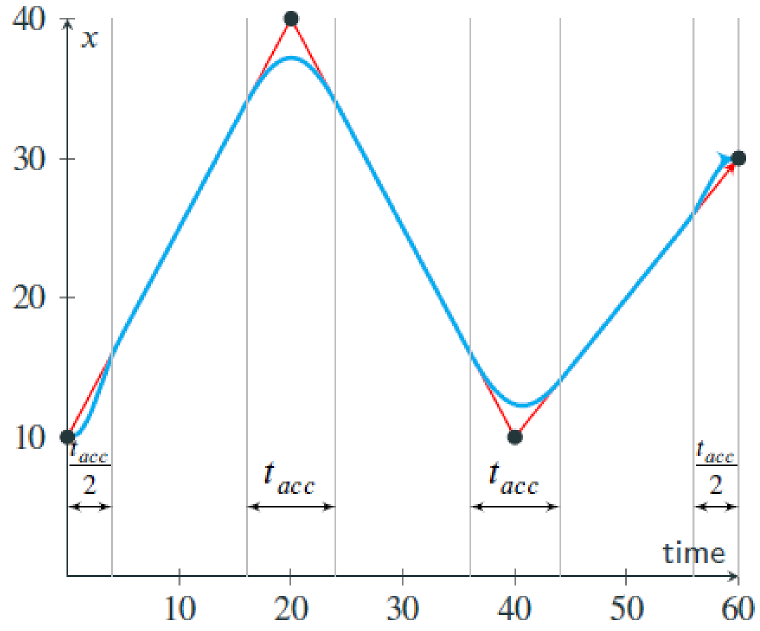
Example:

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

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


LUND UNIVERSITY

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)

LUND UNIVERSITY

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 $\dot{\boldsymbol{\varphi}}$

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

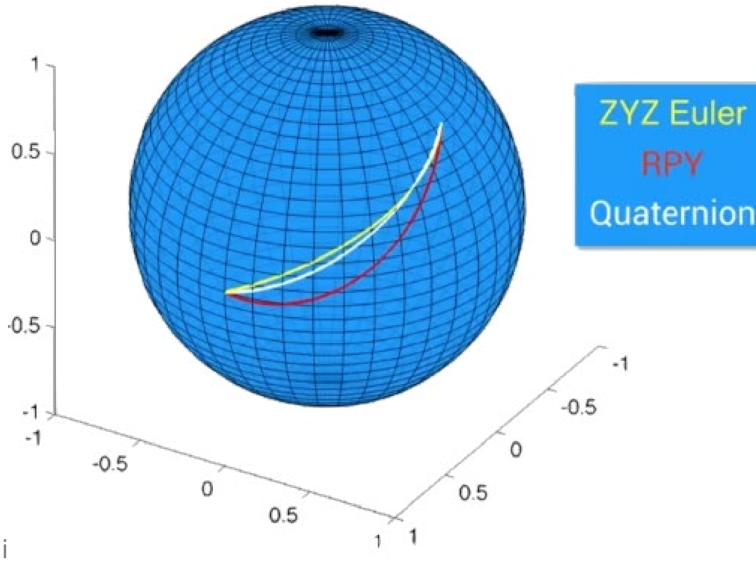
Poor predictability of the intermediate orientation

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

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

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



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