ME 598: Introduction to Robotics

Lecture 4: Path and Trajectory Planning

Stevens Institute of Technology
Dr. Mishah U. Salman
Fall 2013

Date:
By:

Slides adapted from Dr. David J. Cappelleri
Some slides courtesy of Jonathan Fiene, University of Pennsylvania

Review: Jacobian

Review: Differential Motion

The Instantaneous Position Jacobian

$$\dot{\mathbf{p}} = \mathbf{J}_p(q)\ \dot{\mathbf{q}}$$

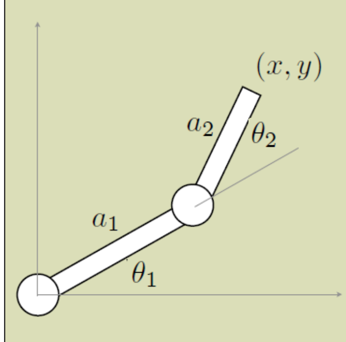
endpoint velocity — Jacobian matrix — joint velocity

For an n-dimensional joint variable space and a cartesian workspace, the Jacobian is a 3xn matrix composed of the partial derivatives of the end-effector position with respect to each joint variable.

$$\mathbf{J}_p = \begin{bmatrix} \frac{\delta x}{\delta q_1} & \frac{\delta x}{\delta q_2} & \cdots & \frac{\delta x}{\delta q_n} \\ \frac{\delta y}{\delta q_1} & \frac{\delta y}{\delta q_2} & \cdots & \frac{\delta y}{\delta q_n} \\ \frac{\delta z}{\delta q_1} & \frac{\delta z}{\delta q_2} & \cdots & \frac{\delta z}{\delta q_n} \end{bmatrix}$$

Reveiw: Position Jacobian

Example 1: Planar RR



From the forward kinematics, we can extract the position vector from the last column of the transform matrix:

$$\mathbf{d}_2^0 = \begin{bmatrix} a_2 c_{12} + a_1 c_1 \\ a_2 s_{12} + a_1 s_1 \\ 0 \end{bmatrix}$$

Taking the partial derivative with respect to each joint variable produces the Jacobian:

Review: Position Jacobian

Example 1: Planar RR

From the forward kinematics, we can extract the position vector from the last column of the transform matrix:

$$\mathbf{d}_2^0 = \begin{bmatrix} a_2 c_{12} + a_1 c_1 \\ a_2 s_{12} + a_1 s_1 \\ 0 \end{bmatrix}$$

Taking the partial derivative with respect to each joint variable produces the Jacobian:

$$= \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \end{bmatrix}$$

Review: Singularities

Singularities are points in the configuration space where infinitesimal motion in a certain direction is not possible and the manipulator loses one or more degrees of freedom

when operating at a singular point, bounded end-effector velocities may correspond to unbounded joint velocities

singularities are often found on the extents of the workspace,
and also relate to the nonuniqueness of solution to inverse kinematics

Mathematically, singularities exist at any point in the workspace where the Jacobian matrix loses rank.

[i.e. all columns of J are not linearly independent]

Review: Identifying Singularities

a matrix is singular if and only if it's determinant is zero:

$$\det(\mathbf{J}) = 0$$

The 2×2 matrix,

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

has determinant

$$\det(A) = ad - bc.$$

The 3×3 matrix:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}.$$

Using the cofactor expansion on the first row of the matrix we get:

$$\begin{aligned} \det(A) &= a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix} \\ &= aei - afh - bdi + bfg + cdh - ceg \\ &= (aei + bfg + cdh) - (gec + hfa + idb) \end{aligned}$$

[http://en.wikipedia.org/wiki/Determinant]

Review: Singularities

Example 1: Planar RR

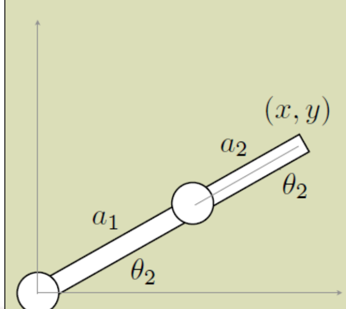
The 2×2 matrix,

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

has determinant

$$\det(A) = ad - bc.$$

$$\mathbf{J} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix}$$



Review: Jacobian Transpose

The transpose of the Jacobian relates joint torques and forces to cartesian end-effector forces

$$\tau = \mathbf{J}^{\top}(q)\ \mathbf{F}$$

joint torques

Jacobian matrix transpose

endpoint forces

Review: Inverse Jacobian

The Jacobian relationship:

$$\dot{\mathbf{p}} = \mathbf{J}_p(q)\ \dot{\mathbf{q}}$$

Specifies the end-effector velocity that will result when the joints move with velocity $\dot{\mathbf{q}}$

Inverse problem: Find the joint velocities $\dot{\mathbf{q}}$ that produce the desired end-effector velocity

$$\dot{\mathbf{q}} = \mathbf{J}_p(q)^{-1}\ \dot{\mathbf{p}}$$

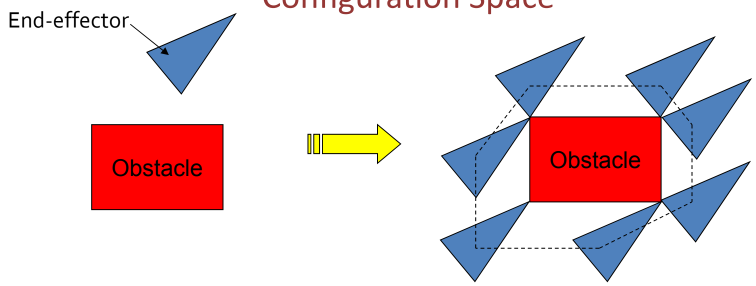
[Hard if have non-square J → pseudo-inverse (pinv)]

Path and Trajectory Planning

Path and Trajectory Planning: Introduction

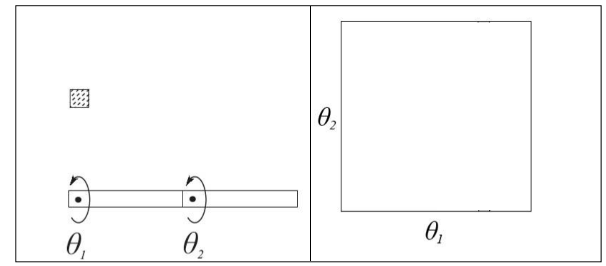
- Given:
  - Initial configuration of robot, q_s (initial joint coordinates)
  - Final configuration of robot, q_f (final joint coordinates)
- Goal:
  - Find a collision free path connecting q_s and q_f
- Path Planning
  - Provides geometric description (q) of the robot motion (no dynamics)
- Trajectory Planning
  - Provides time function to specify velocities and accelerations as robot moves along path q

Path and Trajectory Planning: Configuration Space



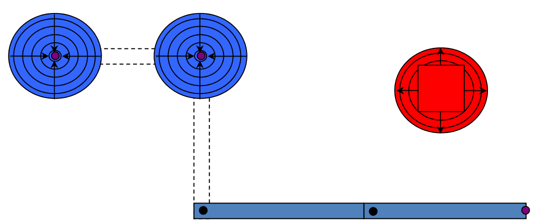
- Configuration Space = C
- Obstacle Configuration Space = CO
- Free Configuration Space = C – CO = C_{free}

Path and Trajectory Planning: Configuration Space



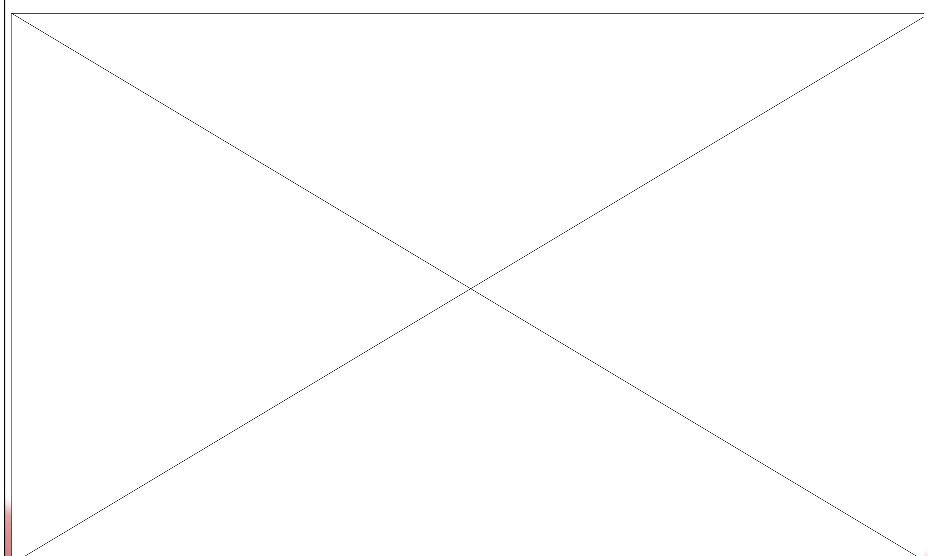
- Very expensive to compute exactly
- Complexity grow exponentially with degrees of freedom → intractable
- Need methods that avoid explicit construction of CO or C_{free}

Path and Trajectory Planning: Path Planning Using Potential Fields

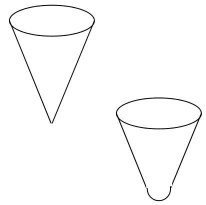


- Workspace potential fields
  - Attract the origins of the DH frames to goal locations while repelling them from obstacles
  - Used to define motions in configuration space with the manipulator Jacobians

Path and Trajectory Planning: Artificial Potential Fields



Path and Trajectory Planning: The Attractive Field



- Conic well potential – far away from goal
- Parabolic well potential – close to goal
- Workspace attractive Force = negative gradient of U_{att}

$$U_{\mathrm{att},i}(q) = \begin{cases} \frac{1}{2}\zeta_i\|o_i(q) - o_i(q_f)\|^2 & ; \ \|o_i(q) - o_i(q_f)\| \le d \\ d\zeta_i\|o_i(q) - o_i(q_f)\| - \frac{1}{2}\zeta_i d^2 & ; \ \|o_i(q) - o_i(q_f)\| > d \end{cases} \quad (5.3)$$

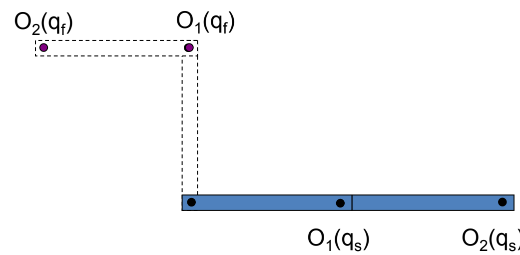
in which d is the distance that defines the transition from conic to parabolic well. In this case the workspace force for o_i is given by

$$F_{\mathrm{att},i}(q) = \begin{cases} -\zeta_i(o_i(q) - o_i(q_f)) & : \ \|o_i(q) - o_i(q_f)\| \le d \\ -d\zeta_i\frac{(o_i(q) - o_i(q_f))}{\|o_i(q) - o_i(q_f)\|} & : \ \|o_i(q) - o_i(q_f)\| > d \end{cases} \quad (5.4)$$

The gradient is well defined at the boundary of the two fields since at the boundary $d = \|o_i(q) - o_i(q_f)\|$ and the gradient of the quadratic potential is equal to the gradient of the conic potential $F_{\mathrm{att},i}(q) = -\zeta_i(o_i(q) - o_i(q_f))$.

Path and Trajectory Planning: Example- Attractive Field

$$q_s = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad q_f = \begin{bmatrix} \frac{\pi}{2} \\ \frac{\pi}{2} \end{bmatrix}$$



$$F_{\mathrm{att},i}(q) = \begin{cases} -\zeta_i(o_i(q) - o_i(q_f)) & : \ \|o_i(q) - o_i(q_f)\| \le d \\ -d\zeta_i\frac{(o_i(q) - o_i(q_f))}{\|o_i(q) - o_i(q_f)\|} & : \ \|o_i(q) - o_i(q_f)\| > d \end{cases} \quad (5.4)$$

Path and Trajectory Planning: The Repulsive Field

- Properties
  - Repel robot from obstacles, never allowing collisions
  - When robot far away, little/no influence on motion

ρ_o = distance of influence of an obstacle

$$U_{\mathrm{rep},i}(q) = \begin{cases} \frac{1}{2}\eta_i\left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0}\right)^2 & ; \ \rho(o_i(q)) \le \rho_0 \\ 0 & ; \ \rho(o_i(q)) > \rho_0 \end{cases} \quad (5.5)$$

in which $\rho(o_i(q))$ is the shortest distance between o_i and any workspace obstacle. The workspace repulsive force is equal to the negative gradient of $U_{\mathrm{rep},i}$. For $\rho(o_i(q)) \le \rho_0$, this force is given by (Problem 5-11)

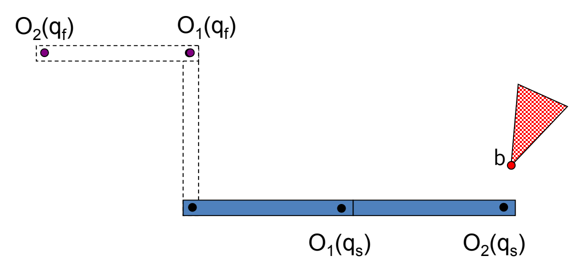
$$F_{\mathrm{rep},i}(q) = \eta_i\left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(o_i(q))}\nabla\rho(o_i(q)) \quad (5.6)$$

in which the notation $\nabla\rho(o_i(q))$ indicates the gradient $\nabla\rho(x)$ evaluated at $x = o_i(q)$. If the obstacle region is convex and b is the point on the obstacle boundary that is closest to o_i, then $\rho(o_i(q)) = \|o_i(q) - b\|$, and its gradient is

$$\nabla\rho(x)\Big|_{x=o_i(q)} = \frac{o_i(q) - b}{\|o_i(q) - b\|} \quad (5.7)$$

that is, the unit vector directed from b toward $o_i(q)$.

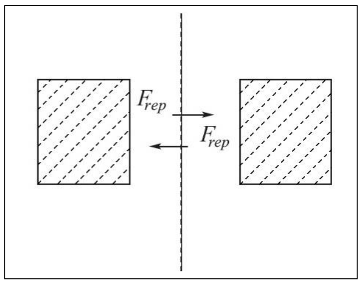
Path and Trajectory Planning: Example- Repulsive Field

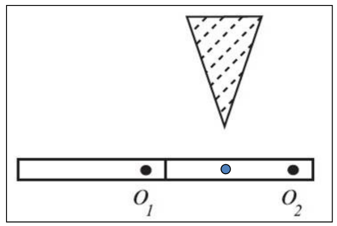


$$F_{\mathrm{rep},i}(q) = \eta_i\left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(o_i(q))}\nabla\rho(o_i(q)) \quad (5.6)$$

Path and Trajectory Planning: Repulsive Field

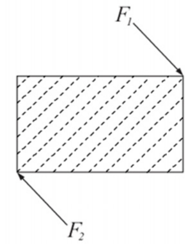
- Special cases to consider
  - Repulsive force discontinuities
  - Floating control points





Path and Trajectory Planning: Workspace Forces → Joint Torques

- Map workspace forces to configuration space ***before*** combining them



- Use Jacobians at each O_i

$$\tau = J^T F$$

$$\tau(q) = \sum_i J^T{}_{O_i}(q) F_{att,i}(q) + J^T{}_{O_i}(q) F_{rep,i}(q)$$

Path and Trajectory Planning: Gradient Descent Planning Algorithm

1. $q^0 = q_s$, $i = 0$
2. WHILE $||q^i - q_f|| > \varepsilon$

$$q^{i+1} = q^i + \alpha^i \frac{\tau(q^i)}{\|\tau(q^i)\|}$$

$$i = i + 1$$

3. END
4. Return [$q^0, q^1, \ldots, q^i$]

α = step size

ζ_i = controls the relative influence of attractive potential for O_i

η_i = controls the relative influence of repulsive potential for O_i

ρ_o = defines the distance of influence for obstacles

Path and Trajectory Planning: Escaping Local Minima

1. $q^0 = q_s$, $i = 0$
2. WHILE $||q^i - q_f|| > \varepsilon$

$$q^{i+1} = q^i + \alpha^i \frac{\tau(q^i)}{\|\tau(q^i)\|}$$

$$i = i + 1$$

- If stuck in local minimum
  - Execute Random walk, ending at q'

$$q^{i+1} = q'$$

3. END
4. Return [$q^0, q^1, \ldots, q^i$]

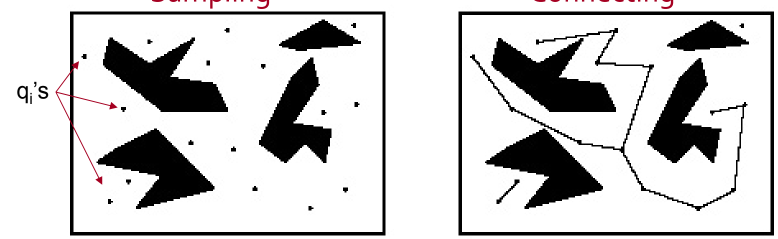
MATLAB demo

Path and Trajectory Planning: Probabilistic Roadmap Methods

- Potential Field approaches incrementally explore C_{free}
  - Single path
  - New goal location → entirely new path
- Alternative approach
  - Construct representation of C_{free} that can be quickly used to generate new paths
    - →Robots working long periods in single workspace

Path and Trajectory Planning: Probabilistic Roadmap Methods

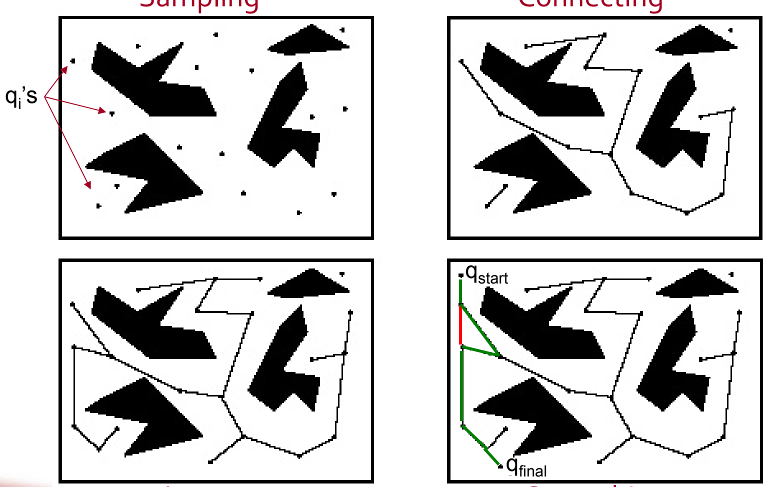
Sampling Connecting



2-norm in $\mathcal{Q}$:	$\lVert q' - q \rVert = \left[\sum_{i=1}^{n}(q'_i - q_i)^2\right]^{\frac{1}{2}}$
∞-norm in $\mathcal{Q}$:	$\max_n \lvert q'_i - q_i \rvert$
2-norm in workspace:	$\left[\sum_{p \in \mathcal{A}} \lVert p(q') - p(q) \rVert^2\right]^{\frac{1}{2}}$
∞-norm in workspace:	$\max_{p \in \mathcal{A}} \lVert p(q') - p(q) \rVert$

Path and Trajectory Planning: Probabilistic Roadmap Methods

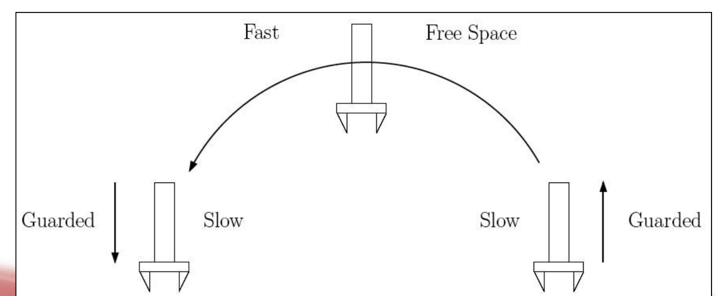
Sampling Connecting



Enhancing Smoothing

Path and Trajectory Planning: Trajectory Planning

- Path from q_s to q_f in C :
  - continuous map γ, with γ(0) = q_s and γ(1) = q_f
- Trajectory:
  - function of time q(t) such that $q(t_o) = q_s$ and $q(t_f) = q_f$
  - $t_f - t_0$ = time to execute trajectory
  - q'(t), q''(t) = velocity, acceleration
  - path planning only give sequence of points along q



Path and Trajectory Planning: Trajectories- Point to Point Motion

- n = # of constraints (pos/velocity/accel)
- Trajectory function = polynomial of degree n-1
- Cubic Polynomial Trajectories:

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

Constraints, n = 4:

q_o = initial position

v_o = initial velocity

q_f = final position

v_f = final velocity

Then the desired velocity is given as

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 \quad (5.20)$$

Combining Equations (5.19) and (5.20) with the four constraints yields four equations in four unknowns

$$\begin{aligned} q_0 &= a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \\ v_0 &= a_1 + 2a_2 t_0 + 3a_3 t_0^2 \\ q_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ v_f &= a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{aligned}$$

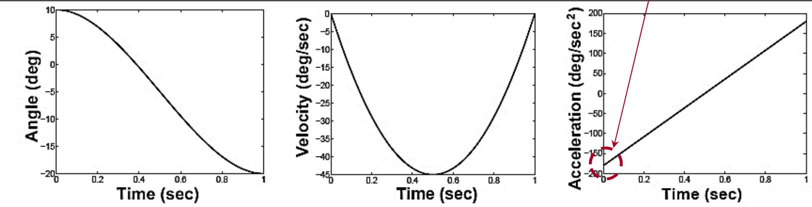
These four equations can be combined into a single matrix equation

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} \quad (5.21)$$

Path and Trajectory Planning: Example- Cubic Polynomial Trajectories

Derivative of acceleration = "jerk"

Discontinuities in acceleration



(a) (b) (c)

Figure 5.13: (a) Cubic polynomial trajectory. (b) Velocity profile for cubic polynomial trajectory. (c) Acceleration profile for cubic polynomial trajectory.

Path and Trajectory Planning: Quintic Polynomial Trajectories

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

Constraints, n = 6:

q_o = initial position

v_o = initial velocity

α_0 = initial acceleration

q_f = final position

v_f = final velocity

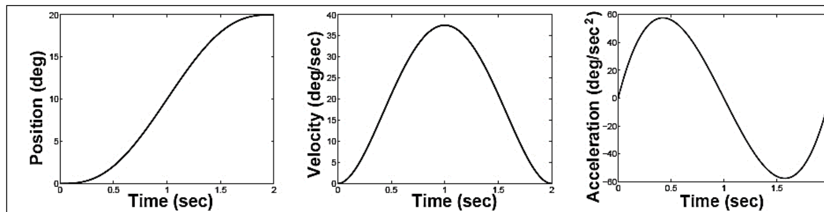
α_f = final acceleration

$$\begin{aligned} q_0 &= a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5 \\ v_0 &= a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4 \\ \alpha_0 &= 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3 \\ q_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \\ v_f &= a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \\ \alpha_f &= 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \end{aligned}$$

which can be written as

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ \alpha_0 \\ q_f \\ v_f \\ \alpha_f \end{bmatrix} \quad (5.23)$$

Path and Trajectory Planning: Quintic Polynomial Trajectories



(a) (b) (c)

Figure 5.14: (a) Quintic polynomial trajectory, (b) its velocity profile, and (c) its acceleration profile.

Path and Trajectory Planning: Linear Segments with Parabolic Blends

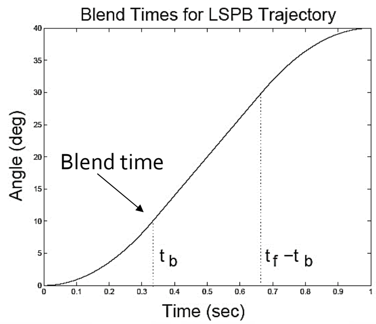
- Used when want constant velocities along portion of a path
- Trapezoidal velocity profile
  - Velocity initially ramped up to desired value
  - Ramped down when close to goal position

Quadratic polynomial

Linear

$$q(t) = \begin{cases} q_0 + \frac{\alpha}{2}t^2 & 0 \le t \le t_b \\ \frac{q_f + q_0 - Vt_f}{2} + Vt & t_b < t \le t_f - t_b \\ q_f - \frac{\alpha t_f^2}{2} + \alpha t_f t - \frac{\alpha}{2}t^2 & t_f - t_b < t \le t_f \end{cases} \quad (5.25)$$

Quadratic polynomial



Path and Trajectory Planning: Linear Segments with Parabolic Blends

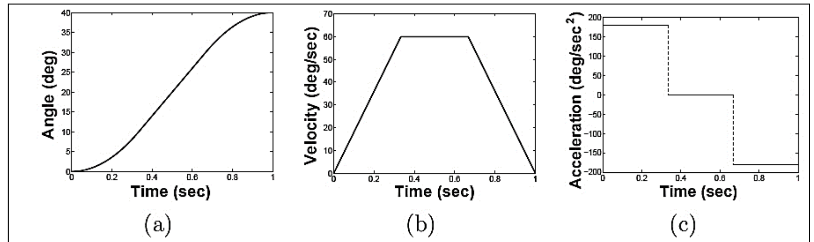


Figure 5.16: (a) LSPB trajectory. (b) Velocity profile for LSPB trajectory. (c) Acceleration profile for LSPB trajectory.

Path and Trajectory Planning: Minimum Time Trajectories

- Variation of LSPB
- Leave t_f unspecified, seek fastest trajectory between q_0 and q_f with given constant acceleration α
  - Trajectory with minimum t_f
  - Max acceleration (+)α until t_{switch}
  - Min acceleration (-)α from t_{switch} to t_f

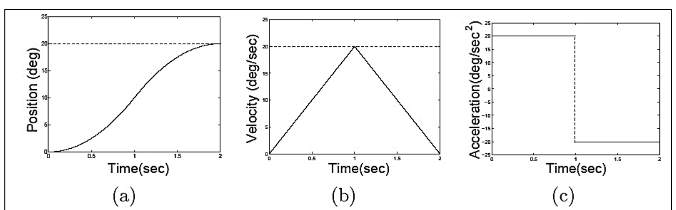


Figure 5.17: (a) Minimum-time trajectory. (b) Velocity profile for minimum-time trajectory. (c) Acceleration profile for minimum-time trajectory.

Path and Trajectory Planning: Trajectories for Paths Specified by multiple (Via) Points

- Path specified by three configurations, q_0, q_1, and q_2 such that they are reached at times t_o, t_1, and t_2
- Additional constraints on initial and final velocities

Constraints, n=7

$q(t_o) = q_o$
$q'(t_o) = v_o$
$q''(t_o) = \alpha_o$
$q(t_1) = q_1$
$q(t_2) = q_2$
$q'(t_2) = v_2$
$q''(t_2) = \alpha_2$

Sixth order polynomial trajectory:

$q(t) = a_o + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + a_6 t^6$

(+) Continuous everywhere

(-) Solve 7-dimensional linear system

number of dimensions scales with number of q's

Path and Trajectory Planning: Trajectory For Paths W/ Multiple Points

- Use lower order polynomials for trajectory segments between adjacent points
- Require velocity and acceleration constraints at points where switch from one polynomial to another
- For each segment:

Constraints, n=4

$q(t_o) = q_o$

$q'(t_o) = v_o$

$q(t_f) = q_1$

$q'(t_f) = v_2$

For sequence of moves:

Use end conditions q_f and v_f of the i^{th} move as initial conditions for next move

Cubic polynomial trajectory:

$q(t) = a_o + a_1 (t-t_o) + a_2 (t-t_o)^2 + a_3 (t-t_o)^3$

where:

$$a_0 = q_0$$

$$a_1 = v_0$$

$$a_2 = \frac{3(q_1 - q_0) - (2v_0 + v_1)(t_f - t_o)}{(t_f - t_o)^2}$$

$$a_3 = \frac{2(q_0 - q_1) + (v_0 + v_1)(t_f - t_o)}{(t_f - t_o)^3}$$