ME 598: Introduction to Robotics

Lecture 5: Control of Manipulators

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

Date:
By:

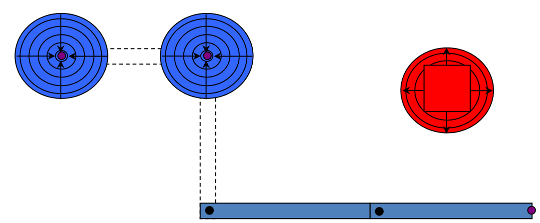
Slides adapted from Dr. David J. Cappelleri

Review: Trajectory & Path Planning

Review: Path and Trajectory Planning

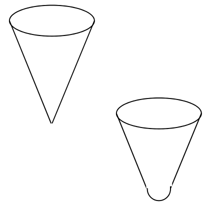
- Given:
  - Initial configuration of robot, q_s (initial joint coordinates)
  - Final configuration of robot, q_f (final joint coordinates)
- Goal:
  - Find a collision free path connecting q_s and q_f
- Path Planning
  - Provides geometric description (q) of the robot motion (no dynamics)
- Trajectory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

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_{att}

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

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

$$F_{\mathrm{att},i}(q) = \begin{cases} -\zeta_i(o_i(q) - o_i(q_{\mathrm{f}})) & : \ \|o_i(q) - o_i(q_{\mathrm{f}})\| \le d \\ -d\zeta_i\dfrac{(o_i(q) - o_i(q_{\mathrm{f}}))}{\|o_i(q) - o_i(q_{\mathrm{f}})\|} & : \ \|o_i(q) - o_i(q_{\mathrm{f}})\| > d \end{cases} \qquad (5.4)$$

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

Review:
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

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

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)

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

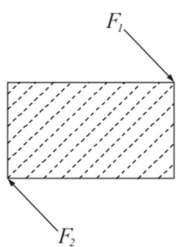
in which the notation $\nabla\rho(o_i(q))$ indicates the gradient $\nabla\rho(x)$ evaluated at $x = o_i(q)$. If the obstacle region is convex and b is the point on the obstacle boundary that is closest to o_i, then $\rho(o_i(q)) = \|o_i(q) - b\|$, and its gradient is

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

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

Review:
Workspace Forces → Joint Torques

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



- Use Jacobians at each O_i

$$\tau = J^T F$$

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

Review:
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 [$q^0, q^1, \ldots, q^i$]

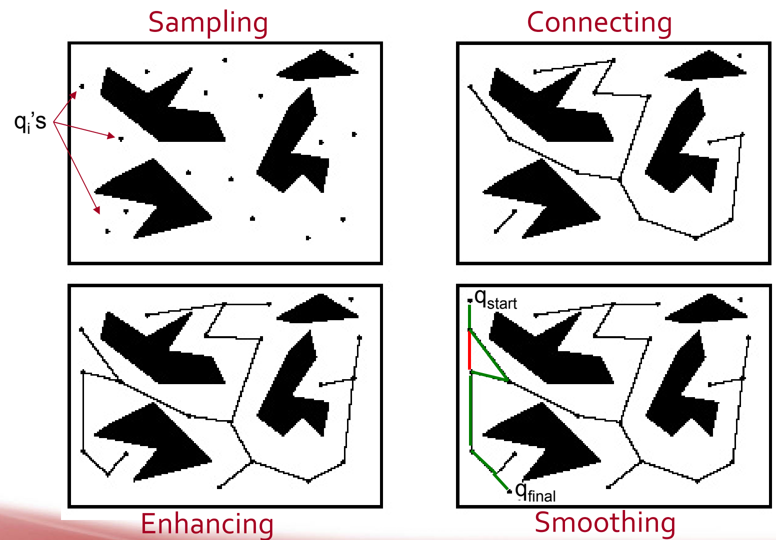
α = step size

ζ_i = controls the relative influence of attractive potential for O_i

η_i = controls the relative influence of repulsive potential for O_i

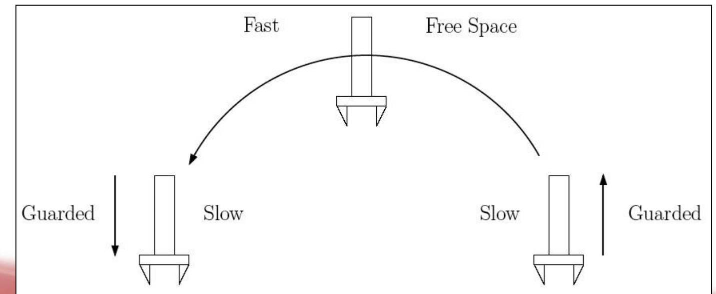
ρ_o = defines the distance of influence for obstacles

Review:
Probabilistic Roadmap Methods



Review: 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



Review:
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

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

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

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

Review:
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

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

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:

Constraints, n=4

$q(t_o) = q_o$

$q'(t_o) = v_o$

$q(t_f) = q_1$

$q'(t_f) = v_2$

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:

$$a_0 = q_0$$
$$a_1 = v_0$$
$$a_2 = \frac{3(q_1 - q_0) - (2v_0 + v_1)(t_f - t_o)}{(t_f - t_o)^2}$$
$$a_3 = \frac{2(q_0 - q_1) + (v_0 + v_1)(t_f - t_o)}{(t_f - t_o)^3}$$

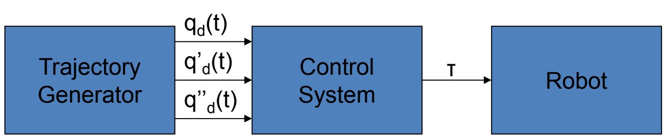
For sequence of moves:

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

Control of Manipulators

Reference:
J. Craig, Chapter 9: Linear Control of Manipulators, Introduction to Robotics: Mechanics and Control, 2nd Edition, Addison-Wesley Publishing Company, New York, 1989.

Control of Manipulators: Open-Loop Introduction

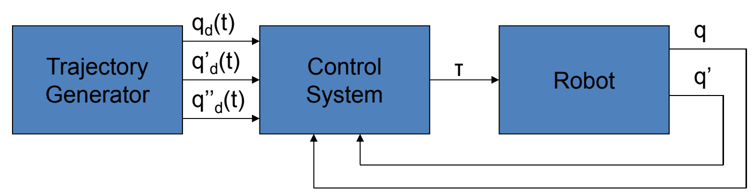


- Linear control – system modeled by linear differential equations

$$\tau = M(q_d)\ddot{q}_d + V(q_d, \dot{q}_d) + G(q_d)$$

- 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:

$$E = q_d - q$$
$$\dot{E} = \dot{q}_d - \dot{q}$$

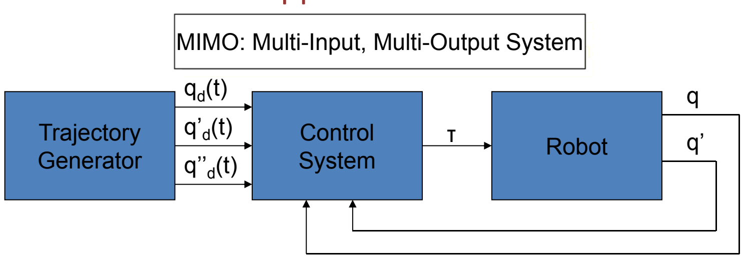
- Control system computes how much torque to send actuators as a function of E, E'

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

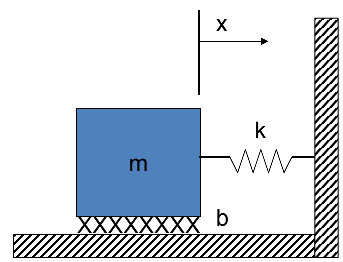
Control of Manipulators: Instability Demonstration

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):

$$s_1 = -\frac{b}{2m} + \frac{\sqrt{b^2 - 4mk}}{2m}$$

$$s_2 = -\frac{b}{2m} - \frac{\sqrt{b^2 - 4mk}}{2m}$$

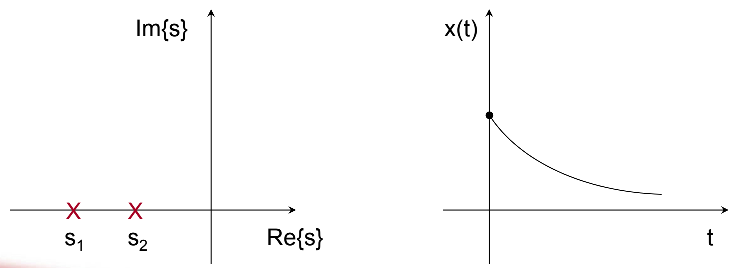
- Three cases for roots (if system is stable!):
  1. Real and Unequal: $b^2 > 4mk$, friction dominates, sluggish behavior results → **overdamped**
  2. Complex: $b^2 < 4mk$, stiffness dominates, oscillatory behavior results → **underdamped**
  3. Real and equal: $b^2 = 4mk$, friction and stiffness are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

$$s_1 = \lambda + \mu i, \quad s_2 = \lambda - \mu i$$

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}$$

$$e^{ix} = \cos(x) + i\sin(x)$$

$$\therefore \quad x(t) = c_1 e^{\lambda t}\cos(\mu t) + c_2 e^{\lambda t}\sin(\mu t)$$

  - 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

$$c_1 = r\cos(\delta), \quad c_2 = r\sin(\delta)$$

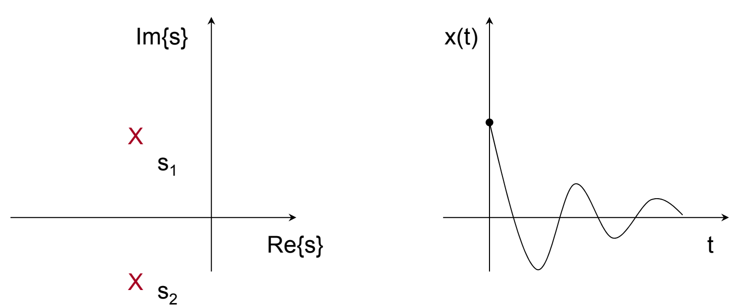
$$x(t) = re^{\lambda t}\cos(\mu t - \delta)$$

$$where \quad r = \sqrt{c_1^{\,2} + c_2^{\,2}}, \quad \delta = a\tan 2(c_2, c_1)$$

Control of Manipulators: Case 2- Underdamped System

$$s_1 = \lambda + \mu i, \quad s_2 = \lambda - \mu i$$

$$x(t) = c_1 e^{\lambda t}\cos(\mu t) + c_2 e^{\lambda t}\sin(\mu t)$$



Control of Manipulators: Case 3- Critically Damped System

- Solution:

$$x(t) = c_1 e^{s_1 t} + c_2 t e^{s_2 t}$$

$$s_1 = s_2 = -\frac{b}{2m}$$

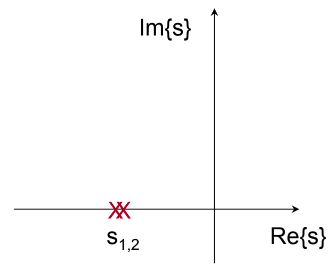
$$\therefore \quad x(t) = (c_1 + c_2 t) e^{-\frac{b}{2m}t}$$

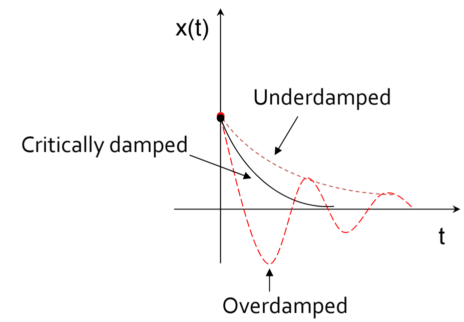
  - S_1 and S_2 are real and equal (repeated roots)
  - c_1 and c_2 constants determined from initial conditions, i.e. initial position and velocity of block

Control of Manipulators: Case 3- Critically Damped System

$$s_1 = s_2 = -\frac{b}{2m}$$

$$x(t) = (c_1 + c_2 t) e^{-\frac{b}{2m}t}$$





Control of Manipulators: Second Order System

- Alternative representation:
  - Parameterize characteristic equation by:

$$\boxed{\begin{aligned} s^2 + 2\varsigma\omega_n s + \omega_n{}^2 &= 0 \\ \varsigma &= \text{damping ratio} \\ \omega_n &= \text{natural frequency} \end{aligned}}$$

  - Relationship to pole locations: $s_1 = \lambda + \mu j, \quad s_2 = \lambda - \mu j$

$$\lambda = -\varsigma\omega_n, \quad \mu = \omega_n\sqrt{1-\varsigma^2} = \text{damped natural frequency}$$

  - For this spring-mass-damp system:

$$\varsigma = \frac{b}{2\sqrt{km}}, \quad \omega_n = \sqrt{k/m}$$

No damping: b = 0, ς = 0

Critically damped, (b^2=4km), ς = 1

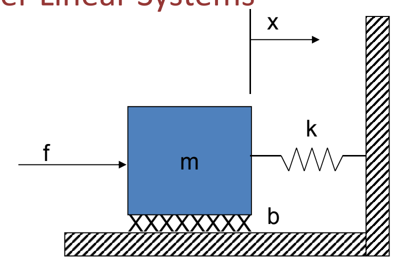
Control of Manipulators: Control of Second-Order Linear Systems

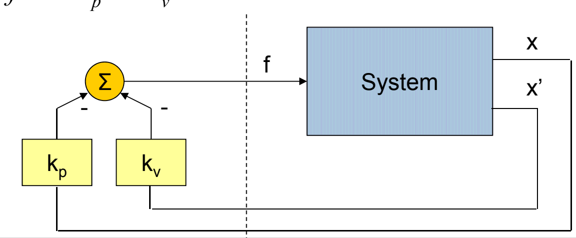
- Equation of motion:

$$m\ddot{x} + b\dot{x} + kx = 0$$

- Control law as a function of sensed feedback:

$$f = -k_p x - k_v \dot{x}$$





<u>Position regulation system</u>: 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

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

- Plugging (1) into (2):

$$m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x}$$
$$m\ddot{x} + (b + k_v)\dot{x} + (k + k_p)x = 0$$
$$m\ddot{x} + b'\dot{x} + k'x = 0$$
$$\text{where} \quad b' = b + k_v \text{ and } k' = k + k_p$$

- Choose control gains, k_v and k_p, to cause system to have any second order system behavior that is desired:

$$\text{critically damped: } b' = 2\sqrt{mk'}$$
$$\text{closed loop stiffness: } k'$$

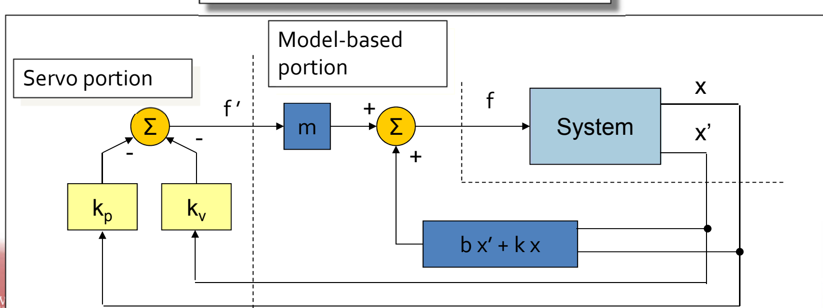
Control of Manipulators: Control Law Partitioning

- 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

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

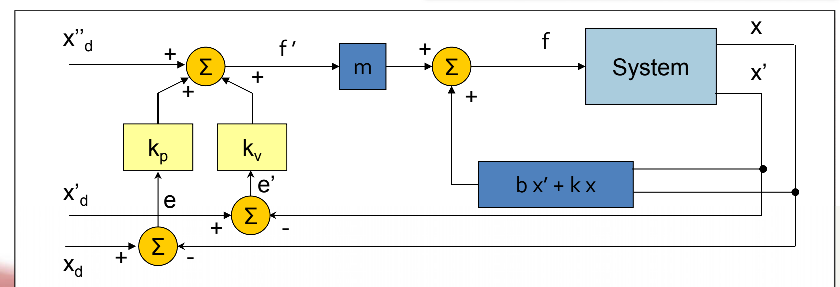
Control of Manipulators: Control Law Partitioning

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



Control of Manipulators: Trajectory Following-Control

$$\text{Desired Trajectory: } x_d(t)$$
$$\text{Given: } x_d, \dot{x}_d, \ddot{x}_d$$
$$\text{Servo Error: } e = x_d - x$$
$$\text{Control law: } f' = \ddot{x}_d + k_v \dot{e} + k_p e$$
$$\text{After substitution: } \ddot{e} + k_v \dot{e} + k_p e = 0$$
$$\boxed{\text{For critical damping: } k_v = 2\sqrt{k_p}}$$

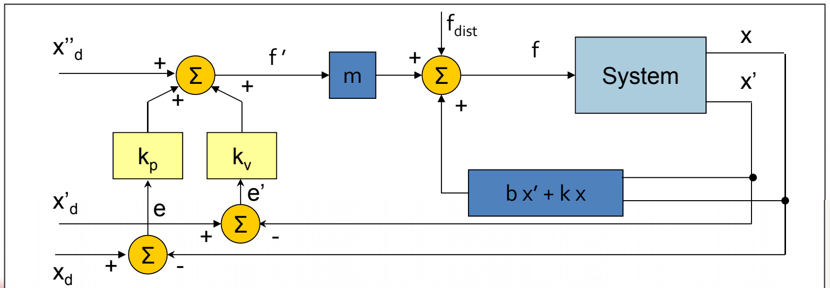


Control of Manipulators: Disturbance Rejection

$$\ddot{e} + k_v \dot{e} + k_p e = f_{dist}$$

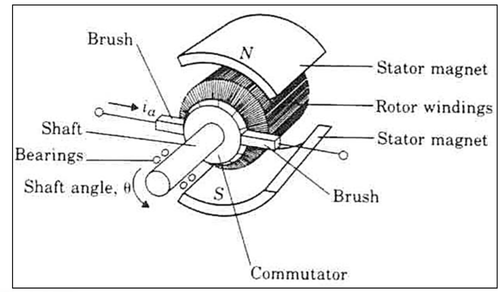
$$\text{Control law}: f' = \ddot{x}_d + k_v \dot{e} + k_p e + k_i \int e\,dt$$

$$\text{At steady-state}: e = 0$$



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: Modeling and Control of a Single Joint

- Motor torque constant relates armature current to output torque:

$$\tau_m = k_m i_a$$

- Rotating motor acts like generator and develops voltage across armature
  - Back emf constant relates voltage generated for a given rotational velocity

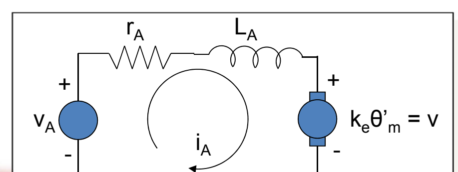
$$v = k_e \dot{\theta}_m$$

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 → actuator acts as pure torque source that we can command directly



Control of Manipulators: Mechanical Model of DC Motor Rotor

τ_m = torque applied to rotor $\qquad \tau = \eta \tau_m$

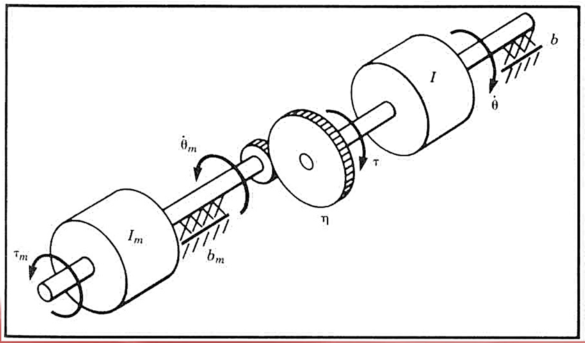
τ = torque applied to load $\qquad \dot{\theta} = \left(\frac{1}{\eta}\right)\dot{\theta}_m$

i_a = armature current

η = gear ratio

I_m and I = motor and load inertias

b_m and b = rotor and load bearings viscous friction coefficients



Control of Manipulators: Second-Order Model for DC Motor

Torque balance:

$$\tau_m = I_m\ddot{\theta}_m + b_m\dot{\theta}_m + \left(\tfrac{1}{\eta}\right)\left(I\ddot{\theta} + b\dot{\theta}\right)$$

$$\tau = \eta\tau_m \qquad \dot{\theta} = \left(\tfrac{1}{\eta}\right)\dot{\theta}_m$$

In terms of motor variables:

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

In terms of load variables:

$$\tau = \underbrace{\left(I + \eta^2 I_m\right)}_{\text{Effective inertia}}\ddot{\theta} + \underbrace{\left(b + \eta^2 b_m\right)}_{\text{Effective damping}}\dot{\theta}$$

For highly geared joints (η>>1), I_m dominates → can assume effective inertia term is a constant.

To ensure link motion is never underdamped, set I to I_{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 ω_{res}, need to limit closed-loop natural frequency:

$$\omega_n \le \tfrac{1}{2}\omega_{res}$$

  - This will limit the magnitudes for some of the gains that we choose in our controller design
  - For k = stiffness of flexible member, m = equivalent mass, estimate ω_{res} as:

$$\omega_{res} = \sqrt{k/m}$$

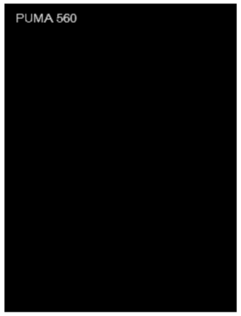
Control of Manipulators: Control of a Single Joint

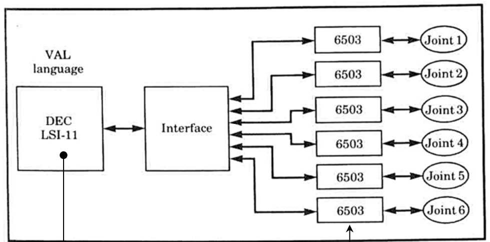
- Assumptions:
  1. Neglect motor inductance L_a
  2. High gearing, effective inertia is constant: $I_{max} + \eta^2 I_m$
  3. Structural flexibilities are neglected; use the lowest one, ω_{res}, to set the servo gains
- Use partitioned controller design:

$$\boxed{\begin{gathered}\alpha = I_{max} + \eta^2 I_m \\ \beta = b + \eta^2 b_m \\ \text{control law} = \tau' = \ddot{\theta}_d + k_v\dot{e} + k_p e\end{gathered}}$$

- Closed-loop dynamics: $\boxed{\ddot{e} + k_v\dot{e} + k_p e = \tau_{dist}}$
- Gains: $\boxed{k_p = \omega_n^2 = \frac{1}{4}\omega_{res}^2, \quad k_v = 2\sqrt{k_p} = \omega_{res}}$

Control of Manipulators:
Unimation PUMA 560 Control System





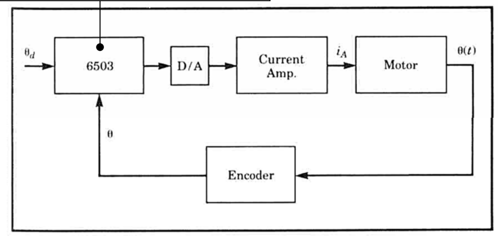
Computer - Interprets motion command, perform inverse kinematic computations, plan desired trajectory, generate trajectory via points every 28 ms

Microprocessors – get position commands every 28 ms

Control of Manipulators:
Unimation PUMA 560 Control System

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