Review: Path and Trajectory Planning

- Given:
  - Initial configuration of robot, $q_i$ (initial joint coordinates)
  - Final configuration of robot, $q_f$ (final joint coordinates)
- Goal:
  - Find a collision free path connecting $q_i$ 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 $q$

Review: 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
Review: The Attractive Field

- Conic well potential – far away from goal
- Parabolic well potential – close to goal

Workspace attractive Force = negative gradient of $U_{\text{attr}}$

$$U_{\text{attr}}(q) = \begin{cases} \frac{1}{2}d([q] - [\bar{q}])^2 & : ||[q] - [\bar{q}]|| \leq d \\ \frac{1}{2}d^2 & : ||[q] - [\bar{q}]|| > d \end{cases}$$

(5.3)

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

$$F_{\text{attr}}(q) = \begin{cases} -\zeta([q] - [\bar{q}]) & : ||[q] - [\bar{q}]|| \leq d \\ -\eta([q] - [\bar{q}]) & : ||[q] - [\bar{q}]|| > d \end{cases}$$

(5.4)

The gradient is well defined at the boundary of the two fields since at the boundary $d = ||[q] - [\bar{q}]||$ and the gradient of the quadratic potential is equal to the gradient of the conic potential $F_{\text{attr}}(q) = -\zeta([q] - [\bar{q}])$.

Review: The Repulsive Field

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

$$p_o = \text{distance of influence of an obstacle}$$

$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2p_o}(\frac{1}{\rho_o(q)} - 1)^2 & : \rho_o(q) \leq \rho_o \\ 0 & : \rho_o(q) > \rho_o \end{cases}$$

(5.5)

in which $\rho_o(q)$ is the shortest distance between $a_i$ and any workspace obstacle. The workspace repulsive force is equal to the negative gradient of $U_{\text{rep}}$. For $\rho_o(q) \leq \rho_o$, this force is given by

$$F_{\text{rep}}(q) = \eta \frac{1}{\rho_o(q)} \frac{1}{\rho_o(q)} \nabla \rho_o(q)$$

(5.6)

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

$$\nabla \rho_o(q) = \frac{a_i - b}{||b - q||}$$

(5.7)

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

Review: Workspace Forces $\rightarrow$ Joint Torques

- Map workspace forces to configuration space before combining them

$$\tau = J^T F(q)$$

$$\tau(q) = \sum_i J^T \alpha_i(q) F_{\text{attr}}(q) + J^T \alpha_i(q) F_{\text{rep}}(q)$$

Review: Gradient Descent Planning Algorithm

1. $q^0 = q_{s_i}$, $i = 0$
2. WHILE $||q^i - q_f|| > \varepsilon$

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

$i = i + 1$
3. END
4. Return $[q^0, q^1, ..., q^i]$
Review: Probabilistic Roadmap Methods

Sampling

Connecting

Enhancing

Smoothing

Review: Trajectory Planning

- Path from $q_s$ to $q_f$ in $C$
  - continuous map $\gamma$, with $\gamma(0) = q_s$ and $\gamma(1) = q_f$
- Trajectory:
  - function of time $q(t)$ such that $q(t_0) = q_s$ 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 $q$

Review: Trajectory Planning

- Path from $q_s$ to $q_f$ in $C$
  - continuous map $\gamma$, with $\gamma(0) = q_s$ and $\gamma(1) = q_f$
- Trajectory:
  - function of time $q(t)$ such that $q(t_0) = q_s$ 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 $q$

Review: Trajectory Planning

- Path from $q_s$ to $q_f$ in $C$
  - continuous map $\gamma$, with $\gamma(0) = q_s$ and $\gamma(1) = q_f$
- Trajectory:
  - function of time $q(t)$ such that $q(t_0) = q_s$ 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 $q$

Review: Trajectory Planning

- Path from $q_s$ to $q_f$ in $C$
  - continuous map $\gamma$, with $\gamma(0) = q_s$ and $\gamma(1) = q_f$
- Trajectory:
  - function of time $q(t)$ such that $q(t_0) = q_s$ 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 $q$

Review: Trajectory Planning

- Path from $q_s$ to $q_f$ in $C$
  - continuous map $\gamma$, with $\gamma(0) = q_s$ and $\gamma(1) = q_f$
- Trajectory:
  - function of time $q(t)$ such that $q(t_0) = q_s$ 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 $q$

Review: Trajectory Planning

- Path from $q_s$ to $q_f$ in $C$
  - continuous map $\gamma$, with $\gamma(0) = q_s$ and $\gamma(1) = q_f$
- Trajectory:
  - function of time $q(t)$ such that $q(t_0) = q_s$ 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 $q$

Review: Trajectory Planning

- Path from $q_s$ to $q_f$ in $C$
  - continuous map $\gamma$, with $\gamma(0) = q_s$ and $\gamma(1) = q_f$
- Trajectory:
  - function of time $q(t)$ such that $q(t_0) = q_s$ 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 $q$
Review:
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:

    Cubic polynomial trajectory:
    \[ q(t) = a_0 + a_1 (t-t_0) + a_2 (t-t_0)^2 + a_3 (t-t_0)^3 \]
    where:
    \[ a_0 = q_0 \]
    \[ a_1 = v_0 \]
    \[ a_2 = \frac{3(q_i - q_0) - (2v_0 + v_i)(t_f - t_i)}{(t_f - t_i)^2} \]
    \[ a_3 = \frac{2(q_i - q_0) + (v_0 + v_i)(t_f - t_i)}{(t_f - t_i)^3} \]

For sequence of moves:
Use end conditions \( q_i \) and \( v_i \) of the \( i^{th} \) move as initial conditions for next move

Control of Manipulators

Reference:

Control of Manipulators:
Open-Loop Introduction

- Linear control – system modeled by linear differential equations
  \[ \tau = M(q_d)\ddot{q}_d + V(q_d, \dot{q}_d) + G(q_d) \]
- Open-loop
  - Only function of \( q_d \)
  - Not a function of \( q \), actual trajectory

Control of Manipulators:
Closed-Loop Introduction

Closed-loop
- Use feedback from joint sensors, \( q \) and \( q' \)
- Feedback used to compute servo error:
  \[ E = q_d - q \]
  \[ \dot{E} = \dot{q}_d - \dot{q} \]
- Control system computes how much torque to send actuators as a function of \( E \), \( E' \)
Control of Manipulators: Introduction

- **Goal:**
  - Design a closed-loop system that is stable
  - Errors remain “small” when tracking various desired trajectories even in presence of “moderate” disturbances
  - Meets performance objectives for particular application

Control of Manipulators: Approximation

- **Approximation:**
  - Treat each joint as separate system to be controlled
  - N-jointed manipulator, N-independent single-input, single-output (SISO) control systems

Control of Manipulators: Second-Order Linear Systems

- **Equation of motion:**
  \[ m\ddot{x} + b\dot{x} + kx = 0 \]
- **Find solution:** \( x(t) \)
  - Form of solution depends on roots of characteristic equation:
  \[ ms^2 + bs + k = 0 \]
Control of Manipulators: (Stable) Second-Order Linear Systems

- Roots (poles):
  \[ s_1 = -\frac{b}{2m} + \frac{\sqrt{b^2 - 4mk}}{2m} \]
  \[ s_2 = -\frac{b}{2m} - \frac{\sqrt{b^2 - 4mk}}{2m} \]

- Three cases for roots (if system is stable!):
  1. Real and Unequal: \( b^2 > 4mk \), friction dominates, sluggish behavior results \( \rightarrow \) **overdamped**
  2. Complex: \( b^2 < 4mk \), stiffness dominates, oscillatory behavior results \( \rightarrow \) **underdamped**
  3. Real and equal: \( b^2 = 4mk \), friction and stiffness are balanced, fastest possible nonoscillatory response \( \rightarrow \) **critically damped**

Control of Manipulators: Case 1- Overdamped System

- Solution:
  \[ x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t} \]
  - \( S_1 \) and \( S_2 \) are real and unequal
  - \( c_1 \) and \( c_2 \) constants determined from initial conditions, i.e. initial position and velocity of block

Control of Manipulators: Case 2- Underdamped System

- Solution:
  \[ s_1 = \lambda + \mu i, \quad s_2 = \lambda - \mu i \]
  \[ x(t) = c_1 e^{\lambda t} + c_2 e^{\mu i} \]
  \[ e^{\mu i} = \cos(\mu t) + i \sin(\mu t) \]
  \[ \therefore \quad x(t) = c_1 e^{\lambda t} \cos(\mu t) + c_2 e^{\mu i} \sin(\mu t) \]
  - \( S_1 \) and \( S_2 \) are complex (conjugate pair)
  - \( c_1 \) and \( c_2 \) constants determined from initial conditions, i.e. initial position and velocity of block
  \[ c_1 = r \cos(\delta), \quad c_2 = r \sin(\delta) \]
  \[ x(t) = re^{\lambda t} \cos(\mu t - \delta) \]
  where \( r = \sqrt{c_1^2 + c_2^2} \), \( \delta = \tan^{-1}(c_2, c_1) \)
Control of Manipulators:
Case 3- Critically Damped System

• Solution:

\[ x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t} \]
\[ s_1 = s_2 = -\frac{b}{2m} \]
\[ \therefore x(t) = (c_1 + c_2 t) e^{-\frac{b}{2m} t} \]

– \( s_1 \) and \( s_2 \) are real and equal (repeated roots)

– \( c_1 \) and \( c_2 \) constants determined from initial conditions,
i.e. initial position and velocity of block

Control of Manipulators:
Second Order System

• Alternative representation:
  – Parameterize characteristic equation by:
    \[ s^2 + 2\zeta \omega_n s + \omega_n^2 = 0 \]
    \[ \zeta = \text{damping ratio} \]
    \[ \omega_n = \text{natural frequency} \]

  – Relationship to pole locations: \( s_1 = \lambda + \mu j \), \( s_2 = \lambda - \mu j \)
    \[ \lambda = -\zeta \omega_n, \quad \mu = \omega_n \sqrt{1 - \zeta^2} = \text{damped natural frequency} \]

  – For this spring-mass-damp system:
    \[ \zeta = \frac{b}{2\sqrt{km}}, \quad \omega_n = \sqrt{\frac{\sqrt{km}}{m}} \]
    No damping: \( b = 0, \zeta = 0 \)
    Critically damped, \( (b^2=4km), \zeta = 1 \)

Control of Manipulators:
Control of Second Order Linear Systems

• Equation of motion:
  \[ m\ddot{x} + b\dot{x} + kx = 0 \]

• Control law as a function of sensed feedback:
  \[ f = -k_p x - k_v \dot{x} \]

Position regulation system: maintains the position of the block in one fixed place regardless of disturbance forces applied to the block.
Control of Manipulators:
Control of Second-Order Linear Systems

(1) \[ m\ddot{x} + b\dot{x} + kx = f \]
(2) \[ f = -k_p x - k_v \dot{x} \]

• Plugging (1) into (2):
  \[ m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x} \]
  \[ m\ddot{x} + (b + k_v)\dot{x} + (k + k_p)x = 0 \]
  \[ m\ddot{x} + b'\dot{x} + k'x = 0 \]
  where \( b' = b + k_v \) and \( k' = k + k_p \)

• Choose control gains, \( k_v \) and \( k_p \), to cause system to have any second order system behavior that is desired:
  critically damped: \( b' = 2\sqrt{mk'} \)
  closed loop stiffness: \( k' \)

---

Control of Manipulators:
Control Law Partitioning

- Model-based portion
  - Contains system parameters (m, b, and k)
  - Reduces system so it appears to be a unit mass
- Servo portion
  - Independent of system parameters
  - Uses feedback to modify behavior of system

Equation of motion: \( m\ddot{x} + b\dot{x} + kx = f \)
Control law: \( f = \alpha f' + \beta \)
  \( f' \) = new input to system
\( \alpha \) and \( \beta \) chosen so system appears to be a unit mass

---

Control of Manipulators:
Control Law Partitioning

\[ m\dddot{x} + b\ddot{x} + kx = \alpha f' + \beta \]
Choose: \( \alpha = m, \quad \beta = b\dot{x} + kx \)
After substitution: \( \ddot{x} = f' \rightarrow \) Equation of motion for unit mass
Control law: \( f' = -k_v \dot{x} - k_p x \)
After substitution: \( \ddot{x} + k_v \dot{x} + k_p x = 0 \)
For critical damping: \( k_v = 2\sqrt{k_p} \)

---

Control of Manipulators:
Trajectory Following-Control

Desired Trajectory: \( x_d(t) \)
Given: \( x_d, \dot{x}_d, \ddot{x}_d \)

Servo Error: \( e = x_d - x \)
Control law: \( f' = \ddot{x}_d + k_v \dot{e} + k_p e \)
After substitution: \( \ddot{e} + k_v \dot{e} + k_p e = 0 \)
For critical damping: \( k_v = 2\sqrt{k_p} \)
Control of Manipulators: Disturbance Rejection

\[ \ddot{e} + k_v \dot{e} + k_p e = f_{\text{dist}} \]

Control law: \( f = \ddot{x}_d + k_v \dot{e} + k_p e + k_i \int e \, dt \)

At steady-state: \( e = 0 \)

Control of Manipulators: Modeling and Control of a Single Joint

- Model single rotary joint of manipulator as second-order linear system

DC torque motor

Control of Manipulators: Electrical Model of DC Motor Armature

- Armature circuit modeled by first-order differential equation
  \[ L_a \dot{i}_a + r_a i_a = v_a - k_e \dot{\theta}_m \]

- Use circuitry to control motor torque (rather than velocity)
  - Current amplifier motor driver: sense \( i_a \) and adjust \( v_a \) to get desired \( i_a \)
  - Rate at which \( i_a \) can be commanded is limited by \( L_a \) and \( v_a \)

- Simplifying assumption: neglect \( L_a \rightarrow \) actuator acts as pure torque source that we can command directly
Control of Manipulators: Mechanical Model of DC Motor Rotor

\[ \tau_m = \text{torque applied to rotor} \]
\[ \tau = \eta \tau_m \]
\[ i_a = \text{armature current} \]
\[ \eta = \text{gear ratio} \]
\[ I_m, I = \text{motor and load inertias} \]
\[ b_m, b = \text{rotor and load bearings viscous friction coefficients} \]

\[ \tau_m = I_m \ddot{\theta}_m + b_m \dot{\theta}_m + (\gamma \eta) (I \dot{\theta} + b \dot{\theta}) \]
\[ \dot{\theta} = (\gamma \eta) \dot{\theta}_m \]

In terms of load variables:

\[ \tau_m = \left( I_m + \frac{I}{\eta^2} \right) \ddot{\theta}_m + \left( b_m + b \frac{\eta^2}{\eta^2} \right) \dot{\theta}_m \]

Effective inertia 
Effective damping

For highly geared joints (\( \eta \gg 1 \)), \( I_m \) dominates \( \rightarrow \) can assume effective inertia term is a constant.

To ensure link motion is never underdamped, set \( I \) to \( I_{\text{max}} \) for application

Control of Manipulators: Unmodeled Resonances

- Assumption: gearings, shafts, bearings, and link are rigid, not flexible
  - If system is sufficiently stiff, natural freq of these unmodeled resonances are very high and can be neglected compared to influence of the second-order poles
- If lowest structural resonance is \( \omega_{\text{res}} \), need to limit closed-loop natural frequency:
  \[ \omega_n \leq \frac{1}{2} \omega_{\text{res}} \]
  - This will limit the magnitudes for some of the gains that we choose in our controller design
  - For \( k = \text{stiffness of flexible member}, m = \text{equivalent mass} \), estimate \( \omega_{\text{res}} \) as:
    \[ \omega_{\text{res}} = \sqrt{\frac{k}{m}} \]

Control of Manipulators: Control of a Single Joint

- Assumptions:
  1. Neglect motor inductance \( L_a \)
  2. High gearing, effective inertia is constant: \( I_{\text{max}} + \eta^2 I_m \)
  3. Structural flexibilities are neglected; use the lowest one, \( \omega_{\text{res}} \) to set the servo gains
- Use partitioned controller design:

\[ \alpha = I_{\text{max}} + \eta^2 I_m \]
\[ \beta = b + \eta^2 b_m \]

control law: \( \tau^* = \ddot{\theta}_d + k \dot{e} + k_p e \)

Closed-loop dynamics:

\[ \dot{e} + k \dot{e} + k_p e = \tau_{\text{dist}} \]

Gains:

\[ k_p = \omega_{\text{res}}^2 = \frac{1}{4} \omega_{\text{res}}^2, \quad k_v = 2 \sqrt{k_p} = \omega_{\text{res}} \]
Control of Manipulators: Unimation PUMA 560 Control System

Computer - Interprets motion command, perform inverse kinematic computations, plan desired trajectory, generate trajectory via points every 28 ms

Microprocessors – get position commands every 28 ms

Microprocessors – run at 0.875 ms cycle
Interpolate desired position, compute servo error, PID control law, command new torque value

Optical encoder – measures joint position; joint position differenced on subsequent cycles to estimate velocity

D/A chip converts processor commands to signal for current driver circuits
Current is controlled by adjusting voltage across the armature as needed