Robot Control Basics
CS 685

Jana Kosecka
George Mason University

1

Control basics

- Use some concepts from control theory to understand and learn how to control robots
- Control Theory – general field studies control and understanding of behavior of dynamical systems
  (robots, epidemics, biological systems, stock markets etc.)

2

Control basics

- Basic ingredients
  - state of the system $\vec{x} = [x, y, \theta]$ current position of the robot
  - dynamics behavior of the systems as a function of time (description how system state changes as a function of time)
  - system of differential equations $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$

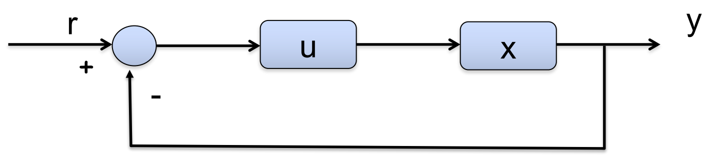
$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = \omega$$

  - control input which can affect the behavior $u = [v, \omega]$
  - controller which takes some function of the goal, the state

3

Control basics

- Basic ingredients
  - controller which takes some function of the goal, the state
  - y output, measurement of some aspect of the state
- Feedback control – how to compute the control based on output (state) and the desired objective



- Difference equations (examples)

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$$

4

Simple control strategies

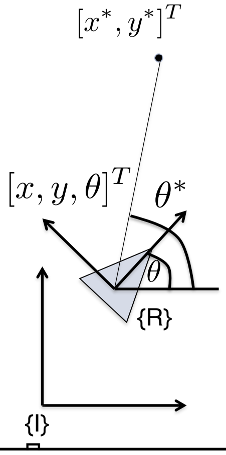
- Moving to a point – go to a point
- Consider a problem of moving to a point (x,y)
- How to control angular and linear velocity of the mobile robot
- Linear velocity – proportional to distance
- Angular velocity – steer towards the goal

- Following a line – steer toward a line
- Angular velocity proportional to the combination distance from the line and also to alignment with the line

5

Moving to a point

- Differential drive robot – go from the current pose $[x, y, \theta]^T$ to desired point with coordinates $[x^*, y^*]^T$



$$\theta^* = \tan^{-1}\frac{y^* - y}{x^* - x}$$

$$v = K_v\sqrt{(x^* - x)^2 + (y^* - y)^2)}$$

$$\omega = K_h(\theta^* - \theta)$$

Source P. Corke: Robotics, Vision and Control. Springer

6

Moving to a line

- Equation of a line

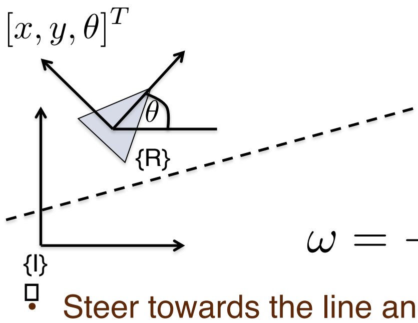
$$ax + by + c = 0$$

- Shortest distance of the robot the line

$$d = \frac{[a, b, c][x, y, 1]^T}{\sqrt{a^2 + b^2}}$$

- Orientation of the line

$$\theta^* = \tan^{-1} \frac{-a}{b}$$



$$\alpha_d = -K_d d \quad K_d > 0$$

$$\alpha_h = K_h(\theta^* - \theta)$$

$$\omega = -K_d d + K_h(\theta^* - \theta)$$

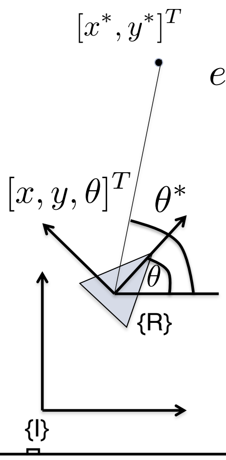
- Steer towards the line and align the robot with the line

7

Following a path

- Same as going to the point – now sequence of waypoints $x(t), y(t)$

$$\theta^* = \tan^{-1} \frac{y^* - y}{x^* - x}$$



$$e = K_v \sqrt{(x^* - x)^2 + (y^* - y)^2)} - d^*$$

d^* distance behind the pursuit point

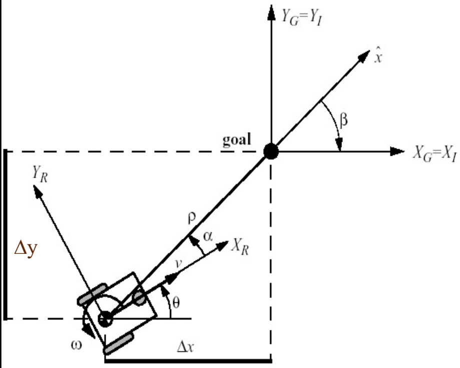
$$v = K_v e + K_i \int e dt$$

$$\omega = K_h(\theta^* - \theta)$$

Source P. Corke: Robotics, Vision and Control, Springer

8

Kinematic Position Control



The kinematic of a differential drive mobile robot described in the initial frame $\{x_I, y_I, \theta\}$ is given by,

$$ {}^I\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} $$

relating the linear velocities in the direction of the x_I and y_I of the initial frame.

Let α denote the angle between the x_R axis of the robots reference frame and the vector connecting the center of the axle of the wheels with the final position.

10

3.6.1

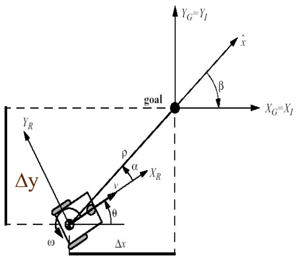
Move to Pose

- Set intermediate positions lying on the requested path.
- Given a goal how to compute the control commands for
- linear and angular velocities to reach the desired configuration

Problem statement

- Given arbitrary position and orientation of the robot $[x, y, \theta]$

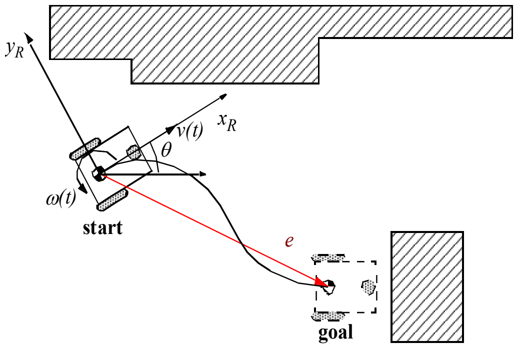
  how to reach desired goal orientation and position $[x_g, y_g, \theta_g]$



11

Move to Pose: Feedback Control, Problem Statement

3.6.2



- Find a control matrix K, if exists

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{bmatrix}$$

- with $k_{ij}=k(t,e)$
- such that the control of $v(t)$ and $\omega(t)$

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = K \cdot e = K \cdot {}^{R}\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

- drives the error e to zero.

$$\lim_{t \to \infty} e(t) = 0$$

- note previous slide the goal is set at zero

© R. Siegwart, I. Nourbakhsh

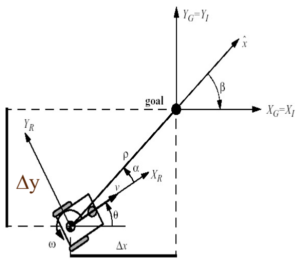
12

Move to Pose

3.6.2

• The kinematic of a differential drive mobile robot described in the initial frame $\{x_I, y_I, \theta\}$ is given by,

$${}^{I}\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$



where and are the linear velocities in the direction of the x_I and y_I of the initial frame.

Let α denote the angle between the x_R axis of the robots reference frame and the vector connecting the center of the axle of the wheels with the final position.

13

3.6.2

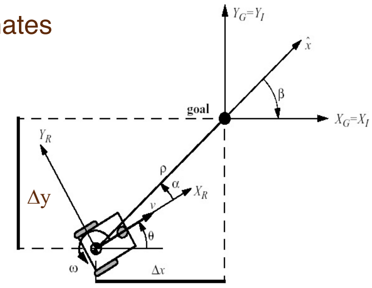
Move to Pose: Coordinates Transformation

Coordinates transformation into polar coordinates with its origin at goal position:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2}$$

$$\alpha = -\theta + \operatorname{atan2}(\Delta y, \Delta x)$$

$$\beta = -\theta - \alpha$$



System description, in the new polar coordinates

$$\begin{bmatrix} \dot\rho \\ \dot\alpha \\ \dot\beta \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ -\frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \qquad \begin{bmatrix} \dot\rho \\ \dot\alpha \\ \dot\beta \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 \\ -\frac{\sin\alpha}{\rho} & 1 \\ \frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

For α from $I_1 = \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$ for $I_2 = (-\pi, -\pi/2] \cup (\pi/2, \pi]$

14

3.6.2

Move to Pose: Remarks

- The coordinates transformation is not defined at $x = y = 0$; as in such a point the determinant of the Jacobian matrix of the transformation is not defined, i.e. it is unbounded
- For $\alpha \in I_1$ the forward direction of the robot points toward the goal, for $\alpha \in I_2$ it is the backward direction.
- By properly defining the forward direction of the robot at its initial configuration, it is always possible to have $\alpha \in I_1$ at $t = 0$. However this does not mean that α remains in I_1 for all time t.

15

3.6.2

Move to Pose: The Control Law

- It can be shown, that with

$$v = k_\rho \rho \qquad \omega = k_\alpha \alpha + k_\beta \beta$$

the feedback controlled system

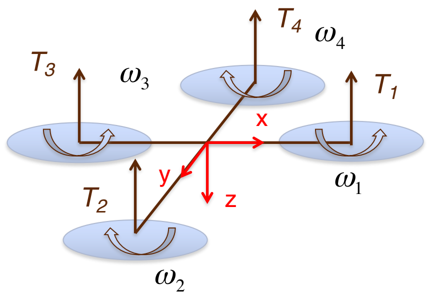
$$\begin{bmatrix} \dot\rho \\ \dot\alpha \\ \dot\beta \end{bmatrix} = \begin{bmatrix} -k_\rho \rho \cos\alpha \\ k_\rho \sin\alpha - k_\alpha \alpha - k_\beta \beta \\ -k_\rho \sin\alpha \end{bmatrix}$$

- will drive the robot to $(\rho, \alpha, \beta) = (0,0,0)$
- The control signal *v* has always constant sign,
  - the direction of movement is kept positive or negative during movement
  - parking maneuver is performed always in the most natural way and without ever inverting its motion.
  - Further details : How to select the constant parameters k's so as to achieve that the error will go to zero

16

Quadcopters model

- Popular unmanned areal vehicles (description adopted from (Robotics, Vision and control book, P. Corke http://www.petercorke.com/RVC/)



- Upward thrust $T_i = b\omega_i^2$ moving up in the negative z dir.
- Lift const. *b* depends on air density, blade radius and chord length

30

Quadcopters

- Translational dynamics (Newton's law – includes mass/acceleration/ forces) (Gravity – Total thrust (rotated to the world frame)

$$m\dot{v} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - R_B^0 \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}, \quad T = \sum_{i=1..4} T_i$$

- Rotations are generated by pairwise differences in rotor thrusts (*d* distance from the center)
- Rolling and pitching torques around *x* and *y*
- Torque in *z* – yaw torque

$$Q_i = k\omega_i^2$$

Torque applied by the motor as opposed to Aerodynamic drag

$$\tau_z = (Q_1 - Q_2 + Q_3 - Q_4)$$

$$\tau_x = dT_4 - dT_2$$

$$\tau_x = db(\omega_4^2 - \omega_2^2)$$

$$\tau_y = db(\omega_1^2 - \omega_3^2)$$

31

Quadcopter dynamics

- Rotational Dynamics, rot. acceleration in the airframe, Euler's eq. of motion

$$J\dot{\omega} = -\vec{\omega} \times J\vec{\omega} + \Gamma, \quad \Gamma = [\tau_x, \tau_y, \tau_z]^T$$

- Where *J is 3x3* inertia matrix
- Rotational Inertia of a body in 3D is represented by a 3x3 symmetric matrix J
- Diagonal elements are moments of inertia and off-diagonal are products of inertia
- Inertia matrix is a constant and depends on the mass and the shape of the body

$$J = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{xy} & J_{yy} & J_{yz} \\ J_{xz} & J_{yz} & J_{zz} \end{bmatrix}$$

32

Quadcopter dynamics

- Forces and torques acting of the airframe obtained integrating forward the eq. above and Newton's second law (prev. slide)

$$\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} -b & -b & -b & -b \\ 0 & -db & 0 & db \\ db & 0 & -db & 0 \\ k & -k & k & -k \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \qquad \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = A^{-1} \begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

- The goal of control is then derive proper thrust and torque to achieve desired goal – compute the rotor speeds
- Substitute these to translational and rotational dynamics and get forward dynamics equations of quadropter

33

Paths and Trajectories

- In general – control problem – need to generate set of control commands to accomplish the task
- In an open loop setting there are two components

1. Geometric Path Generation
2. Trajectory tracking (trajectory – time indexed path)

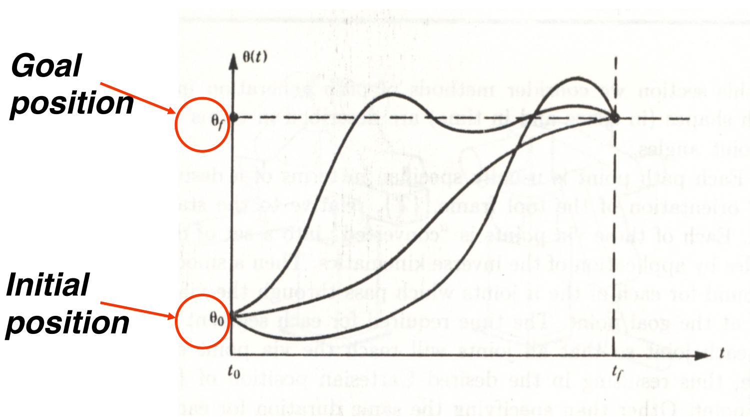
34

1D trajectories

- Trajectory, scalar function of time
- We want smooth trajectories
- Temporal derivatives
- Continuous velocity and acceleration profiles

35

Several Possible Path Shapes for a Single Joint



36

Cubic Polynomials

4 constraints on $\theta(t)$

$$\theta(0)=\theta_0,\quad \theta(t_f)=\theta_f,$$: initial and final values

$$\dot{\theta}(0)=0,\quad \dot{\theta}(t_f)=0.$$: the function is continuous in velocity

These 4 constraints can be satisfied by a polynomial of at least third degree.

$$\theta(t)=a_0+a_1t+a_2t^2+a_3t^3$$
$$\dot{\theta}(t)=a_1+2a_2t+3a_3t^2$$

37

Cubic Polynomials

$$\theta_0=a_0,$$
$$\theta_f=a_0+a_1t_f+a_2t_f^2+a_3t_f^3,$$
$$0=a_1,$$
$$0=a_1+2a_2t_f+3a_3t_f^2.$$

: combining with constraints

$$\boxed{\begin{aligned} a_0&=\theta_0,\\ a_1&=0,\\ a_2&=\frac{3}{t_f^2}(\theta_f-\theta_0),\\ a_3&=-\frac{2}{t_f^3}(\theta_f-\theta_0).\end{aligned}}$$

The cubic polynomial that connects any initial joint angle position with any desired final position when the joint starts and Finishes at zero velocity.

38

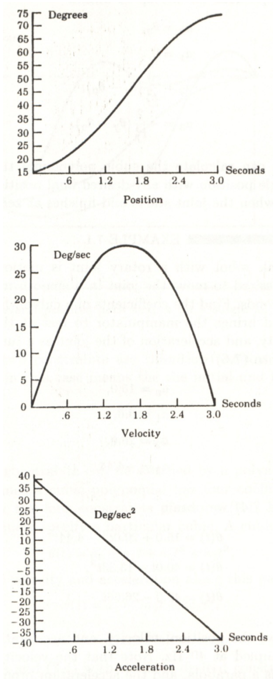
Example

A single-link robot with a rotary joint:

Move the joint in a smooth manner from θ=15 to θ=75 in 3 seconds

$$a_0 = 15.0$$
$$a_1 = 0.0$$
$$a_2 = 20.0$$
$$a_3 = -4.44$$

$$\theta(t) = 15.0 + 20.0t^2 - 4.44t^3$$
$$\dot{\theta}(t) = 40.0\,t - 13.33t^2$$
$$\ddot{\theta}(t) = 40.0 - 26.66\,t$$



1D trajectories

- Acceleration profile not smooth – higher order polynomial
- Continuous velocity and acceleration profiles

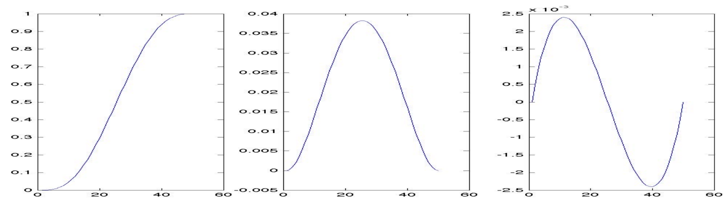
$$\theta(t) = at^5 + bt^4 + ct^3 + dt^2 + et + f$$
$$\dot{\theta}(t) = 5at^4 + 4bt^3 + 3ct^2 + 2dt + e$$
$$\ddot{\theta}(t) = 20at^3 + 12bt^2 + 5ct + 2d$$

- Given initial and final conditions for *t=0* and *t=T*

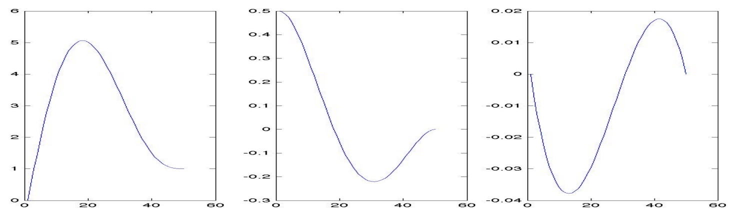
θ	$\dot{\theta}$	$\ddot{\theta}$
θ_0	$\dot{\theta}_0$	$\ddot{\theta}_0$
θ_T	$\dot{\theta}_T$	$\ddot{\theta}_T$

1-D trajectories

- Solve for coefficients, plot trajectories



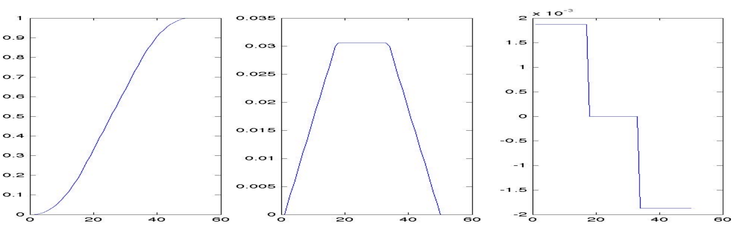
- Non-zero initial values – velocity overshoot at T



41

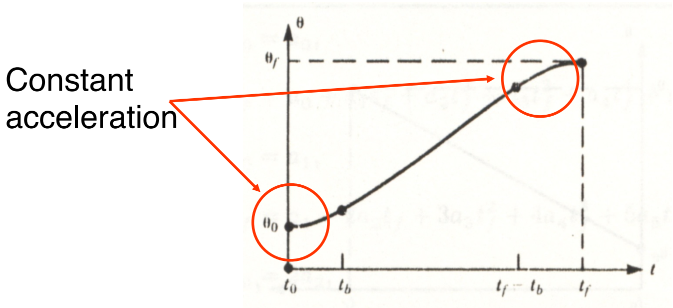
Problems with polynomials

- Overshoots velocity at final value T
- Velocity peaks in the middle, otherwise is far less then maximum
- We should like to have a flatter velocity profiles
- Solution: hybrid trajectories with polynomial segments for acceleration and de=acceleration
- Linear segments with parabolic blends (trapezoidal velocity profiles)



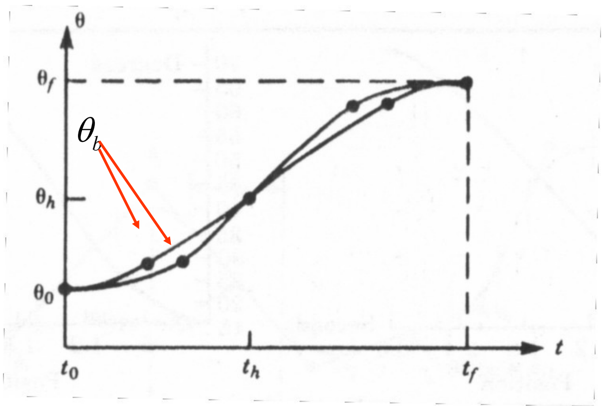
42

Linear Function With Parabolic Blends



The linear function and the two parabolic functions are splined together so that the entire path is continuous in position and velocity.

Linear Function With Parabolic Blends



The parabolic blends have the same duration, are symmetric about the halfway point in time, and the halfway point in position.

Linear Function With Parabolic Blends

$$\ddot{\theta} t_b = \frac{\theta_h - \theta_b}{t_h - t_b},$$

$$\theta_b = \theta_0 + \frac{1}{2}\ddot{\theta} t_b^2.$$

$$t = 2t_h$$

$$\ddot{\theta} t_b^2 - \ddot{\theta} t t_b + (\theta_f - \theta_0) = 0$$

$$t_b = \frac{t}{2} - \frac{\sqrt{\ddot{\theta}^2 t^2 - 4\ddot{\theta}(\theta_f - \theta_0)}}{2\ddot{\theta}}$$

$$\ddot{\theta} \geq \frac{4(\theta_f - \theta_0)}{t^2}.$$

: the constraints on acceleration

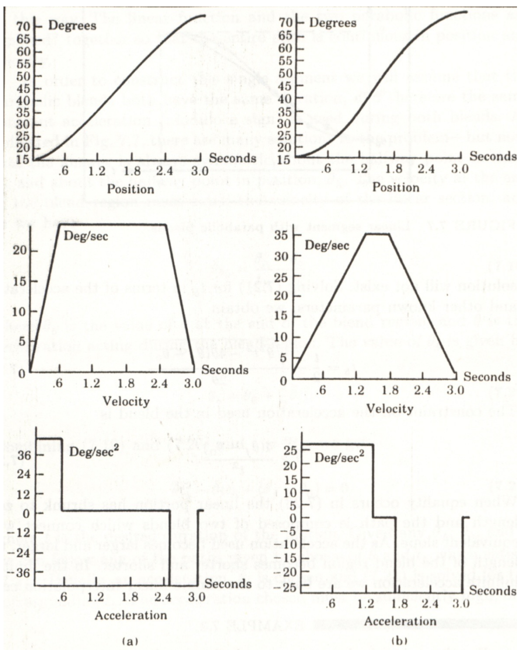
The velocity at the end of the blend region must equal the velocity of the linear section.

Usually acceleration is chosen and then solve for t_b

When equality occurs, the linear portion has shrunk to zero length.

45

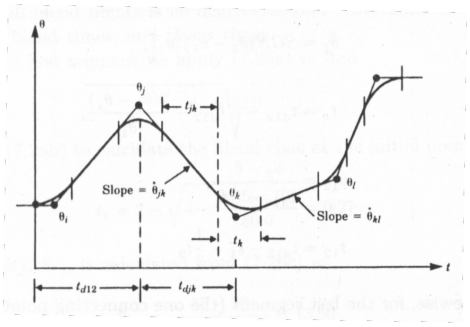
Two possible choices of linear path with parabolic blends



46

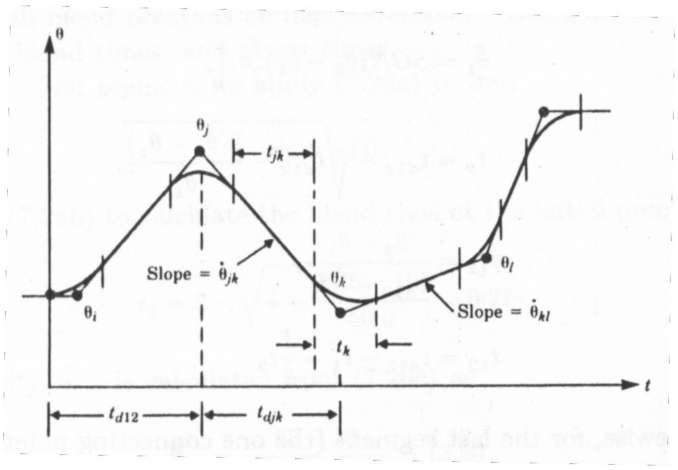
Multi-segment trajectories

- Often need to move through set of way points without stopping
- E.g. to avoid obstacles, or perform a task
- Over constrained problem, we need to surrender ability to reach every point



47

Linear Function With Parabolic Blends for a Path With Via Points



Linear function connects the via points and parabolic blend regions are around each via point

48

The Path Generator

- The results of computations constitute a plan for the trajectory. At execution time the path generator will use these numbers to compute θ, $\dot{\theta}$, $\ddot{\theta}$

49

Paths and Trajectories

- In general – control problem – need to generate set of control commands to accomplish the task
- In an open loop setting there are two components

1. Geometric Path Generation
2. Trajectory tracking (trajectory – time indexed path)

- Example omni-directional robot – can control all degress of freedom independently

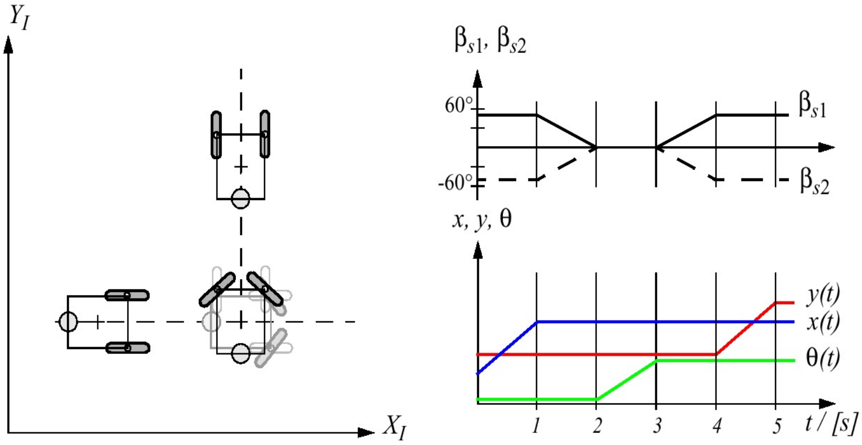
$$\delta_M = \delta_m + \delta_s = 3 + 0 = 3$$

50

3.4.3

Path / Trajectory Considerations: Two-Steer

Move for 1s with constant speed along X, rotate steered wheels by -50/50 degrees; change orientation counterclockwise by 90 degrees in 1s, move for 1s with constant speed along Y



51

Pose trajectories

- Examples so far: 1D trajectories (and velocity and acceleration profiles)
- Multi-segment 1D trajectories
- Multi-segment 2D trajectories comprised of lines and circles

- How to generate trajectory for rigid body so as to move from initial pose (R_0, T_0) to final pose (R_1, T_1)
- Interpolation
- Translation only case for $s = [0,1]$ generate intermediate translations as:

$$T = (1-s)T_0 + sT_1$$

52

Interpolation of rotations

- Interpolation of rotations

$$R = (1-s)R_0 + sR_1$$

- This won't work, rotation matrix properties are violated
- Spherical interpolation using quaternions
- Interpolation using exponential parametrization

$$\vec{\omega} = (1-s)\vec{\omega}_0 + s\vec{\omega}_1$$

- Similarly for full Rigid Body Motion

53

Incremental Motion

- Small incremental rotations

$$R_1 = (\hat{\omega}\sigma_t + I)R_0$$

- Inertial Navigation Systems
- Estimate velocity, orientation, and position wrt to inertial frame (frame of reference with respect to which is motion described)
- IMU – inertial measurement unit - measures accelerations and angular velocities and integrate them over time (3 orthogonally mounted gyros measure the angular velocity of the body)

54