Robotics: Advanced Concepts & Analysis

Module 7 - Motion Planning and Control

Ashitava Ghosal[1]

[1]Department of Mechanical Engineering
&
Centre for Product Design and Manufacture
Indian Institute of Science
Bangalore 560 012, India
Email: asitava@mecheng.iisc.ernet.in

NPTEL, 2010

Contents

OUTLINE

- Trajectory of a robot manipulator.
- Time history of position, velocity and acceleration of *actuated joints* or the *end-effector*.
- Algorithms for *planning* and *generation*.
- Main issues:
  - Ease and flexibility of planning.
  - Planned trajectories must be *sufficiently smooth* so as not to cause vibrations or jerky motion.
  - Efficient representation of trajectory in a computer and generation of desired trajectory in *real time*.

- Trajectory of a robot manipulator.
- Time history of position, velocity and acceleration of *actuated joints* or the *end-effector*.
- Algorithms for *planning* and *generation*.
- Main issues:
  - Ease and flexibility of planning.
  - Planned trajectories must be *sufficiently smooth* so as not to cause vibrations or jerky motion.
  - Efficient representation of trajectory in a computer and generation of desired trajectory in *real time*.

- Trajectory of a robot manipulator.
- Time history of position, velocity and acceleration of *actuated joints* or the *end-effector*.
- Algorithms for *planning* and *generation*.
- Main issues:
  - Ease and flexibility of planning.
  - Planned trajectories must be *sufficiently smooth* so as not to cause vibrations or jerky motion.
  - Efficient representation of trajectory in a computer and generation of desired trajectory in *real time*.

- Trajectory of a robot manipulator.
- Time history of position, velocity and acceleration of *actuated joints* or the *end-effector*.
- Algorithms for *planning* and *generation*.
- Main issues:
  - Ease and flexibility of planning.
  - Planned trajectories must be *sufficiently smooth* so as not to cause vibrations or jerky motion.
  - Efficient representation of trajectory in a computer and generation of desired trajectory in *real time*.

- Two main ways a robot trajectory is specified:
  - *Joint space schemes* – time history of a single or multiple joints.
  - *Cartesian space schemes* – time history of position and/or orientation of end-effector.
- Initial and final points (in joint space or Cartesian space) is specified.
- Initial and final *desired* velocity is often specified.
- Often *via* or intermediate point(s) are specified with or without desired velocity at via point(s).
- Most robots require at least $\mathscr{C}^2$ trajectories – second derivative or acceleration is continuous between initial and final points.
- Trajectory updates at rates between 50 and 200 Hz – Representation and computations of trajectories must be efficient – Not a very serious issue with modern processors!!

INTRODUCTION

OVERVIEW (CONTD.)

- Two main ways a robot trajectory is specified:
  - *Joint space schemes* – time history of a single or multiple joints.
  - *Cartesian space schemes* – time history of position and/or orientation of end-effector.
- Initial and final points (in joint space or Cartesian space) is specified.
- Initial and final *desired* velocity is often specified.
- Often *via* or intermediate point(s) are specified with or without desired velocity at via point(s).
- Most robots require at least $\mathscr{C}^2$ trajectories – second derivative or acceleration is continuous between initial and final points.
- Trajectory updates at rates between 50 and 200 Hz – Representation and computations of trajectories must be efficient – Not a very serious issue with modern processors!!

INTRODUCTION

OVERVIEW (CONTD.)

- Two main ways a robot trajectory is specified:
  - *Joint space schemes* – time history of a single or multiple joints.
  - *Cartesian space schemes* – time history of position and/or orientation of end-effector.
- Initial and final points (in joint space or Cartesian space) is specified.
- Initial and final *desired* velocity is often specified.
- Often *via* or intermediate point(s) are specified with or without desired velocity at via point(s).
- Most robots require at least $\mathscr{C}^2$ trajectories – second derivative or acceleration is continuous between initial and final points.
- Trajectory updates at rates between 50 and 200 Hz – Representation and computations of trajectories must be efficient – Not a very serious issue with modern processors!!

- Two main ways a robot trajectory is specified:
  - *Joint space schemes* – time history of a single or multiple joints.
  - *Cartesian space schemes* – time history of position and/or orientation of end-effector.
- Initial and final points (in joint space or Cartesian space) is specified.
- Initial and final *desired* velocity is often specified.
- Often *via* or intermediate point(s) are specified with or without desired velocity at via point(s).
- Most robots require at least $\mathscr{C}^2$ trajectories – second derivative or acceleration is continuous between initial and final points.
- Trajectory updates at rates between 50 and 200 Hz – Representation and computations of trajectories must be efficient – Not a very serious issue with modern processors!!

INTRODUCTION

OVERVIEW (CONTD.)

- Two main ways a robot trajectory is specified:
  - *Joint space schemes* – time history of a single or multiple joints.
  - *Cartesian space schemes* – time history of position and/or orientation of end-effector.
- Initial and final points (in joint space or Cartesian space) is specified.
- Initial and final *desired* velocity is often specified.
- Often *via* or intermediate point(s) are specified with or without desired velocity at via point(s).
- Most robots require at least $\mathscr{C}^2$ trajectories – second derivative or acceleration is continuous between initial and final points.
- Trajectory updates at rates between 50 and 200 Hz – Representation and computations of trajectories must be efficient – Not a very serious issue with modern processors!!

- Two main ways a robot trajectory is specified:
  - *Joint space schemes* – time history of a single or multiple joints.
  - *Cartesian space schemes* – time history of position and/or orientation of end-effector.
- Initial and final points (in joint space or Cartesian space) is specified.
- Initial and final *desired* velocity is often specified.
- Often *via* or intermediate point(s) are specified with or without desired velocity at via point(s).
- Most robots require at least $\mathscr{C}^2$ trajectories – second derivative or acceleration is continuous between initial and final points.
- Trajectory updates at rates between 50 and 200 Hz – Representation and computations of trajectories must be efficient – Not a very serious issue with modern processors!!

JOINT SPACE SCHEMES

- Planning trajectory of $\theta_1 - \theta_1(t_0)$ to final $\theta_1(t_f) - t_0$, t_f initial and final time.
- *Infinite* number of smooth curves can connect $\theta_i(t_0)$ to $\theta_i(t_f)$.
- **Interpolation** – Choosing a smooth curve between two points – Very well studied in *CAD* and *Geometric Modeling*.
- In robotics – simple polynomials $\rightarrow$ Simplest

$$\theta_1(t) = \frac{\theta_1(t_f) - \theta_1(t_0)}{t_f - t_0}(t - t_f) + \theta_1(t_f)$$

- Not very smooth!!

JOINT SPACE SCHEMES

- Planning trajectory of θ_1 – $\theta_1(t_0)$ to final $\theta_1(t_f)$ – t_0, t_f initial and final time.
- *Infinite* number of smooth curves can connect $\theta_i(t_0)$ to $\theta_i(t_f)$.
- **Interpolation** – Choosing a smooth curve between two points – Very well studied in *CAD* and *Geometric Modeling*.
- In robotics – simple polynomials → Simplest

$$\theta_1(t) = \frac{\theta_1(t_f) - \theta_1(t_0)}{t_f - t_0}(t - t_f) + \theta_1(t_f)$$

- Not very smooth!!

Joint Space Schemes

- Planning trajectory of θ_1 – $\theta_1(t_0)$ to final $\theta_1(t_f)$ – t_0, t_f initial and final time.
- *Infinite* number of smooth curves can connect $\theta_i(t_0)$ to $\theta_i(t_f)$.
- **Interpolation** – Choosing a smooth curve between two points – Very well studied in *CAD* and *Geometric Modeling*.
- In robotics – simple polynomials $\rightarrow$ Simplest

$$\theta_1(t) = \frac{\theta_1(t_f) - \theta_1(t_0)}{t_f - t_0}(t - t_f) + \theta_1(t_f)$$

- Not very smooth!!

Joint Space Schemes

- Planning trajectory of θ_1 – $\theta_1(t_0)$ to final $\theta_1(t_f)$ – t_0, t_f initial and final time.
- *Infinite* number of smooth curves can connect $\theta_i(t_0)$ to $\theta_i(t_f)$.
- **Interpolation** – Choosing a smooth curve between two points – Very well studied in *CAD* and *Geometric Modeling*.
- In robotics – simple polynomials $\rightarrow$ Simplest

$$\theta_1(t) = \frac{\theta_1(t_f) - \theta_1(t_0)}{t_f - t_0}(t - t_f) + \theta_1(t_f)$$

- Not very smooth!!

- Planning trajectory of θ_1 – $\theta_1(t_0)$ to final $\theta_1(t_f)$ – t_0, t_f initial and final time.
- *Infinite* number of smooth curves can connect $\theta_i(t_0)$ to $\theta_i(t_f)$.
- **Interpolation** – Choosing a smooth curve between two points – Very well studied in *CAD* and *Geometric Modeling*.
- In robotics – simple polynomials $\rightarrow$ Simplest

$$\theta_1(t) = \frac{\theta_1(t_f) - \theta_1(t_0)}{t_f - t_0}(t - t_f) + \theta_1(t_f)$$

- Not very smooth!!

JOINT SPACE SCHEMES

PIECE-WISE LINEAR

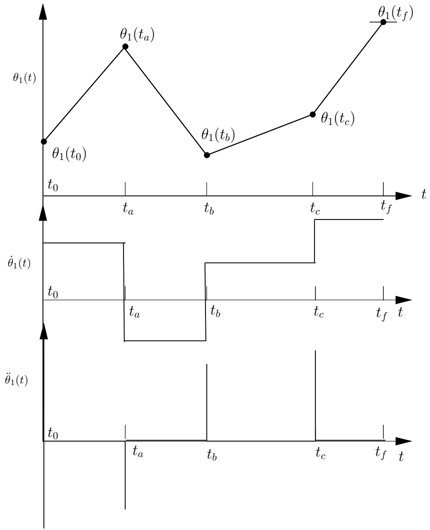


Figure 1: Piece-wise linear joint trajectory

- 4 piece-wise linear segment – trajectory through **3** via points.
- Sign changes in $\dot{\theta}_1(t)$ between segments.
- Plot of $\ddot{\theta}_1(t)$ even worse!!
- Not even $\mathscr{C}^1$ continuity.

Joint Space Schemes

Cubic Trajectory

- Simplest polynomial trajectory with $\mathscr{C}^2$ continuity
- Cubic trajectory

$$\theta_1(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (1)$$

a_0, a_1, a_2 and a_3 are four constant coefficients.

- To obtain a_0, a_1, a_2 and a_3 use given θ_1 and $\dot{\theta}_1$ at t_0 and t_f.

$$\begin{aligned} \theta_1(t_0) &= \theta_1(0), \qquad \theta_1(t_f) = \theta_1(f) \\ \dot{\theta}_1(t_0) &= \dot{\theta}_1(0), \qquad \dot{\theta}_1(t_f) = \dot{\theta}_1(f) \end{aligned} \quad (2)$$

- Four *linear* equations in four unknowns a_0, a_1, a_2 and a_3 – for $t_0 = 0$

$$\begin{aligned} a_0 &= \theta_1(0), \qquad a_1 = \dot{\theta}_1(0) \\ a_2 &= \frac{3}{t_f^2}(\theta_1(f) - \theta_1(0)) - \frac{2}{t_f}\dot{\theta}_1(0) - \frac{1}{t_f}\dot{\theta}_1(f) \\ a_3 &= -\frac{2}{t_f^3}(\theta_1(f) - \theta_1(0)) + \frac{1}{t_f^2}(\dot{\theta}_1(0) + \dot{\theta}_1(f)) \end{aligned} \quad (3)$$

Joint Space Schemes

Cubic Trajectory

- Simplest polynomial trajectory with $\mathscr{C}^2$ continuity
- Cubic trajectory

$$\theta_1(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{1}$$

a_0, a_1, a_2 and a_3 are four constant coefficients.

- To obtain a_0, a_1, a_2 and a_3 use given θ_1 and $\dot{\theta}_1$ at t_0 and t_f.

$$\begin{aligned} \theta_1(t_0) &= \theta_1(0), \qquad \theta_1(t_f) = \theta_1(f) \\ \dot{\theta}_1(t_0) &= \dot{\theta}_1(0), \qquad \dot{\theta}_1(t_f) = \dot{\theta}_1(f) \end{aligned} \tag{2}$$

- Four *linear* equations in four unknowns a_0, a_1, a_2 and a_3 – for $t_0 = 0$

$$\begin{aligned} a_0 &= \theta_1(0), \qquad a_1 = \dot{\theta}_1(0) \\ a_2 &= \frac{3}{t_f^2}(\theta_1(f) - \theta_1(0)) - \frac{2}{t_f}\dot{\theta}_1(0) - \frac{1}{t_f}\dot{\theta}_1(f) \\ a_3 &= -\frac{2}{t_f^3}(\theta_1(f) - \theta_1(0)) + \frac{1}{t_f^2}(\dot{\theta}_1(0) + \dot{\theta}_1(f)) \end{aligned} \tag{3}$$

Joint Space Schemes

Cubic Trajectory

- Simplest polynomial trajectory with $\mathscr{C}^2$ continuity
- Cubic trajectory

$$\theta_1(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{1}$$

a_0, a_1, a_2 and a_3 are four constant coefficients.

- To obtain a_0, a_1, a_2 and a_3 use given θ_1 and $\dot{\theta}_1$ at t_0 and t_f.

$$\begin{aligned} \theta_1(t_0) &= \theta_1(0), \qquad \theta_1(t_f) = \theta_1(f) \\ \dot{\theta}_1(t_0) &= \dot{\theta}_1(0), \qquad \dot{\theta}_1(t_f) = \dot{\theta}_1(f) \end{aligned} \tag{2}$$

- Four *linear* equations in four unknowns a_0, a_1, a_2 and a_3 – for $t_0 = 0$

$$\begin{aligned} a_0 &= \theta_1(0), \qquad a_1 = \dot{\theta}_1(0) \\ a_2 &= \frac{3}{t_f^2}(\theta_1(f) - \theta_1(0)) - \frac{2}{t_f}\dot{\theta}_1(0) - \frac{1}{t_f}\dot{\theta}_1(f) \\ a_3 &= -\frac{2}{t_f^3}(\theta_1(f) - \theta_1(0)) + \frac{1}{t_f^2}(\dot{\theta}_1(0) + \dot{\theta}_1(f)) \end{aligned} \tag{3}$$

Joint Space Schemes

Cubic Trajectory

- Simplest polynomial trajectory with $\mathscr{C}^2$ continuity
- Cubic trajectory

$$\theta_1(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{1}$$

a_0, a_1, a_2 and a_3 are four constant coefficients.

- To obtain a_0, a_1, a_2 and a_3 use given θ_1 and $\dot{\theta}_1$ at t_0 and t_f.

$$\begin{aligned} \theta_1(t_0) &= \theta_1(0), \qquad \theta_1(t_f) = \theta_1(f) \\ \dot{\theta}_1(t_0) &= \dot{\theta}_1(0), \qquad \dot{\theta}_1(t_f) = \dot{\theta}_1(f) \end{aligned} \tag{2}$$

- Four *linear* equations in four unknowns a_0, a_1, a_2 and a_3 – for $t_0 = 0$

$$\begin{aligned} a_0 &= \theta_1(0), \qquad a_1 = \dot{\theta}_1(0) \\ a_2 &= \frac{3}{t_f^2}(\theta_1(f) - \theta_1(0)) - \frac{2}{t_f}\dot{\theta}_1(0) - \frac{1}{t_f}\dot{\theta}_1(f) \\ a_3 &= -\frac{2}{t_f^3}(\theta_1(f) - \theta_1(0)) + \frac{1}{t_f^2}(\dot{\theta}_1(0) + \dot{\theta}_1(f)) \end{aligned} \tag{3}$$

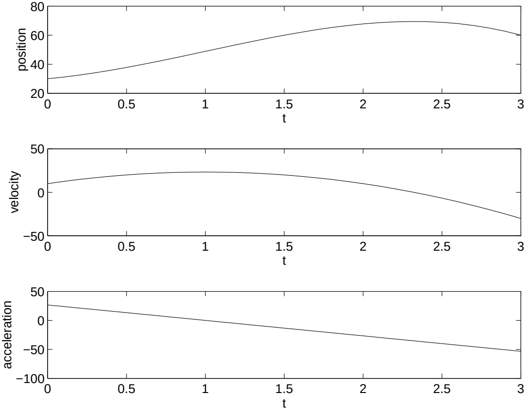


Figure 2: Cubic joint trajectory

- Given $\theta_1(0) = 30^\circ$, $\theta_1(3) = 60^\circ$, $\dot{\theta}_1(0) = 10\text{deg/sec}$ and $\dot{\theta}_1(3) = -30\text{deg/sec}$.
- Cubic coefficients are $a_0 = 30$, $a_1 = 10$, $a_2 = 13.34$ and $a_3 = -4.45$
- The expressions for $\theta_1(t)$

$$\theta_1(t) = 30 + 10t + 13.34t^2 - 4.45t^3$$

- Continuous $\theta_1(t)$, $\dot{\theta}_1(t)$ and $\ddot{\theta}_1(t)$ between $t = 0$ and $t = 3$ seconds.

Joint Space Schemes

Cubic Trajectory – Non-dimensional form

- a_2 and a_3 require division by ${t_f}^2$ and ${t_f}^3$ $\rightarrow$ error prone for large t_f.
- Use scaling of t as in geometric modeling (Mortenson, 1985).
- Define $u = t/t_f$, $u \in [0,1]$ & derivative of $(\cdot)$ with respect to u by $(\cdot)'$
- Cubic – $\theta_1(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$, coefficients of cubic are

$$\begin{aligned} a_0 &= \theta_1(0), \quad a_1 = \theta_1'(0) \\ a_2 &= -3\theta_1(0) + 3\theta_1(1) - 2\theta_1'(0) - \theta_1'(1) \\ a_3 &= 2\theta_1(0) - 2\theta_1(1) + \theta_1'(0) + \theta_1'(1) \end{aligned}$$

and

$$\begin{aligned} \theta_1(u) &= (2u^3 - 3u^2 + 1)\theta_1(0) + (-2u^3 + 3u^2)\theta_1(1) + \\ &\quad (u^3 - 2u^2 + u)\theta_1'(0) + (u^3 - u^2)\theta_1'(1) \end{aligned} \tag{4}$$

- Compute $\theta_1(u)$ and transform back to t.

Joint Space Schemes

Cubic Trajectory – Non-dimensional form

- a_2 and a_3 require division by ${t_f}^2$ and ${t_f}^3$ $\rightarrow$ error prone for large t_f.
- Use scaling of t as in geometric modeling (Mortenson, 1985).
- Define $u = t/t_f$, $u \in [0,1]$ & derivative of $(\cdot)$ with respect to u by $(\cdot)'$
- Cubic – $\theta_1(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$, coefficients of cubic are

$$\begin{aligned} a_0 &= \theta_1(0), \quad a_1 = \theta_1'(0) \\ a_2 &= -3\theta_1(0) + 3\theta_1(1) - 2\theta_1'(0) - \theta_1'(1) \\ a_3 &= 2\theta_1(0) - 2\theta_1(1) + \theta_1'(0) + \theta_1'(1) \end{aligned}$$

and

$$\begin{aligned} \theta_1(u) &= (2u^3 - 3u^2 + 1)\theta_1(0) + (-2u^3 + 3u^2)\theta_1(1) + \\ &\quad (u^3 - 2u^2 + u)\theta_1'(0) + (u^3 - u^2)\theta_1'(1) \end{aligned} \quad (4)$$

- Compute $\theta_1(u)$ and transform back to t.

Joint Space Schemes

Cubic Trajectory – Non-dimensional form

- a_2 and a_3 require division by ${t_f}^2$ and ${t_f}^3 \rightarrow$ error prone for large t_f.
- Use scaling of t as in geometric modeling (Mortenson, 1985).
- Define $u = t/t_f,\ u \in [0,1]$ & derivative of $(\cdot)$ with respect to u by $(\cdot)'$
- Cubic – $\theta_1(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$, coefficients of cubic are

$$\begin{aligned} a_0 &= \theta_1(0), \quad a_1 = \theta_1'(0) \\ a_2 &= -3\theta_1(0) + 3\theta_1(1) - 2\theta_1'(0) - \theta_1'(1) \\ a_3 &= 2\theta_1(0) - 2\theta_1(1) + \theta_1'(0) + \theta_1'(1) \end{aligned}$$

and

$$\begin{aligned} \theta_1(u) = & (2u^3 - 3u^2 + 1)\theta_1(0) + (-2u^3 + 3u^2)\theta_1(1) + \\ & (u^3 - 2u^2 + u)\theta_1'(0) + (u^3 - u^2)\theta_1'(1) \end{aligned} \quad (4)$$

- Compute $\theta_1(u)$ and transform back to t.

Joint Space Schemes

Cubic Trajectory – Non-dimensional form

- a_2 and a_3 require division by ${t_f}^2$ and ${t_f}^3$ $\rightarrow$ error prone for large t_f.
- Use scaling of t as in geometric modeling (Mortenson, 1985).
- Define $u = t/t_f, \; u \in [0,1]$ & derivative of $(\cdot)$ with respect to u by $(\cdot)'$
- Cubic – $\theta_1(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$, coefficients of cubic are

$$\begin{aligned}
a_0 &= \theta_1(0), \quad a_1 = \theta_1'(0) \\
a_2 &= -3\theta_1(0) + 3\theta_1(1) - 2\theta_1'(0) - \theta_1'(1) \\
a_3 &= 2\theta_1(0) - 2\theta_1(1) + \theta_1'(0) + \theta_1'(1)
\end{aligned}$$

and

$$\begin{aligned}
\theta_1(u) &= (2u^3 - 3u^2 + 1)\theta_1(0) + (-2u^3 + 3u^2)\theta_1(1) + \\
&\quad (u^3 - 2u^2 + u)\theta_1'(0) + (u^3 - u^2)\theta_1'(1) \qquad (4)
\end{aligned}$$

- Compute $\theta_1(u)$ and transform back to t.

Joint Space Schemes

Cubic Trajectory – Non-dimensional form

- a_2 and a_3 require division by ${t_f}^2$ and ${t_f}^3$ $\rightarrow$ error prone for large t_f.
- Use scaling of t as in geometric modeling (Mortenson, 1985).
- Define $u = t/t_f$, $u \in [0,1]$ & derivative of $(\cdot)$ with respect to u by $(\cdot)'$
- Cubic – $\theta_1(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$, coefficients of cubic are

$$
\begin{aligned}
a_0 &= \theta_1(0), \quad a_1 = \theta_1'(0) \\
a_2 &= -3\theta_1(0) + 3\theta_1(1) - 2\theta_1'(0) - \theta_1'(1) \\
a_3 &= 2\theta_1(0) - 2\theta_1(1) + \theta_1'(0) + \theta_1'(1)
\end{aligned}
$$

and

$$
\begin{aligned}
\theta_1(u) = {} & (2u^3 - 3u^2 + 1)\theta_1(0) + (-2u^3 + 3u^2)\theta_1(1) + \\
& (u^3 - 2u^2 + u)\theta_1'(0) + (u^3 - u^2)\theta_1'(1) \qquad (4)
\end{aligned}
$$

- Compute $\theta_1(u)$ and transform back to t.

- Cubic can be written in nested form

$$\theta_1(u) = a_0 + u(a_1 + u(a_2 + a_3 u))$$

- Once coefficients are computed (*offline and only once!*)
  - Only 3 multiplications and 3 additions required for $\theta_1(u)$!
  - Only 3 additional multiplications and 3 additions for $\theta_1'(u)$ and $\theta_1''(u)$[1]
- For n jointed robot, multiply by n → Cubic joint space scheme very efficient!!
- Cubic can satisfy at most 4 constraint → No control over initial and final acceleration!
- Higher-order polynomial such as *quintic* for control of acceleration → more computations.

[1] Advanced control of robots use desired position, velocity and acceleration (see Lecture 3 in this module).

- Cubic can be written in nested form

$$\theta_1(u) = a_0 + u(a_1 + u(a_2 + a_3 u))$$

- Once coefficients are computed (*offline and only once!*)
  - Only 3 multiplications and 3 additions required for $\theta_1(u)$!
  - Only 3 additional multiplications and 3 additions for $\theta_1'(u)$ and $\theta_1''(u)$[1]
- For n jointed robot, multiply by $n \rightarrow$ Cubic joint space scheme very efficient!!
- Cubic can satisfy at most 4 constraint $\rightarrow$ No control over initial and final acceleration!
- Higher-order polynomial such as *quintic* for control of acceleration $\rightarrow$ more computations.

[1] Advanced control of robots use desired position, velocity and acceleration (see Lecture 3 in this module).

Cubic Trajectory – Non-dimensional form

- Cubic can be written in nested form

$$\theta_1(u) = a_0 + u(a_1 + u(a_2 + a_3 u))$$

- Once coefficients are computed (*offline and only once!*)
  - Only 3 multiplications and 3 additions required for $\theta_1(u)$!
  - Only 3 additional multiplications and 3 additions for $\theta_1'(u)$ and $\theta_1''(u)$[1]
- For n jointed robot, multiply by $n \rightarrow$ Cubic joint space scheme very efficient!!
- Cubic can satisfy at most 4 constraint $\rightarrow$ No control over initial and final acceleration!
- Higher-order polynomial such as *quintic* for control of acceleration $\rightarrow$ more computations.

[1] Advanced control of robots use desired position, velocity and acceleration (see Lecture 3 in this module).

Joint Space Schemes

Cubic Trajectory – Non-dimensional form

- Cubic can be written in nested form

$$\theta_1(u) = a_0 + u(a_1 + u(a_2 + a_3 u))$$

- Once coefficients are computed (*offline and only once!*)
  - Only 3 multiplications and 3 additions required for $\theta_1(u)$!
  - Only 3 additional multiplications and 3 additions for $\theta_1'(u)$ and $\theta_1''(u)$[1]
- For n jointed robot, multiply by $n \rightarrow$ Cubic joint space scheme very efficient!!
- Cubic can satisfy at most 4 constraint $\rightarrow$ No control over initial and final acceleration!
- Higher-order polynomial such as *quintic* for control of acceleration $\rightarrow$ more computations.

[1] Advanced control of robots use desired position, velocity and acceleration (see Lecture 3 in this module).

Joint Space Schemes

Cubic Trajectory – Non-dimensional form

- Cubic can be written in nested form

$$\theta_1(u) = a_0 + u(a_1 + u(a_2 + a_3 u))$$

- Once coefficients are computed (*offline and only once!*)
    - Only 3 multiplications and 3 additions required for $\theta_1(u)$!
    - Only 3 additional multiplications and 3 additions for $\theta_1'(u)$ and $\theta_1''(u)$[1]
- For n jointed robot, multiply by $n \rightarrow$ Cubic joint space scheme very efficient!!
- Cubic can satisfy at most 4 constraint $\rightarrow$ No control over initial and final acceleration!
- Higher-order polynomial such as *quintic* for control of acceleration $\rightarrow$ more computations.

---

[1] Advanced control of robots use desired position, velocity and acceleration (see Lecture 3 in this module).

- k via points specified with one of two options:
  - Case 1: Velocities at the k via point(s) specified.
  - Case 2: Velocities at the k via point(s) *not* specified.
- Case 1: Plan trajectories for $k+1$ segments as $k+1$ cubics.
- Solve for a_{0i}, a_{1i}, a_{2i}, and a_{3i} $(i = 1, 2, ..., k+1)$ for each of the $k+1$ segments by using equation (3).
- $\mathscr{C}^1$ continuity *ensured* – No control on acceleration, i.e. not $\mathscr{C}^2$.

- k via points specified with one of two options:
  - Case 1: Velocities at the k via point(s) specified.
  - Case 2: Velocities at the k via point(s) *not* specified.
- Case 1: Plan trajectories for $k+1$ segments as $k+1$ cubics.
- Solve for a_{0i}, a_{1i}, a_{2i}, and a_{3i} $(i = 1, 2, ..., k+1)$ for each of the $k+1$ segments by using equation (3).
- $\mathscr{C}^1$ continuity *ensured* – No control on acceleration, i.e. not $\mathscr{C}^2$.

- k via points specified with one of two options:
  - Case 1: Velocities at the k via point(s) specified.
  - Case 2: Velocities at the k via point(s) *not* specified.
- Case 1: Plan trajectories for $k+1$ segments as $k+1$ cubics.
- Solve for a_{0i}, a_{1i}, a_{2i}, and a_{3i} $(i = 1, 2, ..., k+1)$ for each of the $k+1$ segments by using equation (3).
- $\mathscr{C}^1$ continuity *ensured* – No control on acceleration, i.e. not $\mathscr{C}^2$.

- k via points specified with one of two options:
    - Case 1: Velocities at the k via point(s) specified.
    - Case 2: Velocities at the k via point(s) *not* specified.
- Case 1: Plan trajectories for $k+1$ segments as $k+1$ cubics.
- Solve for a_{0i}, a_{1i}, a_{2i}, and a_{3i} $(i = 1, 2, ..., k+1)$ for each of the $k+1$ segments by using equation (3).
- $\mathscr{C}^1$ continuity *ensured* – No control on acceleration, i.e. not $\mathscr{C}^2$.

Joint Space Schemes

Cubic Trajectory with via point - Example

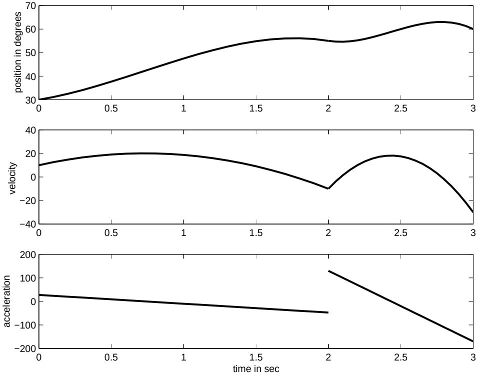


Figure 3: Cubic joint trajectory with via point

- $\theta_1(0) = 30°$, $\theta_1(3) = 60°$, $\dot{\theta}_1(0) = 10\text{deg/sec}$ and $\dot{\theta}_1(3) = -30\text{deg/sec}$.
- $\theta_1(2) = 55°$, $\dot{\theta}_1(2) = -10\text{deg/sec}$
- For segment 1: $a_{01} = 30$, $a_{11} = 10$, $a_{21} = 13.75$ and $a_{31} = -6.25$
- For segment 2: $a_{02} = 55$, $a_{12} = -10$, $a_{22} = 65$ and $a_{32} = -50$

$$\theta_1(t) = 30 + 10t + 13.75t^2 - 6.25t^3, \; 0 \le t \le 2$$
$$\theta_1(t) = 55 - 10t + 65t^2 - 50t^3, \; 2 \le t \le 3$$

Clearly as expected $\ddot{\theta}_1(t)$ is discontinuous!

Joint Space Schemes

Cubic Trajectory with via points: Case 2

- k via points specified – Velocities at the k via point(s) *not* specified.
- Free choices can be used to match velocity and acceleration at via points.
- Two cubics, each $0 \leq t \leq t_{f_i}$, $i = 1, 2$

$$\theta_1(t) = a_{0i} + a_{1i}t + a_{2i}t^2 + a_{3i}t^3, \; i = 1, 2$$

- From given initial, final, via point, and the initial and final velocities

$$\begin{aligned}
\theta_1(0) &= a_{01}, \; \dot{\theta}_1(0) = a_{11} \\
\theta_1(v) &= a_{01} + a_{11}t_{f_1} + a_{21}t_{f_1}^2 + a_{31}t_{f_1}^3, \; \theta_1(v) = a_{02} \\
\theta_1(f) &= a_{02} + a_{12}t_{f_2} + a_{22}t_{f_2}^2 + a_{32}t_{f_2}^3 \\
\dot{\theta}_1(f) &= a_{12} + 2a_{22}t_{f_2} + 3a_{32}t_{f_2}^2 \\
a_{12} &= a_{11} + 2a_{21}t_{f_1} + 3a_{31}t_{f_1}^2, \; 2a_{22} = 2a_{21} + 6a_{31}t_{f_1}
\end{aligned}$$

- 8 equations in 8 unknowns → solve for 8 coefficients of 2 cubics

- k via points specified – Velocities at the k via point(s) *not* specified.
- Free choices can be used to match velocity and acceleration at via points.
- Two cubics, each $0 \leq t \leq t_{f_i}$, $i = 1, 2$

$$\theta_1(t) = a_{0i} + a_{1i}t + a_{2i}t^2 + a_{3i}t^3, \; i = 1, 2$$

- From given initial, final, via point, and the initial and final velocities

$$\begin{aligned}
\theta_1(0) &= a_{01}, \; \dot{\theta}_1(0) = a_{11} \\
\theta_1(v) &= a_{01} + a_{11}t_{f_1} + a_{21}t_{f_1}^2 + a_{31}t_{f_1}^3, \; \theta_1(v) = a_{02} \\
\theta_1(f) &= a_{02} + a_{12}t_{f_2} + a_{22}t_{f_2}^2 + a_{32}t_{f_2}^3 \\
\dot{\theta}_1(f) &= a_{12} + 2a_{22}t_{f_2} + 3a_{32}t_{f_2}^2 \\
a_{12} &= a_{11} + 2a_{21}t_{f_1} + 3a_{31}t_{f_1}^2, \; 2a_{22} = 2a_{21} + 6a_{31}t_{f_1}
\end{aligned}$$

- 8 equations in 8 unknowns → solve for 8 coefficients of 2 cubics

Joint Space Schemes

Cubic Trajectory with via points: Case 2

- k via points specified – Velocities at the k via point(s) *not* specified.
- Free choices can be used to match velocity and acceleration at via points.
- Two cubics, each $0 \leq t \leq t_{f_i}, \; i = 1,2$

$$\theta_1(t) = a_{0i} + a_{1i}t + a_{2i}t^2 + a_{3i}t^3, \; i = 1,2$$

- From given initial, final, via point, and the initial and final velocities

$$\begin{aligned}
\theta_1(0) &= a_{01}, \; \dot{\theta}_1(0) = a_{11} \\
\theta_1(v) &= a_{01} + a_{11}t_{f_1} + a_{21}t_{f_1}^2 + a_{31}t_{f_1}^3, \; \theta_1(v) = a_{02} \\
\theta_1(f) &= a_{02} + a_{12}t_{f_2} + a_{22}t_{f_2}^2 + a_{32}t_{f_2}^3 \\
\dot{\theta}_1(f) &= a_{12} + 2a_{22}t_{f_2} + 3a_{32}t_{f_2}^2 \\
a_{12} &= a_{11} + 2a_{21}t_{f_1} + 3a_{31}t_{f_1}^2, \; 2a_{22} = 2a_{21} + 6a_{31}t_{f_1}
\end{aligned}$$

- 8 equations in 8 unknowns → solve for 8 coefficients of 2 cubics

Joint Space Schemes

Cubic Trajectory with via points: Case 2

- k via points specified – Velocities at the k via point(s) *not* specified.
- Free choices can be used to match velocity and acceleration at via points.
- Two cubics, each $0 \leq t \leq t_{f_i}, \; i = 1, 2$

$$\theta_1(t) = a_{0i} + a_{1i}t + a_{2i}t^2 + a_{3i}t^3, \; i = 1, 2$$

- From given initial, final, via point, and the initial and final velocities

$$\begin{aligned}
\theta_1(0) &= a_{01}, \; \dot{\theta}_1(0) = a_{11} \\
\theta_1(v) &= a_{01} + a_{11}t_{f_1} + a_{21}t_{f_1}^2 + a_{31}t_{f_1}^3, \; \theta_1(v) = a_{02} \\
\theta_1(f) &= a_{02} + a_{12}t_{f_2} + a_{22}t_{f_2}^2 + a_{32}t_{f_2}^3 \\
\dot{\theta}_1(f) &= a_{12} + 2a_{22}t_{f_2} + 3a_{32}t_{f_2}^2 \\
a_{12} &= a_{11} + 2a_{21}t_{f_1} + 3a_{31}t_{f_1}^2, \; 2a_{22} = 2a_{21} + 6a_{31}t_{f_1}
\end{aligned}$$

- 8 equations in 8 unknowns → solve for 8 coefficients of 2 cubics

Joint Space Schemes

Cubic Trajectory with via points: Case 2

- k via points specified – Velocities at the k via point(s) *not* specified.
- Free choices can be used to match velocity and acceleration at via points.
- Two cubics, each $0 \leq t \leq t_{f_i}, \ i = 1,2$

$$\theta_1(t) = a_{0i} + a_{1i}t + a_{2i}t^2 + a_{3i}t^3, \ i = 1,2$$

- From given initial, final, via point, and the initial and final velocities

$$\begin{aligned}
\theta_1(0) &= a_{01}, \ \dot{\theta}_1(0) = a_{11} \\
\theta_1(v) &= a_{01} + a_{11}t_{f_1} + a_{21}t_{f_1}^2 + a_{31}t_{f_1}^3, \ \theta_1(v) = a_{02} \\
\theta_1(f) &= a_{02} + a_{12}t_{f_2} + a_{22}t_{f_2}^2 + a_{32}t_{f_2}^3 \\
\dot{\theta}_1(f) &= a_{12} + 2a_{22}t_{f_2} + 3a_{32}t_{f_2}^2 \\
a_{12} &= a_{11} + 2a_{21}t_{f_1} + 3a_{31}t_{f_1}^2, \ 2a_{22} = 2a_{21} + 6a_{31}t_{f_1}
\end{aligned}$$

- 8 equations in 8 unknowns → solve for 8 coefficients of 2 cubics

Joint Space Schemes

Cubic Trajectory with via point - Case 2 Example

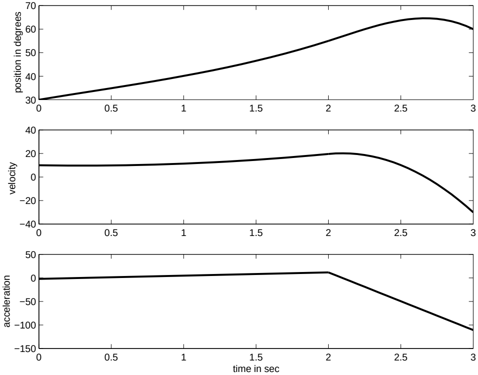


Figure 4: Cubic joint trajectory with continuous acceleration

- $\theta_1(0) = 30^\circ$, $\theta_1(3) = 60^\circ$, $\dot{\theta}_1(0) = 10\text{deg/sec}$, $\dot{\theta}_1(3) = -30\text{deg/sec}$, and $\theta_1(2) = 55^\circ$.
- For segment 1: $a_{01} = 30$, $a_{11} = 10$, $a_{21} = -1.04$ and $a_{31} = 1.15$
- For segment 2: $a_{02} = 55$, $a_{12} = 19.58$, $a_{22} = 5.83$ and $a_{32} = -20.42$

- Clearly as expected $\dot{\theta}_1(t)$ and $\ddot{\theta}_1(t)$ are continuous!
- For k via points $4 + 4k$ equations – sparse matrix and can be solved!!

- Joint space schemes useful if a joint or a group of joints are to be moved.
- Motion of end-effector $\rightarrow$ motion planning in terms of *position* and *orientation* $\rightarrow$ Cartesian Space schemes or motion planning.
  - More natural for the robot operator to specify.
  - Easier to see, visualize and check for obstacles.
  - Difficulty in planning orientation due to representation issues (See Module 2, Lecture 1).
- Traditionally two important Cartesian space paths used for position.
  - Linear interpolation –straight line path between two given positions
  - Circular interpolation –circular arcs between three given positions.
- All paths must be $\mathscr{C}^2$ continuous in time t.

- Joint space schemes useful if a joint or a group of joints are to be moved.
- Motion of end-effector $\rightarrow$ motion planning in terms of *position* and *orientation* $\rightarrow$ Cartesian Space schemes or motion planning.
  - More natural for the robot operator to specify.
  - Easier to see, visualize and check for obstacles.
  - Difficulty in planning orientation due to representation issues (See Module 2, Lecture 1).
- Traditionally two important Cartesian space paths used for position.
  - Linear interpolation –straight line path between two given positions
  - Circular interpolation –circular arcs between three given positions.
- All paths must be $\mathscr{C}^2$ continuous in time t.

- Joint space schemes useful if a joint or a group of joints are to be moved.
- Motion of end-effector $\rightarrow$ motion planning in terms of *position* and *orientation* $\rightarrow$ Cartesian Space schemes or motion planning.
  - More natural for the robot operator to specify.
  - Easier to see, visualize and check for obstacles.
  - Difficulty in planning orientation due to representation issues (See Module 2, Lecture 1).
- Traditionally two important Cartesian space paths used for position.
  - Linear interpolation –straight line path between two given positions
  - Circular interpolation –circular arcs between three given positions.
- All paths must be $\mathscr{C}^2$ continuous in time t.

- Joint space schemes useful if a joint or a group of joints are to be moved.
- Motion of end-effector $\rightarrow$ motion planning in terms of *position* and *orientation* $\rightarrow$ Cartesian Space schemes or motion planning.
  - More natural for the robot operator to specify.
  - Easier to see, visualize and check for obstacles.
  - Difficulty in planning orientation due to representation issues (See Module 2, Lecture 1).
- Traditionally two important Cartesian space paths used for position.
  - Linear interpolation –straight line path between two given positions
  - Circular interpolation –circular arcs between three given positions.
- All paths must be $\mathscr{C}^2$ continuous in time t.

Cartesian Space Schemes

Straight Line Motion

- Given $(x_0, y_0, z_0)^T$, $(\dot{x}_0, \dot{y}_0, \dot{z}_0)^T$ & $(x_f, y_f, z_f)^T$, $(\dot{x}_f, \dot{y}_f, \dot{z}_f)^T$
- Equation of a straight line in the 3D Cartesian space

$$\begin{aligned} y(t) &= \left(\frac{y_f - y_0}{x_f - x_0}\right)(x(t) - x_f) + y_f \\ z(t) &= \left(\frac{z_f - z_0}{x_f - x_0}\right)(x(t) - x_f) + z_f \end{aligned} \tag{5}$$

- Plan smooth cubic trajectory for $x(t)$ as $x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
- Compute coefficients of cubic from given initial and final conditions

$$\begin{aligned} a_0 &= x_0, \; a_1 = \dot{x}_0 \\ a_2 &= \frac{3}{t_f^2}(x_f - x_0) - \frac{2}{t_f}\dot{x}_0 - \frac{1}{t_f}\dot{x}_f \\ a_3 &= -\frac{2}{t_f^3}(x_f - x_0) + \frac{1}{t_f^2}(\dot{x}_0 + \dot{x}_f) \end{aligned} \tag{6}$$

- Compute $y(t)$ and $z(t)$ from equation (5) $\rightarrow$ $x(t)$, $y(t)$ and $z(t)$ are all $\mathscr{C}^2$.

Cartesian Space Schemes

Straight Line Motion

- Given $(x_0, y_0, z_0)^T$, $(\dot{x}_0, \dot{y}_0, \dot{z}_0)^T$ & $(x_f, y_f, z_f)^T$, $(\dot{x}_f, \dot{y}_f, \dot{z}_f)^T$
- Equation of a straight line in the 3D Cartesian space

$$
\begin{aligned}
y(t) &= (\frac{y_f - y_0}{x_f - x_0})(x(t) - x_f) + y_f \\
z(t) &= (\frac{z_f - z_0}{x_f - x_0})(x(t) - x_f) + z_f
\end{aligned}
\tag{5}
$$

- Plan smooth cubic trajectory for $x(t)$ as $x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
- Compute coefficients of cubic from given initial and final conditions

$$
\begin{aligned}
a_0 &= x_0, \; a_1 = \dot{x}_0 \\
a_2 &= \frac{3}{t_f^2}(x_f - x_0) - \frac{2}{t_f}\dot{x}_0 - \frac{1}{t_f}\dot{x}_f \\
a_3 &= -\frac{2}{t_f^3}(x_f - x_0) + \frac{1}{t_f^2}(\dot{x}_0 + \dot{x}_f)
\end{aligned}
\tag{6}
$$

- Compute $y(t)$ and $z(t)$ from equation (5) $\rightarrow$ $x(t)$, $y(t)$ and $z(t)$ are all $\mathscr{C}^2$.

Cartesian Space Schemes

Straight Line Motion

- Given $(x_0, y_0, z_0)^T$, $(\dot{x}_0, \dot{y}_0, \dot{z}_0)^T$ & $(x_f, y_f, z_f)^T$, $(\dot{x}_f, \dot{y}_f, \dot{z}_f)^T$
- Equation of a straight line in the 3D Cartesian space

$$\begin{aligned} y(t) &= \left(\frac{y_f - y_0}{x_f - x_0}\right)(x(t) - x_f) + y_f \\ z(t) &= \left(\frac{z_f - z_0}{x_f - x_0}\right)(x(t) - x_f) + z_f \end{aligned} \quad (5)$$

- Plan smooth cubic trajectory for $x(t)$ as $x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
- Compute coefficients of cubic from given initial and final conditions

$$\begin{aligned} a_0 &= x_0, \; a_1 = \dot{x}_0 \\ a_2 &= \frac{3}{t_f^2}(x_f - x_0) - \frac{2}{t_f}\dot{x}_0 - \frac{1}{t_f}\dot{x}_f \\ a_3 &= -\frac{2}{t_f^3}(x_f - x_0) + \frac{1}{t_f^2}(\dot{x}_0 + \dot{x}_f) \end{aligned} \quad (6)$$

- Compute $y(t)$ and $z(t)$ from equation (5) $\rightarrow$ $x(t)$, $y(t)$ and $z(t)$ are all $\mathscr{C}^2$.

- Given $(x_0, y_0, z_0)^T$, $(\dot{x}_0, \dot{y}_0, \dot{z}_0)^T$ & $(x_f, y_f, z_f)^T$, $(\dot{x}_f, \dot{y}_f, \dot{z}_f)^T$
- Equation of a straight line in the 3D Cartesian space

$$
\begin{aligned}
y(t) &= \left(\frac{y_f - y_0}{x_f - x_0}\right)(x(t) - x_f) + y_f \\
z(t) &= \left(\frac{z_f - z_0}{x_f - x_0}\right)(x(t) - x_f) + z_f
\end{aligned}
\tag{5}
$$

- Plan smooth cubic trajectory for $x(t)$ as $x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
- Compute coefficients of cubic from given initial and final conditions

$$
\begin{aligned}
a_0 &= x_0,\ a_1 = \dot{x}_0 \\
a_2 &= \frac{3}{t_f^2}(x_f - x_0) - \frac{2}{t_f}\dot{x}_0 - \frac{1}{t_f}\dot{x}_f \\
a_3 &= -\frac{2}{t_f^3}(x_f - x_0) + \frac{1}{t_f^2}(\dot{x}_0 + \dot{x}_f)
\end{aligned}
\tag{6}
$$

- Compute $y(t)$ and $z(t)$ from equation (5) $\rightarrow$ $x(t)$, $y(t)$ and $z(t)$ are all $\mathscr{C}^2$.

Cartesian Space Schemes

Straight Line Motion

- Given $(x_0, y_0, z_0)^T$, $(\dot{x}_0, \dot{y}_0, \dot{z}_0)^T$ & $(x_f, y_f, z_f)^T$, $(\dot{x}_f, \dot{y}_f, \dot{z}_f)^T$
- Equation of a straight line in the 3D Cartesian space

$$\begin{aligned} y(t) &= \left(\frac{y_f - y_0}{x_f - x_0}\right)(x(t) - x_f) + y_f \\ z(t) &= \left(\frac{z_f - z_0}{x_f - x_0}\right)(x(t) - x_f) + z_f \end{aligned} \qquad (5)$$

- Plan smooth cubic trajectory for $x(t)$ as $x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
- Compute coefficients of cubic from given initial and final conditions

$$\begin{aligned} a_0 &= x_0, \; a_1 = \dot{x}_0 \\ a_2 &= \frac{3}{t_f^2}(x_f - x_0) - \frac{2}{t_f}\dot{x}_0 - \frac{1}{t_f}\dot{x}_f \\ a_3 &= -\frac{2}{t_f^3}(x_f - x_0) + \frac{1}{t_f^2}(\dot{x}_0 + \dot{x}_f) \end{aligned} \qquad (6)$$

- Compute $y(t)$ and $z(t)$ from equation (5) $\rightarrow$ $x(t)$, $y(t)$ and $z(t)$ are all $\mathscr{C}^2$.

- For smoothness *circular* arcs as opposed to *piece-wise straight lines* are desired.
- Given points ${}^0\mathbf{p}_1$, ${}^0\mathbf{p}_2$, ${}^0\mathbf{p}_3$, in $\Re^3$, and velocities at these points.
- **Algorithm for circular interpolation**
  - Compute the normal to the plane as

$$ {}^0\hat{\mathbf{n}} = \frac{({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1) \times ({}^0\mathbf{p}_3 - {}^0\mathbf{p}_1)}{|({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1) \times ({}^0\mathbf{p}_3 - {}^0\mathbf{p}_1)|} $$

  - Compute ${}^0\hat{\mathbf{X}}$, ${}^0\hat{\mathbf{Y}}$ and ${}^0\hat{\mathbf{Z}}$ as

$$ \begin{aligned} {}^0\hat{\mathbf{Z}} &= {}^0\hat{\mathbf{n}} \\ {}^0\hat{\mathbf{X}} &= \frac{({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1)}{|({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1)|} \\ {}^0\hat{\mathbf{Y}} &= {}^0\mathbf{n} \times {}^0\hat{\mathbf{X}} \end{aligned} $$

    to define coordinate system $\{CIRC\}$.
  - Obtain rotation matrix ${}^0_{CIRC}[R]$ with ${}^0\hat{\mathbf{X}}$, ${}^0\hat{\mathbf{Y}}$ and ${}^0\hat{\mathbf{n}}$.

Cartesian Space Schemes

Circular Motion

- For smoothness *circular* arcs as opposed to *piece-wise straight lines* are desired.
- Given points ${}^0\mathbf{p}_1$, ${}^0\mathbf{p}_2$, ${}^0\mathbf{p}_3$, in $\Re^3$, and velocities at these points.
- Algorithm for circular interpolation
  - Compute the normal to the plane as

$$ {}^0\hat{\mathbf{n}} = \frac{({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1) \times ({}^0\mathbf{p}_3 - {}^0\mathbf{p}_1)}{|({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1) \times ({}^0\mathbf{p}_3 - {}^0\mathbf{p}_1)|} $$

  - Compute ${}^0\hat{\mathbf{X}}$, ${}^0\hat{\mathbf{Y}}$ and ${}^0\hat{\mathbf{Z}}$ as

$$ \begin{aligned} {}^0\hat{\mathbf{Z}} &= {}^0\hat{\mathbf{n}} \\ {}^0\hat{\mathbf{X}} &= \frac{({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1)}{|({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1)|} \\ {}^0\hat{\mathbf{Y}} &= {}^0\mathbf{n} \times {}^0\hat{\mathbf{X}} \end{aligned} $$

    to define coordinate system $\{CIRC\}$.
  - Obtain rotation matrix ${}^0_{CIRC}[R]$ with ${}^0\hat{\mathbf{X}}$, ${}^0\hat{\mathbf{Y}}$ and ${}^0\hat{\mathbf{n}}$.

- For smoothness *circular* arcs as opposed to *piece-wise straight lines* are desired.
- Given points ${}^0\mathbf{p}_1$, ${}^0\mathbf{p}_2$, ${}^0\mathbf{p}_3$, in $\Re^3$, and velocities at these points.
- **Algorithm for circular interpolation**
  - Compute the normal to the plane as

$$ {}^0\hat{\mathbf{n}} = \frac{({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1) \times ({}^0\mathbf{p}_3 - {}^0\mathbf{p}_1)}{|({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1) \times ({}^0\mathbf{p}_3 - {}^0\mathbf{p}_1)|} $$

  - Compute ${}^0\hat{\mathbf{X}}$, ${}^0\hat{\mathbf{Y}}$ and ${}^0\hat{\mathbf{Z}}$ as

$$ \begin{aligned} {}^0\hat{\mathbf{Z}} &= {}^0\hat{\mathbf{n}} \\ {}^0\hat{\mathbf{X}} &= \frac{({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1)}{|({}^0\mathbf{p}_2 - {}^0\mathbf{p}_1)|} \\ {}^0\hat{\mathbf{Y}} &= {}^0\mathbf{n} \times {}^0\hat{\mathbf{X}} \end{aligned} $$

    to define coordinate system $\{CIRC\}$.
  - Obtain rotation matrix ${}^0_{CIRC}[R]$ with ${}^0\hat{\mathbf{X}}$, ${}^0\hat{\mathbf{Y}}$ and ${}^0\hat{\mathbf{n}}$.

- **Algorithm for circular interpolation (Contd.)**
  - Transform ${}^0\mathbf{p}_1$, ${}^0\mathbf{p}_2$, ${}^0\mathbf{p}_3$ to $\{CIRC\}$ using ${}^{CIRC}_{0}[R]$.
  - In $\{CIRC\}$ points become (x_1, y_1, c), (x_2, y_2, c) and (x_3, y_3, c).
  - Compute centre, (a, b), and radius r of the circular arc in $\{CIRC\}$.
  - Compute angle made by line from centre to 3 points with $\hat{\mathbf{X}}$ axis in $\{CIRC\}$. Denote by ϕ_1, ϕ_2 and ϕ_3.
  - Plan a $\mathscr{C}^2$ (cubic trajectory) for $\phi(t)$ such that ϕ_1, ϕ_2 and ϕ_3 are reached at the specified t and order – joint space trajectory with via points.
  - Circular arc in $\{CIRC\}$ described by

$$
\begin{aligned}
x(t) &= a + r\cos(\phi(t)) \\
y(t) &= b + r\sin(\phi(t)),\ z(t) = c
\end{aligned}
$$

  - Since $\phi(t)$ is $\mathscr{C}^2 \rightarrow x(t)$, $y(t)$ and $z(t)$ is $\mathscr{C}^2$.
  - To obtain path of end-effector in $\{0\}$ use ${}^{0}_{CIRC}[R]$.
- Alternate: use *inverse kinematics* and plan trajectory in joint space $\rightarrow$ *Approximate* straight line or circular trajectory in Cartesian space.

Cartesian Space Schemes

Circular Motion (Contd.)

- **Algorithm for circular interpolation (Contd.)**
  - Transform ${}^0\mathbf{p}_1$, ${}^0\mathbf{p}_2$, ${}^0\mathbf{p}_3$ to $\{CIRC\}$ using ${}^{CIRC}_{0}[R]$.
  - In $\{CIRC\}$ points become (x_1, y_1, c), (x_2, y_2, c) and (x_3, y_3, c).
  - Compute centre, (a, b), and radius r of the circular arc in $\{CIRC\}$.
  - Compute angle made by line from centre to 3 points with $\hat{\mathbf{X}}$ axis in $\{CIRC\}$. Denote by ϕ_1, ϕ_2 and ϕ_3.
  - Plan a $\mathscr{C}^2$ (cubic trajectory) for $\phi(t)$ such that ϕ_1, ϕ_2 and ϕ_3 are reached at the specified t and order – joint space trajectory with via points.
  - Circular arc in $\{CIRC\}$ described by

$$
\begin{aligned}
x(t) &= a + r\cos(\phi(t)) \\
y(t) &= b + r\sin(\phi(t)),\ z(t) = c
\end{aligned}
$$

  - Since $\phi(t)$ is $\mathscr{C}^2 \rightarrow x(t)$, $y(t)$ and $z(t)$ is $\mathscr{C}^2$.
  - To obtain path of end-effector in $\{0\}$ use ${}^{0}_{CIRC}[R]$.
- Alternate: use *inverse kinematics* and plan trajectory in joint space $\rightarrow$ *Approximate* straight line or circular trajectory in Cartesian space.

- Various representation of orientation (see Module 2, Lecture 1)– all with their own advantages and disadvantages!!
- Euler parameters (see Module 2, Lecture 1) – 4 parameters + 1 constraint.
  - Given: $({}^0\varepsilon_{Tool}(0), \varepsilon_4(0))^T$ and $({}^0\varepsilon_{Tool}(t_f), \varepsilon_4(t_f))^T$.
  - Constraint: $\varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2 = 1$
  - Interpolation **must** satisfy constraint at *all* t.
- Given: Initial angular velocity ${}^0\omega_{Tool}(0)$ and final angular velocity of end-effector ${}^0\omega_{Tool}(t_f)$.
- Need relationship between angular velocity and Euler parameters – not as simple as $x(t)$ and $\dot{x}(t)$!

- Various representation of orientation (see Module 2, Lecture 1)– all with their own advantages and disadvantages!!
- Euler parameters (see Module 2, Lecture 1) – 4 parameters + 1 constraint.
  - Given: $({}^0\varepsilon_{Tool}(0), \varepsilon_4(0))^T$ and $({}^0\varepsilon_{Tool}(t_f), \varepsilon_4(t_f))^T$.
  - Constraint: $\varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2 = 1$
  - Interpolation **must** satisfy constraint at *all* t.
- Given: Initial angular velocity ${}^0\omega_{Tool}(0)$ and final angular velocity of end-effector ${}^0\omega_{Tool}(t_f)$.
- Need relationship between angular velocity and Euler parameters – not as simple as $x(t)$ and $\dot{x}(t)$!

- Various representation of orientation (see Module 2, Lecture 1)– all with their own advantages and disadvantages!!
- Euler parameters (see Module 2, Lecture 1) – 4 parameters + 1 constraint.
  - Given: $({}^0\varepsilon_{Tool}(0), \varepsilon_4(0))^T$ and $({}^0\varepsilon_{Tool}(t_f), \varepsilon_4(t_f))^T$.
  - Constraint: $\varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2 = 1$
  - Interpolation **must** satisfy constraint at *all* t.
- Given: Initial angular velocity ${}^0\omega_{Tool}(0)$ and final angular velocity of end-effector ${}^0\omega_{Tool}(t_f)$.
- Need relationship between angular velocity and Euler parameters – not as simple as $x(t)$ and $\dot{x}(t)$!

- Various representation of orientation (see Module 2, Lecture 1)– all with their own advantages and disadvantages!!
- Euler parameters (see Module 2, Lecture 1) – 4 parameters + 1 constraint.
  - Given: $({}^0\varepsilon_{Tool}(0), \varepsilon_4(0))^T$ and $({}^0\varepsilon_{Tool}(t_f), \varepsilon_4(t_f))^T$.
  - Constraint: $\varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2 = 1$
  - Interpolation **must** satisfy constraint at *all* t.
- Given: Initial angular velocity ${}^0\omega_{Tool}(0)$ and final angular velocity of end-effector ${}^0\omega_{Tool}(t_f)$.
- Need relationship between angular velocity and Euler parameters – not as simple as $x(t)$ and $\dot{x}(t)$!

Trajectory Planning for Orientation (Contd.)

- Relationships between ${}^0\omega_{Tool}(t)$ and Euler parameters

$$\begin{aligned} {}^0\omega_{Tool}(t) &= 2[E(t)]({}^0\dot{\varepsilon}_{Tool}(t), \dot{\varepsilon}_4(t))^T \\ ({}^0\dot{\varepsilon}_{Tool}(t), \dot{\varepsilon}_4(t))^T &= \frac{1}{2}[E(t)]^T\,{}^0\omega_{Tool}(t) \end{aligned}$$

where $[E(t)]$ is given

$$[E(t)] = \begin{pmatrix} -\varepsilon_1 & \varepsilon_4 & -\varepsilon_3 & \varepsilon_2 \\ -\varepsilon_2 & \varepsilon_3 & \varepsilon_4 & -\varepsilon_1 \\ -\varepsilon_3 & -\varepsilon_2 & \varepsilon_1 & \varepsilon_4 \end{pmatrix}$$

- Plan $\mathscr{C}^2$ trajectories from given ${}^0\varepsilon_{Tool}$ and ${}^0\dot{\varepsilon}_{Tool}$ at $t = 0$ and $t = t_f$.
- Compute the trajectory for $\varepsilon_4(t)$ from

$$\varepsilon_4(t) = \pm\sqrt{1 - ({}^0\varepsilon_{Tool}(t) \cdot {}^0\varepsilon_{Tool}(t))}$$

- From $(\varepsilon(t), \varepsilon_4(t))$ obtain any required representation of the orientation of the end-effector at each instant of time.

Cartesian Space Schemes

Trajectory Planning for Orientation (Contd.)

- Relationships between ${}^0\omega_{Tool}(t)$ and Euler parameters

$$
\begin{aligned}
{}^0\omega_{Tool}(t) &= 2[E(t)]({}^0\dot{\varepsilon}_{Tool}(t), \dot{\varepsilon}_4(t))^T \\
({}^0\dot{\varepsilon}_{Tool}(t), \dot{\varepsilon}_4(t))^T &= \frac{1}{2}[E(t)]^{T\,0}\omega_{Tool}(t)
\end{aligned}
$$

where $[E(t)]$ is given

$$
[E(t)] = \begin{pmatrix} -\varepsilon_1 & \varepsilon_4 & -\varepsilon_3 & \varepsilon_2 \\ -\varepsilon_2 & \varepsilon_3 & \varepsilon_4 & -\varepsilon_1 \\ -\varepsilon_3 & -\varepsilon_2 & \varepsilon_1 & \varepsilon_4 \end{pmatrix}
$$

- Plan $\mathscr{C}^2$ trajectories from given ${}^0\varepsilon_{Tool}$ and ${}^0\dot{\varepsilon}_{Tool}$ at $t = 0$ and $t = t_f$.
- Compute the trajectory for $\varepsilon_4(t)$ from

$$
\varepsilon_4(t) = \pm\sqrt{1 - ({}^0\varepsilon_{Tool}(t) \cdot {}^0\varepsilon_{Tool}(t))}
$$

- From $(\varepsilon(t), \varepsilon_4(t))$ obtain any required representation of the orientation of the end-effector at each instant of time.

Cartesian Space Schemes

Trajectory Planning for Orientation (Contd.)

- Relationships between ${}^0\omega_{Tool}(t)$ and Euler parameters

$$
\begin{aligned}
{}^0\omega_{Tool}(t) &= 2[E(t)]({}^0\dot{\varepsilon}_{Tool}(t), \dot{\varepsilon}_4(t))^T \\
({}^0\dot{\varepsilon}_{Tool}(t), \dot{\varepsilon}_4(t))^T &= \frac{1}{2}[E(t)]^{T\,0}\omega_{Tool}(t)
\end{aligned}
$$

where $[E(t)]$ is given

$$
[E(t)] = \begin{pmatrix} -\varepsilon_1 & \varepsilon_4 & -\varepsilon_3 & \varepsilon_2 \\ -\varepsilon_2 & \varepsilon_3 & \varepsilon_4 & -\varepsilon_1 \\ -\varepsilon_3 & -\varepsilon_2 & \varepsilon_1 & \varepsilon_4 \end{pmatrix}
$$

- Plan $\mathscr{C}^2$ trajectories from given ${}^0\varepsilon_{Tool}$ and ${}^0\dot{\varepsilon}_{Tool}$ at $t=0$ and $t=t_f$.
- Compute the trajectory for $\varepsilon_4(t)$ from

$$
\varepsilon_4(t) = \pm\sqrt{1 - ({}^0\varepsilon_{Tool}(t)\cdot{}^0\varepsilon_{Tool}(t))}
$$

- From $(\varepsilon(t), \varepsilon_4(t))$ obtain any required representation of the orientation of the end-effector at each instant of time.

Cartesian Space Schemes

Trajectory Planning for Orientation (contd.)

- Relationships between ${}^0\omega_{Tool}(t)$ and Euler parameters

$$
\begin{aligned}
{}^0\omega_{Tool}(t) &= 2[E(t)]({}^0\dot{\varepsilon}_{Tool}(t), \dot{\varepsilon}_4(t))^T \\
({}^0\dot{\varepsilon}_{Tool}(t), \dot{\varepsilon}_4(t))^T &= \frac{1}{2}[E(t)]^{T}\,{}^0\omega_{Tool}(t)
\end{aligned}
$$

where $[E(t)]$ is given

$$
[E(t)] = \begin{pmatrix} -\varepsilon_1 & \varepsilon_4 & -\varepsilon_3 & \varepsilon_2 \\ -\varepsilon_2 & \varepsilon_3 & \varepsilon_4 & -\varepsilon_1 \\ -\varepsilon_3 & -\varepsilon_2 & \varepsilon_1 & \varepsilon_4 \end{pmatrix}
$$

- Plan $\mathscr{C}^2$ trajectories from given ${}^0\varepsilon_{Tool}$ and ${}^0\dot{\varepsilon}_{Tool}$ at $t = 0$ and $t = t_f$.
- Compute the trajectory for $\varepsilon_4(t)$ from

$$
\varepsilon_4(t) = \pm\sqrt{1 - ({}^0\varepsilon_{Tool}(t) \cdot {}^0\varepsilon_{Tool}(t))}
$$

- From $(\varepsilon(t), \varepsilon_4(t))$ obtain any required representation of the orientation of the end-effector at each instant of time.

Summary of Motion Planning

- Joint space schemes can be applied for all *actuated* joints in a robot, independently.
- In parallel manipulators with passive joints, interpolated actuated joint values *must satisfy* constraint equations containing passive and actuated joints.
- Straight line or circular trajectories may pass through singularities or points not in workspace *even though* initial and final points are in workspace or far away from singularities!
- Straight line and circular trajectories must be checked for singularities, workspace and joint limits!!
- End-effector trajectories need to take into account dynamics and torque limits at joints (Bobrow et al., 1983).

Summary of Motion Planning

- Joint space schemes can be applied for all *actuated* joints in a robot, independently.
- In parallel manipulators with passive joints, interpolated actuated joint values *must satisfy* constraint equations containing passive and actuated joints.
- Straight line or circular trajectories may pass through singularities or points not in workspace *even though* initial and final points are in workspace or far away from singularities!
- Straight line and circular trajectories must be checked for singularities, workspace and joint limits!!
- End-effector trajectories need to take into account dynamics and torque limits at joints (Bobrow et al., 1983).

Summary of Motion Planning

- Joint space schemes can be applied for all *actuated* joints in a robot, independently.
- In parallel manipulators with passive joints, interpolated actuated joint values *must satisfy* constraint equations containing passive and actuated joints.
- Straight line or circular trajectories may pass through singularities or points not in workspace *even though* initial and final points are in workspace or far away from singularities!
- Straight line and circular trajectories must be checked for singularities, workspace and joint limits!!
- End-effector trajectories need to take into account dynamics and torque limits at joints (Bobrow et al., 1983).

Summary of Motion Planning

- Joint space schemes can be applied for all *actuated* joints in a robot, independently.
- In parallel manipulators with passive joints, interpolated actuated joint values *must satisfy* constraint equations containing passive and actuated joints.
- Straight line or circular trajectories may pass through singularities or points not in workspace *even though* initial and final points are in workspace or far away from singularities!
- Straight line and circular trajectories must be checked for singularities, workspace and joint limits!!
- End-effector trajectories need to take into account dynamics and torque limits at joints (Bobrow et al., 1983).

Summary of Motion Planning

- Joint space schemes can be applied for all *actuated* joints in a robot, independently.
- In parallel manipulators with passive joints, interpolated actuated joint values *must satisfy* constraint equations containing passive and actuated joints.
- Straight line or circular trajectories may pass through singularities or points not in workspace *even though* initial and final points are in workspace or far away from singularities!
- Straight line and circular trajectories must be checked for singularities, workspace and joint limits!!
- End-effector trajectories need to take into account dynamics and torque limits at joints (Bobrow et al., 1983).

OUTLINE

- Desired joint motion $\theta_d(t)$ available from motion planning.
- Goal of control
  - Make the joint *follow* desired $\theta_d(t)$ *accurately*.
  - *In spite* of *external* disturbances and *internal* parameter changes.
- To minimise error between *desired* and *actual or measured* motion *feedback* used.
- Feedback requires use of sensors to measure actual motion and a control scheme.
- Linear control very well known and studied – often a basis for advanced *nonlinear* control schemes.

- Desired joint motion $\theta_d(t)$ available from motion planning.
- Goal of control
  - Make the joint *follow* desired $\theta_d(t)$ *accurately*.
  - *In spite* of *external* disturbances and *internal* parameter changes.
- To minimise error between *desired* and *actual or measured* motion *feedback* used.
- Feedback requires use of sensors to measure actual motion and a control scheme.
- Linear control very well known and studied – often a basis for advanced *nonlinear* control schemes.

INTRODUCTION

OVERVIEW

- Desired joint motion $\theta_d(t)$ available from motion planning.
- Goal of control
  - Make the joint *follow* desired $\theta_d(t)$ *accurately*.
  - *In spite* of *external* disturbances and *internal* parameter changes.
- To minimise error between *desired* and *actual or measured* motion *feedback* used.
- Feedback requires use of sensors to measure actual motion and a control scheme.
- Linear control very well known and studied – often a basis for advanced *nonlinear* control schemes.

- Desired joint motion $\theta_d(t)$ available from motion planning.
- Goal of control
  - Make the joint *follow* desired $\theta_d(t)$ *accurately*.
  - *In spite* of *external* disturbances and *internal* parameter changes.
- To minimise error between *desired* and *actual or measured* motion *feedback* used.
- Feedback requires use of sensors to measure actual motion and a control scheme.
- Linear control very well known and studied – often a basis for advanced *nonlinear* control schemes.

- Desired joint motion $\theta_d(t)$ available from motion planning.
- Goal of control
  - Make the joint *follow* desired $\theta_d(t)$ *accurately*.
  - *In spite* of *external* disturbances and *internal* parameter changes.
- To minimise error between *desired* and *actual or measured* motion *feedback* used.
- Feedback requires use of sensors to measure actual motion and a control scheme.
- Linear control very well known and studied – often a basis for advanced *nonlinear* control schemes.

Control of a single link

Model

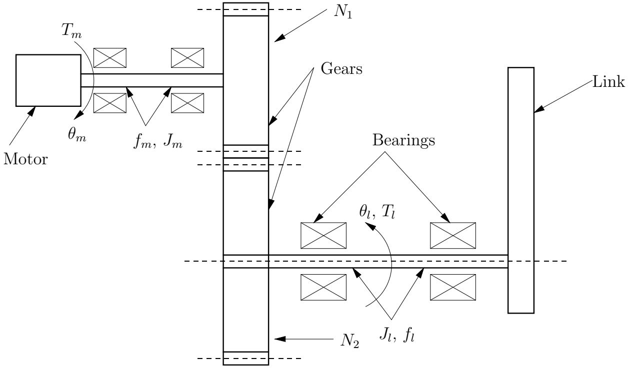


Figure 5: Model of a single link

- Single link driven by a DC motor through a gear shown in Figure 5.
  - Rated speed of typical DC motor → 2000 rpm or more.
  - Required speed about 60 rpm → need large speed reduction!
  - Analysis *assume* two spur gears giving the required speed reduction → gear ratio $n << 1$.

- Link rotation θ_l related to motor rotation θ_m by $\frac{\theta_l}{\theta_m} = n$
- One- degree-of-freedomsystem

$$\theta_l = n\theta_m, \quad \dot{\theta}_l = n\dot{\theta}_m, \quad \ddot{\theta}_l = n\ddot{\theta}_m$$

- Equation of motion of Gear 1

$$J_m\ddot{\theta}_m + f_m\dot{\theta}_m + T_1 = T_m$$

J_m, f_m and T_m are the inertia of the motor, the viscous friction at the motor shaft, and the torque output of the motor, respectively. T_1 denotes the torque acting on gear 1 from gear 2 and the link.

- Equation of motion of link + Gear 2

$$J_l\ddot{\theta}_l + f_l\dot{\theta}_l = T_2 + T_l$$

where J_l, f_l and T_l are the inertia of the load (link and gear), the viscous friction at the load, and any external disturbance torque acting on the link, respectively. T_2 denotes the torque transmitted to gear 2 by gear 1.

CONTROL OF A SINGLE LINK

MODEL (CONTD.)

- Link rotation θ_l related to motor rotation θ_m by $\frac{\theta_l}{\theta_m} = n$
- One- degree-of-freedomsystem

$$\theta_l = n\theta_m, \quad \dot{\theta}_l = n\dot{\theta}_m, \quad \ddot{\theta}_l = n\ddot{\theta}_m$$

- Equation of motion of Gear 1

$$J_m\ddot{\theta}_m + f_m\dot{\theta}_m + T_1 = T_m$$

J_m, f_m and T_m are the inertia of the motor, the viscous friction at the motor shaft, and the torque output of the motor, respectively. T_1 denotes the torque acting on gear 1 from gear 2 and the link.

- Equation of motion of link + Gear 2

$$J_l\ddot{\theta}_l + f_l\dot{\theta}_l = T_2 + T_l$$

where J_l, f_l and T_l are the inertia of the load (link and gear), the viscous friction at the load, and any external disturbance torque acting on the link, respectively. T_2 denotes the torque transmitted to gear 2 by gear 1.

Control of a single link

Model (contd.)

- Link rotation θ_l related to motor rotation θ_m by $\frac{\theta_l}{\theta_m} = n$
- One- degree-of-freedomsystem

$$\theta_l = n\theta_m, \quad \dot{\theta}_l = n\dot{\theta}_m, \quad \ddot{\theta}_l = n\ddot{\theta}_m$$

- Equation of motion of Gear 1

$$J_m\ddot{\theta}_m + f_m\dot{\theta}_m + T_1 = T_m$$

J_m, f_m and T_m are the inertia of the motor, the viscous friction at the motor shaft, and the torque output of the motor, respectively. T_1 denotes the torque acting on gear 1 from gear 2 and the link.

- Equation of motion of link + Gear 2

$$J_l\ddot{\theta}_l + f_l\dot{\theta}_l = T_2 + T_l$$

where J_l, f_l and T_l are the inertia of the load (link and gear), the viscous friction at the load, and any external disturbance torque acting on the link, respectively. T_2 denotes the torque transmitted to gear 2 by gear 1.

Control of a single link

Model (Contd.)

- Link rotation θ_l related to motor rotation θ_m by $\frac{\theta_l}{\theta_m} = n$
- One- degree-of-freedomsystem

$$\theta_l = n\theta_m, \quad \dot{\theta}_l = n\dot{\theta}_m, \quad \ddot{\theta}_l = n\ddot{\theta}_m$$

- Equation of motion of Gear 1

$$J_m\ddot{\theta}_m + f_m\dot{\theta}_m + T_1 = T_m$$

  J_m, f_m and T_m are the inertia of the motor, the viscous friction at the motor shaft, and the torque output of the motor, respectively. T_1 denotes the torque acting on gear 1 from gear 2 and the link.
- Equation of motion of link + Gear 2

$$J_l\ddot{\theta}_l + f_l\dot{\theta}_l = T_2 + T_l$$

  where J_l, f_l and T_l are the inertia of the load (link and gear), the viscous friction at the load, and any external disturbance torque acting on the link, respectively. T_2 denotes the torque transmitted to gear 2 by gear 1.

- *Assuming* no energy loss at gear tooth contacts $T_1\theta_m = T_2\theta_l$
- Equations of motion for system

$$(J_m + n^2 J_l)\ddot{\theta}_m + (f_m + n^2 f_l)\dot{\theta}_m = T_m + nT_l \quad (7)$$

  - n is small (around 0.01), the effect of the load inertia and load friction, *as seen from the motor*, is reduced by a factor of n^2.
  - Effect of T_l is also reduced by a factor of n.
  - Multi-link robots with gear reduction at joints → effect of the coupling torques from motion of other links (see Module 6, Lecture 2) (contributing to T_l) is reduced.
  - One of the reason why linear control schemes work in industrial robots!!.

- *Assuming* no energy loss at gear tooth contacts $T_1\theta_m = T_2\theta_l$
- Equations of motion for system

$$(J_m + n^2 J_l)\ddot{\theta}_m + (f_m + n^2 f_l)\dot{\theta}_m = T_m + nT_l \quad (7)$$

  - n is small (around 0.01), the effect of the load inertia and load friction, *as seen from the motor*, is reduced by a factor of n^2.
  - Effect of T_l is also reduced by a factor of n.
  - Multi-link robots with gear reduction at joints $\rightarrow$ effect of the coupling torques from motion of other links (see Module 6, Lecture 2) (contributing to T_l) is reduced.
  - One of the reason why linear control schemes work in industrial robots!!.

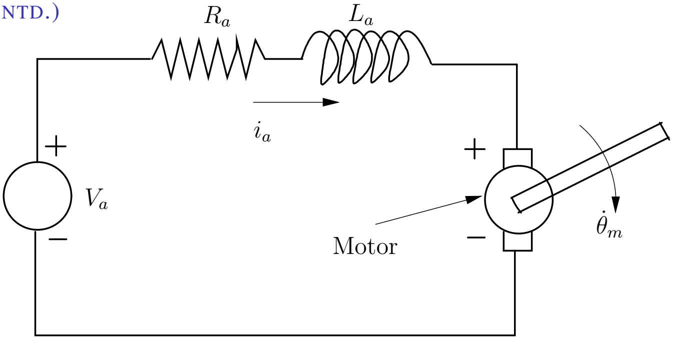


Figure 6: Model of a permanent magnet DC servo-motor

- Model of a permanent magnet DC motor shown in Figure 6.
- Stationary armature of resistance and inductance R_a and L_a respectively.
- Rotor is a permanent magnet (rare earth material).
- Voltage applied V_a and current in coil i_a.

Control of a single link

Model (contd.)

- Torque generated by motor

$$T_m = K_t i_a$$

- *Back emf* generated by coil rotating at $\dot{\theta}_m$

$$V = K_g \dot{\theta}_m$$

K_t and K_g $\rightarrow$ torque and back emf constant (available in motor specifications).

- Dynamics of a motor

$$L_a \dot{i}_a + R_a i_a + K_g \dot{\theta}_m = V_a$$

- For *small* DC servo motors, L_a is small and can be ignored.

CONTROL OF A SINGLE LINK

MODEL (CONTD.)

- Torque generated by motor

$$T_m = K_t i_a$$

- *Back emf* generated by coil rotating at $\dot{\theta}_m$

$$V = K_g \dot{\theta}_m$$

K_t and K_g $\rightarrow$ torque and back emf constant (available in motor specifications).

- Dynamics of a motor

$$L_a \dot{i}_a + R_a i_a + K_g \dot{\theta}_m = V_a$$

- For *small* DC servo motors, L_a is small and can be ignored.

- Torque generated by motor

$$T_m = K_t i_a$$

- *Back emf* generated by coil rotating at $\dot{\theta}_m$

$$V = K_g \dot{\theta}_m$$

K_t and K_g $\rightarrow$ torque and back emf constant (available in motor specifications).

- Dynamics of a motor

$$L_a \dot{i}_a + R_a i_a + K_g \dot{\theta}_m = V_a$$

- For *small* DC servo motors, L_a is small and can be ignored.

- Torque generated by motor

$$T_m = K_t i_a$$

- *Back emf* generated by coil rotating at $\dot{\theta}_m$

$$V = K_g \dot{\theta}_m$$

K_t and K_g $\rightarrow$ torque and back emf constant (available in motor specifications).

- Dynamics of a motor

$$L_a \dot{i}_a + R_a i_a + K_g \dot{\theta}_m = V_a$$

- For *small* DC servo motors, L_a is small and can be ignored.

- Combining equations of motion and the dynamics of motor with $L_a = 0$

$$(J_m + n^2 J_l)\ddot{\theta}_m + (f_m + n^2 f_l)\dot{\theta}_m = K_t \left(\frac{V_a - K_g \dot{\theta}_m}{R_a} \right) + n T_l$$

- In a compact form

$$J\dot{\Omega} + F\Omega = KV_a + T_d \quad (8)$$

$$K = K_t/R_a, \quad F = (f_m + n^2 f_l) + K_t K_g / R_a$$
$$J = J_m + n^2 J_l, \quad T_d = nT_l, \quad \Omega = \dot{\theta}_m$$

- Equation (8) describes the *mechatronic* behavior of the single-link manipulator.
  - Dynamics *in terms of angular velocity* → linear first-order ODE.
  - Back emf → increases the damping of the system.
  - Link will rotate if a) voltage is applied or b) an external disturbance torque acts on the link.

Control of a single link

Model (contd.)

- Combining equations of motion and the dynamics of motor with $L_a = 0$

$$(J_m + n^2 J_l)\ddot{\theta}_m + (f_m + n^2 f_l)\dot{\theta}_m = K_t \left(\frac{V_a - K_g \dot{\theta}_m}{R_a} \right) + n T_l$$

- In a compact form

$$J\dot{\Omega} + F\Omega = KV_a + T_d \tag{8}$$

$$\begin{aligned} K &= K_t/R_a, \quad F = (f_m + n^2 f_l) + K_t K_g/R_a \\ J &= J_m + n^2 J_l, \quad T_d = nT_l, \quad \Omega = \dot{\theta}_m \end{aligned}$$

- Equation (8) describes the *mechatronic* behavior of the single-link manipulator.
  - Dynamics *in terms of angular velocity* → linear first-order ODE.
  - Back emf → increases the damping of the system.
  - Link will rotate if a) voltage is applied or b) an external disturbance torque acts on the link.

Control of a single link

Model (Contd.)

- Combining equations of motion and the dynamics of motor with $L_a = 0$

$$(J_m + n^2 J_l)\ddot{\theta}_m + (f_m + n^2 f_l)\dot{\theta}_m = K_t \left(\frac{V_a - K_g \dot{\theta}_m}{R_a} \right) + n T_l$$

- In a compact form

$$J\dot{\Omega} + F\Omega = KV_a + T_d \tag{8}$$

$$\begin{aligned} K &= K_t/R_a, \quad F = (f_m + n^2 f_l) + K_t K_g/R_a \\ J &= J_m + n^2 J_l, \quad T_d = nT_l, \quad \Omega = \dot{\theta}_m \end{aligned}$$

- Equation (8) describes the *mechatronic* behavior of the single-link manipulator.
  - Dynamics *in terms of angular velocity* → linear first-order ODE.
  - Back emf → increases the damping of the system.
  - Link will rotate if a) voltage is applied or b) an external disturbance torque acts on the link.

- Laplace Transforms (See any undergraduate mathematics textbook)
  - Definition – $F(s) = \mathscr{L}\{f(t)\} = \int_0^\infty e^{-st} f(t)dt$
  - Laplace of derivative: $\mathscr{L}\{\frac{d}{dt} f(t)\} = sF(s) - F(0)$
  - For zero initial conditions, converts ODE to polynomial in s – ODE in equation (8) in Laplace domain is

$$Js\Omega(s) + F\Omega(s) = KV_a(s) + T_d(s)$$

- Transfer Function → Ratio of *output* to *input* in Laplace domain
- Two inputs $V_a(s)$ and $T_d(s)$ → two transfer functions

$$\frac{\Omega(s)}{V_a(s)} = \frac{K}{Js + F}, \quad \frac{\Omega(s)}{T_d(s)} = \frac{1}{Js + F}$$

with $\Omega(s)$ as output.

- Laplace Transforms (See any undergraduate mathematics textbook)
  - Definition – $F(s) = \mathscr{L}\{f(t)\} = \int_0^\infty e^{-st} f(t) dt$
  - Laplace of derivative: $\mathscr{L}\{\frac{d}{dt} f(t)\} = sF(s) - F(0)$
  - For zero initial conditions, converts ODE to polynomial in s – ODE in equation (8) in Laplace domain is

$$Js\Omega(s) + F\Omega(s) = KV_a(s) + T_d(s)$$

- Transfer Function → Ratio of *output* to *input* in Laplace domain
- Two inputs $V_a(s)$ and $T_d(s)$ → two transfer functions

$$\frac{\Omega(s)}{V_a(s)} = \frac{K}{Js+F}, \quad \frac{\Omega(s)}{T_d(s)} = \frac{1}{Js+F}$$

with $\Omega(s)$ as output.

- Laplace Transforms (See any undergraduate mathematics textbook)
  - Definition – $F(s) = \mathscr{L}\{f(t)\} = \int_0^\infty e^{-st} f(t)dt$
  - Laplace of derivative: $\mathscr{L}\{\frac{d}{dt}f(t)\} = sF(s) - F(0)$
  - For zero initial conditions, converts ODE to polynomial in s – ODE in equation (8) in Laplace domain is

$$Js\Omega(s) + F\Omega(s) = KV_a(s) + T_d(s)$$

- Transfer Function → Ratio of *output* to *input* in Laplace domain
- Two inputs $V_a(s)$ and $T_d(s)$ → two transfer functions

$$\frac{\Omega(s)}{V_a(s)} = \frac{K}{Js+F}, \quad \frac{\Omega(s)}{T_d(s)} = \frac{1}{Js+F}$$

with $\Omega(s)$ as output.

Control of a single link

Analysis – s–domain Approach (contd.)

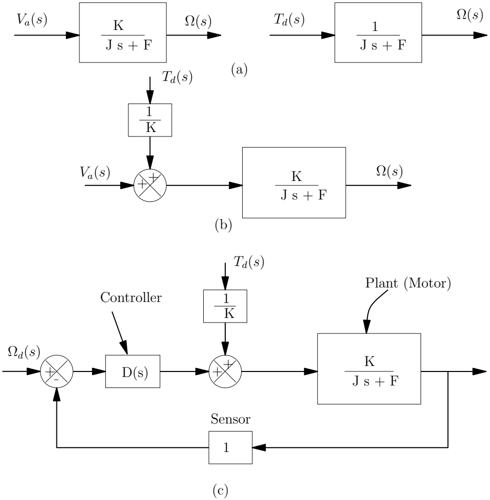


Figure 7: Transfer functions of a single link manipulator

- Figures (7) (a) & (b) are called *Open-loop Transfer Functions*.
- Figure (7) (c) is called *Closed-loop Transfer Functions* – motor output is measured and *fed back* as another input to *controller*.
- *Feedback* $\rightarrow$ *robustness* to internal parameter change and external disturbances.
  - *Assume* $V_a(s) = K_p(\Omega_d(s) - \Omega(s))$ – simplest possible controller, $D(s) = K_p$ a constant!
  - Controller gain, K_p, can be chosen but *once chosen is fixed* (factory setting!)
  - For *open-loop* (without feedback), $V_a(s) = K_p\Omega_d(s)$.
  - Open-loop – Choose $K_p = 1/K_0$ where $K_0 = K/F$.

- Figures (7) (a) & (b) are called *Open-loop Transfer Functions*.
- Figure (7) (c) is called *Closed-loop Transfer Functions* – motor output is measured and *fed back* as another input to *controller*.
- *Feedback* $\rightarrow$ *robustness* to internal parameter change and external disturbances.
  - *Assume* $V_a(s) = K_p(\Omega_d(s) - \Omega(s))$ – simplest possible controller, $D(s) = K_p$ a constant!
  - Controller gain, K_p, can be chosen but *once chosen is fixed* (factory setting!)
  - For *open-loop* (without feedback), $V_a(s) = K_p \Omega_d(s)$.
  - Open-loop – Choose $K_p = 1/K_0$ where $K_0 = K/F$.

- Figures (7) (a) & (b) are called *Open-loop Transfer Functions*.
- Figure (7) (c) is called *Closed-loop Transfer Functions* – motor output is measured and *fed back* as another input to *controller*.
- *Feedback* $\rightarrow$ *robustness* to internal parameter change and external disturbances.
  - *Assume* $V_a(s) = K_p(\Omega_d(s) - \Omega(s))$ – simplest possible controller, $D(s) = K_p$ a constant!
  - Controller gain, K_p, can be chosen but *once chosen is fixed* (factory setting!)
  - For *open-loop* (without feedback), $V_a(s) = K_p\Omega_d(s)$.
  - Open-loop – Choose $K_p = 1/K_0$ where $K_0 = K/F$.

- With $T_d = 0$ and steady state, i.e., $s \to 0$,

$$\lim_{s\to 0} \Omega(s) = \lim_{s\to 0} \frac{K}{Js+F} V_a(s) \Rightarrow \Omega = (K/F)V_a = K_0 K_p \Omega_d$$

- For $K_p = 1/K_0$, $\Omega = \Omega_d$ as desired in any controller!
- For closed-loop $V_a(s) = K_p(\Omega_d(s) - \Omega(s))$ and for $s \to 0$

$$\lim_{s\to 0} \Omega(s) = \lim_{s\to 0} \frac{KK_p}{Js+F+KK_p} V_a(s) \Rightarrow \Omega = \frac{K_0 K_p}{1+K_0 K_p} \Omega_d$$

- Best possible choice $K_0 K_p >> 1$ and best possible outcome $\Omega \approx \Omega_d$.
- Apparently with feedback, the situation is worse!

- With $T_d = 0$ and steady state, i.e., $s \to 0$,

$$\lim_{s \to 0} \Omega(s) = \lim_{s \to 0} \frac{K}{Js + F} V_a(s) \Rightarrow \Omega = (K/F)V_a = K_0 K_p \Omega_d$$

- For $K_p = 1/K_0$, $\Omega = \Omega_d$ as desired in any controller!
- For closed-loop $V_a(s) = K_p(\Omega_d(s) - \Omega(s))$ and for $s \to 0$

$$\lim_{s \to 0} \Omega(s) = \lim_{s \to 0} \frac{KK_p}{Js + F + KK_p} V_a(s) \Rightarrow \Omega = \frac{K_0 K_p}{1 + K_0 K_p} \Omega_d$$

- Best possible choice $K_0 K_p >> 1$ and best possible outcome $\Omega \approx \Omega_d$.
- Apparently with feedback, the situation is worse!

- With $T_d = 0$ and steady state, i.e., $s \to 0$,

$$\lim_{s\to 0} \Omega(s) = \lim_{s\to 0} \frac{K}{Js+F} V_a(s) \Rightarrow \Omega = (K/F)V_a = K_0 K_p \Omega_d$$

- For $K_p = 1/K_0$, $\Omega = \Omega_d$ as desired in any controller!
- For closed-loop $V_a(s) = K_p(\Omega_d(s) - \Omega(s))$ and for $s \to 0$

$$\lim_{s\to 0} \Omega(s) = \lim_{s\to 0} \frac{KK_p}{Js+F+KK_p} V_a(s) \Rightarrow \Omega = \frac{K_0 K_p}{1+K_0 K_p}\Omega_d$$

- Best possible choice $K_0 K_p >> 1$ and best possible outcome $\Omega \approx \Omega_d$.
- Apparently with feedback, the situation is worse!

Control of a single link

Analysis – s–domain Approach(Contd.)

- With $T_d = 0$ and steady state, i.e., $s \rightarrow 0$,

$$\lim_{s \to 0} \Omega(s) = \lim_{s \to 0} \frac{K}{Js + F} V_a(s) \Rightarrow \Omega = (K/F)V_a = K_0 K_p \Omega_d$$

- For $K_p = 1/K_0$, $\Omega = \Omega_d$ as desired in any controller!
- For closed-loop $V_a(s) = K_p(\Omega_d(s) - \Omega(s))$ and for $s \rightarrow 0$

$$\lim_{s \to 0} \Omega(s) = \lim_{s \to 0} \frac{KK_p}{Js + F + KK_p} V_a(s) \Rightarrow \Omega = \frac{K_0 K_p}{1 + K_0 K_p} \Omega_d$$

- Best possible choice $K_0 K_p >> 1$ and best possible outcome $\Omega \approx \Omega_d$.
- Apparently with feedback, the situation is worse!

- With $T_d = 0$ and steady state, i.e., $s \to 0$,

$$\lim_{s\to 0} \Omega(s) = \lim_{s\to 0} \frac{K}{Js+F} V_a(s) \Rightarrow \Omega = (K/F)V_a = K_0 K_p \Omega_d$$

- For $K_p = 1/K_0$, $\Omega = \Omega_d$ as desired in any controller!
- For closed-loop $V_a(s) = K_p(\Omega_d(s) - \Omega(s))$ and for $s \to 0$

$$\lim_{s\to 0} \Omega(s) = \lim_{s\to 0} \frac{KK_p}{Js+F+KK_p} V_a(s) \Rightarrow \Omega = \frac{K_0 K_p}{1+K_0 K_p} \Omega_d$$

- Best possible choice $K_0 K_p >> 1$ and best possible outcome $\Omega \approx \Omega_d$.
- Apparently with feedback, the situation is worse!

- Consider *change* in *internal* parameter due to environmental changes – K_0 changes to $K_0 + \delta K_0$
- For open-loop in steady-state
  - $\Omega + \delta\Omega = (K_0 + \delta K_0) K_p \Omega_d$
  - Since K_p is *set* to $1/K_0$, $\delta\Omega = (\delta K_0 / K_0)\Omega_d$
- For closed-loop with $K_0 K_p >> 1$,

$$\delta\Omega' / \Omega' = \frac{1}{1 + K_0 K_p}(\delta K_0 / K_0)$$

  where $\Omega' = \frac{K_0 K_p}{1+K_0 K_p}\Omega_d \approx \Omega_d$
- An x% change in $K_0 \rightarrow \frac{1}{1+K_0 K_p} \times x$% change in Ω'.
- Since $1 + K_0 K_p >> 1$, the change in output *greatly reduced* by feedback $\rightarrow$ *Robustness*!

- Consider *change* in *internal* parameter due to environmental changes – K_0 changes to $K_0 + \delta K_0$
- For open-loop in steady-state
  - $\Omega + \delta\Omega = (K_0 + \delta K_0) K_p \Omega_d$
  - Since K_p is *set* to $1/K_0$, $\delta\Omega = (\delta K_0/K_0)\Omega_d$
- For closed-loop with $K_0 K_p >> 1$,

$$\delta\Omega'/\Omega' = \frac{1}{1 + K_0 K_p}(\delta K_0/K_0)$$

  where $\Omega' = \frac{K_0 K_p}{1+K_0 K_p}\Omega_d \approx \Omega_d$
- An $x\%$ change in $K_0 \rightarrow \frac{1}{1+K_0 K_p} \times x\%$ change in Ω'.
- Since $1 + K_0 K_p >> 1$, the change in output *greatly reduced* by feedback $\rightarrow$ *Robustness*!

CONTROL OF A SINGLE LINK

ANALYSIS – $s-$DOMAIN APPROACH(CONTD.)

- Consider *change* in *internal* parameter due to environmental changes – K_0 changes to $K_0 + \delta K_0$
- For open-loop in steady-state
  - $\Omega + \delta\Omega = (K_0 + \delta K_0) K_p \Omega_d$
  - Since K_p is *set* to $1/K_0$, $\delta\Omega = (\delta K_0/K_0)\Omega_d$
- For closed-loop with $K_0 K_p >> 1$,

$$\delta\Omega'/\Omega' = \frac{1}{1 + K_0 K_p}(\delta K_0/K_0)$$

where $\Omega' = \frac{K_0 K_p}{1+K_0 K_p}\Omega_d \approx \Omega_d$

- An x% change in $K_0 \rightarrow \frac{1}{1+K_0 K_p} \times x$% change in Ω'.
- Since $1 + K_0 K_p >> 1$, the change in output *greatly reduced* by feedback $\rightarrow$ *Robustness*!

- Consider *change* in *internal* parameter due to environmental changes – K_0 changes to $K_0 + \delta K_0$
- For open-loop in steady-state
  - $\Omega + \delta\Omega = (K_0 + \delta K_0)K_p\Omega_d$
  - Since K_p is *set* to $1/K_0$, $\delta\Omega = (\delta K_0/K_0)\Omega_d$
- For closed-loop with $K_0 K_p >> 1$,

$$\delta\Omega'/\Omega' = \frac{1}{1 + K_0 K_p}(\delta K_0/K_0)$$

  where $\Omega' = \frac{K_0 K_p}{1+K_0 K_p}\Omega_d \approx \Omega_d$
- An x% change in $K_0 \rightarrow \frac{1}{1+K_0 K_p} \times x$% change in Ω'.
- Since $1 + K_0 K_p >> 1$, the change in output *greatly reduced* by feedback $\rightarrow$ *Robustness*!

CONTROL OF A SINGLE LINK

ANALYSIS – s–DOMAIN APPROACH(CONTD.)

- Consider *change* in *internal* parameter due to environmental changes – K_0 changes to $K_0 + \delta K_0$
- For open-loop in steady-state
  - $\Omega + \delta\Omega = (K_0 + \delta K_0) K_p \Omega_d$
  - Since K_p is *set* to $1/K_0$, $\delta\Omega = (\delta K_0 / K_0)\Omega_d$
- For closed-loop with $K_0 K_p >> 1$,

$$\delta\Omega' / \Omega' = \frac{1}{1 + K_0 K_p} (\delta K_0 / K_0)$$

where $\Omega' = \frac{K_0 K_p}{1 + K_0 K_p} \Omega_d \approx \Omega_d$

- An x% change in $K_0 \rightarrow \frac{1}{1+K_0 K_p} \times x$% change in Ω'.
- Since $1 + K_0 K_p >> 1$, the change in output *greatly reduced* by feedback $\rightarrow$ *Robustness*!

- If $T_d \neq 0$

$$\Omega = K_0 K_c \Omega_d + K_0 (T_d / K), \quad \text{Controller gain is } K_c$$

- For $K_0 K_c = 1$, $\Omega = \Omega_d + K_0(T_d/K) \rightarrow$ Change in output proportional to T_d
- With feedback, steady-state output

$$\Omega = \frac{K_0 K_c}{1 + K_0 K_c} \Omega_d + \frac{K_0}{1 + K_0 K_c} (T_d / K)$$

- Choose $K_0 K_c >> 1$ and $K_0 K_c >> (K_0/K)$ (or $K_c >> 1/K$).
- Effect of T_d is *reduced* due to feedback!!

- If $T_d \neq 0$

$$\Omega = K_0 K_c \Omega_d + K_0 (T_d / K), \quad \text{Controller gain is } K_c$$

- For $K_0 K_c = 1$, $\Omega = \Omega_d + K_0 (T_d / K) \rightarrow$ Change in output proportional to T_d
- With feedback, steady-state output

$$\Omega = \frac{K_0 K_c}{1 + K_0 K_c} \Omega_d + \frac{K_0}{1 + K_0 K_c} (T_d / K)$$

- Choose $K_0 K_c >> 1$ and $K_0 K_c >> (K_0 / K)$ (or $K_c >> 1/K$).
- Effect of T_d is *reduced* due to feedback!!

- If $T_d \neq 0$

$$\Omega = K_0 K_c \Omega_d + K_0(T_d/K), \quad \text{Controller gain is } K_c$$

- For $K_0 K_c = 1$, $\Omega = \Omega_d + K_0(T_d/K) \rightarrow$ Change in output proportional to T_d
- With feedback, steady-state output

$$\Omega = \frac{K_0 K_c}{1 + K_0 K_c}\Omega_d + \frac{K_0}{1 + K_0 K_c}(T_d/K)$$

- Choose $K_0 K_c >> 1$ and $K_0 K_c >> (K_0/K)$ (or $K_c >> 1/K$).
- Effect of T_d is *reduced* due to feedback!!

- If $T_d \neq 0$

$$\Omega = K_0 K_c \Omega_d + K_0(T_d/K), \quad \text{Controller gain is } K_c$$

- For $K_0 K_c = 1$, $\Omega = \Omega_d + K_0(T_d/K) \rightarrow$ Change in output proportional to T_d
- With feedback, steady-state output

$$\Omega = \frac{K_0 K_c}{1 + K_0 K_c}\Omega_d + \frac{K_0}{1 + K_0 K_c}(T_d/K)$$

- Choose $K_0 K_c >> 1$ and $K_0 K_c >> (K_0/K)$ (or $K_c >> 1/K$).
- Effect of T_d is *reduced* due to feedback!!

- If $T_d \neq 0$

$$\Omega = K_0 K_c \Omega_d + K_0(T_d/K), \quad \text{Controller gain is } K_c$$

- For $K_0 K_c = 1$, $\Omega = \Omega_d + K_0(T_d/K) \rightarrow$ Change in output proportional to T_d
- With feedback, steady-state output

$$\Omega = \frac{K_0 K_c}{1 + K_0 K_c} \Omega_d + \frac{K_0}{1 + K_0 K_c} (T_d/K)$$

- Choose $K_0 K_c >> 1$ and $K_0 K_c >> (K_0/K)$ (or $K_c >> 1/K$).
- Effect of T_d is *reduced* due to feedback!!

Control of a single link

Analysis – first-order system

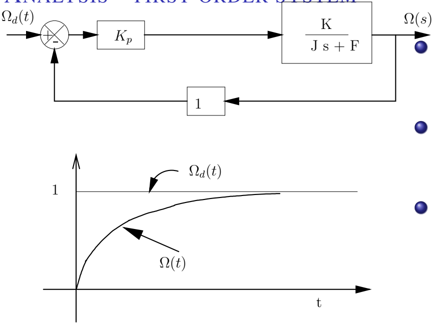


Figure 8: Block diagram of single link manipulator under feedback and $T_d = 0$

- *First-order system* as governing ODE is first-order.
- Several ways to analyse control systems $\rightarrow s-$ plane analysis
- *Closed-loop* transfer function between output $\Omega(s)$ and desired speed $\Omega_d(s)$

$$\frac{\Omega(s)}{\Omega_d(s)} = (KK_p/J)\left(\frac{1}{s+(F+KK_p)/J}\right)$$

- Step response – $\Omega(s)$ for $\Omega_d(s) = 1/s$.
- $\Omega(t)$ is of the form $1 - e^{-(\frac{F+KK_p}{J})t} \rightarrow F$, K, K_p and J are all positive $\rightarrow \Omega(t)$ always *bounded* and approaches $\Omega_d(t)$ as $t \rightarrow \infty$.
- System stable as *bounded output* for a *bounded input*.
- Increasing K_p makes $\Omega(t)$ approach 1 faster!!

- For control of angular rotation, open-loop transfer function with $T_d = 0$ is

$$\frac{\theta(s)}{V_a(s)} = \frac{K}{s(Js+F)}$$

- Transfer function is *second-order* as the governing ODE is second-order (denominator polynomial is second degree in s).
- Closed-loop transfer function between output $\theta(s)$ and desired input $\theta_d(s)$

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p}{s(Js+F)+KK_p} = \frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$$

where $\omega_n^2 = (KK_p/J)$, $F/J = 2\xi\omega_n$ and $\xi = \frac{F}{2\sqrt{JKK_p}}$.

- For second-order systems, ω_n is called the *natural frequency* of the system and ξ is called the *damping*.
- The parameters ω_n and ξ completely determine the behaviour of a second-order system.

- For control of angular rotation, open-loop transfer function with $T_d = 0$ is

$$\frac{\theta(s)}{V_a(s)} = \frac{K}{s(Js+F)}$$

- Transfer function is *second-order* as the governing ODE is second-order (denominator polynomial is second degree in s).
- Closed-loop transfer function between output $\theta(s)$ and desired input $\theta_d(s)$

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p}{s(Js+F)+KK_p} = \frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$$

where $\omega_n^2 = (KK_p/J)$, $F/J = 2\xi\omega_n$ and $\xi = \frac{F}{2\sqrt{JKK_p}}$.
- For second-order systems, ω_n is called the *natural frequency* of the system and ξ is called the *damping*.
- The parameters ω_n and ξ completely determine the behaviour of a second-order system.

Control of a single link

Analysis – second-order system

- For control of angular rotation, open-loop transfer function with $T_d = 0$ is

$$\frac{\theta(s)}{V_a(s)} = \frac{K}{s(Js+F)}$$

- Transfer function is *second-order* as the governing ODE is second-order (denominator polynomial is second degree in s).
- Closed-loop transfer function between output $\theta(s)$ and desired input $\theta_d(s)$

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p}{s(Js+F)+KK_p} = \frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$$

where $\omega_n^2 = (KK_p/J)$, $F/J = 2\xi\omega_n$ and $\xi = \frac{F}{2\sqrt{JKK_p}}$.

- For second-order systems, ω_n is called the *natural frequency* of the system and ξ is called the *damping*.
- The parameters ω_n and ξ completely determine the behaviour of a second-order system.

Control of a single link

Analysis – second-order system

- For control of angular rotation, open-loop transfer function with $T_d = 0$ is

$$\frac{\theta(s)}{V_a(s)} = \frac{K}{s(Js+F)}$$

- Transfer function is *second-order* as the governing ODE is second-order (denominator polynomial is second degree in s).
- Closed-loop transfer function between output $\theta(s)$ and desired input $\theta_d(s)$

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p}{s(Js+F)+KK_p} = \frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$$

where $\omega_n^2 = (KK_p/J)$, $F/J = 2\xi\omega_n$ and $\xi = \frac{F}{2\sqrt{JKK_p}}$.

- For second-order systems, ω_n is called the *natural frequency* of the system and ξ is called the *damping*.
- The parameters ω_n and ξ completely determine the behaviour of a second-order system.

Control of a single link

Analysis – second-order system

- For control of angular rotation, open-loop transfer function with $T_d = 0$ is
$$\frac{\theta(s)}{V_a(s)} = \frac{K}{s(Js+F)}$$
- Transfer function is *second-order* as the governing ODE is second-order (denominator polynomial is second degree in s).
- Closed-loop transfer function between output $\theta(s)$ and desired input $\theta_d(s)$
$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p}{s(Js+F)+KK_p} = \frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$$
where $\omega_n^2 = (KK_p/J)$, $F/J = 2\xi\omega_n$ and $\xi = \frac{F}{2\sqrt{JKK_p}}$.
- For second-order systems, ω_n is called the *natural frequency* of the system and ξ is called the *damping*.
- The parameters ω_n and ξ completely determine the behaviour of a second-order system.

- Three possible kinds of behaviour
- $0 < \xi < 1$ – *under-damped* systems.
  - Output oscillates about the desired input before settling down in *infinite* time.
  - Settling time t_s – Time taken for output to reach within $\pm 5\%$ (or $\pm 2\%$) of the input → For $\pm 5\%$ $t_s \approx \frac{3}{\xi \omega_n}$ and is $\approx \frac{4}{\xi \omega_n}$ for $\pm 2\%$.
  - The maximum overshoot is large for low damping ξ, and small for high ξ → Peak overshoot is $e^{-(\xi/\sqrt{1-\xi^2})\pi}$.
  - The roots of the denominator closed-loop polynomial are complex with negative real parts.
  - Roots are in the left-half of the s plane → second-order system is stable.
- Output $\Omega(t)$ to a step input shown in Figure 9(b).

- Three possible kinds of behaviour
- $0 < \xi < 1$ – *under-damped* systems.
  - Output oscillates about the desired input before settling down in *infinite* time.
  - Settling time t_s – Time taken for output to reach within $\pm 5\%$ (or $\pm 2\%$) of the input $\rightarrow$ For $\pm 5\%$ $t_s \approx \frac{3}{\xi \omega_n}$ and is $\approx \frac{4}{\xi \omega_n}$ for $\pm 2\%$.
  - The maximum overshoot is large for low damping ξ, and small for high ξ $\rightarrow$ Peak overshoot is $e^{-(\xi/\sqrt{1-\xi^2})\pi}$.
  - The roots of the denominator closed-loop polynomial are complex with negative real parts.
  - Roots are in the left-half of the s plane $\rightarrow$ second-order system is stable.
- Output $\Omega(t)$ to a step input shown in Figure 9(b).

- Three possible kinds of behaviour
- $0 < \xi < 1$ – *under-damped* systems.
  - Output oscillates about the desired input before settling down in *infinite* time.
  - Settling time t_s – Time taken for output to reach within $\pm 5\%$ (or $\pm 2\%$) of the input $\rightarrow$ For $\pm 5\%$ $t_s \approx \frac{3}{\xi \omega_n}$ and is $\approx \frac{4}{\xi \omega_n}$ for $\pm 2\%$.
  - The maximum overshoot is large for low damping ξ, and small for high ξ $\rightarrow$ Peak overshoot is $e^{-(\xi/\sqrt{1-\xi^2})\pi}$.
  - The roots of the denominator closed-loop polynomial are complex with negative real parts.
  - Roots are in the left-half of the s plane $\rightarrow$ second-order system is stable.
- Output $\Omega(t)$ to a step input shown in Figure 9(b).

- $\xi = 1$ – *critically damped* systems.
    - Output shows no oscillations and can cross input at most once.
    - Settling time can be defined similar to the under-damped case.
    - The roots of the denominator polynomial are real and repeated, and lie in the left-half of the s plane.
    - Output $\Omega(t)$ for a step input shown in Figure 9(b).
- $\xi > 1$ – *over-damped* systems.
    - Output $\Omega(t)$ can never cross the input and is the sum of two exponential functions.
    - The roots of the denominator polynomial are real and distinct, and lie on the left-half of the s plane.
    - Figure 9(b) shows a typical response of an over-damped system.

- $\xi = 1$ – *critically damped* systems.
  - Output shows no oscillations and can cross input at most once.
  - Settling time can be defined similar to the under-damped case.
  - The roots of the denominator polynomial are real and repeated, and lie in the left-half of the s plane.
  - Output $\Omega(t)$ for a step input shown in Figure 9(b).
- $\xi > 1$ – *over-damped* systems.
  - Output $\Omega(t)$ can never cross the input and is the sum of two exponential functions.
  - The roots of the denominator polynomial are real and distinct, and lie on the left-half of the s plane.
  - Figure 9(b) shows a typical response of an over-damped system.

Control of a single link

Analysis – second-order system (contd.)

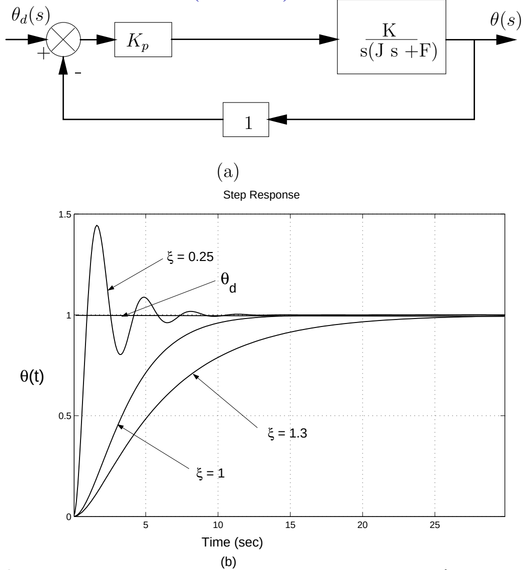


Figure 9: Second-order system and its step response $(K = J = F = 1)$

- For one link manipulator ω_n and ξ depends on controller gain K_p

$$\omega_n^2 = (KK_p/J), \quad \xi = \frac{F}{2\sqrt{JKK_p}}$$

- Changing K_p *changes both* ω_n and ξ.
- Can *make the output* under-damped, critically damped or over-damped by choosing K_p!!
- Simplest possible controller $\rightarrow$ *Proportional Controller*
- To choose ω_n and ξ *arbitrarily*, *two* parameters needed $\rightarrow$ Proportional plus Derivative (PD) controller.

- For one link manipulator ω_n and ξ depends on controller gain K_p

$$\omega_n^2 = (KK_p/J), \quad \xi = \frac{F}{2\sqrt{JKK_p}}$$

- Changing K_p *changes both* ω_n and ξ.
- Can *make the output* under-damped, critically damped or over-damped by choosing K_p!!
- Simplest possible controller $\rightarrow$ *Proportional Controller*
- To choose ω_n and ξ *arbitrarily*, *two* parameters needed $\rightarrow$ Proportional plus Derivative (PD) controller.

CONTROL OF A SINGLE LINK

ANALYSIS – SECOND-ORDER SYSTEM (CONTD.)

- For one link manipulator ω_n and ξ depends on controller gain K_p

$$\omega_n^2 = (KK_p/J), \quad \xi = \frac{F}{2\sqrt{JKK_p}}$$

- Changing K_p *changes both* ω_n and ξ.
- Can *make the output* under-damped, critically damped or over-damped by choosing K_p!!
- Simplest possible controller → *Proportional Controller*
- To choose ω_n and ξ *arbitrarily*, *two* parameters needed → Proportional plus Derivative (PD) controller.

- For one link manipulator ω_n and ξ depends on controller gain K_p

$$\omega_n^2 = (KK_p/J), \quad \xi = \frac{F}{2\sqrt{JKK_p}}$$

- Changing K_p *changes both* ω_n and ξ.
- Can *make the output* under-damped, critically damped or over-damped by choosing K_p!!
- Simplest possible controller $\rightarrow$ *Proportional Controller*
- To choose ω_n and ξ *arbitrarily*, *two* parameters needed $\rightarrow$ Proportional plus Derivative (PD) controller.

- For one link manipulator ω_n and ξ depends on controller gain K_p

$$\omega_n^2 = (KK_p/J), \quad \xi = \frac{F}{2\sqrt{JKK_p}}$$

- Changing K_p *changes both* ω_n and ξ.
- Can *make the output* under-damped, critically damped or over-damped by choosing K_p!!
- Simplest possible controller $\rightarrow$ *Proportional Controller*
- To choose ω_n and ξ *arbitrarily*, *two* parameters needed $\rightarrow$ Proportional plus Derivative (PD) controller.

- Controller transfer function $D(s) = K_p + K_v s$, K_v *derivative* gain.
- The closed-loop transfer function

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p + sKK_v}{Js^2 + s(F + KK_v) + KK_p}$$

- ω_n and ξ related to K_p and K_v and can be set *arbitrarily*.
- Increasing K_v decreases overshoot but t_s becomes larger! For critical damping $K_v = 2\sqrt{K_p}$
- To obtain desired performance, need to use (computer) tools developed by researchers (see Franklin et al., 1991).

- Controller transfer function $D(s) = K_p + K_v s$, K_v *derivative* gain.
- The closed-loop transfer function

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p + sKK_v}{Js^2 + s(F + KK_v) + KK_p}$$

- ω_n and ξ related to K_p and K_v and can be set *arbitrarily*.
- Increasing K_v decreases overshoot but t_s becomes larger! For critical damping $K_v = 2\sqrt{K_p}$
- To obtain desired performance, need to use (computer) tools developed by researchers (see Franklin et al., 1991).

- Controller transfer function $D(s) = K_p + K_v s$, K_v *derivative* gain.
- The closed-loop transfer function

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p + sKK_v}{Js^2 + s(F + KK_v) + KK_p}$$

- ω_n and ξ related to K_p and K_v and can be set *arbitrarily*.
- Increasing K_v decreases overshoot but t_s becomes larger! For critical damping $K_v = 2\sqrt{K_p}$
- To obtain desired performance, need to use (computer) tools developed by researchers (see Franklin et al., 1991).

- Controller transfer function $D(s) = K_p + K_v s$, K_v *derivative* gain.
- The closed-loop transfer function

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p + sKK_v}{Js^2 + s(F + KK_v) + KK_p}$$

- ω_n and ξ related to K_p and K_v and can be set *arbitrarily*.
- Increasing K_v decreases overshoot but t_s becomes larger! For critical damping $K_v = 2\sqrt{K_p}$
- To obtain desired performance, need to use (computer) tools developed by researchers (see Franklin et al., 1991).

- Controller transfer function $D(s) = K_p + K_v s$, K_v *derivative* gain.
- The closed-loop transfer function

$$\frac{\theta(s)}{\theta_d(s)} = \frac{KK_p + sKK_v}{Js^2 + s(F + KK_v) + KK_p}$$

- ω_n and ξ related to K_p and K_v and can be set *arbitrarily*.
- Increasing K_v decreases overshoot but t_s becomes larger! For critical damping $K_v = 2\sqrt{K_p}$
- To obtain desired performance, need to use (computer) tools developed by researchers (see Franklin et al., 1991).

- To decrease steady state error (from backlash, friction/stiction), *integral* term is used.
- Integral term K_i/s – K_i is called the controller gain, must be chosen carefully → large K_i can make system unstable!
- sK_v term is not allowed[2] → PID controller

$$D(s) = K_p + \frac{K_i}{s} + \frac{K_v s}{1 + T_v s}$$

T_v is a (chosen) time constant and $s/(1 + T_v s)$ represents a filter.
- In time domain $V_a(t) = K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(t)dt$.
- Often *feed-forward* term added for improved *trajectory tracking* → modified PID controller

$$V_a(t) = \ddot{\theta}_d(t) + K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(t)dt$$

[2] In classical control, the numerator polynomial degree *must* be less than or equal to the denominator polynomial.

Control of a single link

PID Control

- To decrease steady state error (from backlash, friction/stiction), *integral* term is used.
- Integral term K_i/s – K_i is called the controller gain, must be chosen carefully $\rightarrow$ large K_i can make system unstable!
- sK_v term is not allowed[2] $\rightarrow$ PID controller

$$D(s) = K_p + \frac{K_i}{s} + \frac{K_v s}{1 + T_v s}$$

  T_v is a (chosen) time constant and $s/(1 + T_v s)$ represents a filter.
- In time domain $V_a(t) = K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(t)dt$.
- Often *feed-forward* term added for improved *trajectory tracking* $\rightarrow$ modified PID controller

$$V_a(t) = \ddot{\theta}_d(t) + K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(t)dt$$

[2] In classical control, the numerator polynomial degree *must* be less than or equal to the denominator polynomial.

- To decrease steady state error (from backlash, friction/stiction), *integral* term is used.
- Integral term K_i/s – K_i is called the controller gain, must be chosen carefully $\rightarrow$ large K_i can make system unstable!
- sK_v term is not allowed[2] $\rightarrow$ PID controller

$$D(s) = K_p + \frac{K_i}{s} + \frac{K_v s}{1 + T_v s}$$

T_v is a (chosen) time constant and $s/(1 + T_v s)$ represents a filter.

- In time domain $V_a(t) = K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(t)dt$.
- Often *feed-forward* term added for improved *trajectory tracking* $\rightarrow$ modified PID controller

$$V_a(t) = \ddot{\theta}_d(t) + K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(t)dt$$

[2] In classical control, the numerator polynomial degree *must* be less than or equal to the denominator polynomial.

Control of a single link

PID Control

- To decrease steady state error (from backlash, friction/stiction), *integral* term is used.
- Integral term K_i/s – K_i is called the controller gain, must be chosen carefully $\rightarrow$ large K_i can make system unstable!
- sK_v term is not allowed[2] $\rightarrow$ PID controller

$$D(s) = K_p + \frac{K_i}{s} + \frac{K_v s}{1 + T_v s}$$

  T_v is a (chosen) time constant and $s/(1 + T_v s)$ represents a filter.
- In time domain $V_a(t) = K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(t)dt$.
- Often *feed-forward* term added for improved *trajectory tracking* $\rightarrow$ modified PID controller

$$V_a(t) = \ddot{\theta}_d(t) + K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(t)dt$$

[2] In classical control, the numerator polynomial degree *must* be less than or equal to the denominator polynomial.

- To decrease steady state error (from backlash, friction/stiction), *integral* term is used.
- Integral term K_i/s – K_i is called the controller gain, must be chosen carefully $\rightarrow$ large K_i can make system unstable!
- sK_v term is not allowed[2] $\rightarrow$ PID controller

$$D(s) = K_p + \frac{K_i}{s} + \frac{K_v s}{1 + T_v s}$$

  T_v is a (chosen) time constant and $s/(1+T_v s)$ represents a filter.
- In time domain $V_a(t) = K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(t)dt$.
- Often *feed-forward* term added for improved *trajectory tracking* $\rightarrow$ modified PID controller

$$V_a(t) = \ddot{\theta}_d(t) + K_p e(t) + K_v \dot{e}(t) + K_i \int_0^t e(t)dt$$

[2] In classical control, the numerator polynomial degree *must* be less than or equal to the denominator polynomial.

CONTROL OF A SINGLE LINK

DIGITAL CONTROL

- Most modern controller are implemented using digital microprocessors.
- No longer *continuous* time control → *discrete-time* control – Sampling.

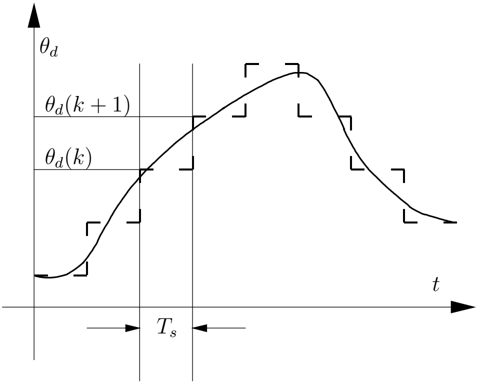


Figure 10: Discretisation of $\theta_d(t)$

- Desired input $\theta_d(t)$ and the output $\theta(t)$ are *not* continuous → only dashed lines available.
- Analog to digital conversion is done electronically
- Typical sampling time, T_s, is between 1 and 10 milli-seconds and typically 8 – 12 bits used in A/D conversion.
- Less difference if number of bits in A/D conversion is more.

Control of a single link

Digital Control (contd.)

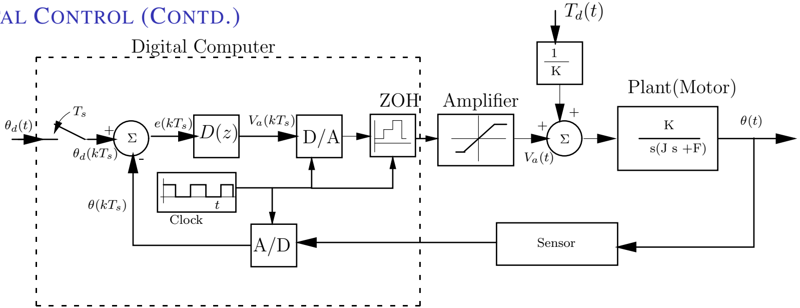


Figure 11: Block diagram of a digital controller

- Sampling performed by an *independent* clock which *interrupts* the microprocessor.
- $\theta_d(kT_s)$ and $\theta(kT_s)$ are the kth desired and measured θ.
- Error $e(kT_s) = \theta_d(kT_s) - \theta(kT_s)$ computed as a digital value.

Control of a single link

Digital Control (contd.)

- Error is input to the controller $D(z) \rightarrow$ output is *discretised* voltage.
- Discretised voltage *converted* to analog in a D/A converter and using a zero order hold *ZOH*.
- The D/A and ZOH introduces *delay* $\rightarrow$ source of many complications!
- Output of microprocessor in milliamperes $\rightarrow$ needs to be amplified to drive motor.
- Controller designed using *discreet controls* and z transform (see textbook by Franklin et al., 1990)

$$D(z) = K_p + \frac{K_i T_s}{1 - z^{-1}} + \frac{K_v(1 - z^{-1})}{T_s + T_v(1 - z^{-1})}$$

Control of a single link

Digital Control (Contd.)

- Error is input to the controller $D(z)$ $\rightarrow$ output is *discretised* voltage.
- Discretised voltage *converted* to analog in a D/A converter and using a zero order hold *ZOH*.
- The D/A and ZOH introduces *delay* $\rightarrow$ source of many complications!
- Output of microprocessor in milliamperes $\rightarrow$ needs to be amplified to drive motor.
- Controller designed using *discreet controls* and z transform (see textbook by Franklin et al., 1990)

$$D(z) = K_p + \frac{K_i T_s}{1 - z^{-1}} + \frac{K_v(1 - z^{-1})}{T_s + T_v(1 - z^{-1})}$$

- Error is input to the controller $D(z)$ $\rightarrow$ output is *discretised* voltage.
- Discretised voltage *converted* to analog in a D/A converter and using a zero order hold *ZOH*.
- The D/A and ZOH introduces *delay* $\rightarrow$ source of many complications!
- Output of microprocessor in milliamperes $\rightarrow$ needs to be amplified to drive motor.
- Controller designed using *discreet controls* and z transform (see textbook by Franklin et al., 1990)

$$D(z) = K_p + \frac{K_i T_s}{1 - z^{-1}} + \frac{K_v(1 - z^{-1})}{T_s + T_v(1 - z^{-1})}$$

- Error is input to the controller $D(z)$ $\rightarrow$ output is *discretised* voltage.
- Discretised voltage *converted* to analog in a D/A converter and using a zero order hold *ZOH*.
- The D/A and ZOH introduces *delay* $\rightarrow$ source of many complications!
- Output of microprocessor in milliamperes $\rightarrow$ needs to be amplified to drive motor.
- Controller designed using *discreet controls* and z transform (see textbook by Franklin et al., 1990)

$$D(z) = K_p + \frac{K_i T_s}{1 - z^{-1}} + \frac{K_v(1 - z^{-1})}{T_s + T_v(1 - z^{-1})}$$

- Error is input to the controller $D(z)$ → output is *discretised* voltage.
- Discretised voltage *converted* to analog in a D/A converter and using a zero order hold *ZOH*.
- The D/A and ZOH introduces *delay* → source of many complications!
- Output of microprocessor in milliamperes → needs to be amplified to drive motor.
- Controller designed using *discreet controls* and z transform (see textbook by Franklin et al., 1990)

$$D(z) = K_p + \frac{K_i T_s}{1 - z^{-1}} + \frac{K_v(1 - z^{-1})}{T_s + T_v(1 - z^{-1})}$$

OUTLINE

1. CONTENTS
2. LECTURE 1
   - Motion planning
3. LECTURE 2
   - Control of a single link
4. LECTURE 3
   - Control of a multi-link serial manipulator
5. LECTURE 4*
   - Control of constrained and parallel manipulator
   - Cartesian control of serial manipulators
6. LECTURE 5*
   - Force control of manipulators
   - Hybrid position/force control of manipulators
7. LECTURE 6*
   - Advanced topics in non-linear control of manipulators
8. MODULE 7 – ADDITIONAL MATERIAL
   - Problems, References and Suggested Reading

- Multi-link $\rightarrow$ n joint variables – $\mathbf{q}$.
- Desired joint motion, $\mathbf{q}_d(t)$, available from motion planning.
- Assume $\dot{\mathbf{q}}_d(t)$ and $\ddot{\mathbf{q}}_d(t)$ also available – see cubic trajectory plan!
- PD control of multi-link manipulator – actual implementation is PID.
- Non-linear control of multi-link manipulator.
- Simulation and experimental results.

- Multi-link $\rightarrow$ n joint variables – $\mathbf{q}$.
- Desired joint motion, $\mathbf{q}_d(t)$, available from motion planning.
- Assume $\dot{\mathbf{q}}_d(t)$ and $\ddot{\mathbf{q}}_d(t)$ also available – see cubic trajectory plan!
- PD control of multi-link manipulator – actual implementation is PID.
- Non-linear control of multi-link manipulator.
- Simulation and experimental results.

- Multi-link $\rightarrow$ n joint variables – $\mathbf{q}$.
- Desired joint motion, $\mathbf{q}_d(t)$, available from motion planning.
- Assume $\dot{\mathbf{q}}_d(t)$ and $\ddot{\mathbf{q}}_d(t)$ also available – see cubic trajectory plan!
- PD control of multi-link manipulator – actual implementation is PID.
- Non-linear control of multi-link manipulator.
- Simulation and experimental results.

- Multi-link $\rightarrow$ n joint variables – $\mathbf{q}$.
- Desired joint motion, $\mathbf{q}_d(t)$, available from motion planning.
- Assume $\dot{\mathbf{q}}_d(t)$ and $\ddot{\mathbf{q}}_d(t)$ also available – see cubic trajectory plan!
- PD control of multi-link manipulator – actual implementation is PID.
- Non-linear control of multi-link manipulator.
- Simulation and experimental results.

- Multi-link $\rightarrow$ n joint variables – $\mathbf{q}$.
- Desired joint motion, $\mathbf{q}_d(t)$, available from motion planning.
- Assume $\dot{\mathbf{q}}_d(t)$ and $\ddot{\mathbf{q}}_d(t)$ also available – see cubic trajectory plan!
- PD control of multi-link manipulator – actual implementation is PID.
- Non-linear control of multi-link manipulator.
- Simulation and experimental results.

- Multi-link $\rightarrow$ n joint variables – $\mathbf{q}$.
- Desired joint motion, $\mathbf{q}_d(t)$, available from motion planning.
- Assume $\dot{\mathbf{q}}_d(t)$ and $\ddot{\mathbf{q}}_d(t)$ also available – see cubic trajectory plan!
- PD control of multi-link manipulator – actual implementation is PID.
- Non-linear control of multi-link manipulator.
- Simulation and experimental results.

- Extend continuous time control of single link manipulator.
- Feed-forward plus PD instead of PID control algorithm for analysis

$$V_a(t) = \ddot{q}_d(t) + K_v \dot{e}(t) + K_p e(t), \quad e(t) = q_d(t) - q(t)$$

  Implemented control will also have a integral term!
- Use torque τ acting at the joint instead of voltage V_a in analysis[3].
- Control law used in analysis

$$\tau(t) = \ddot{q}_d(t) + K_v \dot{e}(t) + K_p e(t), \quad e(t) = q_d(t) - q(t)$$

- Linear control law applied to a non-linear system!

[3] Joint torque is related to the applied voltage at the motor terminals since $T_m = K_t i_a = (K_t/R_a)(V_a - K_g \dot{\theta}_m)$ and $\tau = T_m/n$. One can also find V_a from motor characteristics curves.

- Extend continuous time control of single link manipulator.
- Feed-forward plus PD instead of PID control algorithm for analysis

$$V_a(t) = \ddot{q}_d(t) + K_v \dot{e}(t) + K_p e(t), \quad e(t) = q_d(t) - q(t)$$

  Implemented control will also have a integral term!
- Use torque τ acting at the joint instead of voltage V_a in analysis[3].
- Control law used in analysis

$$\tau(t) = \ddot{q}_d(t) + K_v \dot{e}(t) + K_p e(t), \quad e(t) = q_d(t) - q(t)$$

- Linear control law applied to a non-linear system!

[3] Joint torque is related to the applied voltage at the motor terminals since $T_m = K_t i_a = (K_t/R_a)(V_a - K_g \dot{\theta}_m)$ and $\tau = T_m/n$. One can also find V_a from motor characteristics curves.

- Extend continuous time control of single link manipulator.
- Feed-forward plus PD instead of PID control algorithm for analysis

$$V_a(t) = \ddot{q}_d(t) + K_v \dot{e}(t) + K_p e(t), \quad e(t) = q_d(t) - q(t)$$

  Implemented control will also have a integral term!
- Use torque τ acting at the joint instead of voltage V_a in analysis[3].
- Control law used in analysis

$$\tau(t) = \ddot{q}_d(t) + K_v \dot{e}(t) + K_p e(t), \quad e(t) = q_d(t) - q(t)$$

- Linear control law applied to a non-linear system!

[3] Joint torque is related to the applied voltage at the motor terminals since $T_m = K_t i_a = (K_t/R_a)(V_a - K_g \dot{\theta}_m)$ and $\tau = T_m/n$. One can also find V_a from motor characteristics curves.

- Extend continuous time control of single link manipulator.
- Feed-forward plus PD instead of PID control algorithm for analysis

$$V_a(t) = \ddot{q}_d(t) + K_v \dot{e}(t) + K_p e(t), \quad e(t) = q_d(t) - q(t)$$

Implemented control will also have a integral term!

- Use torque τ acting at the joint instead of voltage V_a in analysis[3].
- Control law used in analysis

$$\tau(t) = \ddot{q}_d(t) + K_v \dot{e}(t) + K_p e(t), \quad e(t) = q_d(t) - q(t)$$

- Linear control law applied to a non-linear system!

[3] Joint torque is related to the applied voltage at the motor terminals since $T_m = K_t i_a = (K_t/R_a)(V_a - K_g \dot{\theta}_m)$ and $\tau = T_m/n$. One can also find V_a from motor characteristics curves.

- Extend continuous time control of single link manipulator.
- Feed-forward plus PD instead of PID control algorithm for analysis

$$V_a(t) = \ddot{q}_d(t) + K_v\dot{e}(t) + K_p e(t), \quad e(t) = q_d(t) - q(t)$$

  Implemented control will also have a integral term!
- Use torque τ acting at the joint instead of voltage V_a in analysis[3].
- Control law used in analysis

$$\tau(t) = \ddot{q}_d(t) + K_v\dot{e}(t) + K_p e(t), \quad e(t) = q_d(t) - q(t)$$

- Linear control law applied to a non-linear system!

[3] Joint torque is related to the applied voltage at the motor terminals since $T_m = K_t i_a = (K_t/R_a)(V_a - K_g\dot{\theta}_m)$ and $\tau = T_m/n$. One can also find V_a from motor characteristics curves.

PD CONTROL OF MULTI-LINK SERIAL MANIPULATOR

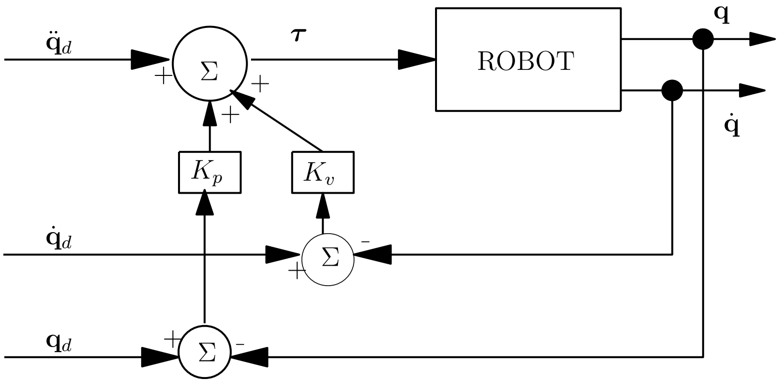


Figure 12: PD control of a multi-link robot

- Each joint or motor independently controlled.
- All quantities, $\mathbf{q}_d$, $\mathbf{q}$, τ are $n \times 1$ vectors (n DOF manipulator)
- $[K_p]$ and $[K_v]$ are $n \times n$ positive-definite proportional and derivative controller gain matrices.

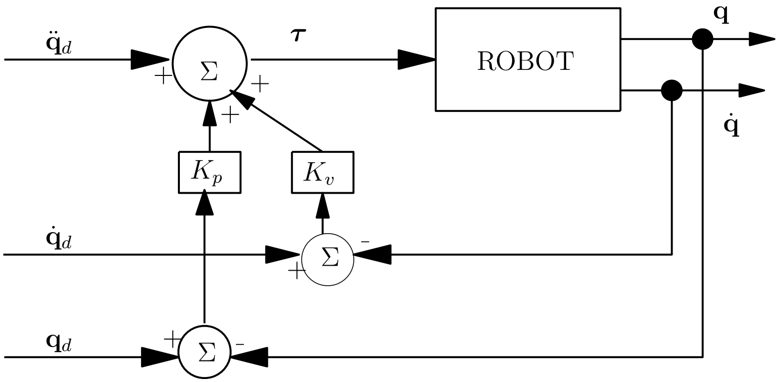


Figure 12: PD control of a multi-link robot

- Each joint or motor independently controlled.
- All quantities, $\mathbf{q}_d$, $\mathbf{q}$, τ are $n \times 1$ vectors (n DOF manipulator)
- $[K_p]$ and $[K_v]$ are $n \times n$ positive-definite proportional and derivative controller gain matrices.

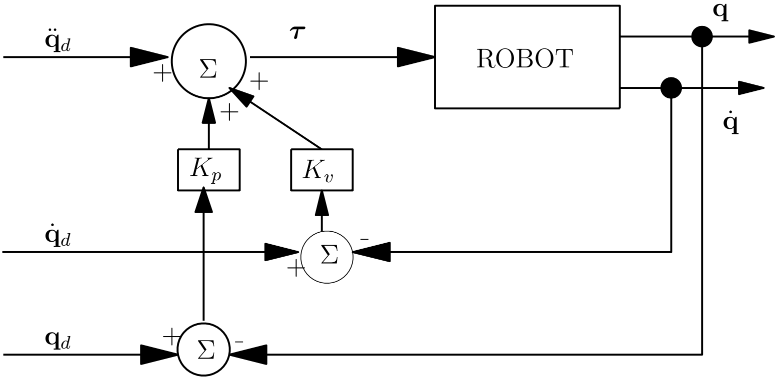


Figure 12: PD control of a multi-link robot

- Each joint or motor independently controlled.
- All quantities, $\mathbf{q}_d$, $\mathbf{q}$, τ are $n \times 1$ vectors (n DOF manipulator)
- $[K_p]$ and $[K_v]$ are $n \times n$ positive-definite proportional and derivative controller gain matrices.

- Multi-link manipulator is a non-linear system → Cannot expect *uniform* damping and settling time *everywhere* in workspace.
- Reason for working – slow speed and large gear ratio at joints!
- Linear control law implemented using one or more microprocessors
- Two main kinds of architecture commonly used.
  - *Joint parallel* – each joint (PID) controlled by a micro-processor & additional master or 'coordinating' processor for GUI, data logging etc.
  - *Functional parallel* – Each/group of function(s)/task(s) handled by a processor.
- Original PUMA robot – 6503 microprocessor at joints and DEC LSI-11 for master, θ_d available every 28 msec and T_s for joint processor was 0.875 msec, high-level language *VAL* for robot programming.
- Modern solution – add-on cards for industrial PC's to control several joints.

- Multi-link manipulator is a non-linear system $\rightarrow$ Cannot expect *uniform* damping and settling time *everywhere* in workspace.
- Reason for working – slow speed and large gear ratio at joints!
- Linear control law implemented using one or more microprocessors
- Two main kinds of architecture commonly used.
  - *Joint parallel* – each joint (PID) controlled by a micro-processor & additional master or 'coordinating' processor for GUI, data logging etc.
  - *Functional parallel* – Each/group of function(s)/task(s) handled by a processor.
- Original PUMA robot – 6503 microprocessor at joints and DEC LSI-11 for master, θ_d available every 28 msec and T_s for joint processor was 0.875 msec, high-level language *VAL* for robot programming.
- Modern solution – add-on cards for industrial PC's to control several joints.

- Multi-link manipulator is a non-linear system $\rightarrow$ Cannot expect *uniform* damping and settling time *everywhere* in workspace.
- Reason for working – slow speed and large gear ratio at joints!
- Linear control law implemented using one or more microprocessors
- Two main kinds of architecture commonly used.
  - *Joint parallel* – each joint (PID) controlled by a micro-processor & additional master or 'coordinating' processor for GUI, data logging etc.
  - *Functional parallel* – Each/group of function(s)/task(s) handled by a processor.
- Original PUMA robot – 6503 microprocessor at joints and DEC LSI-11 for master, θ_d available every 28 msec and T_s for joint processor was 0.875 msec, high-level language *VAL* for robot programming.
- Modern solution – add-on cards for industrial PC's to control several joints.

- Multi-link manipulator is a non-linear system → Cannot expect *uniform* damping and settling time *everywhere* in workspace.
- Reason for working – slow speed and large gear ratio at joints!
- Linear control law implemented using one or more microprocessors
- Two main kinds of architecture commonly used.
  - *Joint parallel* – each joint (PID) controlled by a micro-processor & additional master or 'coordinating' processor for GUI, data logging etc.
  - *Functional parallel* – Each/group of function(s)/task(s) handled by a processor.
- Original PUMA robot – 6503 microprocessor at joints and DEC LSI-11 for master, θ_d available every 28 msec and T_s for joint processor was 0.875 msec, high-level language *VAL* for robot programming.
- Modern solution – add-on cards for industrial PC's to control several joints.

- Multi-link manipulator is a non-linear system → Cannot expect *uniform* damping and settling time *everywhere* in workspace.
- Reason for working – slow speed and large gear ratio at joints!
- Linear control law implemented using one or more microprocessors
- Two main kinds of architecture commonly used.
  - *Joint parallel* – each joint (PID) controlled by a micro-processor & additional master or 'coordinating' processor for GUI, data logging etc.
  - *Functional parallel* – Each/group of function(s)/task(s) handled by a processor.
- Original PUMA robot – 6503 microprocessor at joints and DEC LSI-11 for master, θ_d available every 28 msec and T_s for joint processor was 0.875 msec, high-level language *VAL* for robot programming.
- Modern solution – add-on cards for industrial PC's to control several joints.

- Multi-link manipulator is a non-linear system → Cannot expect *uniform* damping and settling time *everywhere* in workspace.
- Reason for working – slow speed and large gear ratio at joints!
- Linear control law implemented using one or more microprocessors
- Two main kinds of architecture commonly used.
  - *Joint parallel* – each joint (PID) controlled by a micro-processor & additional master or 'coordinating' processor for GUI, data logging etc.
  - *Functional parallel* – Each/group of function(s)/task(s) handled by a processor.
- Original PUMA robot – 6503 microprocessor at joints and DEC LSI-11 for master, θ_d available every 28 msec and T_s for joint processor was 0.875 msec, high-level language *VAL* for robot programming.
- Modern solution – add-on cards for industrial PC's to control several joints.

- Non-linear control – a vast field!
- One particular kind of non-linear controller – *computed torque* (also called *feedback linearizing*) control scheme.
- In ideal situations can give *uniform* performance *everywhere* in workspace!
- Uses dynamic model in the control scheme.
- The better the estimate of the dynamic model, better the performance.
- Large amount of literature – first popularised by Freund (1982) in turn uses results of Singh and Rugh(1972)

- Non-linear control – a vast field!
- One particular kind of non-linear controller – *computed torque* (also called *feedback linearizing*) control scheme.
- In ideal situations can give *uniform* performance *everywhere* in workspace!
- Uses dynamic model in the control scheme.
- The better the estimate of the dynamic model, better the performance.
- Large amount of literature – first popularised by Freund (1982) in turn uses results of Singh and Rugh(1972)

Non-linear control of multi-link serial manipulator

Introduction

- Non-linear control – a vast field!
- One particular kind of non-linear controller – *computed torque* (also called *feedback linearizing*) control scheme.
- In ideal situations can give *uniform* performance *everywhere* in workspace!
- Uses dynamic model in the control scheme.
- The better the estimate of the dynamic model, better the performance.
- Large amount of literature – first popularised by Freund (1982) in turn uses results of Singh and Rugh(1972)

- Non-linear control – a vast field!
- One particular kind of non-linear controller – *computed torque* (also called *feedback linearizing*) control scheme.
- In ideal situations can give *uniform* performance *everywhere* in workspace!
- Uses dynamic model in the control scheme.
- The better the estimate of the dynamic model, better the performance.
- Large amount of literature – first popularised by Freund (1982) in turn uses results of Singh and Rugh(1972)

Non-linear control of multi-link serial manipulator

Introduction

- Non-linear control – a vast field!
- One particular kind of non-linear controller – *computed torque* (also called *feedback linearizing*) control scheme.
- In ideal situations can give *uniform* performance *everywhere* in workspace!
- Uses dynamic model in the control scheme.
- The better the estimate of the dynamic model, better the performance.
- Large amount of literature – first popularised by Freund (1982) in turn uses results of Singh and Rugh(1972)

- Non-linear control – a vast field!
- One particular kind of non-linear controller – *computed torque* (also called *feedback linearizing*) control scheme.
- In ideal situations can give *uniform* performance *everywhere* in workspace!
- Uses dynamic model in the control scheme.
- The better the estimate of the dynamic model, better the performance.
- Large amount of literature – first popularised by Freund (1982) in turn uses results of Singh and Rugh(1972)

Non-linear control of multi-link serial manipulator

Control law partitioning

- Dynamic equations of motion for a serial manipulator (see Module 6, Lecture 1)

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$

  $[\mathbf{M}(\mathbf{q})]$ is an $n \times n$ mass matrix and $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$, and $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$ are $n \times 1$ vectors representing Coriolis/centripetal, gravity, and friction terms, respectively.

- Write $n \times 1$ vector τ of joint torques as,

$$\tau = [\alpha]\tau' + \beta$$

- Choose

$$[\alpha] = [\mathbf{M}(\mathbf{q})], \quad \beta = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$

- Using equations of motion,

$$\tau' = \ddot{\mathbf{q}}$$

Non-linear control of multi-link serial manipulator

Control law partitioning

- Dynamic equations of motion for a serial manipulator (see Module 6, Lecture 1)

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$$

  $[\mathbf{M}(\mathbf{q})]$ is an $n \times n$ mass matrix and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$, and $\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$ are $n \times 1$ vectors representing Coriolis/centripetal, gravity, and friction terms, respectively.
- Write $n \times 1$ vector τ of joint torques as,

$$\tau = [\alpha]\tau' + \beta$$

- Choose

$$[\alpha] = [\mathbf{M}(\mathbf{q})], \quad \beta = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$$

- Using equations of motion,

$$\tau' = \ddot{\mathbf{q}}$$

Non-linear control of multi-link serial manipulator

Control law partitioning

- Dynamic equations of motion for a serial manipulator (see Module 6, Lecture 1)

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$

  $[\mathbf{M}(\mathbf{q})]$ is an $n \times n$ mass matrix and $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$, and $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$ are $n \times 1$ vectors representing Coriolis/centripetal, gravity, and friction terms, respectively.

- Write $n \times 1$ vector τ of joint torques as,

$$\tau = [\alpha]\tau' + \beta$$

- Choose

$$[\alpha] = [\mathbf{M}(\mathbf{q})], \quad \beta = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$

- Using equations of motion,

$$\tau' = \ddot{\mathbf{q}}$$

Non-linear control of multi-link serial manipulator

Control law partitioning

- Dynamic equations of motion for a serial manipulator (see Module 6, Lecture 1)

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$

  $[\mathbf{M}(\mathbf{q})]$ is an $n \times n$ mass matrix and $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$, and $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$ are $n \times 1$ vectors representing Coriolis/centripetal, gravity, and friction terms, respectively.
- Write $n \times 1$ vector τ of joint torques as,

$$\tau = [\alpha]\tau' + \beta$$

- Choose

$$[\alpha] = [\mathbf{M}(\mathbf{q})], \quad \beta = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$

- Using equations of motion,

$$\tau' = \ddot{\mathbf{q}}$$

NON-LINEAR CONTROL OF MULTI-LINK SERIAL MANIPULATOR

CONTROL LAW PARTITIONING (CONTD.)

- The equation $\tau' = \ddot{\mathbf{q}}$ represents a *unit* inertia system with input τ'.
- The dynamics represented by $[\alpha]$ and β are used.
- All *non-linearities* & *coupling* are 'canceled' and original non-linear equations *transformed* to *n decoupled linear* equations.
- Choose

$$\tau' = \ddot{\mathbf{q}}_d(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t)$$

- Error equation becomes

$$\ddot{\mathbf{e}}(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t) = \mathbf{0}$$

- Choose positive-definite, diagonal matrices $[K_p]$ and $[K_v]$, to get *critical* damping at *every* point in the workspace!!

Non-linear control of multi-link serial manipulator

Control law partitioning (Contd.)

- The equation $\tau' = \ddot{\mathbf{q}}$ represents a *unit* inertia system with input τ'.
- The dynamics represented by $[\alpha]$ and β are used.
- All *non-linearities & coupling* are 'canceled' and original non-linear equations *transformed* to *n decoupled linear* equations.
- Choose

$$\tau' = \ddot{\mathbf{q}}_d(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t)$$

- Error equation becomes

$$\ddot{\mathbf{e}}(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t) = \mathbf{0}$$

- Choose positive-definite, diagonal matrices $[K_p]$ and $[K_v]$, to get *critical* damping at *every* point in the workspace!!

Non-linear control of multi-link serial manipulator

Control law partitioning (Contd.)

- The equation $\tau' = \ddot{\mathbf{q}}$ represents a *unit* inertia system with input τ'.
- The dynamics represented by $[\alpha]$ and β are used.
- All *non-linearities* & *coupling* are 'canceled' and original non-linear equations *transformed* to *n decoupled linear* equations.
- Choose

$$\tau' = \ddot{\mathbf{q}}_d(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t)$$

- Error equation becomes

$$\ddot{\mathbf{e}}(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t) = \mathbf{0}$$

- Choose positive-definite, diagonal matrices $[K_p]$ and $[K_v]$, to get *critical* damping at *every* point in the workspace!!

Non-linear control of multi-link serial manipulator

Control law partitioning (Contd.)

- The equation $\tau' = \ddot{\mathbf{q}}$ represents a *unit* inertia system with input τ'.
- The dynamics represented by $[\alpha]$ and β are used.
- All *non-linearities* & *coupling* are 'canceled' and original non-linear equations *transformed* to *n decoupled linear* equations.
- Choose

$$\tau' = \ddot{\mathbf{q}}_d(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t)$$

- Error equation becomes

$$\ddot{\mathbf{e}}(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t) = \mathbf{0}$$

- Choose positive-definite, diagonal matrices $[K_p]$ and $[K_v]$, to get *critical* damping at *every* point in the workspace!!

Non-linear control of multi-link serial manipulator

Control law partitioning (Contd.)

- The equation $\tau' = \ddot{\mathbf{q}}$ represents a *unit* inertia system with input τ'.
- The dynamics represented by $[\alpha]$ and β are used.
- All *non-linearities* & *coupling* are 'canceled' and original non-linear equations *transformed* to n *decoupled linear* equations.
- Choose

$$\tau' = \ddot{\mathbf{q}}_d(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t)$$

- Error equation becomes

$$\ddot{\mathbf{e}}(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t) = \mathbf{0}$$

- Choose positive-definite, diagonal matrices $[K_p]$ and $[K_v]$, to get *critical* damping at *every* point in the workspace!!

- The equation $\tau' = \ddot{\mathbf{q}}$ represents a *unit* inertia system with input τ'.
- The dynamics represented by $[\alpha]$ and β are used.
- All *non-linearities* & *coupling* are 'canceled' and original non-linear equations *transformed* to *n decoupled linear* equations.
- Choose

$$\tau' = \ddot{\mathbf{q}}_d(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t)$$

- Error equation becomes

$$\ddot{\mathbf{e}}(t) + [K_v]\dot{\mathbf{e}}(t) + [K_p]\mathbf{e}(t) = \mathbf{0}$$

- Choose positive-definite, diagonal matrices $[K_p]$ and $[K_v]$, to get *critical* damping at *every* point in the workspace!!

Non-linear control of multi-link serial manipulator

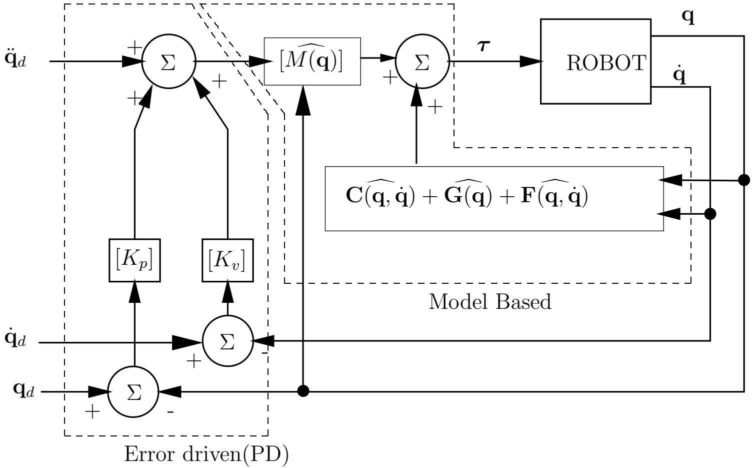


Figure 13: Computed torque control scheme for robots

- Two partitions – Error driven PD control and Model-based

Non-linear control of multi-link serial manipulator

Control law partitioning (Contd.)

- "Ideal" performance not possible
  - Time required to compute $[\alpha]$ and β $\rightarrow$ during this time $\mathbf{q}$ changes!
  - Manipulator parameters such as mass, inertia etc. not known *exactly*!
- Only estimates of $[\mathbf{M}(\mathbf{q})]$, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$ and $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$ available $\rightarrow$ symbol $[\widehat{\mathbf{M}(\mathbf{q})}]$ etc. used in figure.
- Estimates $\rightarrow$ Error equation no longer linear and decoupled.
- If $[\alpha] = [\widehat{\mathbf{M}(\mathbf{q})}]$ and $\beta = \widehat{\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})} + \widehat{\mathbf{G}(\mathbf{q})} + \widehat{\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})}$, then error equation

$$\begin{aligned} \ddot{\mathbf{e}} &+ [K_v]\dot{\mathbf{e}} + [K_p]\mathbf{e} \\ &= [\widehat{\mathbf{M}}]^{-1}\left[([\mathbf{M}] - [\widehat{\mathbf{M}}])\ddot{\mathbf{q}} + (\mathbf{C} - \widehat{\mathbf{C}}) + (\mathbf{G} - \widehat{\mathbf{G}}) + (\mathbf{F} - \widehat{\mathbf{F}})\right] \end{aligned}$$

- If $([\mathbf{M}] - [\widehat{\mathbf{M}}]) = [0]$ etc. then "exact cancellation".

Non-linear control of multi-link serial manipulator

Control law partitioning (Contd.)

- "Ideal" performance not possible
  - Time required to compute $[\alpha]$ and β $\rightarrow$ during this time $\mathbf{q}$ changes!
  - Manipulator parameters such as mass, inertia etc. not known *exactly*!
- Only estimates of $[\mathbf{M}(\mathbf{q})]$, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$ and $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$ available $\rightarrow$ symbol $[\widehat{\mathbf{M}(\mathbf{q})}]$ etc. used in figure.
- Estimates $\rightarrow$ Error equation no longer linear and decoupled.
- If $[\alpha] = [\widehat{\mathbf{M}(\mathbf{q})}]$ and $\beta = \widehat{\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})} + \widehat{\mathbf{G}(\mathbf{q})} + \widehat{\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})}$, then error equation

$$\begin{aligned} \ddot{\mathbf{e}} &+ [K_v]\dot{\mathbf{e}} + [K_p]\mathbf{e} \\ &= [\widehat{\mathbf{M}}]^{-1}\left[([\mathbf{M}] - [\widehat{\mathbf{M}}])\ddot{\mathbf{q}} + (\mathbf{C} - \widehat{\mathbf{C}}) + (\mathbf{G} - \widehat{\mathbf{G}}) + (\mathbf{F} - \widehat{\mathbf{F}})\right] \end{aligned}$$

- If $([\mathbf{M}] - [\widehat{\mathbf{M}}]) = [0]$ etc. then "exact cancellation".

Non-linear control of multi-link serial manipulator

Control law partitioning (Contd.)

- "Ideal" performance not possible
  - Time required to compute $[\alpha]$ and β $\rightarrow$ during this time $\mathbf{q}$ changes!
  - Manipulator parameters such as mass, inertia etc. not known *exactly*!
- Only estimates of $[\mathbf{M}(\mathbf{q})]$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$ and $\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$ available $\rightarrow$ symbol $[\widehat{\mathbf{M}(\mathbf{q})}]$ etc. used in figure.
- Estimates $\rightarrow$ Error equation no longer linear and decoupled.
- If $[\alpha] = [\widehat{\mathbf{M}(\mathbf{q})}]$ and $\beta = \widehat{\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})} + \widehat{\mathbf{G}(\mathbf{q})} + \widehat{\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})}$, then error equation

$$\begin{aligned} \ddot{\mathbf{e}} &+ [K_v]\dot{\mathbf{e}} + [K_p]\mathbf{e} \\ &= [\widehat{\mathbf{M}}]^{-1}\left[([\mathbf{M}] - [\widehat{\mathbf{M}}])\ddot{\mathbf{q}} + (\mathbf{C} - \widehat{\mathbf{C}}) + (\mathbf{G} - \widehat{\mathbf{G}}) + (\mathbf{F} - \widehat{\mathbf{F}})\right] \end{aligned}$$

- If $([\mathbf{M}] - [\widehat{\mathbf{M}}]) = [0]$ etc. then "exact cancellation".

Control law partitioning (Contd.)

- "Ideal" performance not possible
  - Time required to compute $[\alpha]$ and β $\rightarrow$ during this time $\mathbf{q}$ changes!
  - Manipulator parameters such as mass, inertia etc. not known *exactly*!
- Only estimates of $[\mathbf{M}(\mathbf{q})]$, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$ and $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$ available $\rightarrow$ symbol $[\widehat{\mathbf{M}(\mathbf{q})}]$ etc. used in figure.
- Estimates $\rightarrow$ Error equation no longer linear and decoupled.
- If $[\alpha] = [\widehat{\mathbf{M}(\mathbf{q})}]$ and $\beta = \widehat{\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})} + \widehat{\mathbf{G}(\mathbf{q})} + \widehat{\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})}$, then error equation

$$
\begin{aligned}
\ddot{\mathbf{e}} &+ [K_v]\dot{\mathbf{e}} + [K_p]\mathbf{e} \\
&= [\widehat{\mathbf{M}}]^{-1}\left[([\mathbf{M}] - [\widehat{\mathbf{M}}])\ddot{\mathbf{q}} + (\mathbf{C} - \widehat{\mathbf{C}}) + (\mathbf{G} - \widehat{\mathbf{G}}) + (\mathbf{F} - \widehat{\mathbf{F}})\right]
\end{aligned}
$$

- If $([\mathbf{M}] - [\widehat{\mathbf{M}}]) = [0]$ etc. then "exact cancellation".

Non-linear control of multi-link serial manipulator

Control law partitioning (Contd.)

- "Ideal" performance not possible
  - Time required to compute $[\alpha]$ and β $\rightarrow$ during this time $\mathbf{q}$ changes!
  - Manipulator parameters such as mass, inertia etc. not known *exactly*!
- Only estimates of $[\mathbf{M}(\mathbf{q})]$, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$ and $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$ available $\rightarrow$ symbol $[\widehat{\mathbf{M}(\mathbf{q})}]$ etc. used in figure.
- Estimates $\rightarrow$ Error equation no longer linear and decoupled.
- If $[\alpha] = [\widehat{\mathbf{M}(\mathbf{q})}]$ and $\beta = \widehat{\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})} + \widehat{\mathbf{G}(\mathbf{q})} + \widehat{\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})}$, then error equation

$$\begin{aligned}\ddot{\mathbf{e}} &+ [K_v]\dot{\mathbf{e}} + [K_p]\mathbf{e} \\ &= [\widehat{\mathbf{M}}]^{-1}\left[([\mathbf{M}] - [\widehat{\mathbf{M}}])\ddot{\mathbf{q}} + (\mathbf{C} - \widehat{\mathbf{C}}) + (\mathbf{G} - \widehat{\mathbf{G}}) + (\mathbf{F} - \widehat{\mathbf{F}})\right]\end{aligned}$$

- If $([\mathbf{M}] - [\widehat{\mathbf{M}}]) = [\mathbf{0}]$ etc. then "exact cancellation".

NON-LINEAR CONTROL OF MULTI-LINK SERIAL MANIPULATOR

CONTROL LAW PARTITIONING (CONTD.)

- Special cases of computed torque scheme
  - $[\alpha] = [U]$ and $\beta = \mathbf{G}(\mathbf{q}) \rightarrow$ Gravity compensation.
  - No model used $\rightarrow [\alpha] = [U]$ and $\beta = \mathbf{0} \rightarrow$ PD control scheme.
  - *Feed-forward* control law

$$[\alpha] = [\widehat{\mathbf{M}(\mathbf{q}_d)}], \quad \beta = \mathbf{C}\widehat{(\mathbf{q}_d, \dot{\mathbf{q}}_d)} + \widehat{\mathbf{G}(\mathbf{q}_d)} + \mathbf{F}\widehat{(\mathbf{q}_d, \dot{\mathbf{q}}_d)}$$

  - Model terms computed according to *desired* trajectory and *not* in the feed-back loop.
  - Model terms can be computed off-line $\rightarrow$ Almost no issue of computation time.
- No "exact" cancellation in special cases $\rightarrow$ No decoupling or linearity.
- If estimates are good, then right-hand side is small! $\rightarrow$ performance better than PD.
- Borne out by simulations and experiments.

Non-linear control of multi-link serial manipulator

Control law partitioning (Contd.)

- Special cases of computed torque scheme
  - $[\alpha] = [U]$ and $\beta = \mathbf{G}(\mathbf{q}) \rightarrow$ Gravity compensation.
  - No model used $\rightarrow [\alpha] = [U]$ and $\beta = \mathbf{0} \rightarrow$ PD control scheme.
  - *Feed-forward* control law

$$[\alpha] = [\widehat{\mathbf{M}(\mathbf{q}_d)}], \quad \beta = \mathbf{C}\widehat{(\mathbf{q}_d, \dot{\mathbf{q}}_d)} + \widehat{\mathbf{G}(\mathbf{q}_d)} + \mathbf{F}\widehat{(\mathbf{q}_d, \dot{\mathbf{q}}_d)}$$

  - Model terms computed according to *desired* trajectory and *not* in the feed-back loop.
  - Model terms can be computed off-line $\rightarrow$ Almost no issue of computation time.
- No "exact" cancellation in special cases $\rightarrow$ No decoupling or linearity.
- If estimates are good, then right-hand side is small! $\rightarrow$ performance better than PD.
- Borne out by simulations and experiments.

CONTROL LAW PARTITIONING (CONTD.)

- Special cases of computed torque scheme
  - $[\alpha] = [U]$ and $\beta = \mathbf{G}(\mathbf{q}) \rightarrow$ Gravity compensation.
  - No model used $\rightarrow [\alpha] = [U]$ and $\beta = \mathbf{0} \rightarrow$ PD control scheme.
  - *Feed-forward* control law

$$[\alpha] = [\widehat{\mathbf{M}(\mathbf{q}_d)}], \quad \beta = \mathbf{C}\widehat{(\mathbf{q}_d, \dot{\mathbf{q}}_d)} + \widehat{\mathbf{G}(\mathbf{q}_d)} + \mathbf{F}\widehat{(\mathbf{q}_d, \dot{\mathbf{q}}_d)}$$

  - Model terms computed according to *desired* trajectory and *not* in the feed-back loop.
  - Model terms can be computed off-line $\rightarrow$ Almost no issue of computation time.
- No "exact" cancellation in special cases $\rightarrow$ No decoupling or linearity.
- If estimates are good, then right-hand side is small! $\rightarrow$ performance better than PD.
- Borne out by simulations and experiments.

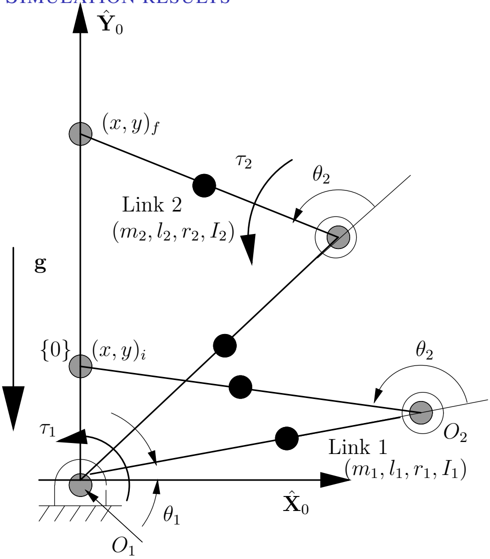
NON-LINEAR CONTROL OF MULTI-LINK SERIAL MANIPULATOR

CONTROL LAW PARTITIONING (CONTD.)

- Special cases of computed torque scheme
  - $[\alpha] = [U]$ and $\beta = \mathbf{G}(\mathbf{q}) \rightarrow$ Gravity compensation.
  - No model used $\rightarrow [\alpha] = [U]$ and $\beta = \mathbf{0} \rightarrow$ PD control scheme.
  - *Feed-forward* control law

$$[\alpha] = [\widehat{\mathbf{M}(\mathbf{q}_d)}], \quad \beta = \mathbf{C}\widehat{(\mathbf{q}_d, \dot{\mathbf{q}}_d)} + \widehat{\mathbf{G}(\mathbf{q}_d)} + \mathbf{F}\widehat{(\mathbf{q}_d, \dot{\mathbf{q}}_d)}$$

  - Model terms computed according to *desired* trajectory and *not* in the feed-back loop.
  - Model terms can be computed off-line $\rightarrow$ Almost no issue of computation time.
- No "exact" cancellation in special cases $\rightarrow$ No decoupling or linearity.
- If estimates are good, then right-hand side is small! $\rightarrow$ performance better than PD.
- Borne out by simulations and experiments.



- Planar 2R robot shown in 2 configurations.
- Link 1 parameters – $l_1 = 1m$, $r_1 = 0.773m$, $m_1 = 12.456kg$ and $I_1 = 1.042 \text{ kg} - \text{m}^2$.
- Link 2 parameters – $l_1 = 1m$, $r_1 = 0.583m$, $m_1 = 12.456kg$ and $I_1 = 1.042 \text{ kg} - \text{m}^2$.
- Payload at the end 2.5 kg.

Non-linear control of multi-link serial manipulator

- Tip moves up from $(0, 0.55m)$ to $(0, 1.45m)$ and back to $(0, 0.55m)$.
- Two cases: (a) *fast*: total time is 2 sec, (b) *slow*: total time is 2 min.
- Smooth Cartesian cubic trajectories generated.

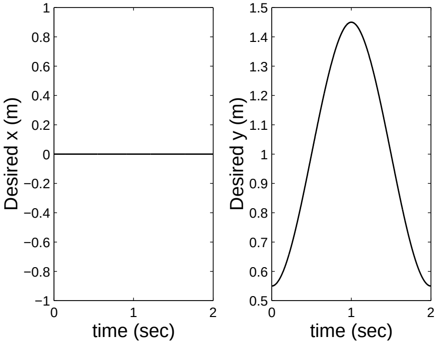


Figure 15: Desired Cartesian trajectory

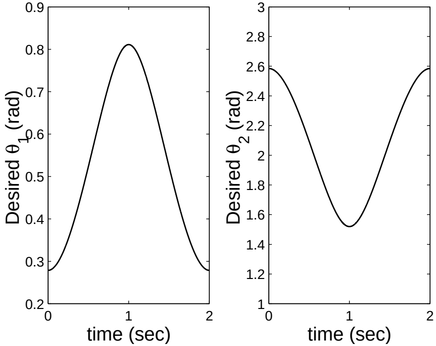


Figure 16: Desired $\theta_1(t)$ and $\theta_2(t)$

- Desired $\theta_{i_d}(t),\ i = 1,2$ and derivatives obtained using *inverse kinematics*
- Simulation results presented for
  - PD control scheme
  - Feed-forward controller with an *exact* knowledge of the model parameters,
  - Model-based controller with 10% error in m_i and 5% error in r_i
  - Cartesian control scheme (discussed later).
- Gain values K_{p_i}, K_{v_i} are chosen such that $\omega_1 = 85.0$, $\omega_2 = 75.0$, and ξ_i are 2.0 $\rightarrow$ system over-damped.

- Desired $\theta_{i\,d}(t),\ i = 1,2$ and derivatives obtained using *inverse kinematics*
- Simulation results presented for
  - PD control scheme
  - Feed-forward controller with an *exact* knowledge of the model parameters,
  - Model-based controller with 10% error in m_i and 5% error in r_i
  - Cartesian control scheme (discussed later).
- Gain values K_{p_i}, K_{v_i} are chosen such that $\omega_1 = 85.0$, $\omega_2 = 75.0$, and ξ_i are 2.0 $\rightarrow$ system over-damped.

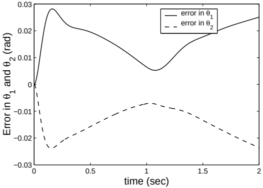
NON-LINEAR CONTROL OF MULTI-LINK SERIAL MANIPULATOR

SIMULATION RESULTS(CONTD.)

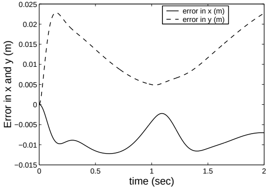
- Desired $\theta_{i\,d}(t),\ i = 1,2$ and derivatives obtained using *inverse kinematics*
- Simulation results presented for
  - PD control scheme
  - Feed-forward controller with an *exact* knowledge of the model parameters,
  - Model-based controller with 10% error in m_i and 5% error in r_i
  - Cartesian control scheme (discussed later).
- Gain values K_{p_i}, K_{v_i} are chosen such that $\omega_1 = 85.0$, $\omega_2 = 75.0$, and ξ_i are 2.0 $\rightarrow$ system over-damped.

Non-linear control of multi-link serial manipulator

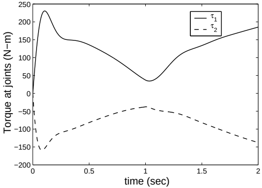
Simulation results – PD control



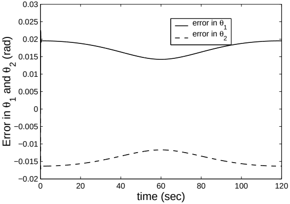
(a) *Error in θ_1, θ_2 for fast motion*



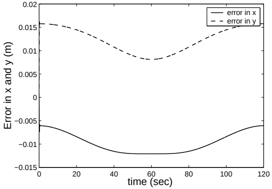
(b) *Error in x, y for fast motion*



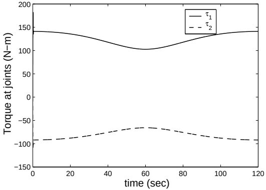
(c) *Torque at two joints for fast motion*



(d) *Error in θ_1, θ_2 for slow motion*



(e) *Error in x, y for slow motion*



(f) *Torque at two joints for slow motion*

Non-linear control of multi-link serial manipulator

Simulation results – PD control

- Maximum error in joint variables larger in case of *fast* motion.
  - Approximately 0.03 rad in fast versus 0.02 rad in slow motion.
  - Approximately 0.023 m in fast versus 0.016 m in slow motion.
- Fast motion $\rightarrow$ Non-linear inertia, centripetal/Coriolis terms larger
- Linear PD control less effective *as expected*!
- Maximum torque at the joints is larger – Approximately 225 N-m versus 145 N-m
- Torque larger in fast motion due to non-linear terms in equations of motion!
- Curves much smoother in slow motion.
- Non-linear controller results next!!

- Maximum error in joint variables larger in case of *fast* motion.
  - Approximately 0.03 rad in fast versus 0.02 rad in slow motion.
  - Approximately 0.023 m in fast versus 0.016 m in slow motion.
- Fast motion → Non-linear inertia, centripetal/Coriolis terms larger
- Linear PD control less effective *as expected*!
- Maximum torque at the joints is larger – Approximately 225 N-m versus 145 N-m
- Torque larger in fast motion due to non-linear terms in equations of motion!
- Curves much smoother in slow motion.
- Non-linear controller results next!!

Non-linear control of multi-link serial manipulator

Simulation results – PD control

- Maximum error in joint variables larger in case of *fast* motion.
  - Approximately 0.03 rad in fast versus 0.02 rad in slow motion.
  - Approximately 0.023 m in fast versus 0.016 m in slow motion.
- Fast motion → Non-linear inertia, centripetal/Coriolis terms larger
- Linear PD control less effective *as expected*!
- Maximum torque at the joints is larger – Approximately 225 N-m versus 145 N-m
- Torque larger in fast motion due to non-linear terms in equations of motion!
- Curves much smoother in slow motion.
- Non-linear controller results next!!

Non-linear control of multi-link serial manipulator

Simulation results – PD control

- Maximum error in joint variables larger in case of *fast* motion.
  - Approximately 0.03 rad in fast versus 0.02 rad in slow motion.
  - Approximately 0.023 m in fast versus 0.016 m in slow motion.
- Fast motion $\rightarrow$ Non-linear inertia, centripetal/Coriolis terms larger
- Linear PD control less effective *as expected*!
- Maximum torque at the joints is larger – Approximately 225 N-m versus 145 N-m
- Torque larger in fast motion due to non-linear terms in equations of motion!
- Curves much smoother in slow motion.
- Non-linear controller results next!!

Non-linear control of multi-link serial manipulator

Simulation results – PD control

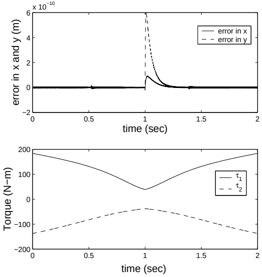
- Maximum error in joint variables larger in case of *fast* motion.
  - Approximately 0.03 rad in fast versus 0.02 rad in slow motion.
  - Approximately 0.023 m in fast versus 0.016 m in slow motion.
- Fast motion $\rightarrow$ Non-linear inertia, centripetal/Coriolis terms larger
- Linear PD control less effective *as expected*!
- Maximum torque at the joints is larger – Approximately 225 N-m versus 145 N-m
- Torque larger in fast motion due to non-linear terms in equations of motion!
- Curves much smoother in slow motion.
- Non-linear controller results next!!

- Maximum error in joint variables larger in case of *fast* motion.
  - Approximately 0.03 rad in fast versus 0.02 rad in slow motion.
  - Approximately 0.023 m in fast versus 0.016 m in slow motion.
- Fast motion $\rightarrow$ Non-linear inertia, centripetal/Coriolis terms larger
- Linear PD control less effective *as expected*!
- Maximum torque at the joints is larger – Approximately 225 N-m versus 145 N-m
- Torque larger in fast motion due to non-linear terms in equations of motion!
- Curves much smoother in slow motion.
- Non-linear controller results next!!

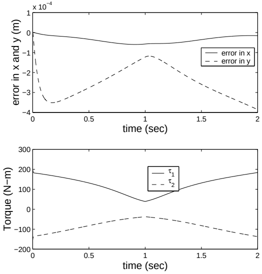
- Maximum error in joint variables larger in case of *fast* motion.
  - Approximately 0.03 rad in fast versus 0.02 rad in slow motion.
  - Approximately 0.023 m in fast versus 0.016 m in slow motion.
- Fast motion $\rightarrow$ Non-linear inertia, centripetal/Coriolis terms larger
- Linear PD control less effective *as expected*!
- Maximum torque at the joints is larger – Approximately 225 N-m versus 145 N-m
- Torque larger in fast motion due to non-linear terms in equations of motion!
- Curves much smoother in slow motion.
- Non-linear controller results next!!

NON-LINEAR CONTROL OF MULTI-LINK SERIAL MANIPULATOR

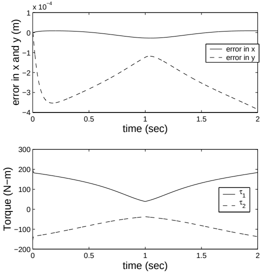
SIMULATION RESULTS – NON-LINEAR CONTROLLERS (FAST MOTIONS)



(g) Trajectory errors and torques for feed-forward controller



(h) Trajectory errors and torques for computed torque controller with uncertainties



(i) Trajectory errors and torques for Cartesian controller

- Feed-forward controller *without* model uncertainties is very accurate.
- Computed torque *with* 10% uncertainities more accurate than PD.
- Torque profiles are smoother – similar to PD control for *slow motion* $\rightarrow$ effect of non-linearities reduced!!

- The PD (PID) control scheme is not suitable for high-speed applications and the errors can be large. To reduce errors, we need to perform trial and error. The performance for slow-speed operation is better and one can get smooth torque profiles.
- Model-based schemes show improved performance in simulation. The torques are lower and the profile is also smoother. The lack of the knowledge of parameters degrades the performance only to a small extent.
- The computation times for the model-based control are larger, but can be easily handled by newer processors.

Non-linear control of multi-link serial manipulator

Experimental results

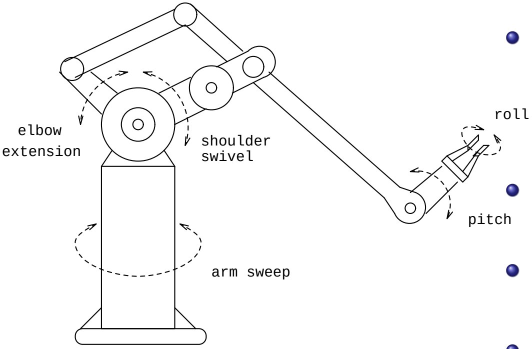


Figure 17: Schematic of a five-axis servo manipulator

- Five DOF pink-and-place robot, all DOF rotary, $\theta_i, \; i = 1, ..., 5$.
- A four-bar linkage drive joint 3 – Motors for joint 2 and 3 are on platform rotated by Motor 1 $\rightarrow$ Motor 2 "see" less inertia
- All motors are two-phase AC motors with large gear reduction.
- Significant backlash and friction in the gears.
- Encoders and tacho-generators measure joint rotation and velocity.

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Existing control law $V_i(t) = K_{p_i}(\theta_{i\,d} - \theta_i) - K_{v_i}\dot{\theta}_i, \quad i = 1,...,5$
- Voltage $V_i(t)$ applied at motor i.
- Subset of PD control law – available $\dot{\theta}_{i\,d}$ and $\ddot{\theta}_{i\,d}$ not used.
- Modify *existing* desired joint rotation to

$$\theta_{i\,d}^{*} = \theta_{i\,d} + \frac{1}{K_{p_i}}\ddot{\theta}_{i\,d} + \frac{K_{v_i}}{K_{p_i}}\dot{\theta}_{i\,d}, \quad i = 1,...,5$$

- Modified control law with $\theta_{i\,d}^{*} \rightarrow$ PD Control Law.

$$\begin{aligned} V_i(t) &= K_{p_i}(\theta_{i\,d}{}^{*} - \theta_i) - K_{v_i}\dot{\theta}_i \\ &= \ddot{\theta}_{i\,d} + K_{p_i}(\theta_{i\,d} - \theta_i) + K_{v_i}(\dot{\theta}_{i\,d} - \dot{\theta}_i), \; i = 1,...,5 \end{aligned}$$

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Existing control law $V_i(t) = K_{p_i}(\theta_{i\,d} - \theta_i) - K_{v_i}\dot{\theta}_i, \quad i = 1, ..., 5$
- Voltage $V_i(t)$ applied at motor i.
- Subset of PD control law – available $\dot{\theta}_{i\,d}$ and $\ddot{\theta}_{i\,d}$ not used.
- Modify *existing* desired joint rotation to

$$\theta_{i\,d}^{*} = \theta_{i\,d} + \frac{1}{K_{p_i}}\ddot{\theta}_{i\,d} + \frac{K_{v_i}}{K_{p_i}}\dot{\theta}_{i\,d}, \quad i = 1, ..., 5$$

- Modified control law with $\theta_{i\,d}^{*} \rightarrow$ PD Control Law.

$$\begin{aligned} V_i(t) &= K_{p_i}(\theta_{i\,d}{}^{*} - \theta_i) - K_{v_i}\dot{\theta}_i \\ &= \ddot{\theta}_{i\,d} + K_{p_i}(\theta_{i\,d} - \theta_i) + K_{v_i}(\dot{\theta}_{i\,d} - \dot{\theta}_i), \; i = 1, ..., 5 \end{aligned}$$

- Existing control law $V_i(t) = K_{p_i}(\theta_{id} - \theta_i) - K_{v_i}\dot{\theta}_i, \quad i = 1, ..., 5$
- Voltage $V_i(t)$ applied at motor i.
- Subset of PD control law – available $\dot{\theta}_{id}$ and $\ddot{\theta}_{id}$ not used.
- Modify *existing* desired joint rotation to

$$\theta_{id}^* = \theta_{id} + \frac{1}{K_{p_i}}\ddot{\theta}_{id} + \frac{K_{v_i}}{K_{p_i}}\dot{\theta}_{id}, \quad i = 1, ..., 5$$

- Modified control law with $\theta_{id}^* \rightarrow$ PD Control Law.

$$\begin{aligned} V_i(t) &= K_{p_i}(\theta_{id}{}^* - \theta_i) - K_{v_i}\dot{\theta}_i \\ &= \ddot{\theta}_{id} + K_{p_i}(\theta_{id} - \theta_i) + K_{v_i}(\dot{\theta}_{id} - \dot{\theta}_i), \; i = 1, ..., 5 \end{aligned}$$

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Existing control law $V_i(t) = K_{p_i}(\theta_{i\,d} - \theta_i) - K_{v_i}\dot{\theta}_i, \quad i = 1, ..., 5$
- Voltage $V_i(t)$ applied at motor i.
- Subset of PD control law – available $\dot{\theta}_{i\,d}$ and $\ddot{\theta}_{i\,d}$ not used.
- Modify *existing* desired joint rotation to

$$\theta_{i\,d}^{*} = \theta_{i\,d} + \frac{1}{K_{p_i}}\ddot{\theta}_{i\,d} + \frac{K_{v_i}}{K_{p_i}}\dot{\theta}_{i\,d}, \quad i = 1, ..., 5$$

- Modified control law with $\theta_{i\,d}^{*} \rightarrow$ PD Control Law.

$$\begin{aligned} V_i(t) &= K_{p_i}(\theta_{i\,d}{}^{*} - \theta_i) - K_{v_i}\dot{\theta}_i \\ &= \ddot{\theta}_{i\,d} + K_{p_i}(\theta_{i\,d} - \theta_i) + K_{v_i}(\dot{\theta}_{i\,d} - \dot{\theta}_i), \; i = 1, ..., 5 \end{aligned}$$

NON-LINEAR CONTROL OF MULTI-LINK SERIAL MANIPULATOR

EXPERIMENTAL RESULTS (CONTD.)

- Existing control law $V_i(t) = K_{p_i}(\theta_{i\,d} - \theta_i) - K_{v_i}\dot{\theta}_i, \quad i = 1, \ldots, 5$
- Voltage $V_i(t)$ applied at motor i.
- Subset of PD control law – available $\dot{\theta}_{i\,d}$ and $\ddot{\theta}_{i\,d}$ not used.
- Modify *existing* desired joint rotation to

$$\theta_{i\,d}^{*} = \theta_{i\,d} + \frac{1}{K_{p_i}}\ddot{\theta}_{i\,d} + \frac{K_{v_i}}{K_{p_i}}\dot{\theta}_{i\,d}, \quad i = 1, \ldots, 5$$

- Modified control law with $\theta_{i\,d}^{*} \rightarrow$ PD Control Law.

$$\begin{aligned} V_i(t) &= K_{p_i}(\theta_{i\,d}{}^{*} - \theta_i) - K_{v_i}\dot{\theta}_i \\ &= \ddot{\theta}_{i\,d} + K_{p_i}(\theta_{i\,d} - \theta_i) + K_{v_i}(\dot{\theta}_{i\,d} - \dot{\theta}_i), \; i = 1, \ldots, 5 \end{aligned}$$

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Similar idea used to modify existing controller to a *model-based* control scheme – Modify desired θ_{id} with

$$\theta_{id}^{*} = \frac{V_{i_{mdl}}}{K_{p_i}} + \theta_{id} + \frac{1}{K_{p_i}}\ddot{\theta}_{id} + \frac{K_{v_i}}{K_{p_i}}\dot{\theta}_{id}, \quad i = 1, ..., 5$$

where $V_{i_{mdl}}$, corresponding to $\tau_{i_{mdl}}$ computed from

$$\tau_{mdl} = [\mathbf{M}(\theta_d)]\ddot{\theta}_d + \mathbf{C}(\theta_d, \dot{\theta}_d) + \mathbf{G}(\theta_d)$$

with available motor characteristics chart.

- Above control law is analogous to *feed-forward* law

$$\tau = \tau_{model} + \ddot{\theta}_d + [K_p](\theta_d - \theta) + [K_v](\dot{\theta}_d - \dot{\theta})$$

- Model parameters required for θ_{id}^{*} from CAD model of robot.
- Computed θ_{id}^{*} *instead* of θ_{id} used as reference input.
- Above approach *does not* change any electronics or hardware!

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Similar idea used to modify existing controller to a *model-based* control scheme – Modify desired $\theta_{i\,d}$ with

$$\theta_{i\,d}^{*} = \frac{V_{i_{mdl}}}{K_{p_i}} + \theta_{i\,d} + \frac{1}{K_{p_i}}\ddot{\theta}_{i\,d} + \frac{K_{v_i}}{K_{p_i}}\dot{\theta}_{i\,d}, \quad i = 1, \ldots, 5$$

  where $V_{i_{mdl}}$, corresponding to $\tau_{i_{mdl}}$ computed from

$$\tau_{mdl} = [\mathbf{M}(\theta_d)]\ddot{\theta}_d + \mathbf{C}(\theta_d, \dot{\theta}_d) + \mathbf{G}(\theta_d)$$

  with available motor characteristics chart.
- Above control law is analogous to *feed-forward* law

$$\tau = \tau_{model} + \ddot{\theta}_d + [K_p](\theta_d - \theta) + [K_v](\dot{\theta}_d - \dot{\theta})$$

- Model parameters required for $\theta_{i\,d}^{*}$ from CAD model of robot.
- Computed $\theta_{i\,d}^{*}$ *instead* of $\theta_{i\,d}$ used as reference input.
- Above approach *does not* change any electronics or hardware!

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Similar idea used to modify existing controller to a *model-based* control scheme – Modify desired $\theta_{i\,d}$ with

$$\theta^*_{i\,d} = \frac{V_{i_{mdl}}}{K_{p_i}} + \theta_{i\,d} + \frac{1}{K_{p_i}}\ddot{\theta}_{i\,d} + \frac{K_{v_i}}{K_{p_i}}\dot{\theta}_{i\,d}, \quad i = 1, ..., 5$$

  where $V_{i_{mdl}}$, corresponding to $\tau_{i_{mdl}}$ computed from

$$\tau_{mdl} = [\mathbf{M}(\theta_d)]\ddot{\theta}_d + \mathbf{C}(\theta_d, \dot{\theta}_d) + \mathbf{G}(\theta_d)$$

  with available motor characteristics chart.
- Above control law is analogous to *feed-forward* law

$$\tau = \tau_{model} + \ddot{\theta}_d + [K_p](\theta_d - \theta) + [K_v](\dot{\theta}_d - \dot{\theta})$$

- Model parameters required for $\theta^*_{i\,d}$ from CAD model of robot.
- Computed $\theta^*_{i\,d}$ *instead* of $\theta_{i\,d}$ used as reference input.
- Above approach *does not* change any electronics or hardware!

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Similar idea used to modify existing controller to a *model-based* control scheme – Modify desired $\theta_{i\,d}$ with

$$\theta_{i\,d}^{*} = \frac{V_{i_{mdl}}}{K_{p_i}} + \theta_{i\,d} + \frac{1}{K_{p_i}}\ddot{\theta}_{i\,d} + \frac{K_{v_i}}{K_{p_i}}\dot{\theta}_{i\,d}, \quad i = 1, \ldots, 5$$

  where $V_{i_{mdl}}$, corresponding to $\tau_{i_{mdl}}$ computed from

$$\tau_{mdl} = [\mathbf{M}(\theta_d)]\ddot{\theta}_d + \mathbf{C}(\theta_d, \dot{\theta}_d) + \mathbf{G}(\theta_d)$$

  with available motor characteristics chart.
- Above control law is analogous to *feed-forward* law

$$\tau = \tau_{model} + \ddot{\theta}_d + [K_p](\theta_d - \theta) + [K_v](\dot{\theta}_d - \dot{\theta})$$

- Model parameters required for $\theta_{i\,d}^{*}$ from CAD model of robot.
- Computed $\theta_{i\,d}^{*}$ *instead* of $\theta_{i\,d}$ used as reference input.
- Above approach *does not* change any electronics or hardware!

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Similar idea used to modify existing controller to a *model-based* control scheme – Modify desired $\theta_{i\,d}$ with

$$\theta_{i\,d}^{*} = \frac{V_{i_{mdl}}}{K_{p_i}} + \theta_{i\,d} + \frac{1}{K_{p_i}}\ddot{\theta}_{i\,d} + \frac{K_{v_i}}{K_{p_i}}\dot{\theta}_{i\,d}, \quad i = 1, \ldots, 5$$

where $V_{i_{mdl}}$, corresponding to $\tau_{i_{mdl}}$ computed from

$$\tau_{mdl} = [\mathbf{M}(\theta_d)]\ddot{\theta}_d + \mathbf{C}(\theta_d, \dot{\theta}_d) + \mathbf{G}(\theta_d)$$

with available motor characteristics chart.

- Above control law is analogous to *feed-forward* law

$$\tau = \tau_{model} + \ddot{\theta}_d + [K_p](\theta_d - \theta) + [K_v](\dot{\theta}_d - \dot{\theta})$$

- Model parameters required for $\theta_{i\,d}^{*}$ from CAD model of robot.
- Computed $\theta_{i\,d}^{*}$ *instead* of $\theta_{i\,d}$ used as reference input.
- Above approach *does not* change any electronics or hardware!

- Desired trajectory – traverse from (0°, 0°,-90°, 180°, 0°) to (30°, 40°,-60°, 180°, 0°) and back
- Total time 4 seconds – going 2 seconds and coming back 2 seconds.
- Initial 2 seconds against gravity and final two seconds aided by gravity.
- Smooth cubic trajectories generated (see Lecture 1) with zero initial and final velocity.
- Sampling time is 5 ms or a set-points generated at a frequency of 200 Hz.
- Trajectory faster than typical usage for the robot.

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Desired trajectory – traverse from (0°, 0°,-90°, 180°, 0°) to (30°, 40°,-60°, 180°, 0°) and back
- Total time 4 seconds – going 2 seconds and coming back 2 seconds.
- Initial 2 seconds against gravity and final two seconds aided by gravity.
- Smooth cubic trajectories generated (see Lecture 1) with zero initial and final velocity.
- Sampling time is 5 ms or a set-points generated at a frequency of 200 Hz.
- Trajectory faster than typical usage for the robot.

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Desired trajectory – traverse from (0°, 0°,-90°, 180°, 0°) to (30°, 40°,-60°, 180°, 0°) and back
- Total time 4 seconds – going 2 seconds and coming back 2 seconds.
- Initial 2 seconds against gravity and final two seconds aided by gravity.
- Smooth cubic trajectories generated (see Lecture 1) with zero initial and final velocity.
- Sampling time is 5 ms or a set-points generated at a frequency of 200 Hz.
- Trajectory faster than typical usage for the robot.

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Desired trajectory – traverse from (0°, 0°,-90°, 180°, 0°) to (30°, 40°,-60°, 180°, 0°) and back
- Total time 4 seconds – going 2 seconds and coming back 2 seconds.
- Initial 2 seconds against gravity and final two seconds aided by gravity.
- Smooth cubic trajectories generated (see Lecture 1) with zero initial and final velocity.
- Sampling time is 5 ms or a set-points generated at a frequency of 200 Hz.
- Trajectory faster than typical usage for the robot.

NON-LINEAR CONTROL OF MULTI-LINK SERIAL MANIPULATOR

EXPERIMENTAL RESULTS (CONTD.)

- Desired trajectory – traverse from (0°, 0°,-90°, 180°, 0°) to (30°, 40°,-60°, 180°, 0°) and back
- Total time 4 seconds – going 2 seconds and coming back 2 seconds.
- Initial 2 seconds against gravity and final two seconds aided by gravity.
- Smooth cubic trajectories generated (see Lecture 1) with zero initial and final velocity.
- Sampling time is 5 ms or a set-points generated at a frequency of 200 Hz.
- Trajectory faster than typical usage for the robot.

EXPERIMENTAL RESULTS (CONTD.)

- Desired trajectory – traverse from (0°, 0°,-90°, 180°, 0°) to (30°, 40°,-60°, 180°, 0°) and back
- Total time 4 seconds – going 2 seconds and coming back 2 seconds.
- Initial 2 seconds against gravity and final two seconds aided by gravity.
- Smooth cubic trajectories generated (see Lecture 1) with zero initial and final velocity.
- Sampling time is 5 ms or a set-points generated at a frequency of 200 Hz.
- Trajectory faster than typical usage for the robot.

Non-linear control of multi-link serial manipulator

Experimental results (Contd.)

- Solid line is θ_{1_d}, Dotted line is θ_1 using PD control.
- Dashed line is achieved trajectory of joint 1 using model-based control.

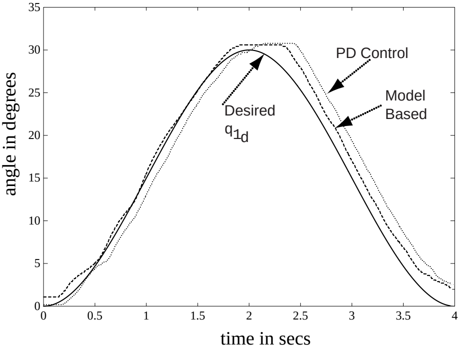


Figure 18: Controller performance in following the desired trajectory of joint 1

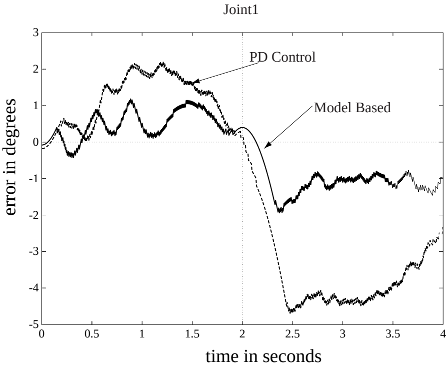


Figure 19: Comparison of errors at joint 1

Non-linear control of multi-link serial manipulator

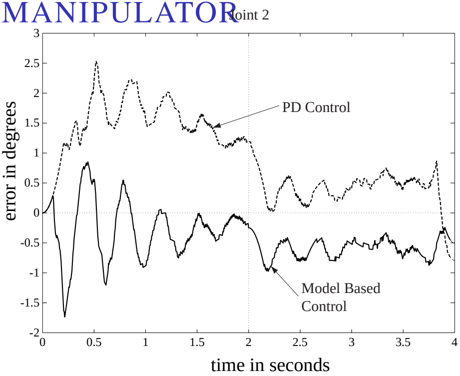


Figure 20: Comparison of errors at joint 2

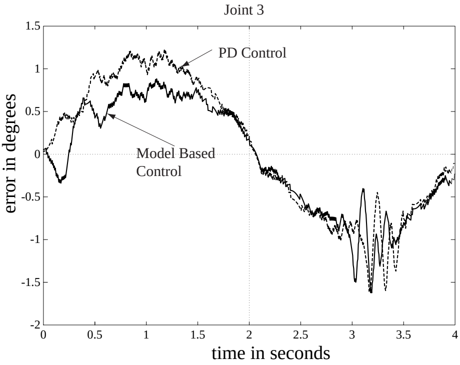


Figure 21: Comparison of errors at joint 3

- Maximum θ_1 error reduce from -5° to 2°.
- θ_2 error also reduces for model-based, not much difference in θ_3.
- In joint 4 and 5 (not shown), there is almost no difference!
- Joints 4 and 5 "see" less inertial, centripetal/Coriolis effects!

Outline

- Till now – control of serial manipulator *without* any constraint on joint trajectory $\mathbf{q}(t)$.
- End-effector of a serial manipulator tracing a desired path *while maintaining* contact with a surface.
- Parallel manipulators – passive and active variables related by loop-closure equations.
- Joint space and Cartesian space approaches.
- Leads to *force* and *hybrid* position/force control – End-effector of a serial manipulator tracing a path on a surface and *applying* a force.

CONTROL OF CONSTRAINED AND PARALLEL MANIPULATOR

OVERVIEW

- Till now – control of serial manipulator *without* any constraint on joint trajectory $\mathbf{q}(t)$.
- End-effector of a serial manipulator tracing a desired path *while maintaining* contact with a surface.
- Parallel manipulators – passive and active variables related by loop-closure equations.
- Joint space and Cartesian space approaches.
- Leads to *force* and *hybrid* position/force control – End-effector of a serial manipulator tracing a path on a surface and *applying* a force.

- Till now – control of serial manipulator *without* any constraint on joint trajectory $\mathbf{q}(t)$.
- End-effector of a serial manipulator tracing a desired path *while maintaining* contact with a surface.
- Parallel manipulators – passive and active variables related by loop-closure equations.
- Joint space and Cartesian space approaches.
- Leads to *force* and *hybrid* position/force control – End-effector of a serial manipulator tracing a path on a surface and *applying* a force.

- Till now – control of serial manipulator *without* any constraint on joint trajectory $\mathbf{q}(t)$.
- End-effector of a serial manipulator tracing a desired path *while maintaining* contact with a surface.
- Parallel manipulators – passive and active variables related by loop-closure equations.
- Joint space and Cartesian space approaches.
- Leads to *force* and *hybrid* position/force control – End-effector of a serial manipulator tracing a path on a surface and *applying* a force.

Control of constrained and parallel manipulator

Overview

- Till now – control of serial manipulator *without* any constraint on joint trajectory $\mathbf{q}(t)$.
- End-effector of a serial manipulator tracing a desired path *while maintaining* contact with a surface.
- Parallel manipulators – passive and active variables related by loop-closure equations.
- Joint space and Cartesian space approaches.
- Leads to *force* and *hybrid* position/force control – End-effector of a serial manipulator tracing a path on a surface and *applying* a force.

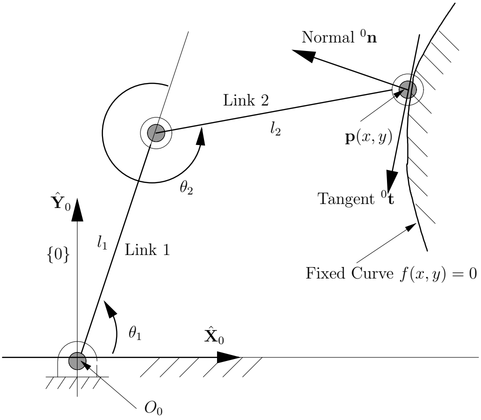


Figure 22: Constrained motion of a 2R planar manipulator

- Tip of planar 2R manipulator to keep in contact with the curve $f(x,y)=0$.
- In joint space

$$F(\theta_1,\theta_2) = f(l_1c_1 + l_2c_{12}, l_1s_1 + l_2s_{12}) = 0$$

Since $x = l_1c_1 + l_2c_{12}$ and $y = l_1s_1 + l_2s_{12}$

- See direct kinematic equations for the planar 2R manipulator (See Module 3, Lecture 1).

Control of constrained and parallel manipulator

Analysis of constrained motion of planar 2R manipulator

- From $f(x,y)=0$ obtain $x=f_1(\phi)$ and $y=f_2(\phi)$ → parametric equation of the curve $f(x,y)=0$ in terms of parameter ϕ.
- Obtain from the parametric form

$$\theta_1 = h_1(\phi), \quad \theta_2 = h_2(\phi), \quad \text{or } \Theta = \mathbf{h}(\phi),\ \Theta = (\theta_1, \theta_2)^T$$

- *Inverse* kinematics of the planar 2R manipulator[4].
- If $f(x,y)=0$ is a simple curve such as circle, then possible to use *direct kinematics* of a parallel manipulator/mechanism.
- For a circle centered at $(l_0, 0)$ and radius l_3, parametric equations (from the equations of a four-bar) are

$$x = l_1 c_1 + l_2 c_{12} = l_0 + l_3 \cos\phi,\ \ y = l_1 s_1 + l_2 s_{12} = l_3 \sin\phi$$

[4] For other manipulators, it may not be easy to obtain analytical expressions.

Control of constrained and parallel manipulator

Analysis of constrained motion of planar 2R manipulator

- From $f(x,y)=0$ obtain $x=f_1(\phi)$ and $y=f_2(\phi)$ $\rightarrow$ parametric equation of the curve $f(x,y)=0$ in terms of parameter ϕ.
- Obtain from the parametric form

$$\theta_1 = h_1(\phi), \quad \theta_2 = h_2(\phi), \quad \text{or } \Theta = \mathbf{h}(\phi),\ \Theta = (\theta_1, \theta_2)^T$$

- *Inverse* kinematics of the planar 2R manipulator[4].
- If $f(x,y)=0$ is a simple curve such as circle, then possible to use *direct kinematics* of a parallel manipulator/mechanism.
- For a circle centered at $(l_0, 0)$ and radius l_3, parametric equations (from the equations of a four-bar) are

$$x = l_1 c_1 + l_2 c_{12} = l_0 + l_3 \cos\phi, \ \ y = l_1 s_1 + l_2 s_{12} = l_3 \sin\phi$$

[4]For other manipulators, it may not be easy to obtain analytical expressions.

- From $f(x,y)=0$ obtain $x=f_1(\phi)$ and $y=f_2(\phi)$ $\rightarrow$ parametric equation of the curve $f(x,y)=0$ in terms of parameter ϕ.
- Obtain from the parametric form

$$\theta_1 = h_1(\phi), \quad \theta_2 = h_2(\phi), \quad \text{or } \Theta = \mathbf{h}(\phi),\ \Theta = (\theta_1, \theta_2)^T$$

- *Inverse* kinematics of the planar 2R manipulator[4].
- If $f(x,y)=0$ is a simple curve such as circle, then possible to use *direct kinematics* of a parallel manipulator/mechanism.
- For a circle centered at $(l_0, 0)$ and radius l_3, parametric equations (from the equations of a four-bar) are

$$x = l_1 c_1 + l_2 c_{12} = l_0 + l_3 \cos\phi,\ \ y = l_1 s_1 + l_2 s_{12} = l_3 \sin\phi$$

[4] For other manipulators, it may not be easy to obtain analytical expressions.

- From $f(x,y)=0$ obtain $x=f_1(\phi)$ and $y=f_2(\phi)$ $\rightarrow$ parametric equation of the curve $f(x,y)=0$ in terms of parameter ϕ.
- Obtain from the parametric form

$$\theta_1 = h_1(\phi), \quad \theta_2 = h_2(\phi), \quad \text{or } \Theta = \mathbf{h}(\phi),\ \Theta = (\theta_1, \theta_2)^T$$

- *Inverse* kinematics of the planar 2R manipulator[4].
- If $f(x,y)=0$ is a simple curve such as circle, then possible to use *direct kinematics* of a parallel manipulator/mechanism.
- For a circle centered at $(l_0, 0)$ and radius l_3, parametric equations (from the equations of a four-bar) are

$$x = l_1 c_1 + l_2 c_{12} = l_0 + l_3 \cos\phi,\ \ y = l_1 s_1 + l_2 s_{12} = l_3 \sin\phi$$

[4] For other manipulators, it may not be easy to obtain analytical expressions.

- From $f(x,y)=0$ obtain $x=f_1(\phi)$ and $y=f_2(\phi) \rightarrow$ parametric equation of the curve $f(x,y)=0$ in terms of parameter ϕ.
- Obtain from the parametric form

$$\theta_1 = h_1(\phi), \quad \theta_2 = h_2(\phi), \quad \text{or } \Theta = \mathbf{h}(\phi),\ \Theta = (\theta_1, \theta_2)^T$$

- *Inverse* kinematics of the planar 2R manipulator[4].
- If $f(x,y)=0$ is a simple curve such as circle, then possible to use *direct kinematics* of a parallel manipulator/mechanism.
- For a circle centered at $(l_0, 0)$ and radius l_3, parametric equations (from the equations of a four-bar) are

$$x = l_1 c_1 + l_2 c_{12} = l_0 + l_3 \cos\phi,\ \ y = l_1 s_1 + l_2 s_{12} = l_3 \sin\phi$$

[4] For other manipulators, it may not be easy to obtain analytical expressions.

- From $\theta_1 = h_1(\phi)$, $\theta_2 = h_2(\phi)$, obtain

$$\dot{\theta}_i = \frac{\partial h_i}{\partial \phi}\dot{\phi}, \quad i = 1,2$$
$$\ddot{\theta}_i = \frac{\partial h_i}{\partial \phi}\ddot{\phi} + (\frac{\partial^2 h_i}{\partial \phi^2}\dot{\phi})\dot{\phi} \quad i = 1,2$$

- Substitute θ_i, $\dot{\theta}_i$ and $\ddot{\theta}_i$ $(i = 1,2)$ in the equations of motion of a planar 2R manipulator (see Module 6, Lecture 2) to get

$$[\mathbf{M}(\Theta)][J_{\mathbf{h}}]\ddot{\phi} + (\mathbf{C}(\Theta,\dot{\Theta}) + [\mathbf{M}(\Theta)][\dot{J}_{\mathbf{h}}]\dot{\phi}) + \mathbf{G}(\Theta) = \tau$$

$[J_{\mathbf{h}}]$ denotes the Jacobian of the transformation $\Theta = \mathbf{h}(\phi)$ and $[\dot{J}_{\mathbf{h}}]$ is its time derivative.

- From $\theta_1 = h_1(\phi), \quad \theta_2 = h_2(\phi)$, obtain

$$
\begin{aligned}
\dot{\theta}_i &= \frac{\partial h_i}{\partial \phi}\dot{\phi}, \quad i = 1,2 \\
\ddot{\theta}_i &= \frac{\partial h_i}{\partial \phi}\ddot{\phi} + (\frac{\partial^2 h_i}{\partial \phi^2}\dot{\phi})\dot{\phi} \quad i = 1,2
\end{aligned}
$$

- Substitute θ_i, $\dot{\theta}_i$ and $\ddot{\theta}_i$ $(i = 1,2)$ in the equations of motion of a planar 2R manipulator (see Module 6, Lecture 2) to get

$$[\mathbf{M}(\Theta)][J_{\mathbf{h}}]\ddot{\phi} + (\mathbf{C}(\Theta,\dot{\Theta}) + [\mathbf{M}(\Theta)][\dot{J}_{\mathbf{h}}]\dot{\phi}) + \mathbf{G}(\Theta) = \tau$$

$[J_{\mathbf{h}}]$ denotes the Jacobian of the transformation $\Theta = \mathbf{h}(\phi)$ and $[\dot{J}_{\mathbf{h}}]$ is its time derivative.

- Pre-multiply the left- and the right-hand side by $[J_{\mathbf{h}}]^T$ to get

$$\bar{M}(\phi)\ddot{\phi} + \bar{C}(\phi,\dot{\phi}) + \bar{G}(\phi) = [J_{\mathbf{h}}]^T \tau$$

where

$$\begin{aligned} \bar{M}(\phi) &= [J_{\mathbf{h}}]^T [\mathbf{M}(\mathbf{h}(\phi))][J_{\mathbf{h}}] \\ \bar{C}(\phi,\dot{\phi}) &= \mathbf{C}(\mathbf{h}(\phi),[J_{\mathbf{h}}]\dot{\phi}) + [\mathbf{M}(\mathbf{h}(\phi))][\dot{J}_{\mathbf{h}}]\dot{\phi} \\ \bar{G}(\phi) &= \mathbf{G}(\mathbf{h}(\phi)) \end{aligned}$$

- Above represents a *unconstrained* one DOF system which *satisfies* $f(x,y) = 0$.
- The single ODE can be used to “design” model-based control schemes.

- Pre-multiply the left- and the right-hand side by $[J_{\mathbf{h}}]^T$ to get

$$\bar{M}(\phi)\ddot{\phi} + \bar{C}(\phi,\dot{\phi}) + \bar{G}(\phi) = [J_{\mathbf{h}}]^T \tau$$

where

$$\begin{aligned}
\bar{M}(\phi) &= [J_{\mathbf{h}}]^T [\mathbf{M}(\mathbf{h}(\phi))][J_{\mathbf{h}}] \\
\bar{C}(\phi,\dot{\phi}) &= \mathbf{C}(\mathbf{h}(\phi),[J_{\mathbf{h}}]\dot{\phi}) + [\mathbf{M}(\mathbf{h}(\phi))][\dot{J}_{\mathbf{h}}]\dot{\phi} \\
\bar{G}(\phi) &= \mathbf{G}(\mathbf{h}(\phi))
\end{aligned}$$

- Above represents a *unconstrained* one DOF system which *satisfies* $f(x,y)=0$.
- The single ODE can be used to "design" model-based control schemes.

- Pre-multiply the left- and the right-hand side by $[J_{\mathbf{h}}]^T$ to get

$$\bar{M}(\phi)\ddot{\phi} + \bar{C}(\phi, \dot{\phi}) + \bar{G}(\phi) = [J_{\mathbf{h}}]^T \tau$$

where

$$\begin{aligned} \bar{M}(\phi) &= [J_{\mathbf{h}}]^T [\mathbf{M}(\mathbf{h}(\phi))][J_{\mathbf{h}}] \\ \bar{C}(\phi, \dot{\phi}) &= \mathbf{C}(\mathbf{h}(\phi), [J_{\mathbf{h}}]\dot{\phi}) + [\mathbf{M}(\mathbf{h}(\phi))][\dot{J}_{\mathbf{h}}]\dot{\phi} \\ \bar{G}(\phi) &= \mathbf{G}(\mathbf{h}(\phi)) \end{aligned}$$

- Above represents a *unconstrained* one DOF system which *satisfies* $f(x, y) = 0$.
- The single ODE can be used to "design" model-based control schemes.

Analysis of constrained motion of planar 2R manipulator

- $[J_{\mathbf{h}}]^T$ removes all information about the force normal to curve $\rightarrow$ Single ODE not useful to "design" control scheme for applying force.
- The normal is along gradient $\nabla f(x,y)$.
- Force normal to $f(x,y)=0$ is of the form $\tau_{\mathbf{n}} = \lambda \nabla F(\theta_1, \theta_2)$ where $\lambda(t)$ is the desired force.
- $\tau_{\mathbf{n}}$ does not do any work while tracing $f(x,y)=0$

$$\begin{aligned}\tau_{\mathbf{n}} \cdot \dot{\Theta} &= \lambda\left(\frac{\partial F(\theta_1,\theta_2)}{\partial \theta_1}\dot{\theta}_1 + \frac{\partial F(\theta_1,\theta_2)}{\partial \theta_2}\dot{\theta}_2\right) \\ &= \lambda \frac{d}{dt}(F(\theta_1,\theta_2)) = 0\end{aligned}$$

- Combined joint torque

$$\tau = \lambda(t)\nabla F(\theta_1,\theta_2) + \tau_\phi$$

τ_ϕ can be utilised to trace a desired path without violating the constraint $f(x,y)=0$.

- $[J_{\mathbf{h}}]^T$ removes all information about the force normal to curve $\rightarrow$ Single ODE not useful to "design" control scheme for applying force.
- The normal is along gradient $\nabla f(x, y)$.
- Force normal to $f(x, y) = 0$ is of the form $\tau_{\mathbf{n}} = \lambda \nabla F(\theta_1, \theta_2)$ where $\lambda(t)$ is the desired force.
- $\tau_{\mathbf{n}}$ does not do any work while tracing $f(x, y) = 0$

$$\begin{aligned} \tau_{\mathbf{n}} \cdot \dot{\Theta} &= \lambda\left(\frac{\partial F(\theta_1, \theta_2)}{\partial \theta_1}\dot{\theta}_1 + \frac{\partial F(\theta_1, \theta_2)}{\partial \theta_2}\dot{\theta}_2\right) \\ &= \lambda \frac{d}{dt}(F(\theta_1, \theta_2)) = 0 \end{aligned}$$

- Combined joint torque

$$\tau = \lambda(t)\nabla F(\theta_1, \theta_2) + \tau_\phi$$

τ_ϕ can be utilised to trace a desired path without violating the constraint $f(x, y) = 0$.

- $[J_{\mathbf{h}}]^T$ removes all information about the force normal to curve $\rightarrow$ Single ODE not useful to "design" control scheme for applying force.
- The normal is along gradient $\nabla f(x,y)$.
- Force normal to $f(x,y)=0$ is of the form $\tau_{\mathbf{n}} = \lambda \nabla F(\theta_1, \theta_2)$ where $\lambda(t)$ is the desired force.
- $\tau_{\mathbf{n}}$ does not do any work while tracing $f(x,y)=0$

$$\begin{aligned}\tau_{\mathbf{n}} \cdot \dot{\Theta} &= \lambda\left(\frac{\partial F(\theta_1,\theta_2)}{\partial \theta_1}\dot{\theta}_1 + \frac{\partial F(\theta_1,\theta_2)}{\partial \theta_2}\dot{\theta}_2\right) \\ &= \lambda \frac{d}{dt}(F(\theta_1,\theta_2)) = 0\end{aligned}$$

- Combined joint torque

$$\tau = \lambda(t)\nabla F(\theta_1,\theta_2) + \tau_\phi$$

τ_ϕ can be utilised to trace a desired path without violating the constraint $f(x,y)=0$.

- $[J_{\mathbf{h}}]^T$ removes all information about the force normal to curve $\rightarrow$ Single ODE not useful to "design" control scheme for applying force.
- The normal is along gradient $\nabla f(x,y)$.
- Force normal to $f(x,y)=0$ is of the form $\tau_{\mathbf{n}} = \lambda \nabla F(\theta_1, \theta_2)$ where $\lambda(t)$ is the desired force.
- $\tau_{\mathbf{n}}$ does not do any work while tracing $f(x,y)=0$

$$\begin{aligned} \tau_{\mathbf{n}} \cdot \dot{\Theta} &= \lambda\left(\frac{\partial F(\theta_1,\theta_2)}{\partial \theta_1}\dot{\theta}_1 + \frac{\partial F(\theta_1,\theta_2)}{\partial \theta_2}\dot{\theta}_2\right) \\ &= \lambda \frac{d}{dt}(F(\theta_1,\theta_2)) = 0 \end{aligned}$$

- Combined joint torque

$$\tau = \lambda(t)\nabla F(\theta_1,\theta_2) + \tau_\phi$$

τ_ϕ can be utilised to trace a desired path without violating the constraint $f(x,y)=0$.

- $[J_{\mathbf{h}}]^T$ removes all information about the force normal to curve $\rightarrow$ Single ODE not useful to "design" control scheme for applying force.
- The normal is along gradient $\nabla f(x,y)$.
- Force normal to $f(x,y)=0$ is of the form $\tau_{\mathbf{n}} = \lambda \nabla F(\theta_1, \theta_2)$ where $\lambda(t)$ is the desired force.
- $\tau_{\mathbf{n}}$ does not do any work while tracing $f(x,y)=0$

$$\begin{aligned} \tau_{\mathbf{n}} \cdot \dot{\Theta} &= \lambda\left(\frac{\partial F(\theta_1,\theta_2)}{\partial \theta_1}\dot{\theta}_1 + \frac{\partial F(\theta_1,\theta_2)}{\partial \theta_2}\dot{\theta}_2\right) \\ &= \lambda \frac{d}{dt}(F(\theta_1,\theta_2)) = 0 \end{aligned}$$

- Combined joint torque

$$\tau = \lambda(t)\nabla F(\theta_1,\theta_2) + \tau_\phi$$

τ_ϕ can be utilised to trace a desired path without violating the constraint $f(x,y)=0$.

- Using concept of computed torque control

$$\tau_\phi = [\alpha]_\phi \tau'_\phi + \beta_\phi$$

with

$$\begin{aligned}
[\alpha]_\phi &= [\mathbf{M}(\Theta)][J_\mathbf{h}] \\
\beta_\phi &= (\mathbf{C}(\Theta,\dot{\Theta}) + [\mathbf{M}(\Theta)][\dot{J}_\mathbf{h}]\dot{\phi}) + \mathbf{G}(\Theta) \\
\tau'_\phi &= \ddot{\phi}_d + K_v(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi)
\end{aligned}$$

- Choose controller gains K_p and K_v to meet performance requirement.
- Manipulator alway keeps in contact with $f(x,y) = 0$.
- The terms $\lambda(t)\nabla F(\theta_1,\theta_2)$ and τ_ϕ *do not* affect each other!
- Fairly complicated – not practical for 6 DOF manipulator → Cartesian control schemes much better!

- Using concept of computed torque control

$$\tau_\phi = [\alpha]_\phi \tau'_\phi + \beta_\phi$$

with

$$\begin{aligned}
[\alpha]_\phi &= [\mathbf{M}(\Theta)][J_{\mathbf{h}}] \\
\beta_\phi &= (\mathbf{C}(\Theta, \dot{\Theta}) + [\mathbf{M}(\Theta)][\dot{J}_{\mathbf{h}}]\dot{\phi}) + \mathbf{G}(\Theta) \\
\tau'_\phi &= \ddot{\phi}_d + K_v(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi)
\end{aligned}$$

- Choose controller gains K_p and K_v to meet performance requirement.
- Manipulator alway keeps in contact with $f(x, y) = 0$.
- The terms $\lambda(t)\nabla F(\theta_1, \theta_2)$ and τ_ϕ *do not* affect each other!
- Fairly complicated – not practical for 6 DOF manipulator $\rightarrow$ Cartesian control schemes much better!

Control of constrained and parallel manipulator

Analysis of constrained motion of planar 2R manipulator

- Using concept of computed torque control

$$\tau_\phi = [\alpha]_\phi \tau'_\phi + \beta_\phi$$

with

$$\begin{aligned}
[\alpha]_\phi &= [\mathbf{M}(\Theta)][J_\mathbf{h}] \\
\beta_\phi &= (\mathbf{C}(\Theta, \dot{\Theta}) + [\mathbf{M}(\Theta)][\dot{J}_\mathbf{h}]\dot{\phi}) + \mathbf{G}(\Theta) \\
\tau'_\phi &= \ddot{\phi}_d + K_v(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi)
\end{aligned}$$

- Choose controller gains K_p and K_v to meet performance requirement.
- Manipulator alway keeps in contact with $f(x, y) = 0$.
- The terms $\lambda(t)\nabla F(\theta_1, \theta_2)$ and τ_ϕ *do not* affect each other!
- Fairly complicated – not practical for 6 DOF manipulator → Cartesian control schemes much better!

CONTROL OF CONSTRAINED AND PARALLEL MANIPULATOR

ANALYSIS OF CONSTRAINED MOTION OF PLANAR 2R MANIPULATOR

- Using concept of computed torque control

$$\tau_\phi = [\alpha]_\phi \tau'_\phi + \beta_\phi$$

with

$$\begin{aligned} [\alpha]_\phi &= [\mathbf{M}(\Theta)][J_\mathbf{h}] \\ \beta_\phi &= (\mathbf{C}(\Theta, \dot{\Theta}) + [\mathbf{M}(\Theta)][\dot{J}_\mathbf{h}]\dot{\phi}) + \mathbf{G}(\Theta) \\ \tau'_\phi &= \ddot{\phi}_d + K_v(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi) \end{aligned}$$

- Choose controller gains K_p and K_v to meet performance requirement.
- Manipulator alway keeps in contact with $f(x,y) = 0$.
- The terms $\lambda(t)\nabla F(\theta_1, \theta_2)$ and τ_ϕ *do not* affect each other!
- Fairly complicated – not practical for 6 DOF manipulator → Cartesian control schemes much better!

Control of constrained and parallel manipulator

Analysis of constrained motion of planar 2R manipulator

- Using concept of computed torque control

$$\tau_\phi = [\alpha]_\phi {\tau'}_\phi + \beta_\phi$$

with

$$\begin{aligned} [\alpha]_\phi &= [\mathbf{M}(\Theta)][J_{\mathbf{h}}] \\ \beta_\phi &= (\mathbf{C}(\Theta,\dot{\Theta}) + [\mathbf{M}(\Theta)][\dot{J}_{\mathbf{h}}]\dot{\phi}) + \mathbf{G}(\Theta) \\ {\tau'}_\phi &= \ddot{\phi}_d + K_v(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi) \end{aligned}$$

- Choose controller gains K_p and K_v to meet performance requirement.
- Manipulator alway keeps in contact with $f(x,y) = 0$.
- The terms $\lambda(t)\nabla F(\theta_1,\theta_2)$ and τ_ϕ *do not* affect each other!
- Fairly complicated – not practical for 6 DOF manipulator → Cartesian control schemes much better!

- In parallel manipulator *loop-closure* constraint.
- Equations of motion can be derived using Lagrange multipliers (see Module 6, Lecture 1).

$$[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau + [\Psi(\mathbf{q})]^T \lambda$$

- $[\Psi(\mathbf{q})]$ and λ are similar to the Jacobian matrix $[J_{\mathbf{h}}]$ and λ for 2R serial manipulators with constraints.
- Key difference – no need to control constraint forces arising out of loop-closure constraints!

- In parallel manipulator *loop-closure* constraint.
- Equations of motion can be derived using Lagrange multipliers (see Module 6, Lecture 1).

$$[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau + [\Psi(\mathbf{q})]^T \lambda$$

- $[\Psi(\mathbf{q})]$ and λ are similar to the Jacobian matrix $[J_{\mathbf{h}}]$ and λ for 2R serial manipulators with constraints.
- Key difference – no need to control constraint forces arising out of loop-closure constraints!

- In parallel manipulator *loop-closure* constraint.
- Equations of motion can be derived using Lagrange multipliers (see Module 6, Lecture 1).

$$[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})]\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau + [\Psi(\mathbf{q})]^T \lambda$$

- $[\Psi(\mathbf{q})]$ and λ are similar to the Jacobian matrix $[J_{\mathbf{h}}]$ and λ for 2R serial manipulators with constraints.
- Key difference – no need to control constraint forces arising out of loop-closure constraints!

- In parallel manipulator *loop-closure* constraint.
- Equations of motion can be derived using Lagrange multipliers (see Module 6, Lecture 1).

$$[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau + [\Psi(\mathbf{q})]^T \lambda$$

- $[\Psi(\mathbf{q})]$ and λ are similar to the Jacobian matrix $[J_{\mathbf{h}}]$ and λ for 2R serial manipulators with constraints.
- Key difference – no need to control constraint forces arising out of loop-closure constraints!

- τ has non-zero elements *only* for the n actuated joints.
- Can directly use the equations obtained after eliminating λ (see Module 6, Lecture 1).

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\}$$

  $\mathbf{f}$ denotes $(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$.
- The $n+m$ equations of motion can be written as

$$[\mathbf{M}]\ddot{\mathbf{q}} + \mathbf{B}(\mathbf{q},\dot{\mathbf{q}}) = [\mathbf{A}(\mathbf{q})]\tau$$

- From control law partitioning

$$[\mathbf{A}(\mathbf{q})]\tau = [\alpha]\tau' + \beta$$

  Choose $[\alpha]$ and β as $[\mathbf{M}(\mathbf{q})]$ and $\mathbf{B}(\mathbf{q},\dot{\mathbf{q}})$, respectively, for the model based control part.

Control of constrained and parallel manipulator

Parallel manipulators (Contd.)

- τ has non-zero elements *only* for the n actuated joints.
- Can directly use the equations obtained after eliminating λ (see Module 6, Lecture 1).

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\}$$

  $\mathbf{f}$ denotes $(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$.

- The $n+m$ equations of motion can be written as

$$[\mathbf{M}]\ddot{\mathbf{q}} + \mathbf{B}(\mathbf{q},\dot{\mathbf{q}}) = [\mathbf{A}(\mathbf{q})]\tau$$

- From control law partitioning

$$[\mathbf{A}(\mathbf{q})]\tau = [\alpha]\tau' + \beta$$

  Choose $[\alpha]$ and β as $[\mathbf{M}(\mathbf{q})]$ and $\mathbf{B}(\mathbf{q},\dot{\mathbf{q}})$, respectively, for the model based control part.

- τ has non-zero elements *only* for the n actuated joints.
- Can directly use the equations obtained after eliminating λ (see Module 6, Lecture 1).

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\}$$

  $\mathbf{f}$ denotes $(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$.
- The $n+m$ equations of motion can be written as

$$[\mathbf{M}]\ddot{\mathbf{q}} + \mathbf{B}(\mathbf{q},\dot{\mathbf{q}}) = [\mathbf{A}(\mathbf{q})]\tau$$

- From control law partitioning

$$[\mathbf{A}(\mathbf{q})]\tau = [\alpha]\tau' + \beta$$

  Choose $[\alpha]$ and β as $[\mathbf{M}(\mathbf{q})]$ and $\mathbf{B}(\mathbf{q},\dot{\mathbf{q}})$, respectively, for the model based control part.

- τ has non-zero elements *only* for the n actuated joints.
- Can directly use the equations obtained after eliminating λ (see Module 6, Lecture 1).

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\}$$

  $\mathbf{f}$ denotes $(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$.
- The $n+m$ equations of motion can be written as

$$[\mathbf{M}]\ddot{\mathbf{q}} + \mathbf{B}(\mathbf{q},\dot{\mathbf{q}}) = [\mathbf{A}(\mathbf{q})]\tau$$

- From control law partitioning

$$[\mathbf{A}(\mathbf{q})]\tau = [\alpha]\tau' + \beta$$

  Choose $[\alpha]$ and β as $[\mathbf{M}(\mathbf{q})]$ and $\mathbf{B}(\mathbf{q},\dot{\mathbf{q}})$, respectively, for the model based control part.

Control of constrained and parallel manipulator

Parallel manipulators (Contd.)

- Choose non-zero elements of τ' for PD control with appropriate gain matrices $[K_p]$ and $[K_v]$.
- Motion of actuated joints will not violate loop-closure constraints!
- Model-based terms involve *active* and *passive* variables!
- Typically *passive* variables not measured $\rightarrow$ Passive variables must be estimated using direct-kinematics equations
- Use of direct kinematics for estimating passive joint variables and their rates make model-based control of parallel manipulators much more complex.
- As in serial manipulators cannot avoid issues arising out "lack of knowledge" of parameters.

Control of constrained and parallel manipulator

Parallel manipulators (contd.)

- Choose non-zero elements of τ' for PD control with appropriate gain matrices $[K_p]$ and $[K_v]$.
- Motion of actuated joints will not violate loop-closure constraints!
- Model-based terms involve *active* and *passive* variables!
- Typically *passive* variables not measured $\rightarrow$ Passive variables must be estimated using direct-kinematics equations
- Use of direct kinematics for estimating passive joint variables and their rates make model-based control of parallel manipulators much more complex.
- As in serial manipulators cannot avoid issues arising out "lack of knowledge" of parameters.

- Choose non-zero elements of τ' for PD control with appropriate gain matrices $[K_p]$ and $[K_v]$.
- Motion of actuated joints will not violate loop-closure constraints!
- Model-based terms involve *active* and *passive* variables!
- Typically *passive* variables not measured $\rightarrow$ Passive variables must be estimated using direct-kinematics equations
- Use of direct kinematics for estimating passive joint variables and their rates make model-based control of parallel manipulators much more complex.
- As in serial manipulators cannot avoid issues arising out "lack of knowledge" of parameters.

Control of constrained and parallel manipulator

Parallel manipulators (contd.)

- Choose non-zero elements of τ' for PD control with appropriate gain matrices $[K_p]$ and $[K_v]$.
- Motion of actuated joints will not violate loop-closure constraints!
- Model-based terms involve *active* and *passive* variables!
- Typically *passive* variables not measured $\rightarrow$ Passive variables must be estimated using direct-kinematics equations
- Use of direct kinematics for estimating passive joint variables and their rates make model-based control of parallel manipulators much more complex.
- As in serial manipulators cannot avoid issues arising out "lack of knowledge" of parameters.

- Choose non-zero elements of τ' for PD control with appropriate gain matrices $[K_p]$ and $[K_v]$.
- Motion of actuated joints will not violate loop-closure constraints!
- Model-based terms involve *active* and *passive* variables!
- Typically *passive* variables not measured $\rightarrow$ Passive variables must be estimated using direct-kinematics equations
- Use of direct kinematics for estimating passive joint variables and their rates make model-based control of parallel manipulators much more complex.
- As in serial manipulators cannot avoid issues arising out "lack of knowledge" of parameters.

Control of constrained and parallel manipulator

Parallel manipulators (Contd.)

- Choose non-zero elements of τ' for PD control with appropriate gain matrices $[K_p]$ and $[K_v]$.
- Motion of actuated joints will not violate loop-closure constraints!
- Model-based terms involve *active* and *passive* variables!
- Typically *passive* variables not measured $\rightarrow$ Passive variables must be estimated using direct-kinematics equations
- Use of direct kinematics for estimating passive joint variables and their rates make model-based control of parallel manipulators much more complex.
- As in serial manipulators cannot avoid issues arising out "lack of knowledge" of parameters.

OUTLINE

- Very difficult to implement joint space control of serial manipulators with constraint
  - The constraint is *almost always* in terms of end-effector position and/or orientation.
  - More often than not, closed-form expressions for inverse kinematics do not exist!
  - Except for simple curves, not possible to convert to a simple parallel mechanism.
- Need to develope control schemes which use desired trajectories specified in terms of Cartesian/task space variables.
- Scheme should not use inverse kinematics as its is computationally intensive.
- A model-based/feedback linearization type of control scheme is desirable.

- Very difficult to implement joint space control of serial manipulators with constraint
  - The constraint is *almost always* in terms of end-effector position and/or orientation.
  - More often than not, closed-form expressions for inverse kinematics do not exist!
  - Except for simple curves, not possible to convert to a simple parallel mechanism.
- Need to develope control schemes which use desired trajectories specified in terms of Cartesian/task space variables.
- Scheme should not use inverse kinematics as its is computationally intensive.
- A model-based/feedback linearization type of control scheme is desirable.

- Very difficult to implement joint space control of serial manipulators with constraint
  - The constraint is *almost always* in terms of end-effector position and/or orientation.
  - More often than not, closed-form expressions for inverse kinematics do not exist!
  - Except for simple curves, not possible to convert to a simple parallel mechanism.
- Need to develope control schemes which use desired trajectories specified in terms of Cartesian/task space variables.
- Scheme should not use inverse kinematics as its is computationally intensive.
- A model-based/feedback linearization type of control scheme is desirable.

- Very difficult to implement joint space control of serial manipulators with constraint
  - The constraint is *almost always* in terms of end-effector position and/or orientation.
  - More often than not, closed-form expressions for inverse kinematics do not exist!
  - Except for simple curves, not possible to convert to a simple parallel mechanism.
- Need to develope control schemes which use desired trajectories specified in terms of Cartesian/task space variables.
- Scheme should not use inverse kinematics as its is computationally intensive.
- A model-based/feedback linearization type of control scheme is desirable.

- Equations of motion in terms of Cartesian/task space variables $\mathscr{X}$ (See Module 6, Lecture 1).

$$\mathscr{F} = [M_{\mathscr{X}}(\mathbf{q})]\ddot{\mathscr{X}} + \mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q})$$

where $\mathscr{F}$ is a 6×1 entity of force & moment acting on the end-effector and

$$\begin{aligned} [J(\mathbf{q})]^T \mathscr{F} &= \tau \\ [M_{\mathscr{X}}(\mathbf{q})] &= [J(\mathbf{q})]^{-T}[\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1} \\ \mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) &= [J(\mathbf{q})]^{-T}(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) - [\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1}[\dot{J}(\mathbf{q})]\dot{\mathbf{q}}) \\ \mathbf{G}_{\mathscr{X}}(\mathbf{q}) &= [J(\mathbf{q})]^{-T}\mathbf{G}(\mathbf{q}) \end{aligned}$$

and where $[J(\mathbf{q})]^{-T}$ denotes the inverse of $[J(\mathbf{q})]^T$.

- Inverse kinematics is not required in the control.
- Inverse Jacobian required to obtain Cartesian mass matrix and other model-based terms $\rightarrow$ the model-based terms can be obtained *symbolically* once.

- Equations of motion in terms of Cartesian/task space variables $\mathscr{X}$ (See Module 6, Lecture 1).

$$\mathscr{F} = [M_{\mathscr{X}}(\mathbf{q})]\ddot{\mathscr{X}} + \mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q})$$

where $\mathscr{F}$ is a 6×1 entity of force & moment acting on the end-effector and

$$\begin{aligned}
[J(\mathbf{q})]^T \mathscr{F} &= \tau \\
[M_{\mathscr{X}}(\mathbf{q})] &= [J(\mathbf{q})]^{-T}[\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1} \\
\mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) &= [J(\mathbf{q})]^{-T}(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) - [\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1}[\dot{J}(\mathbf{q})]\dot{\mathbf{q}}) \\
\mathbf{G}_{\mathscr{X}}(\mathbf{q}) &= [J(\mathbf{q})]^{-T}\mathbf{G}(\mathbf{q})
\end{aligned}$$

and where $[J(\mathbf{q})]^{-T}$ denotes the inverse of $[J(\mathbf{q})]^T$.

- Inverse kinematics is not required in the control.
- Inverse Jacobian required to obtain Cartesian mass matrix and other model-based terms $\rightarrow$ the model-based terms can be obtained *symbolically* once.

- Equations of motion in terms of Cartesian/task space variables $\mathscr{X}$ (See Module 6, Lecture 1).

$$\mathscr{F} = [M_{\mathscr{X}}(\mathbf{q})]\ddot{\mathscr{X}} + \mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q})$$

where $\mathscr{F}$ is a 6×1 entity of force & moment acting on the end-effector and

$$\begin{aligned}
[J(\mathbf{q})]^T \mathscr{F} &= \tau \\
[M_{\mathscr{X}}(\mathbf{q})] &= [J(\mathbf{q})]^{-T}[\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1} \\
\mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) &= [J(\mathbf{q})]^{-T}(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) - [\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1}[\dot{J}(\mathbf{q})]\dot{\mathbf{q}}) \\
\mathbf{G}_{\mathscr{X}}(\mathbf{q}) &= [J(\mathbf{q})]^{-T}\mathbf{G}(\mathbf{q})
\end{aligned}$$

and where $[J(\mathbf{q})]^{-T}$ denotes the inverse of $[J(\mathbf{q})]^T$.

- Inverse kinematics is not required in the control.
- Inverse Jacobian required to obtain Cartesian mass matrix and other model-based terms $\rightarrow$ the model-based terms can be obtained *symbolically* once.

- Similar to joint space control scheme, assume a control law of the form

$$\mathscr{F} = [\alpha_{\mathscr{X}}]\mathscr{F}' + \beta_{\mathscr{X}}$$

- Choose $[\alpha_{\mathscr{X}}] = [M_{\mathscr{X}}(\mathbf{q})]$ and $\beta_{\mathscr{X}} = \mathbf{C}_{\mathscr{X}}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q})$.
- To get $\mathscr{F}' = \ddot{\mathscr{X}} \rightarrow$ Unit mass system with new input $\mathscr{F}'$
- Choose

$$\mathscr{F}' = \ddot{\mathscr{X}}_d(t) + [K_v]_{\mathscr{X}}\dot{\mathbf{e}}(t) + [K_p]_{\mathscr{X}}\mathbf{e}(t)$$

  to get *linear*, *decoupled* error equation of the form

$$\ddot{\mathbf{e}}(t) + [K_v]_{\mathscr{X}}\dot{\mathbf{e}}(t) + [K_p]_{\mathscr{X}}\mathbf{e}(t) = \mathbf{0}$$

  and appropriate choice of $[K_p]_{\mathscr{X}}$ and $[K_v]_{\mathscr{X}}$ will give required performance!
- To obtain required Cartesian actuation force & moment $\mathscr{F}$, use joint torque as $\tau = [J(\mathbf{q})]^T\mathscr{F}$

- Similar to joint space control scheme, assume a control law of the form

$$\mathscr{F} = [\alpha_{\mathscr{X}}]\mathscr{F}' + \beta_{\mathscr{X}}$$

- Choose $[\alpha_{\mathscr{X}}] = [M_{\mathscr{X}}(\mathbf{q})]$ and $\beta_{\mathscr{X}} = \mathbf{C}_{\mathscr{X}}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q})$.
- To get $\mathscr{F}' = \ddot{\mathscr{X}} \rightarrow$ Unit mass system with new input $\mathscr{F}'$
- Choose

$$\mathscr{F}' = \ddot{\mathscr{X}}_d(t) + [K_v]_{\mathscr{X}}\dot{\mathbf{e}}(t) + [K_p]_{\mathscr{X}}\mathbf{e}(t)$$

to get *linear*, *decoupled* error equation of the form

$$\ddot{\mathbf{e}}(t) + [K_v]_{\mathscr{X}}\dot{\mathbf{e}}(t) + [K_p]_{\mathscr{X}}\mathbf{e}(t) = \mathbf{0}$$

and appropriate choice of $[K_p]_{\mathscr{X}}$ and $[K_v]_{\mathscr{X}}$ will give required performance!
- To obtain required Cartesian actuation force & moment $\mathscr{F}$, use joint torque as $\tau = [J(\mathbf{q})]^T\mathscr{F}$

- Similar to joint space control scheme, assume a control law of the form

$$\mathscr{F} = [\alpha_{\mathscr{X}}]\mathscr{F}' + \beta_{\mathscr{X}}$$

- Choose $[\alpha_{\mathscr{X}}] = [M_{\mathscr{X}}(\mathbf{q})]$ and $\beta_{\mathscr{X}} = \mathbf{C}_{\mathscr{X}}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q})$.
- To get $\mathscr{F}' = \ddot{\mathscr{X}}$ → Unit mass system with new input $\mathscr{F}'$
- Choose

$$\mathscr{F}' = \ddot{\mathscr{X}}_d(t) + [K_v]_{\mathscr{X}}\dot{\mathbf{e}}(t) + [K_p]_{\mathscr{X}}\mathbf{e}(t)$$

  to get *linear*, *decoupled* error equation of the form

$$\ddot{\mathbf{e}}(t) + [K_v]_{\mathscr{X}}\dot{\mathbf{e}}(t) + [K_p]_{\mathscr{X}}\mathbf{e}(t) = \mathbf{0}$$

  and appropriate choice of $[K_p]_{\mathscr{X}}$ and $[K_v]_{\mathscr{X}}$ will give required performance!
- To obtain required Cartesian actuation force & moment $\mathscr{F}$, use joint torque as $\tau = [J(\mathbf{q})]^T\mathscr{F}$

CARTESIAN CONTROL SCHEMES

MODEL-BASED CARTESIAN CONTROL

- Similar to joint space control scheme, assume a control law of the form

$$\mathscr{F} = [\alpha_{\mathscr{X}}]\mathscr{F}' + \beta_{\mathscr{X}}$$

- Choose $[\alpha_{\mathscr{X}}] = [M_{\mathscr{X}}(\mathbf{q})]$ and $\beta_{\mathscr{X}} = \mathbf{C}_{\mathscr{X}}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q})$.
- To get $\mathscr{F}' = \ddot{\mathscr{X}} \rightarrow$ Unit mass system with new input $\mathscr{F}'$
- Choose

$$\mathscr{F}' = \ddot{\mathscr{X}}_d(t) + [K_v]_{\mathscr{X}}\dot{\mathbf{e}}(t) + [K_p]_{\mathscr{X}}\mathbf{e}(t)$$

  to get *linear*, *decoupled* error equation of the form

$$\ddot{\mathbf{e}}(t) + [K_v]_{\mathscr{X}}\dot{\mathbf{e}}(t) + [K_p]_{\mathscr{X}}\mathbf{e}(t) = \mathbf{0}$$

  and appropriate choice of $[K_p]_{\mathscr{X}}$ and $[K_v]_{\mathscr{X}}$ will give required performance!
- To obtain required Cartesian actuation force & moment $\mathscr{F}$, use joint torque as $\tau = [J(\mathbf{q})]^T \mathscr{F}$

- Similar to joint space control scheme, assume a control law of the form

$$\mathscr{F} = [\alpha_{\mathscr{X}}]\mathscr{F}' + \beta_{\mathscr{X}}$$

- Choose $[\alpha_{\mathscr{X}}] = [M_{\mathscr{X}}(\mathbf{q})]$ and $\beta_{\mathscr{X}} = \mathbf{C}_{\mathscr{X}}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q})$.
- To get $\mathscr{F}' = \ddot{\mathscr{X}} \rightarrow$ Unit mass system with new input $\mathscr{F}'$
- Choose

$$\mathscr{F}' = \ddot{\mathscr{X}}_d(t) + [K_v]_{\mathscr{X}}\dot{\mathbf{e}}(t) + [K_p]_{\mathscr{X}}\mathbf{e}(t)$$

  to get *linear*, *decoupled* error equation of the form

$$\ddot{\mathbf{e}}(t) + [K_v]_{\mathscr{X}}\dot{\mathbf{e}}(t) + [K_p]_{\mathscr{X}}\mathbf{e}(t) = \mathbf{0}$$

  and appropriate choice of $[K_p]_{\mathscr{X}}$ and $[K_v]_{\mathscr{X}}$ will give required performance!
- To obtain required Cartesian actuation force & moment $\mathscr{F}$, use joint torque as $\tau = [J(\mathbf{q})]^T\mathscr{F}$

Cartesian control schemes

Model-based Cartesian control

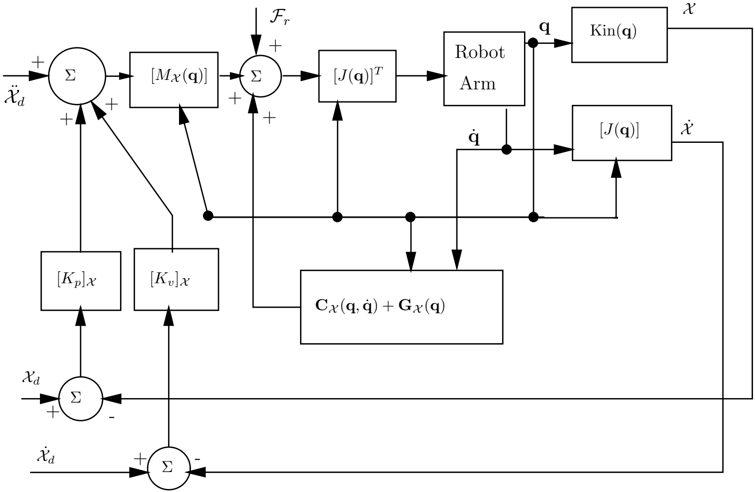


Figure 23: Cartesian model-based control scheme

- No inverse kinematics used → Direct kinematics used to estimate $\mathscr{X}$ and $\dot{\mathscr{X}}$ in figure.
- Vision or other sensors can also be used to measure $\mathscr{X}$ and $\dot{\mathscr{X}}$.
- Khatib (1986) used the Cartesian controller for *real-time* obstacle avoidance – Synthetic force $\mathscr{F}_r$ obtained as

$$\mathscr{F}_r = \sum_i^N \mathscr{F}_i \propto K_i / r_i^n$$

N is the number of obstacles and r_i is the distance from the i^{th} obstacle (see figure).

- $\mathscr{F}_r$ is *repulsive* and K_i and n chosen so that it falls off quickly!
- $\mathscr{F}$ drives the robot along a desired trajectory, when near obstacle $\mathscr{F}_r$ is more dominant → *repels* robot away from obstacles!

- No inverse kinematics used → Direct kinematics used to estimate $\mathscr{X}$ and $\dot{\mathscr{X}}$ in figure.
- Vision or other sensors can also be used to measure $\mathscr{X}$ and $\dot{\mathscr{X}}$.
- Khatib (1986) used the Cartesian controller for *real-time* obstacle avoidance – Synthetic force $\mathscr{F}_r$ obtained as

$$\mathscr{F}_r = \sum_i^N \mathscr{F}_i \propto K_i / r_i^n$$

  N is the number of obstacles and r_i is the distance from the i^{th} obstacle (see figure).
- $\mathscr{F}_r$ is *repulsive* and K_i and n chosen so that it falls off quickly!
- $\mathscr{F}$ drives the robot along a desired trajectory, when near obstacle $\mathscr{F}_r$ is more dominant → *repels* robot away from obstacles!

- No inverse kinematics used → Direct kinematics used to estimate $\mathscr{X}$ and $\dot{\mathscr{X}}$ in figure.
- Vision or other sensors can also be used to measure $\mathscr{X}$ and $\dot{\mathscr{X}}$.
- Khatib (1986) used the Cartesian controller for *real-time* obstacle avoidance – Synthetic force $\mathscr{F}_r$ obtained as

$$\mathscr{F}_r = \sum_i^N \mathscr{F}_i \propto K_i / r_i^n$$

N is the number of obstacles and r_i is the distance from the i^{th} obstacle (see figure).

- $\mathscr{F}_r$ is *repulsive* and K_i and n chosen so that it falls off quickly!
- $\mathscr{F}$ drives the robot along a desired trajectory, when near obstacle $\mathscr{F}_r$ is more dominant → *repels* robot away from obstacles!

- No inverse kinematics used → Direct kinematics used to estimate $\mathscr{X}$ and $\dot{\mathscr{X}}$ in figure.
- Vision or other sensors can also be used to measure $\mathscr{X}$ and $\dot{\mathscr{X}}$.
- Khatib (1986) used the Cartesian controller for *real-time* obstacle avoidance – Synthetic force $\mathscr{F}_r$ obtained as

$$\mathscr{F}_r = \sum_i^N \mathscr{F}_i \propto K_i / r_i^n$$

  N is the number of obstacles and r_i is the distance from the i^{th} obstacle (see figure).
- $\mathscr{F}_r$ is *repulsive* and K_i and n chosen so that it falls off quickly!
- $\mathscr{F}$ drives the robot along a desired trajectory, when near obstacle $\mathscr{F}_r$ is more dominant → *repels* robot away from obstacles!

- No inverse kinematics used → Direct kinematics used to estimate $\mathscr{X}$ and $\dot{\mathscr{X}}$ in figure.
- Vision or other sensors can also be used to measure $\mathscr{X}$ and $\dot{\mathscr{X}}$.
- Khatib (1986) used the Cartesian controller for *real-time* obstacle avoidance – Synthetic force $\mathscr{F}_r$ obtained as

$$\mathscr{F}_r = \sum_i^N \mathscr{F}_i \propto K_i / r_i^n$$

  N is the number of obstacles and r_i is the distance from the i^{th} obstacle (see figure).
- $\mathscr{F}_r$ is *repulsive* and K_i and n chosen so that it falls off quickly!
- $\mathscr{F}$ drives the robot along a desired trajectory, when near obstacle $\mathscr{F}_r$ is more dominant → *repels* robot away from obstacles!

OUTLINE

- Manipulator moving in *free space* $\rightarrow$ position control.
- Robotic *assembly*, *grinding* and manufacturing $\rightarrow$ Position control not enough $\rightarrow$ Need to apply desired force/moment on environment!
- Apply force/moment with *passive* stiffness in end-effector $\rightarrow$ Plan a trajectory such that it is 'just inside' the contacting surface.
- Difficult to apply *desired and changing* force/moment.
  - Error in position control can result in *not touching* or *excessive* interference!
  - Not possible to *apply* desired force/moment to environment if stiffness of environment is high $\rightarrow$ Very small strains and displacements difficult to measure.
- Joint space control, similar to constrained motion, not suitable
- Cartesian control strategy easily extended for force control!

- Manipulator moving in *free space* → position control.
- Robotic *assembly*, *grinding* and manufacturing → Position control not enough → Need to apply desired force/moment on environment!
- Apply force/moment with *passive* stiffness in end-effector → Plan a trajectory such that it is 'just inside' the contacting surface.
- Difficult to apply *desired and changing* force/moment.
  - Error in position control can result in *not touching* or *excessive* interference!
  - Not possible to *apply* desired force/moment to environment if stiffness of environment is high → Very small strains and displacements difficult to measure.
- Joint space control, similar to constrained motion, not suitable
- Cartesian control strategy easily extended for force control!

- Manipulator moving in *free space* → position control.
- Robotic *assembly*, *grinding* and manufacturing → Position control not enough → Need to apply desired force/moment on environment!
- Apply force/moment with *passive* stiffness in end-effector → Plan a trajectory such that it is 'just inside' the contacting surface.
- Difficult to apply *desired and changing* force/moment.
    - Error in position control can result in *not touching* or *excessive* interference!
    - Not possible to *apply* desired force/moment to environment if stiffness of environment is high → Very small strains and displacements difficult to measure.
- Joint space control, similar to constrained motion, not suitable
- Cartesian control strategy easily extended for force control!

- Manipulator moving in *free space* → position control.
- Robotic *assembly*, *grinding* and manufacturing → Position control not enough → Need to apply desired force/moment on environment!
- Apply force/moment with *passive* stiffness in end-effector → Plan a trajectory such that it is 'just inside' the contacting surface.
- Difficult to apply *desired and changing* force/moment.
  - Error in position control can result in *not touching* or *excessive* interference!
  - Not possible to *apply* desired force/moment to environment if stiffness of environment is high → Very small strains and displacements difficult to measure.
- Joint space control, similar to constrained motion, not suitable
- Cartesian control strategy easily extended for force control!

FORCE CONTROL OF MANIPULATORS

OVERVIEW

- Manipulator moving in *free space* $\rightarrow$ position control.
- Robotic *assembly*, *grinding* and manufacturing $\rightarrow$ Position control not enough $\rightarrow$ Need to apply desired force/moment on environment!
- Apply force/moment with *passive* stiffness in end-effector $\rightarrow$ Plan a trajectory such that it is 'just inside' the contacting surface.
- Difficult to apply *desired and changing* force/moment.
  - Error in position control can result in *not touching* or *excessive* interference!
  - Not possible to *apply* desired force/moment to environment if stiffness of environment is high $\rightarrow$ Very small strains and displacements difficult to measure.
- Joint space control, similar to constrained motion, not suitable
- Cartesian control strategy easily extended for force control!

- Manipulator moving in *free space* → position control.
- Robotic *assembly*, *grinding* and manufacturing → Position control not enough → Need to apply desired force/moment on environment!
- Apply force/moment with *passive* stiffness in end-effector → Plan a trajectory such that it is 'just inside' the contacting surface.
- Difficult to apply *desired and changing* force/moment.
  - Error in position control can result in *not touching* or *excessive* interference!
  - Not possible to *apply* desired force/moment to environment if stiffness of environment is high → Very small strains and displacements difficult to measure.
- Joint space control, similar to constrained motion, not suitable
- Cartesian control strategy easily extended for force control!

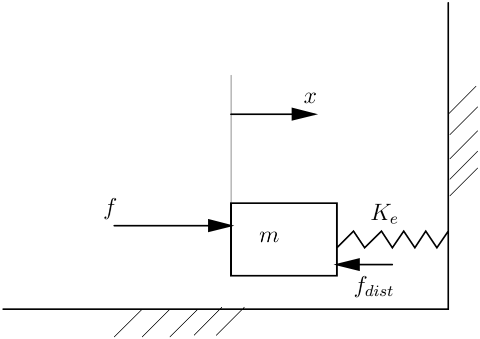


Figure 24: Force control of a mass along one direction

- Applied force from an actuator $f(t)$.
- Disturbance force $f_{\text{dist}}(t)$
- Displacement of mass $x(t)$
- Environment stiffness K_e
- Force exerted by environment $f_e(t) = K_e x(t)$
- Aim is to control $f_e(t)$ to a desired value $f_{e_d}(t)$ by $f(t)$.

- The equation of motion of the system is given by

$$f = m\ddot{x} + K_e x + f_{dist}$$

- Written in terms of f_e,

$$f = mK_e^{-1}\ddot{f}_e + f_e + f_{dist}$$

- Similar to a second-order ODE for a single-link manipulator.
- Can use PD or PID control scheme.
- Model-based control scheme is better!

- The equation of motion of the system is given by

$$f = m\ddot{x} + K_e x + f_{dist}$$

- Written in terms of f_e,

$$f = m K_e^{-1} \ddot{f}_e + f_e + f_{dist}$$

- Similar to a second-order ODE for a single-link manipulator.
- Can use PD or PID control scheme.
- Model-based control scheme is better!

- The equation of motion of the system is given by

$$f = m\ddot{x} + K_e x + f_{dist}$$

- Written in terms of f_e,

$$f = mK_e^{-1}\ddot{f}_e + f_e + f_{dist}$$

- Similar to a second-order ODE for a single-link manipulator.
- Can use PD or PID control scheme.
- Model-based control scheme is better!

- The equation of motion of the system is given by

$$f = m\ddot{x} + K_e x + f_{dist}$$

- Written in terms of f_e,

$$f = mK_e^{-1}\ddot{f}_e + f_e + f_{dist}$$

- Similar to a second-order ODE for a single-link manipulator.
- Can use PD or PID control scheme.
- Model-based control scheme is better!

- The equation of motion of the system is given by

$$f = m\ddot{x} + K_e x + f_{dist}$$

- Written in terms of f_e,

$$f = mK_e^{-1}\ddot{f}_e + f_e + f_{dist}$$

- Similar to a second-order ODE for a single-link manipulator.
- Can use PD or PID control scheme.
- Model-based control scheme is better!

Force control of a single mass (contd.)

- Following control law partitioning concept

$$
\begin{aligned}
f &= \alpha f' + \beta \\
\alpha &= m K_e^{-1} \\
\beta &= f_e + f_{dist} \\
f' &= \ddot{f}_{ed} + K_{v_f} \dot{e}_f + K_{p_f} e_f
\end{aligned}
$$

force error is $e_f = f_{ed} - f_e$ & f_e (measured) force acting on the environment.

- Closed-loop force error equation is

$$\ddot{e}_f + K_{v_f} \dot{e}_f + K_{p_f} e_f = 0$$

- K_{v_f}, K_{p_f} – derivative and proportional gains – set for required performance.

- Following control law partitioning concept

$$\begin{aligned} f &= \alpha f' + \beta \\ \alpha &= m K_e^{-1} \\ \beta &= f_e + f_{dist} \\ f' &= \ddot{f}_{ed} + K_{v_f} \dot{e}_f + K_{p_f} e_f \end{aligned}$$

  force error is $e_f = f_{ed} - f_e$ & f_e (measured) force acting on the environment.

- Closed-loop force error equation is

$$\ddot{e}_f + K_{v_f} \dot{e}_f + K_{p_f} e_f = 0$$

- K_{v_f}, K_{p_f} – derivative and proportional gains – set for required performance.

- Following control law partitioning concept

$$
\begin{aligned}
f &= \alpha f' + \beta \\
\alpha &= m K_e^{-1} \\
\beta &= f_e + f_{dist} \\
f' &= \ddot{f}_{ed} + K_{v_f} \dot{e}_f + K_{p_f} e_f
\end{aligned}
$$

force error is $e_f = f_{ed} - f_e$ & f_e (measured) force acting on the environment.

- Closed-loop force error equation is

$$\ddot{e}_f + K_{v_f} \dot{e}_f + K_{p_f} e_f = 0$$

- K_{v_f}, K_{p_f} – derivative and proportional gains – set for required performance.

FORCE CONTROL OF A SINGLE MASS (CONTD.)

- No knowledge of f_{dist} $\rightarrow$ cannot use in model-based term!
- Set $\beta = f_{e_d}$ $\rightarrow$ Steady-state error not zero!

$$e_f = \frac{f_{dist}}{1 + mK_e^{-1}}$$

- Since K_e is typically large $\rightarrow$ $e_f \simeq f_{\text{dist}}$ – best possible!
- $\dot{f}_{e_d}$ and $\ddot{f}_{e_d}$ not specified – no physical sense in derivative of desired force!
- f_e measured but $\dot{f}_e$ very difficult to measure $\rightarrow$ $\dot{f}_e = K_e \dot{x}$.
- Control law with above constraints

$$f = m[K_{p_f} K_e^{-1} e_f - K_{v_f} \dot{x}] + f_{e_d}$$

Force control of manipulators

Force control of a single mass (contd.)

- No knowledge of f_{dist} $\rightarrow$ cannot use in model-based term!
- Set $\beta = f_{e_d}$ $\rightarrow$ Steady-state error not zero!

$$e_f = \frac{f_{dist}}{1 + mK_e^{-1}}$$

- Since K_e is typically large $\rightarrow$ $e_f \simeq f_{\text{dist}}$ – best possible!
- $\dot{f}_{e_d}$ and $\ddot{f}_{e_d}$ not specified – no physical sense in derivative of desired force!
- f_e measured but $\dot{f}_e$ very difficult to measure $\rightarrow$ $\dot{f}_e = K_e\dot{x}$.
- Control law with above constraints

$$f = m[K_{p_f} K_e^{-1} e_f - K_{v_f} \dot{x}] + f_{e_d}$$

- No knowledge of f_{dist} $\rightarrow$ cannot use in model-based term!
- Set $\beta = f_{e_d}$ $\rightarrow$ Steady-state error not zero!

$$e_f = \frac{f_{dist}}{1 + mK_e^{-1}}$$

- Since K_e is typically large $\rightarrow$ $e_f \simeq f_{\mathrm{dist}}$ – best possible!
- $\dot{f}_{e_d}$ and $\ddot{f}_{e_d}$ not specified – no physical sense in derivative of desired force!
- f_e measured but $\dot{f}_e$ very difficult to measure $\rightarrow$ $\dot{f}_e = K_e \dot{x}$.
- Control law with above constraints

$$f = m[K_{p_f} K_e^{-1} e_f - K_{v_f} \dot{x}] + f_{e_d}$$

- No knowledge of $f_{\rm dist}$ $\rightarrow$ cannot use in model-based term!
- Set $\beta = f_{e_d}$ $\rightarrow$ Steady-state error not zero!

$$e_f = \frac{f_{dist}}{1 + mK_e^{-1}}$$

- Since K_e is typically large $\rightarrow$ $e_f \simeq f_{\rm dist}$ – best possible!
- $\dot{f}_{e_d}$ and $\ddot{f}_{e_d}$ not specified – no physical sense in derivative of desired force!
- f_e measured but $\dot{f}_e$ very difficult to measure $\rightarrow$ $\dot{f}_e = K_e \dot{x}$.
- Control law with above constraints

$$f = m[K_{p_f} K_e^{-1} e_f - K_{v_f} \dot{x}] + f_{e_d}$$

- No knowledge of f_{dist} $\rightarrow$ cannot use in model-based term!
- Set $\beta = f_{e_d}$ $\rightarrow$ Steady-state error not zero!

$$e_f = \frac{f_{dist}}{1 + mK_e^{-1}}$$

- Since K_e is typically large $\rightarrow$ $e_f \simeq f_{\mathrm{dist}}$ – best possible!
- $\dot{f}_{e_d}$ and $\ddot{f}_{e_d}$ not specified – no physical sense in derivative of desired force!
- f_e measured but $\dot{f}_e$ very difficult to measure $\rightarrow$ $\dot{f}_e = K_e \dot{x}$.
- Control law with above constraints

$$f = m[K_{p_f} K_e^{-1} e_f - K_{v_f} \dot{x}] + f_{e_d}$$

- No knowledge of f_{dist} $\rightarrow$ cannot use in model-based term!
- Set $\beta = f_{e_d}$ $\rightarrow$ Steady-state error not zero!

$$e_f = \frac{f_{dist}}{1 + mK_e^{-1}}$$

- Since K_e is typically large $\rightarrow$ $e_f \simeq f_{\text{dist}}$ – best possible!
- $\dot{f}_{e_d}$ and $\ddot{f}_{e_d}$ not specified – no physical sense in derivative of desired force!
- f_e measured but $\dot{f}_e$ very difficult to measure $\rightarrow$ $\dot{f}_e = K_e \dot{x}$.
- Control law with above constraints

$$f = m[K_{p_f} K_e^{-1} e_f - K_{v_f} \dot{x}] + f_{e_d}$$

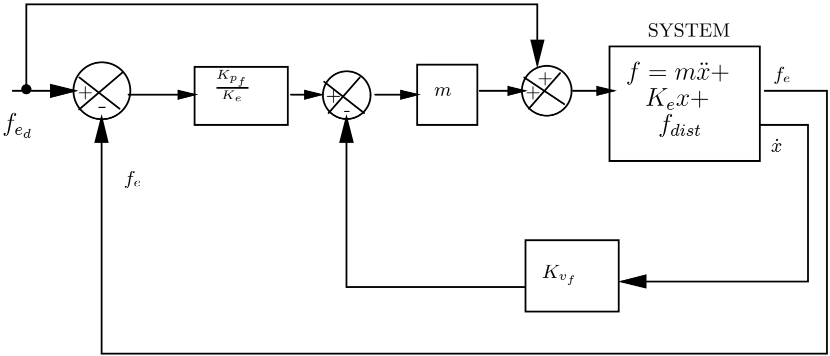


Figure 25: A force control scheme for a spring-mass system

- Difficult to estimate K_e – can change with time.
- Choose K_e large as most environments are “stiff”.
- Terms in β and derivatives of f_{e_d} dropped $\rightarrow$ e_f does not go to zero as in a second-order system!
- Six DOF manipulator, $\mathscr{F}$ and $\mathscr{X}$ are 6×1 entities (not vectors!), m is the Cartesian mass matrix and K_e is a 6×6 positive-definite (diagonal) stiffness matrix.
- The gain matrices $[K_{p_f}]$ and $[K_{v_f}]$ are 6×6 positive definite and diagonal matrices.

- Difficult to estimate K_e – can change with time.
- Choose K_e large as most environments are "stiff".
- Terms in β and derivatives of f_{e_d} dropped $\rightarrow$ e_f does not go to zero as in a second-order system!
- Six DOF manipulator, $\mathscr{F}$ and $\mathscr{X}$ are 6×1 entities (not vectors!), m is the Cartesian mass matrix and K_e is a 6×6 positive-definite (diagonal) stiffness matrix.
- The gain matrices $[K_{p_f}]$ and $[K_{v_f}]$ are 6×6 positive definite and diagonal matrices.

- Difficult to estimate K_e – can change with time.
- Choose K_e large as most environments are "stiff".
- Terms in β and derivatives of f_{e_d} dropped $\rightarrow$ e_f does not go to zero as in a second-order system!
- Six DOF manipulator, $\mathscr{F}$ and $\mathscr{X}$ are 6×1 entities (not vectors!), m is the Cartesian mass matrix and K_e is a 6×6 positive-definite (diagonal) stiffness matrix.
- The gain matrices $[K_{p_f}]$ and $[K_{v_f}]$ are 6×6 positive definite and diagonal matrices.

- Difficult to estimate K_e – can change with time.
- Choose K_e large as most environments are "stiff".
- Terms in β and derivatives of f_{e_d} dropped $\rightarrow$ e_f does not go to zero as in a second-order system!
- Six DOF manipulator, $\mathscr{F}$ and $\mathscr{X}$ are 6×1 entities (not vectors!), m is the Cartesian mass matrix and K_e is a 6×6 positive-definite (diagonal) stiffness matrix.
- The gain matrices $[K_{p_f}]$ and $[K_{v_f}]$ are 6×6 positive definite and diagonal matrices.

- Difficult to estimate K_e – can change with time.
- Choose K_e large as most environments are "stiff".
- Terms in β and derivatives of f_{e_d} dropped $\rightarrow$ e_f does not go to zero as in a second-order system!
- Six DOF manipulator, $\mathscr{F}$ and $\mathscr{X}$ are 6×1 entities (not vectors!), m is the Cartesian mass matrix and K_e is a 6×6 positive-definite (diagonal) stiffness matrix.
- The gain matrices $[K_{p_f}]$ and $[K_{v_f}]$ are 6×6 positive definite and diagonal matrices.

- Principle of *duality* – Cannot control force *and* velocity(or position) in same direction.
- Since force/torque and linear/angular velocity are related through power[5].
- Example – in robotic grinding, force can be controlled normal to surface being ground and velocity can be controlled tangent to the surface being ground[6].
- Duality is analogous to the partitioning of control torque in the planar 2R robot moving while satisfying a constraint (see Lecture 4).
- Cartesian control schemes naturally extends for force control!

[5]A more accurate description can be given using advanced kinematic concepts of screws, wrenches, and the principle of reciprocity (see papers by Mason (1981), Raibert and Craig (1981) and others listed at the end of this module for a more detailed treatment.

[6]Even if friction is taken into account, the force tangent to the surface cannot be arbitrary!

- Principle of *duality* – Cannot control force *and* velocity(or position) in same direction.
- Since force/torque and linear/angular velocity are related through power[5].
- Example – in robotic grinding, force can be controlled normal to surface being ground and velocity can be controlled tangent to the surface being ground[6].
- Duality is analogous to the partitioning of control torque in the planar 2R robot moving while satisfying a constraint (see Lecture 4).
- Cartesian control schemes naturally extends for force control!

[5] A more accurate description can be given using advanced kinematic concepts of screws, wrenches, and the principle of reciprocity (see papers by Mason (1981), Raibert and Craig (1981) and others listed at the end of this module for a more detailed treatment.

[6] Even if friction is taken into account, the force tangent to the surface cannot be arbitrary!

- Principle of *duality* – Cannot control force *and* velocity(or position) in same direction.
- Since force/torque and linear/angular velocity are related through power[5].
- Example – in robotic grinding, force can be controlled normal to surface being ground and velocity can be controlled tangent to the surface being ground[6].
- Duality is analogous to the partitioning of control torque in the planar 2R robot moving while satisfying a constraint (see Lecture 4).
- Cartesian control schemes naturally extends for force control!

[5] A more accurate description can be given using advanced kinematic concepts of screws, wrenches, and the principle of reciprocity (see papers by Mason (1981), Raibert and Craig (1981) and others listed at the end of this module for a more detailed treatment.

[6] Even if friction is taken into account, the force tangent to the surface cannot be arbitrary!

- Principle of *duality* – Cannot control force *and* velocity(or position) in same direction.
- Since force/torque and linear/angular velocity are related through power[5].
- Example – in robotic grinding, force can be controlled normal to surface being ground and velocity can be controlled tangent to the surface being ground[6].
- Duality is analogous to the partitioning of control torque in the planar 2R robot moving while satisfying a constraint (see Lecture 4).
- Cartesian control schemes naturally extends for force control!

---

[5] A more accurate description can be given using advanced kinematic concepts of screws, wrenches, and the principle of reciprocity (see papers by Mason (1981), Raibert and Craig (1981) and others listed at the end of this module for a more detailed treatment.

[6] Even if friction is taken into account, the force tangent to the surface cannot be arbitrary!

- Principle of *duality* – Cannot control force *and* velocity(or position) in same direction.
- Since force/torque and linear/angular velocity are related through power[5].
- Example – in robotic grinding, force can be controlled normal to surface being ground and velocity can be controlled tangent to the surface being ground[6].
- Duality is analogous to the partitioning of control torque in the planar 2R robot moving while satisfying a constraint (see Lecture 4).
- Cartesian control schemes naturally extends for force control!

[5] A more accurate description can be given using advanced kinematic concepts of screws, wrenches, and the principle of reciprocity (see papers by Mason (1981), Raibert and Craig (1981) and others listed at the end of this module for a more detailed treatment.

[6] Even if friction is taken into account, the force tangent to the surface cannot be arbitrary!

- Robotic tasks divided into subtasks – (a) in contact with environment or (b) in free space.
- Tasks in contact with environment – position control and force controlled 'directions".
- *Natural constraint* on position and force when manipulator in contact with a surface[7] – involve variables that *cannot be controlled*.
  - Manipulator cannot go through surface – natural position constraint.
  - Manipulator cannot apply arbitrary force tangent to surface – natural force constraint.
- Natural position constraints *normal* to surface and natural force constraint *tangent* to surface.
- Can generate natural position and force constraints for any robotic task where robot in contact with environment.

[7] We follow Craig (1989) for this treatment.

- Robotic tasks divided into subtasks – (a) in contact with environment or (b) in free space.
- Tasks in contact with environment – position control and force controlled ‘directions”.
- *Natural constraint* on position and force when manipulator in contact with a surface[7] – involve variables that *cannot be controlled*.
  - Manipulator cannot go through surface – natural position constraint.
  - Manipulator cannot apply arbitrary force tangent to surface – natural force constraint.
- Natural position constraints *normal* to surface and natural force constraint *tangent* to surface.
- Can generate natural position and force constraints for any robotic task where robot in contact with environment.

[7] We follow Craig (1989) for this treatment.

- Robotic tasks divided into subtasks – (a) in contact with environment or (b) in free space.
- Tasks in contact with environment – position control and force controlled 'directions".
- *Natural constraint* on position and force when manipulator in contact with a surface[7] – involve variables that *cannot be controlled*.
  - Manipulator cannot go through surface – natural position constraint.
  - Manipulator cannot apply arbitrary force tangent to surface – natural force constraint.
- Natural position constraints *normal* to surface and natural force constraint *tangent* to surface.
- Can generate natural position and force constraints for any robotic task where robot in contact with environment.

[7] We follow Craig (1989) for this treatment.

- Robotic tasks divided into subtasks – (a) in contact with environment or (b) in free space.
- Tasks in contact with environment – position control and force controlled 'directions".
- *Natural constraint* on position and force when manipulator in contact with a surface[7] – involve variables that *cannot be controlled*.
  - Manipulator cannot go through surface – natural position constraint.
  - Manipulator cannot apply arbitrary force tangent to surface – natural force constraint.
- Natural position constraints *normal* to surface and natural force constraint *tangent* to surface.
- Can generate natural position and force constraints for any robotic task where robot in contact with environment.

[7] We follow Craig (1989) for this treatment.

- Robotic tasks divided into subtasks – (a) in contact with environment or (b) in free space.
- Tasks in contact with environment – position control and force controlled 'directions".
- *Natural constraint* on position and force when manipulator in contact with a surface[7] – involve variables that *cannot be controlled*.
  - Manipulator cannot go through surface – natural position constraint.
  - Manipulator cannot apply arbitrary force tangent to surface – natural force constraint.
- Natural position constraints *normal* to surface and natural force constraint *tangent* to surface.
- Can generate natural position and force constraints for any robotic task where robot in contact with environment.

[7] We follow Craig (1989) for this treatment.

- *Artificial constraints* – all position and force variables that *can be controlled*.
- Manipulator in contact with environment
  - Position variables in the tangent direction can be controlled.
  - Force variables in the normal direction can be controlled.
- *Natural* and *Artificial* constraints partition position and force variables in two complementary sets.
- Follows from principle of duality.
- Typical examples shown next!

- *Artificial constraints* – all position and force variables that *can be controlled*.
- Manipulator in contact with environment
  - Position variables in the tangent direction can be controlled.
  - Force variables in the normal direction can be controlled.
- *Natural* and *Artificial* constraints partition position and force variables in two complementary sets.
- Follows from principle of duality.
- Typical examples shown next!

- *Artificial constraints* – all position and force variables that *can be controlled*.
- Manipulator in contact with environment
  - Position variables in the tangent direction can be controlled.
  - Force variables in the normal direction can be controlled.
- *Natural* and *Artificial* constraints partition position and force variables in two complementary sets.
- Follows from principle of duality.
- Typical examples shown next!

- *Artificial constraints* – all position and force variables that *can be controlled*.
- Manipulator in contact with environment
  - Position variables in the tangent direction can be controlled.
  - Force variables in the normal direction can be controlled.
- *Natural* and *Artificial* constraints partition position and force variables in two complementary sets.
- Follows from principle of duality.
- Typical examples shown next!

- *Artificial constraints* – all position and force variables that *can be controlled*.
- Manipulator in contact with environment
  - Position variables in the tangent direction can be controlled.
  - Force variables in the normal direction can be controlled.
- *Natural* and *Artificial* constraints partition position and force variables in two complementary sets.
- Follows from principle of duality.
- Typical examples shown next!

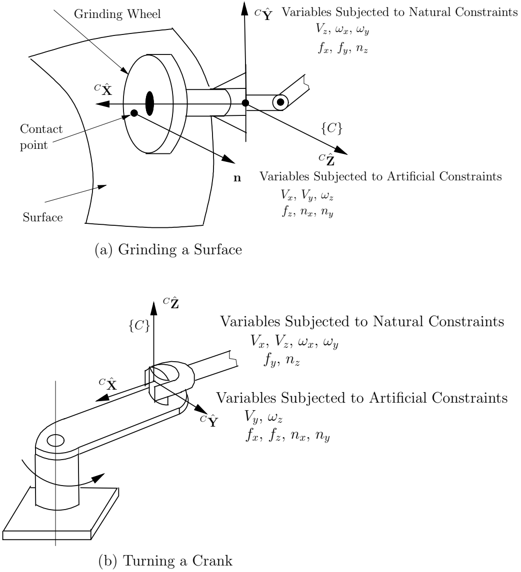


Figure 26: Natural and artificial constraints for two tasks

- Manipulator holding a grinding wheel grinding a surface.
- Define constraint frame $\{C\}$ at end-effector
  - ${}^C\hat{\mathbf{Z}}$ is parallel to the normal $\mathbf{n}$
  - ${}^C\hat{\mathbf{X}}$ and ${}^C\hat{\mathbf{Y}}$ determine the tangent plane at the point of contact on the surface.
- Grinding – a desired force along the normal and a desired trajectory on the surface.
- All constraints described in $\{C\}$ using linear velocity components V_x, V_y, V_z, angular velocity components $\omega_x, \omega_y, \omega_z$, force components f_x, f_y, f_z, and moment components n_x, n_y, n_z.

- Manipulator holding a grinding wheel grinding a surface.
- Define constraint frame $\{C\}$ at end-effector
  - ${}^C\hat{\mathbf{Z}}$ is parallel to the normal $\mathbf{n}$
  - ${}^C\hat{\mathbf{X}}$ and ${}^C\hat{\mathbf{Y}}$ determine the tangent plane at the point of contact on the surface.
- Grinding – a desired force along the normal and a desired trajectory on the surface.
- All constraints described in $\{C\}$ using linear velocity components V_x, V_y, V_z, angular velocity components $\omega_x, \omega_y, \omega_z$, force components f_x, f_y, f_z, and moment components n_x, n_y, n_z.

- Manipulator holding a grinding wheel grinding a surface.
- Define constraint frame $\{C\}$ at end-effector
  - ${}^C\hat{\mathbf{Z}}$ is parallel to the normal $\mathbf{n}$
  - ${}^C\hat{\mathbf{X}}$ and ${}^C\hat{\mathbf{Y}}$ determine the tangent plane at the point of contact on the surface.
- Grinding – a desired force along the normal and a desired trajectory on the surface.
- All constraints described in $\{C\}$ using linear velocity components V_x, V_y, V_z, angular velocity components $\omega_x, \omega_y, \omega_z$, force components f_x, f_y, f_z, and moment components n_x, n_y, n_z.

- Manipulator holding a grinding wheel grinding a surface.
- Define constraint frame $\{C\}$ at end-effector
  - ${}^{C}\hat{\mathbf{Z}}$ is parallel to the normal $\mathbf{n}$
  - ${}^{C}\hat{\mathbf{X}}$ and ${}^{C}\hat{\mathbf{Y}}$ determine the tangent plane at the point of contact on the surface.
- Grinding – a desired force along the normal and a desired trajectory on the surface.
- All constraints described in $\{C\}$ using linear velocity components V_x, V_y, V_z, angular velocity components $\omega_x, \omega_y, \omega_z$, force components f_x, f_y, f_z, and moment components n_x, n_y, n_z.

- Cannot loose contact or interfere $\rightarrow V_z = 0$.
- Grinding wheel has area contact $\rightarrow \omega_x = \omega_y = 0$ so as not too loose contact.
- f_x, f_y and n_z determined by the friction – not arbitrary!
- V_x and V_y determine *desired* trajectory $\rightarrow$ artificial constraint.
- Desired force $f_z \rightarrow$ artificial constraint.
- ω_z, n_x and n_y from principle of duality $\rightarrow$ artificial constraints!

- Cannot loose contact or interfere $\rightarrow V_z = 0$.
- Grinding wheel has area contact $\rightarrow \omega_x = \omega_y = 0$ so as not too loose contact.
- f_x, f_y and n_z determined by the friction – not arbitrary!
- V_x and V_y determine *desired* trajectory $\rightarrow$ artificial constraint.
- Desired force $f_z \rightarrow$ artificial constraint.
- ω_z, n_x and n_y from principle of duality $\rightarrow$ artificial constraints!

- Cannot loose contact or interfere $\rightarrow$ $V_z = 0$.
- Grinding wheel has area contact $\rightarrow$ $\omega_x = \omega_y = 0$ so as not too loose contact.
- f_x, f_y and n_z determined by the friction – not arbitrary!
- V_x and V_y determine *desired* trajectory $\rightarrow$ artificial constraint.
- Desired force f_z $\rightarrow$ artificial constraint.
- ω_z, n_x and n_y from principle of duality $\rightarrow$ artificial constraints!

- Cannot loose contact or interfere $\rightarrow V_z = 0$.
- Grinding wheel has area contact $\rightarrow \omega_x = \omega_y = 0$ so as not too loose contact.
- f_x, f_y and n_z determined by the friction – not arbitrary!
- V_x and V_y determine *desired* trajectory $\rightarrow$ artificial constraint.
- Desired force $f_z \rightarrow$ artificial constraint.
- ω_z, n_x and n_y from principle of duality $\rightarrow$ artificial constraints!

- Cannot loose contact or interfere $\rightarrow V_z = 0$.
- Grinding wheel has area contact $\rightarrow \omega_x = \omega_y = 0$ so as not too loose contact.
- f_x, f_y and n_z determined by the friction – not arbitrary!
- V_x and V_y determine *desired* trajectory $\rightarrow$ artificial constraint.
- Desired force $f_z \rightarrow$ artificial constraint.
- ω_z, n_x and n_y from principle of duality $\rightarrow$ artificial constraints!

- Cannot loose contact or interfere $\rightarrow V_z = 0$.
- Grinding wheel has area contact $\rightarrow \omega_x = \omega_y = 0$ so as not too loose contact.
- f_x, f_y and n_z determined by the friction – not arbitrary!
- V_x and V_y determine *desired* trajectory $\rightarrow$ artificial constraint.
- Desired force $f_z \rightarrow$ artificial constraint.
- ω_z, n_x and n_y from principle of duality $\rightarrow$ artificial constraints!

- Robot turning a crank – $\{C\}$ as shown in figure.
- $V_x = V_z = 0$ – No motion possible along ${}^C\hat{\mathbf{X}}$ or ${}^C\hat{\mathbf{Z}}$ direction.
- $\omega_x = \omega_y = 0$ – No rotation possible along ${}^C\hat{\mathbf{X}}$- and ${}^C\hat{\mathbf{Y}}$-axis.
- Cannot apply any force along the ${}^C\hat{\mathbf{Y}}$-axis or apply moment about the ${}^C\hat{\mathbf{Z}}$-axis.
- Artificial position constraints → Controlled position/orientation variables V_y and ω_z.
- Artificial force constraints → Controlled force/moment variables – f_x, f_z, n_x and n_y.

- Robot turning a crank – $\{C\}$ as shown in figure.
- $V_x = V_z = 0$ – No motion possible along ${}^C\hat{\mathbf{X}}$ or ${}^C\hat{\mathbf{Z}}$ direction.
- $\omega_x = \omega_y = 0$ – No rotation possible along ${}^C\hat{\mathbf{X}}$- and ${}^C\hat{\mathbf{Y}}$-axis.
- Cannot apply any force along the ${}^C\hat{\mathbf{Y}}$-axis or apply moment about the ${}^C\hat{\mathbf{Z}}$-axis.
- Artificial position constraints $\rightarrow$ Controlled position/orientation variables V_y and ω_z.
- Artificial force constraints $\rightarrow$ Controlled force/moment variables – f_x, f_z, n_x and n_y.

- Robot turning a crank – $\{C\}$ as shown in figure.
- $V_x = V_z = 0$ – No motion possible along ${}^C\hat{\mathbf{X}}$ or ${}^C\hat{\mathbf{Z}}$ direction.
- $\omega_x = \omega_y = 0$ – No rotation possible along ${}^C\hat{\mathbf{X}}$- and ${}^C\hat{\mathbf{Y}}$-axis.
- Cannot apply any force along the ${}^C\hat{\mathbf{Y}}$-axis or apply moment about the ${}^C\hat{\mathbf{Z}}$-axis.
- Artificial position constraints $\rightarrow$ Controlled position/orientation variables V_y and ω_z.
- Artificial force constraints $\rightarrow$ Controlled force/moment variables – f_x, f_z, n_x and n_y.

- Robot turning a crank – $\{C\}$ as shown in figure.
- $V_x = V_z = 0$ – No motion possible along ${}^C\hat{\mathbf{X}}$ or ${}^C\hat{\mathbf{Z}}$ direction.
- $\omega_x = \omega_y = 0$ – No rotation possible along ${}^C\hat{\mathbf{X}}$- and ${}^C\hat{\mathbf{Y}}$-axis.
- Cannot apply any force along the ${}^C\hat{\mathbf{Y}}$-axis or apply moment about the ${}^C\hat{\mathbf{Z}}$-axis.
- Artificial position constraints → Controlled position/orientation variables V_y and ω_z.
- Artificial force constraints → Controlled force/moment variables – f_x, f_z, n_x and n_y.

- Robot turning a crank – $\{C\}$ as shown in figure.
- $V_x = V_z = 0$ – No motion possible along ${}^C\hat{\mathbf{X}}$ or ${}^C\hat{\mathbf{Z}}$ direction.
- $\omega_x = \omega_y = 0$ – No rotation possible along ${}^C\hat{\mathbf{X}}$- and ${}^C\hat{\mathbf{Y}}$-axis.
- Cannot apply any force along the ${}^C\hat{\mathbf{Y}}$-axis or apply moment about the ${}^C\hat{\mathbf{Z}}$-axis.
- Artificial position constraints $\rightarrow$ Controlled position/orientation variables V_y and ω_z.
- Artificial force constraints $\rightarrow$ Controlled force/moment variables – f_x, f_z, n_x and n_y.

- Robot turning a crank – $\{C\}$ as shown in figure.
- $V_x = V_z = 0$ – No motion possible along ${}^C\hat{\mathbf{X}}$ or ${}^C\hat{\mathbf{Z}}$ direction.
- $\omega_x = \omega_y = 0$ – No rotation possible along ${}^C\hat{\mathbf{X}}$- and ${}^C\hat{\mathbf{Y}}$-axis.
- Cannot apply any force along the ${}^C\hat{\mathbf{Y}}$-axis or apply moment about the ${}^C\hat{\mathbf{Z}}$-axis.
- Artificial position constraints $\rightarrow$ Controlled position/orientation variables V_y and ω_z.
- Artificial force constraints $\rightarrow$ Controlled force/moment variables – f_x, f_z, n_x and n_y.

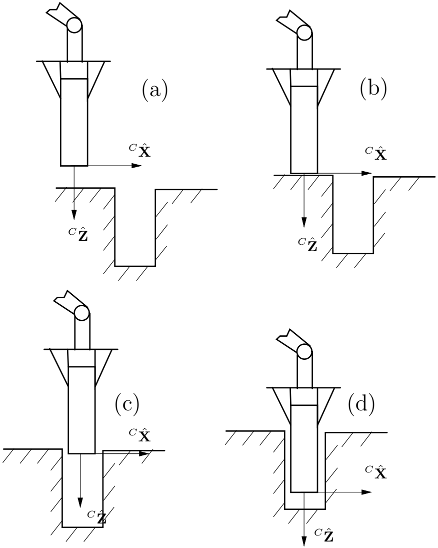


Figure 27: Peg-in-hole assembly

- Classic problem in robotic assembly – Assembly of a peg in a hole.
- Assumptions:
  - 2D motion of peg.
  - No friction between peg and hole surface.
  - Sensors available to find hole.
- Can be divided into 4 stages.
  - Stage 1 – motion in free space – figure (a)
  - Stage 2 – motion while touching surface – figure (b)
  - Stage 3 – insertion of peg in hole – figure (c)
  - Stage 4 – completion of assembly – figure (d)

- Classic problem in robotic assembly – Assembly of a peg in a hole.
- Assumptions:
  - 2D motion of peg.
  - No friction between peg and hole surface.
  - Sensors available to find hole.
- Can be divided into 4 stages.
  - Stage 1 – motion in free space – figure (a)
  - Stage 2 – motion while touching surface – figure (b)
  - Stage 3 – insertion of peg in hole – figure (c)
  - Stage 4 – completion of assembly – figure (d)

- Classic problem in robotic assembly – Assembly of a peg in a hole.
- Assumptions:
  - 2D motion of peg.
  - No friction between peg and hole surface.
  - Sensors available to find hole.
- Can be divided into 4 stages.
  - Stage 1 – motion in free space – figure (a)
  - Stage 2 – motion while touching surface – figure (b)
  - Stage 3 – insertion of peg in hole – figure (c)
  - Stage 4 – completion of assembly – figure (d)

- Stage 1 – natural constraints

$$^{C}\mathscr{F} = [f_x,\ f_y,\ f_z;\ n_x,\ n_y,\ n_z]^T = 0$$

Motion in free space → no forces/moments on end-effector.

- Stage 1 – artificial constraints

$$^{C}\mathscr{V} = [0,\ 0,\ v_a;\ 0,\ 0,\ 0]^T$$

v_a is a desired approach velocity → manipulator under *pure position* control.

- Stage 2 – natural constraints
  - Once peg touches surface, no motion along $^{C}\hat{\mathbf{Z}}$ or rotation about $^{C}\hat{\mathbf{X}}$- or $^{C}\hat{\mathbf{Y}}$-axis.
  - Cannot apply force along the direction of sliding.

$$V_z = 0, \quad \omega_x = 0, \quad \omega_y = 0$$
$$f_x = 0, \quad f_y = 0, \quad n_z = 0$$

- Stage 1 – natural constraints

$$ {}^{C}\mathscr{F} = [f_x,\ f_y,\ f_z;\ n_x,\ n_y,\ n_z]^T = 0 $$

  Motion in free space $\rightarrow$ no forces/moments on end-effector.
- Stage 1 – artificial constraints

$$ {}^{C}\mathscr{V} = [0,\ 0,\ v_a;\ 0,\ 0,\ 0]^T $$

  v_a is a desired approach velocity $\rightarrow$ manipulator under *pure position* control.
- Stage 2 – natural constraints
  - Once peg touches surface, no motion along ${}^{C}\hat{\mathbf{Z}}$ or rotation about ${}^{C}\hat{\mathbf{X}}$- or ${}^{C}\hat{\mathbf{Y}}$-axis.
  - Cannot apply force along the direction of sliding.

$$ \begin{aligned} V_z &= 0, \quad \omega_x = 0, \quad \omega_y = 0 \\ f_x &= 0, \quad f_y = 0, \quad n_z = 0 \end{aligned} $$

Force control of manipulators

Partitioning of tasks – Example of Assembly

- Stage 1 – natural constraints

$$^{C}\mathscr{F} = [f_x,\ f_y,\ f_z;\ n_x,\ n_y,\ n_z]^T = 0$$

Motion in free space $\rightarrow$ no forces/moments on end-effector.

- Stage 1 – artificial constraints

$$^{C}\mathscr{V} = [0,\ 0,\ v_a;\ 0,\ 0,\ 0]^T$$

v_a is a desired approach velocity $\rightarrow$ manipulator under *pure position* control.

- Stage 2 – natural constraints
  - Once peg touches surface, no motion along $^{C}\hat{\mathbf{Z}}$ or rotation about $^{C}\hat{\mathbf{X}}$- or $^{C}\hat{\mathbf{Y}}$-axis.
  - Cannot apply force along the direction of sliding.

$$\begin{aligned} V_z &= 0, \quad \omega_x = 0, \quad \omega_y = 0 \\ f_x &= 0, \quad f_y = 0, \quad n_z = 0 \end{aligned}$$

- Stage 2 – artificial constraints
  - Apply a small force along the ${}^{C}\hat{\mathbf{Z}}$-axis to keep it in contact.
  - Control the velocity along the ${}^{C}\hat{\mathbf{X}}$-axis.

$$V_x = v_s, \quad V_y = 0, \quad \omega_z = 0$$
$$f_z = f_c, \quad n_x = 0, \quad n_y = 0$$

  v_s and f_c are the sliding velocity and the contact force.

- Stage 3 – natural constraints

$$V_x = 0, \quad V_y = 0, \quad \omega_x = 0, \quad \omega_y = 0$$
$$f_z = 0, \quad n_z = 0$$

  After some motion along the ${}^{C}\hat{\mathbf{X}}$-axis, the peg will fall into the hole.

- Stage 3 – artificial constraints or controlled variables are

$$V_z = v_i, \quad \omega_z = 0$$
$$f_x = 0, \quad f_y = 0, \quad n_x = 0, \quad n_y = 0$$

  v_i is the insertion speed of the peg.

- Stage 2 – artificial constraints
  - Apply a small force along the ${}^{C}\hat{\mathbf{Z}}$-axis to keep it in contact.
  - Control the velocity along the ${}^{C}\hat{\mathbf{X}}$-axis.

$$\begin{aligned} V_x &= v_s, \quad V_y = 0, \quad \omega_z = 0 \\ f_z &= f_c, \quad n_x = 0, \quad n_y = 0 \end{aligned}$$

  v_s and f_c are the sliding velocity and the contact force.
- Stage 3 – natural constraints

$$\begin{aligned} V_x &= 0, \quad V_y = 0, \quad \omega_x = 0, \quad \omega_y = 0 \\ f_z &= 0, \quad n_z = 0 \end{aligned}$$

  After some motion along the ${}^{C}\hat{\mathbf{X}}$-axis, the peg will fall into the hole.
- Stage 3 – artificial constraints or controlled variables are

$$\begin{aligned} V_z &= v_i, \quad \omega_z = 0 \\ f_x &= 0, \quad f_y = 0, \quad n_x = 0, \quad n_y = 0 \end{aligned}$$

  v_i is the insertion speed of the peg.

- Stage 2 – artificial constraints
  - Apply a small force along the ${}^C\hat{\mathbf{Z}}$-axis to keep it in contact.
  - Control the velocity along the ${}^C\hat{\mathbf{X}}$-axis.

$$
\begin{aligned}
V_x &= v_s, \quad V_y = 0, \quad \omega_z = 0 \\
f_z &= f_c, \quad n_x = 0, \quad n_y = 0
\end{aligned}
$$

  v_s and f_c are the sliding velocity and the contact force.

- Stage 3 – natural constraints

$$
\begin{aligned}
V_x &= 0, \quad V_y = 0, \quad \omega_x = 0, \quad \omega_y = 0 \\
f_z &= 0, \quad n_z = 0
\end{aligned}
$$

After some motion along the ${}^C\hat{\mathbf{X}}$-axis, the peg will fall into the hole.

- Stage 3 – artificial constraints or controlled variables are

$$
\begin{aligned}
V_z &= v_i, \quad \omega_z = 0 \\
f_x &= 0, \quad f_y = 0, \quad n_x = 0, \quad n_y = 0
\end{aligned}
$$

v_i is the insertion speed of the peg.

- Stage 4 – natural constraints

$$^{C}\mathscr{V} = [V_x,\ V_y,\ V_z; \omega_x,\ \omega_y,\ \omega_z]^T = 0$$

No motion after a full insertion.

- Stage 4 – controlled variables

$$^{C}\mathscr{F} = [f_x,\ f_y,\ f_z;\ n_x,\ n_y,\ n_z]^T = 0$$

No force should be applied after assembly is over!

- Switching between stages decided by monitoring changes in *natural* constraints – not the variable being controlled!
  - Stage 1 to 2 – monitor *force* along $^{C}\hat{\mathbf{Z}}$ $\rightarrow$ This should cross (from $\mathbf{0}$) a chosen threshold value.
  - Stage 2 to 3 – monitor *motion* along $^{C}\hat{\mathbf{Z}}$ $\rightarrow$ This should cross (from $\mathbf{0}$) a chosen threshold value.
  - Stage 3 to 4 – monitor *force* along $^{C}\hat{\mathbf{Z}}$ $\rightarrow$ This should cross (from $\mathbf{0}$) a chosen threshold value.

- Stage 4 – natural constraints

$$^{C}\mathscr{V} = [V_x,\ V_y,\ V_z; \omega_x,\ \omega_y,\ \omega_z]^T = 0$$

No motion after a full insertion.

- Stage 4 – controlled variables

$$^{C}\mathscr{F} = [f_x,\ f_y,\ f_z;\ n_x,\ n_y,\ n_z]^T = 0$$

No force should be applied after assembly is over!

- Switching between stages decided by monitoring changes in *natural* constraints – not the variable being controlled!
  - Stage 1 to 2 – monitor *force* along $^{C}\hat{\mathbf{Z}}$ $\rightarrow$ This should cross (from **0**) a chosen threshold value.
  - Stage 2 to 3 – monitor *motion* along $^{C}\hat{\mathbf{Z}}$ $\rightarrow$ This should cross (from **0**) a chosen threshold value.
  - Stage 3 to 4 – monitor *force* along $^{C}\hat{\mathbf{Z}}$ $\rightarrow$ This should cross (from **0**) a chosen threshold value.

- Stage 4 – natural constraints

$$^{C}\mathscr{V} = [V_x,\ V_y,\ V_z; \omega_x,\ \omega_y,\ \omega_z]^T = 0$$

No motion after a full insertion.

- Stage 4 – controlled variables

$$^{C}\mathscr{F} = [f_x,\ f_y,\ f_z;\ n_x,\ n_y,\ n_z]^T = 0$$

No force should be applied after assembly is over!

- Switching between stages decided by monitoring changes in *natural* constraints – not the variable being controlled!
  - Stage 1 to 2 – monitor *force* along $^{C}\hat{\mathbf{Z}} \rightarrow$ This should cross (from $\mathbf{0}$) a chosen threshold value.
  - Stage 2 to 3 – monitor *motion* along $^{C}\hat{\mathbf{Z}} \rightarrow$ This should cross (from $\mathbf{0}$) a chosen threshold value.
  - Stage 3 to 4 – monitor *force* along $^{C}\hat{\mathbf{Z}} \rightarrow$ This should cross (from $\mathbf{0}$) a chosen threshold value.

Outline

- Many robotic tasks require position and force control at the same time.
- Position and force control not in the same direction!
- Joint space scheme shown for planar 2R with constraint not feasible for spatial and multi-DOF motions.
- Cartesian position and force control algorithms can be combined.
- Choose position and force control variables using a task planner as shown in examples.
- Form a selector switch to select appropriate position and force variables for control!

- Many robotic tasks require position and force control at the same time.
- Position and force control not in the same direction!
- Joint space scheme shown for planar 2R with constraint not feasible for spatial and multi-DOF motions.
- Cartesian position and force control algorithms can be combined.
- Choose position and force control variables using a task planner as shown in examples.
- Form a selector switch to select appropriate position and force variables for control!

- Many robotic tasks require position and force control at the same time.
- Position and force control not in the same direction!
- Joint space scheme shown for planar 2R with constraint not feasible for spatial and multi-DOF motions.
- Cartesian position and force control algorithms can be combined.
- Choose position and force control variables using a task planner as shown in examples.
- Form a selector switch to select appropriate position and force variables for control!

- Many robotic tasks require position and force control at the same time.
- Position and force control not in the same direction!
- Joint space scheme shown for planar 2R with constraint not feasible for spatial and multi-DOF motions.
- Cartesian position and force control algorithms can be combined.
- Choose position and force control variables using a task planner as shown in examples.
- Form a selector switch to select appropriate position and force variables for control!

- Many robotic tasks require position and force control at the same time.
- Position and force control not in the same direction!
- Joint space scheme shown for planar 2R with constraint not feasible for spatial and multi-DOF motions.
- Cartesian position and force control algorithms can be combined.
- Choose position and force control variables using a task planner as shown in examples.
- Form a selector switch to select appropriate position and force variables for control!

- Many robotic tasks require position and force control at the same time.
- Position and force control not in the same direction!
- Joint space scheme shown for planar 2R with constraint not feasible for spatial and multi-DOF motions.
- Cartesian position and force control algorithms can be combined.
- Choose position and force control variables using a task planner as shown in examples.
- Form a selector switch to select appropriate position and force variables for control!

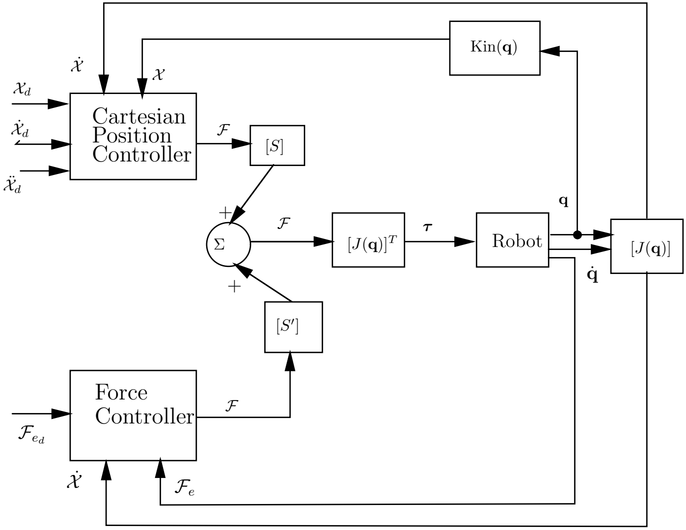


Figure 28: A hybrid position/force controller

- Top half of figure implements Cartesian position control & Bottom half of figure implements force control.
- Output of both controllers are $\mathscr{F}$ and can be combined!
- Matrix $[S]$ and $[S']$ selector switches to select position and force variables, according to principle of duality.
- For Stage 2 in peg-in-hole assembly

$$[S] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad [S'] = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Top half of figure implements Cartesian position control & Bottom half of figure implements force control.
- Output of both controllers are $\mathscr{F}$ and can be combined!
- Matrix $[S]$ and $[S']$ selector switches to select position and force variables, according to principle of duality.
- For Stage 2 in peg-in-hole assembly

$$[S] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad [S'] = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Top half of figure implements Cartesian position control & Bottom half of figure implements force control.
- Output of both controllers are $\mathscr{F}$ and can be combined!
- Matrix $[S]$ and $[S']$ selector switches to select position and force variables, according to principle of duality.
- For Stage 2 in peg-in-hole assembly

$$[S] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad [S'] = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Top half of figure implements Cartesian position control & Bottom half of figure implements force control.
- Output of both controllers are $\mathscr{F}$ and can be combined!
- Matrix $[S]$ and $[S']$ selector switches to select position and force variables, according to principle of duality.
- For Stage 2 in peg-in-hole assembly

$$[S] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad [S'] = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Outline

- Stability – *bounded input* gives *bounded output*.
- Linear controller – easy to analyse
  - Single-link manipulator under a proportional control scheme.
  - $\Omega(t) \rightarrow \Omega_d(t)$ as $t \rightarrow \infty$
  - Proportional controller is stable.
  - PD is also stable but PID can become unstable!
- Non-linear controllers – difficult to analyse for stability.
- Stability analysis using Lyapunov's method.
- Controllability.

- Stability – *bounded input* gives *bounded output*.
- Linear controller – easy to analyse
  - Single-link manipulator under a proportional control scheme.
  - $\Omega(t) \to \Omega_d(t)$ as $t \to \infty$
  - Proportional controller is stable.
  - PD is also stable but PID can become unstable!
- Non-linear controllers – difficult to analyse for stability.
- Stability analysis using Lyapunov's method.
- Controllability.

- Stability – *bounded input* gives *bounded output*.
- Linear controller – easy to analyse
  - Single-link manipulator under a proportional control scheme.
  - $\Omega(t) \rightarrow \Omega_d(t)$ as $t \rightarrow \infty$
  - Proportional controller is stable.
  - PD is also stable but PID can become unstable!
- Non-linear controllers – difficult to analyse for stability.
- Stability analysis using Lyapunov's method.
- Controllability.

- Stability – *bounded input* gives *bounded output*.
- Linear controller – easy to analyse
  - Single-link manipulator under a proportional control scheme.
  - $\Omega(t) \to \Omega_d(t)$ as $t \to \infty$
  - Proportional controller is stable.
  - PD is also stable but PID can become unstable!
- Non-linear controllers – difficult to analyse for stability.
- Stability analysis using Lyapunov's method.
- Controllability.

- Stability – *bounded input* gives *bounded output*.
- Linear controller – easy to analyse
  - Single-link manipulator under a proportional control scheme.
  - $\Omega(t) \rightarrow \Omega_d(t)$ as $t \rightarrow \infty$
  - Proportional controller is stable.
  - PD is also stable but PID can become unstable!
- Non-linear controllers – difficult to analyse for stability.
- Stability analysis using Lyapunov's method.
- Controllability.

- Lyapunov (1892) Russian mathematician.
- One of the few general and widely used result for non-linear systems.
- Non-linear system described in the form

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t)$$

$\mathbf{X} \in \Re^n$, $\mathbf{f}(\mathbf{X}, t)$ are n vector functions, and t denotes the time
- ODE has a unique solution starting at a given initial condition $\mathbf{X}_0$.
- Robot manipulators no *explicit* dependence on t and $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$.

- Lyapunov (1892) Russian mathematician.
- One of the few general and widely used result for non-linear systems.
- Non-linear system described in the form

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t)$$

$\mathbf{X} \in \Re^n$, $\mathbf{f}(\mathbf{X}, t)$ are n vector functions, and t denotes the time
- ODE has a unique solution starting at a given initial condition $\mathbf{X}_0$.
- Robot manipulators no *explicit* dependence on t and $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$.

- Lyapunov (1892) Russian mathematician.
- One of the few general and widely used result for non-linear systems.
- Non-linear system described in the form

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t)$$

$\mathbf{X} \in \Re^n$, $\mathbf{f}(\mathbf{X}, t)$ are n vector functions, and t denotes the time

- ODE has a unique solution starting at a given initial condition $\mathbf{X}_0$.
- Robot manipulators no *explicit* dependence on t and $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$.

- Lyapunov (1892) Russian mathematician.
- One of the few general and widely used result for non-linear systems.
- Non-linear system described in the form

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t)$$

  $\mathbf{X} \in \Re^n$, $\mathbf{f}(\mathbf{X}, t)$ are n vector functions, and t denotes the time
- ODE has a unique solution starting at a given initial condition $\mathbf{X}_0$.
- Robot manipulators no *explicit* dependence on t and $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$.

- Lyapunov (1892) Russian mathematician.
- One of the few general and widely used result for non-linear systems.
- Non-linear system described in the form

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t)$$

  $\mathbf{X} \in \Re^n$, $\mathbf{f}(\mathbf{X}, t)$ are n vector functions, and t denotes the time
- ODE has a unique solution starting at a given initial condition $\mathbf{X}_0$.
- Robot manipulators no *explicit* dependence on t and $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$.

- Stability analysis is performed at *equilibrium* point(s) or state(s).
- $\mathbf{X}_e$ is called an *equilibrium* point or state when it satisfies

$$\mathbf{f}(\mathbf{X}) = \mathbf{0}$$

- $\mathbf{X}_e$ can be solved from n non-linear algebraic/trigonometric equations.
- $\mathbf{f}(\mathbf{X}) = \mathbf{0}$ can have *more than one* solution[8].
- Need to investigate stability at *all* equilibrium points!

[8] In a linear system $\dot{\mathbf{X}} = [A]\mathbf{X}$, only one equilibrium point $\mathbf{X} = \mathbf{0}$ when $[A]$ is a constant non-singular matrix.

Stability analysis using Lyapunov's method

- Stability analysis is performed at *equilibrium* point(s) or state(s).
- $\mathbf{X}_e$ is called an *equilibrium* point or state when it satisfies

$$\mathbf{f}(\mathbf{X}) = \mathbf{0}$$

- $\mathbf{X}_e$ can be solved from n non-linear algebraic/trigonometric equations.
- $\mathbf{f}(\mathbf{X}) = \mathbf{0}$ can have *more than one* solution[8].
- Need to investigate stability at *all* equilibrium points!

[8] In a linear system $\dot{\mathbf{X}} = [A]\mathbf{X}$, only one equilibrium point $\mathbf{X} = \mathbf{0}$ when $[A]$ is a constant non-singular matrix.

Stability analysis using Lyapunov's method

- Stability analysis is performed at *equilibrium* point(s) or state(s).
- $\mathbf{X}_e$ is called an *equilibrium* point or state when it satisfies

$$\mathrm{f}(\mathbf{X}) = \mathbf{0}$$

- $\mathbf{X}_e$ can be solved from n non-linear algebraic/trigonometric equations.
- $\mathrm{f}(\mathbf{X}) = \mathbf{0}$ can have *more than one* solution[8].
- Need to investigate stability at *all* equilibrium points!

[8] In a linear system $\dot{\mathbf{X}} = [A]\mathbf{X}$, only one equilibrium point $\mathbf{X} = \mathbf{0}$ when $[A]$ is a constant non-singular matrix.

- Stability analysis is performed at *equilibrium* point(s) or state(s).
- $\mathbf{X}_e$ is called an *equilibrium* point or state when it satisfies

$$\mathbf{f}(\mathbf{X}) = \mathbf{0}$$

- $\mathbf{X}_e$ can be solved from n non-linear algebraic/trigonometric equations.
- $\mathbf{f}(\mathbf{X}) = \mathbf{0}$ can have *more than one* solution[8].
- Need to investigate stability at *all* equilibrium points!

[8] In a linear system $\dot{\mathbf{X}} = [A]\mathbf{X}$, only one equilibrium point $\mathbf{X} = \mathbf{0}$ when $[A]$ is a constant non-singular matrix.

- Stability analysis is performed at *equilibrium* point(s) or state(s).
- $\mathbf{X}_e$ is called an *equilibrium* point or state when it satisfies

$$\mathbf{f}(\mathbf{X}) = \mathbf{0}$$

- $\mathbf{X}_e$ can be solved from n non-linear algebraic/trigonometric equations.
- $\mathbf{f}(\mathbf{X}) = \mathbf{0}$ can have *more than one* solution[8].
- Need to investigate stability at *all* equilibrium points!

[8] In a linear system $\dot{\mathbf{X}} = [A]\mathbf{X}$, only one equilibrium point $\mathbf{X} = \mathbf{0}$ when $[A]$ is a constant non-singular matrix.

- Statement: A non-linear system $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$ is said to be asymptotically stable (in the sense of Lyapunov) if there exists a *positive-definite*, differentiable, scalar function of the state variables $V(\mathbf{X})$, with $\dot{V}(\mathbf{X})$ being *negative definite*.
- A function $f(\mathbf{x})$ is positive definite if $f(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{0}$ and is zero only when $\mathbf{x} = \mathbf{0}$.
- Positive semi-definite, if $f(\mathbf{x}) \geq 0$ & Negative definite if $f(\mathbf{x}) < 0$.
  - $f(\mathbf{x}) = x_1^2 + x_2^2$ is positive-definite
  - $f(\mathbf{x}) = (x_1 - x_2)^2$ is positive semi-definite, and
  - $f(\mathbf{x} = -(x_1^2 + x_2^2)$ is negative definite.
- Motivation of Lyapunov's theorem: Spring-mass-damper system is *stable*
  - Energy of system is positive-definite
  - Energy continuously *decreases* with time.

- Statement: A non-linear system $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$ is said to be asymptotically stable (in the sense of Lyapunov) if there exists a *positive-definite*, differentiable, scalar function of the state variables $V(\mathbf{X})$, with $\dot{V}(\mathbf{X})$ being *negative definite*.
- A function $f(\mathbf{x})$ is positive definite if $f(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{0}$ and is zero only when $\mathbf{x} = \mathbf{0}$.
- Positive semi-definite, if $f(\mathbf{x}) \geq 0$ & Negative definite if $f(\mathbf{x}) < 0$.
  - $f(\mathbf{x}) = x_1^2 + x_2^2$ is positive-definite
  - $f(\mathbf{x}) = (x_1 - x_2)^2$ is positive semi-definite, and
  - $f(\mathbf{x} = -(x_1^2 + x_2^2)$ is negative definite.
- Motivation of Lyapunov's theorem: Spring-mass-damper system is *stable*
  - Energy of system is positive-definite
  - Energy continuously *decreases* with time.

- Statement: A non-linear system $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$ is said to be asymptotically stable (in the sense of Lyapunov) if there exists a *positive-definite*, differentiable, scalar function of the state variables $V(\mathbf{X})$, with $\dot{V}(\mathbf{X})$ being *negative definite*.
- A function $f(\mathbf{x})$ is positive definite if $f(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{0}$ and is zero only when $\mathbf{x} = \mathbf{0}$.
- Positive semi-definite, if $f(\mathbf{x}) \geq 0$ & Negative definite if $f(\mathbf{x}) < 0$.
  - $f(\mathbf{x}) = x_1^2 + x_2^2$ is positive-definite
  - $f(\mathbf{x}) = (x_1 - x_2)^2$ is positive semi-definite, and
  - $f(\mathbf{x} = -(x_1^2 + x_2^2)$ is negative definite.
- Motivation of Lyapunov's theorem: Spring-mass-damper system is *stable*
  - Energy of system is positive-definite
  - Energy continuously *decreases* with time.

- Statement: A non-linear system $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$ is said to be asymptotically stable (in the sense of Lyapunov) if there exists a *positive-definite*, differentiable, scalar function of the state variables $V(\mathbf{X})$, with $\dot{V}(\mathbf{X})$ being *negative definite*.
- A function $f(\mathbf{x})$ is positive definite if $f(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{0}$ and is zero only when $\mathbf{x} = \mathbf{0}$.
- Positive semi-definite, if $f(\mathbf{x}) \geq 0$ & Negative definite if $f(\mathbf{x}) < 0$.
  - $f(\mathbf{x}) = x_1^2 + x_2^2$ is positive-definite
  - $f(\mathbf{x}) = (x_1 - x_2)^2$ is positive semi-definite, and
  - $f(\mathbf{x} = -(x_1^2 + x_2^2)$ is negative definite.
- Motivation of Lyapunov's theorem: Spring-mass-damper system is *stable*
  - Energy of system is positive-definite
  - Energy continuously *decreases* with time.

- *Sufficient condition* for stability *not a necessary conditions*!.
  - A *single* $V(\mathbf{X}) > 0$ such that $\dot{V}(\mathbf{X}) < 0 \Rightarrow$ Asymptotic stability!
  - For a $V(\mathbf{X}) > 0$, if $\dot{V}(\mathbf{X}) \nless 0 \nRightarrow$ system is *not* stable (or unstable) – Choice of $V(\mathbf{X})$ was not proper!!
- If $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) \leq 0 \rightarrow$ Asymptotic stability under certain conditions (LaSalle and Lefschetz (1961)).
- Local result – $\mathbf{X}_e$ is asymptotically stable if any trajectory *starting in a region around the point* converges to $\mathbf{X}_e$ as $t \rightarrow \infty$ (see Khalil (1992) or Vidyasagar (1993) for a more formal definition). Region of asymptotic stability or *domain of attraction* is more difficult to obtain!
- Lyapunov's method is also applicable for *non-autonomous systems* $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t)$ (see Khalil (1992) and Vidyasagar (1993)).
- Main difficulty – finding appropriate Lyapunov function, $V(\mathbf{X})$.

- *Sufficient condition* for stability *not a necessary conditions*!.
  - A *single* $V(\mathbf{X}) > 0$ such that $\dot{V}(\mathbf{X}) < 0 \Rightarrow$ Asymptotic stability!
  - For a $V(\mathbf{X}) > 0$, if $\dot{V}(\mathbf{X}) \not< 0 \not\Rightarrow$ system is *not* stable (or unstable) – Choice of $V(\mathbf{X})$ was not proper!!
- If $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) \leq 0 \rightarrow$ Asymptotic stability under certain conditions (LaSalle and Lefschetz (1961)).
- Local result – $\mathbf{X}_e$ is asymptotically stable if any trajectory *starting in a region around the point* converges to $\mathbf{X}_e$ as $t \rightarrow \infty$ (see Khalil (1992) or Vidyasagar (1993) for a more formal definition). Region of asymptotic stability or *domain of attraction* is more difficult to obtain!
- Lyapunov's method is also applicable for *non-autonomous systems* $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t)$ (see Khalil (1992) and Vidyasagar (1993)).
- Main difficulty – finding appropriate Lyapunov function, $V(\mathbf{X})$.

- *Sufficient condition* for stability *not a necessary conditions*!.
  - A *single* $V(\mathbf{X}) > 0$ such that $\dot{V}(\mathbf{X}) < 0 \Rightarrow$ Asymptotic stability!
  - For a $V(\mathbf{X}) > 0$, if $\dot{V}(\mathbf{X}) \nless 0 \nRightarrow$ system is *not* stable (or unstable) – Choice of $V(\mathbf{X})$ was not proper!!
- If $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) \leq 0 \rightarrow$ Asymptotic stability under certain conditions (LaSalle and Lefschetz (1961)).
- Local result – $\mathbf{X}_e$ is asymptotically stable if any trajectory *starting in a region around the point* converges to $\mathbf{X}_e$ as $t \rightarrow \infty$ (see Khalil (1992) or Vidyasagar (1993) for a more formal definition). Region of asymptotic stability or *domain of attraction* is more difficult to obtain!
- Lyapunov's method is also applicable for *non-autonomous systems* $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t)$ (see Khalil (1992) and Vidyasagar (1993)).
- Main difficulty – finding appropriate Lyapunov function, $V(\mathbf{X})$.

- *Sufficient condition* for stability *not a necessary conditions*!.
  - A *single* $V(\mathbf{X}) > 0$ such that $\dot{V}(\mathbf{X}) < 0 \Rightarrow$ Asymptotic stability!
  - For a $V(\mathbf{X}) > 0$, if $\dot{V}(\mathbf{X}) \nless 0 \nRightarrow$ system is *not* stable (or unstable) – Choice of $V(\mathbf{X})$ was not proper!!
- If $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) \leq 0 \rightarrow$ Asymptotic stability under certain conditions (LaSalle and Lefschetz (1961)).
- Local result – $\mathbf{X}_e$ is asymptotically stable if any trajectory *starting in a region around the point* converges to $\mathbf{X}_e$ as $t \rightarrow \infty$ (see Khalil (1992) or Vidyasagar (1993) for a more formal definition). Region of asymptotic stability or *domain of attraction* is more difficult to obtain!
- Lyapunov's method is also applicable for *non-autonomous systems* $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t)$ (see Khalil (1992) and Vidyasagar (1993)).
- Main difficulty – finding appropriate Lyapunov function, $V(\mathbf{X})$.

- *Sufficient condition* for stability *not a necessary conditions*!.
  - A *single* $V(\mathbf{X}) > 0$ such that $\dot{V}(\mathbf{X}) < 0 \Rightarrow$ Asymptotic stability!
  - For a $V(\mathbf{X}) > 0$, if $\dot{V}(\mathbf{X}) \nless 0 \nRightarrow$ system is *not* stable (or unstable) – Choice of $V(\mathbf{X})$ was not proper!!
- If $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) \leq 0 \rightarrow$ Asymptotic stability under certain conditions (LaSalle and Lefschetz (1961)).
- Local result – $\mathbf{X}_e$ is asymptotically stable if any trajectory *starting in a region around the point* converges to $\mathbf{X}_e$ as $t \rightarrow \infty$ (see Khalil (1992) or Vidyasagar (1993) for a more formal definition). Region of asymptotic stability or *domain of attraction* is more difficult to obtain!
- Lyapunov's method is also applicable for *non-autonomous systems* $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t)$ (see Khalil (1992) and Vidyasagar (1993)).
- Main difficulty – finding appropriate Lyapunov function, $V(\mathbf{X})$.

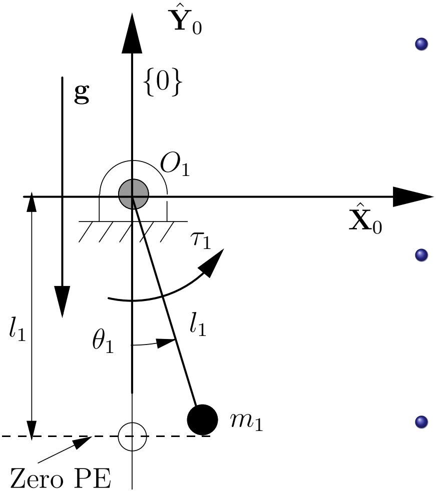


Figure 29: A single link manipulator

- Equation of motion

$$\ddot{\theta}_1 + (g/l_1)\sin\theta_1 = u(t)$$

where $u(t) = \tau_1(t)/(m_1 l_1^2)$ and θ_1 is angle as shown.

- State equation with $(X_1, X_2)^T = (\theta_1, \dot{\theta}_1)^T$

$$\dot{X}_1 = X_2, \quad \dot{X}_2 = u(t) - (g/l_1)\sin(X_1)$$

- Equilibrium points: $u(t) = 0$, $\theta_1 = 0$ and $\theta_1 = \pi$.

- Investigate stability at $\theta_1 = 0$.
- Candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1))$$

- $V(X_1, X_2) = \text{Total Energy} > 0$ & Zero only when $X_1 = X_2 = 0$ – Zero potential energy at $y = -l_1$.
- $\dot{V}(X_1, X_2)$ at equilibrium point $\theta_1 = 0$ is given by

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 \dot{X}_2 + m_1 g l_1 \sin(X_1) \dot{X}_1 = 0$$

- Not asymptotically stable!!

- Investigate stability at $\theta_1 = 0$.
- Candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1))$$

- $V(X_1, X_2) = \text{Total Energy} > 0$ & Zero only when $X_1 = X_2 = 0$ – Zero potential energy at $y = -l_1$.
- $\dot{V}(X_1, X_2)$ at equilibrium point $\theta_1 = 0$ is given by

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 \dot{X}_2 + m_1 g l_1 \sin(X_1) \dot{X}_1 = 0$$

- Not asymptotically stable!!

- Investigate stability at $\theta_1 = 0$.
- Candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1))$$

- $V(X_1, X_2) =$ Total Energy > 0 & Zero only when $X_1 = X_2 = 0$ – Zero potential energy at $y = -l_1$.
- $\dot{V}(X_1, X_2)$ at equilibrium point $\theta_1 = 0$ is given by

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 \dot{X}_2 + m_1 g l_1 \sin(X_1) \dot{X}_1 = 0$$

- Not asymptotically stable!!

Stability analysis using Lyapunov's method

Example: Single link manipulator (contd.)

- Investigate stability at $\theta_1 = 0$.
- Candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1))$$

- $V(X_1, X_2) = \text{Total Energy} > 0$ & Zero only when $X_1 = X_2 = 0$ – Zero potential energy at $y = -l_1$.
- $\dot{V}(X_1, X_2)$ at equilibrium point $\theta_1 = 0$ is given by

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 \dot{X}_2 + m_1 g l_1 \sin(X_1) \dot{X}_1 = 0$$

- Not asymptotically stable!!

- Investigate stability at $\theta_1 = 0$.
- Candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1))$$

- $V(X_1, X_2) = \text{Total Energy} > 0$ & Zero only when $X_1 = X_2 = 0$ – Zero potential energy at $y = -l_1$.
- $\dot{V}(X_1, X_2)$ at equilibrium point $\theta_1 = 0$ is given by

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 \dot{X}_2 + m_1 g l_1 \sin(X_1) \dot{X}_1 = 0$$

- Not asymptotically stable!!

- Consider added damping

$$\dot{X}_1 = X_2, \quad \dot{X}_2 = -(g/l_1)\sin(X_1) - cX_2, \quad c > 0$$

- For above state equations, $\dot{V}(X_1, X_2)$ at $\theta_1 = 0$ is

$$\dot{V}(X_1, X_2) = -m_1 l_1^2 c X_2^2 \; < 0$$

- Single link manipulator *with damping* is asymptotically stable!
- Consider actuator output $\tau_1(t) = K_p(X_{1_d} - X_1), \; K_p > 0$, or $u(t)$ given by

$$u(t) = K_p(X_{1_d} - X_1)/(m_1 l_1^2), \quad K_p > 0$$

where X_{1_d} denotes a desired θ_1.

- Consider added damping

$$\dot{X}_1 = X_2, \quad \dot{X}_2 = -(g/l_1)\sin(X_1) - cX_2, \quad c > 0$$

- For above state equations, $\dot{V}(X_1, X_2)$ at $\theta_1 = 0$ is

$$\dot{V}(X_1, X_2) = -m_1 l_1^2 c X_2^2 \; < 0$$

- Single link manipulator *with damping* is asymptotically stable!
- Consider actuator output $\tau_1(t) = K_p(X_{1_d} - X_1)$, $K_p > 0$, or $u(t)$ given by

$$u(t) = K_p(X_{1_d} - X_1)/(m_1 l_1^2), \quad K_p > 0$$

where X_{1_d} denotes a desired θ_1.

- Consider added damping

$$\dot{X}_1 = X_2, \quad \dot{X}_2 = -(g/l_1)\sin(X_1) - cX_2, \quad c > 0$$

- For above state equations, $\dot{V}(X_1, X_2)$ at $\theta_1 = 0$ is

$$\dot{V}(X_1, X_2) = -m_1 l_1^2 c X_2^2 \ < 0$$

- Single link manipulator *with damping* is asymptotically stable!
- Consider actuator output $\tau_1(t) = K_p(X_{1_d} - X_1)$, $K_p > 0$, or $u(t)$ given by

$$u(t) = K_p(X_{1_d} - X_1)/(m_1 l_1^2), \quad K_p > 0$$

where X_{1_d} denotes a desired θ_1.

- Consider added damping

$$\dot{X}_1 = X_2, \quad \dot{X}_2 = -(g/l_1)\sin(X_1) - cX_2, \quad c > 0$$

- For above state equations, $\dot{V}(X_1, X_2)$ at $\theta_1 = 0$ is

$$\dot{V}(X_1, X_2) = -m_1 l_1^2 c X_2^2 \ < 0$$

- Single link manipulator *with damping* is asymptotically stable!
- Consider actuator output $\tau_1(t) = K_p(X_{1_d} - X_1), \ K_p > 0$, or $u(t)$ given by

$$u(t) = K_p(X_{1_d} - X_1)/(m_1 l_1^2), \quad K_p > 0$$

where X_{1_d} denotes a desired θ_1.

- Investigate stability for $X_{1_d} = 0$[9].
- Choose candidate Lyapunov function as

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- $V(X_1, X_2)$ is positive definite.
- For the *undamped* state equations,

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 u(t) + K_p X_1 X_2$$

- For $u(t) = -K_p X_1 / (m_1 l_1^2) \rightarrow \dot{V}(X_1, X_2) = 0 \Rightarrow$ Not asymptotically stable!
- With damping, $\dot{V}(X_1, X_2) < 0 \Rightarrow$ Asymptotic stability at X_{1_d}.

[9] If $X_{1_d} \neq 0$, perform a change of coordinates $X_1{}' = X_{1_d} - X_1$.

- Investigate stability for $X_{1_d} = 0$[9].
- Choose candidate Lyapunov function as

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- $V(X_1, X_2)$ is positive definite.
- For the *undamped* state equations,

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 u(t) + K_p X_1 X_2$$

- For $u(t) = -K_p X_1/(m_1 l_1^2) \rightarrow \dot{V}(X_1, X_2) = 0 \Rightarrow$ Not asymptotically stable!
- With damping, $\dot{V}(X_1, X_2) < 0 \Rightarrow$ Asymptotic stability at X_{1_d}.

[9] If $X_{1_d} \neq 0$, perform a change of coordinates $X_1{}' = X_{1_d} - X_1$.

- Investigate stability for $X_{1_d} = 0$[9].
- Choose candidate Lyapunov function as

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- $V(X_1, X_2)$ is positive definite.
- For the *undamped* state equations,

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 u(t) + K_p X_1 X_2$$

- For $u(t) = -K_p X_1/(m_1 l_1^2) \rightarrow \dot{V}(X_1, X_2) = 0 \Rightarrow$ Not asymptotically stable!
- With damping, $\dot{V}(X_1, X_2) < 0 \Rightarrow$ Asymptotic stability at X_{1_d}.

[9] If $X_{1_d} \neq 0$, perform a change of coordinates $X_1{}' = X_{1_d} - X_1$.

- Investigate stability for $X_{1_d} = 0$[9].
- Choose candidate Lyapunov function as

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- $V(X_1, X_2)$ is positive definite.
- For the *undamped* state equations,

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 u(t) + K_p X_1 X_2$$

- For $u(t) = -K_p X_1 / (m_1 l_1^2) \rightarrow \dot{V}(X_1, X_2) = 0 \Rightarrow$ Not asymptotically stable!
- With damping, $\dot{V}(X_1, X_2) < 0 \Rightarrow$ Asymptotic stability at X_{1_d}.

[9] If $X_{1_d} \neq 0$, perform a change of coordinates $X_1{}' = X_{1_d} - X_1$.

EXAMPLE: SINGLE LINK MANIPULATOR (CONTD.)

- Investigate stability for $X_{1_d} = 0$[9].
- Choose candidate Lyapunov function as

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- $V(X_1, X_2)$ is positive definite.
- For the *undamped* state equations,

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 u(t) + K_p X_1 X_2$$

- For $u(t) = -K_p X_1 / (m_1 l_1^2) \rightarrow \dot{V}(X_1, X_2) = 0 \Rightarrow$ Not asymptotically stable!
- With damping, $\dot{V}(X_1, X_2) < 0 \Rightarrow$ Asymptotic stability at X_{1_d}.

[9] If $X_{1_d} \neq 0$, perform a change of coordinates $X_1{}' = X_{1_d} - X_1$.

- Investigate stability for $X_{1_d} = 0$[9].
- Choose candidate Lyapunov function as

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- $V(X_1, X_2)$ is positive definite.
- For the *undamped* state equations,

$$\dot{V}(X_1, X_2) = m_1 l_1^2 X_2 u(t) + K_p X_1 X_2$$

- For $u(t) = -K_p X_1 / (m_1 l_1^2) \rightarrow \dot{V}(X_1, X_2) = 0 \Rightarrow$ Not asymptotically stable!
- With damping, $\dot{V}(X_1, X_2) < 0 \Rightarrow$ Asymptotic stability at X_{1_d}.

[9] If $X_{1_d} \neq 0$, perform a change of coordinates $X_1{}' = X_{1_d} - X_1$.

Example: Single link manipulator (contd.)

- Consider a (modified) proportional plus derivative (PD) control

$$\tau_1(t) = -K_p X_1 - K_v \dot{X}_1, \;\; K_p, K_v > 0$$

- Consider the candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- For the undamped system $\dot{V}(X_1, X_2) = -K_v X_2^2 < 0 \Rightarrow$ Asymptotically stable!
- $\dot{X}_{1_d}$ is *assumed* zero – Cannot prove asymptotic stability for trajectory following when $\dot{X}_{1_d}$ is non-zero!
- *Not possible* to prove stability for second equilibrium point $\theta_1 = \pi$ using Lyapunov's second method.
- Recall $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) < 0$ is a *sufficient condition for stability* – $\theta_1 = \pi$ is known to be unstable!!

- Consider a (modified) proportional plus derivative (PD) control

$$\tau_1(t) = -K_p X_1 - K_v \dot{X}_1, \;\; K_p, K_v > 0$$

- Consider the candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- For the undamped system $\dot{V}(X_1, X_2) = -K_v X_2^2 < 0 \Rightarrow$ Asymptotically stable!
- $\dot{X}_{1_d}$ is *assumed* zero – Cannot prove asymptotic stability for trajectory following when $\dot{X}_{1_d}$ is non-zero!
- *Not possible* to prove stability for second equilibrium point $\theta_1 = \pi$ using Lyapunov's second method.
- Recall $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) < 0$ is a *sufficient condition for stability* – $\theta_1 = \pi$ is known to be unstable!!

- Consider a (modified) proportional plus derivative (PD) control

$$\tau_1(t) = -K_p X_1 - K_v \dot{X}_1, \;\; K_p, K_v > 0$$

- Consider the candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- For the undamped system $\dot{V}(X_1, X_2) = -K_v X_2^2 < 0 \Rightarrow$ Asymptotically stable!
- $\dot{X}_{1_d}$ is *assumed* zero – Cannot prove asymptotic stability for trajectory following when $\dot{X}_{1_d}$ is non-zero!
- *Not possible* to prove stability for second equilibrium point $\theta_1 = \pi$ using Lyapunov's second method.
- Recall $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) < 0$ is a *sufficient condition for stability* – $\theta_1 = \pi$ is known to be unstable!!

Example: Single link manipulator (contd.)

- Consider a (modified) proportional plus derivative (PD) control

$$\tau_1(t) = -K_p X_1 - K_v \dot{X}_1, \;\; K_p, K_v > 0$$

- Consider the candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- For the undamped system $\dot{V}(X_1, X_2) = -K_v X_2^2 < 0 \Rightarrow$ Asymptotically stable!
- $\dot{X}_{1_d}$ is *assumed* zero – Cannot prove asymptotic stability for trajectory following when $\dot{X}_{1_d}$ is non-zero!
- *Not possible* to prove stability for second equilibrium point $\theta_1 = \pi$ using Lyapunov's second method.
- Recall $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) < 0$ is a *sufficient condition for stability* – $\theta_1 = \pi$ is known to be unstable!!

- Consider a (modified) proportional plus derivative (PD) control

$$\tau_1(t) = -K_p X_1 - K_v \dot{X}_1, \;\; K_p, K_v > 0$$

- Consider the candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- For the undamped system $\dot{V}(X_1, X_2) = -K_v X_2^2 < 0 \Rightarrow$ Asymptotically stable!
- $\dot{X}_{1_d}$ is *assumed* zero – Cannot prove asymptotic stability for trajectory following when $\dot{X}_{1_d}$ is non-zero!
- *Not possible* to prove stability for second equilibrium point $\theta_1 = \pi$ using Lyapunov's second method.
- Recall $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) < 0$ is a *sufficient condition for stability* – $\theta_1 = \pi$ is known to be unstable!!

EXAMPLE: SINGLE LINK MANIPULATOR (CONTD.)

- Consider a (modified) proportional plus derivative (PD) control

$$\tau_1(t) = -K_p X_1 - K_v \dot{X}_1,\ \ K_p, K_v > 0$$

- Consider the candidate Lyapunov function

$$V(X_1, X_2) = \frac{1}{2} m_1 (l_1 X_2)^2 + m_1 g l_1 (1 - \cos(X_1)) + \frac{1}{2} K_p X_1^2$$

- For the undamped system $\dot{V}(X_1, X_2) = -K_v X_2^2 < 0 \Rightarrow$ Asymptotically stable!
- $\dot{X}_{1_d}$ is *assumed* zero – Cannot prove asymptotic stability for trajectory following when $\dot{X}_{1_d}$ is non-zero!
- *Not possible* to prove stability for second equilibrium point $\theta_1 = \pi$ using Lyapunov's second method.
- Recall $V(\mathbf{X}) > 0$ and $\dot{V}(\mathbf{X}) < 0$ is a *sufficient condition for stability* – $\theta_1 = \pi$ is known to be unstable!!

- Equations of motion of n-DOF manipulator *without* gravity

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})]\dot{\mathbf{q}}$$

- Consider a PD control of the form $\tau = -[K_p]\mathbf{q}(t) - [K_v]\dot{\mathbf{q}}(t)$. Note: $\dot{\mathbf{q}}_d(t) = 0$ and $\mathbf{q}_d = \mathbf{0}$[10].
- Consider a candidate Lyapunov function

$$V(\mathbf{q},\dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\dot{\mathbf{q}} + \frac{1}{2}\mathbf{q}^T[K_p]\mathbf{q}$$

- Evaluate $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ to get

$$\begin{aligned}\dot{V}(\mathbf{q},\dot{\mathbf{q}}) &= \dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T[\dot{\mathbf{M}}(\mathbf{q})]\dot{\mathbf{q}} + \dot{\mathbf{q}}^T[K_p]\mathbf{q} \\ &= -\dot{\mathbf{q}}^T[K_v]\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T([\dot{\mathbf{M}}(\mathbf{q})] - 2[\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})])\dot{\mathbf{q}}\end{aligned}$$

$[\dot{\mathbf{M}}]$ denotes the derivative of $[\mathbf{M}]$ with respect to time.

[10] Setting $\mathbf{q}_d = \mathbf{0}$ is not a serious issue – perform a linear transformation $\mathbf{q}' = \mathbf{q}_d - \mathbf{q}$ and investigate the stability at $\mathbf{q}' = 0$.

- Equations of motion of n-DOF manipulator *without* gravity

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})]\dot{\mathbf{q}}$$

- Consider a PD control of the form $\tau = -[K_p]\mathbf{q}(t) - [K_v]\dot{\mathbf{q}}(t)$. Note: $\dot{\mathbf{q}}_d(t) = 0$ and $\mathbf{q}_d = \mathbf{0}$[10].
- Consider a candidate Lyapunov function

$$V(\mathbf{q},\dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\dot{\mathbf{q}} + \frac{1}{2}\mathbf{q}^T[K_p]\mathbf{q}$$

- Evaluate $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ to get

$$\begin{aligned}\dot{V}(\mathbf{q},\dot{\mathbf{q}}) &= \dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T[\dot{\mathbf{M}}(\mathbf{q})]\dot{\mathbf{q}} + \dot{\mathbf{q}}^T[K_p]\mathbf{q} \\ &= -\dot{\mathbf{q}}^T[K_v]\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T([\dot{\mathbf{M}}(\mathbf{q})] - 2[\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})])\dot{\mathbf{q}}\end{aligned}$$

$[\dot{\mathbf{M}}]$ denotes the derivative of $[\mathbf{M}]$ with respect to time.

[10] Setting $\mathbf{q}_d = \mathbf{0}$ is not a serious issue – perform a linear transformation $\mathbf{q}' = \mathbf{q}_d - \mathbf{q}$ and investigate the stability at $\mathbf{q}' = 0$.

- Equations of motion of n-DOF manipulator *without* gravity

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})]\dot{\mathbf{q}}$$

- Consider a PD control of the form $\tau = -[K_p]\mathbf{q}(t) - [K_v]\dot{\mathbf{q}}(t)$. Note: $\dot{\mathbf{q}}_d(t) = 0$ and $\mathbf{q}_d = \mathbf{0}$[10].
- Consider a candidate Lyapunov function

$$V(\mathbf{q},\dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\dot{\mathbf{q}} + \frac{1}{2}\mathbf{q}^T[K_p]\mathbf{q}$$

- Evaluate $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ to get

$$\begin{aligned}\dot{V}(\mathbf{q},\dot{\mathbf{q}}) &= \dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T[\dot{\mathbf{M}}(\mathbf{q})]\dot{\mathbf{q}} + \dot{\mathbf{q}}^T[K_p]\mathbf{q} \\ &= -\dot{\mathbf{q}}^T[K_v]\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T([\dot{\mathbf{M}}(\mathbf{q})] - 2[\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})])\dot{\mathbf{q}}\end{aligned}$$

$[\dot{\mathbf{M}}]$ denotes the derivative of $[\mathbf{M}]$ with respect to time.

[10] Setting $\mathbf{q}_d = \mathbf{0}$ is not a serious issue – perform a linear transformation $\mathbf{q}' = \mathbf{q}_d - \mathbf{q}$ and investigate the stability at $\mathbf{q}' = 0$.

- Equations of motion of n-DOF manipulator *without* gravity

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}}$$

- Consider a PD control of the form $\tau = -[K_p]\mathbf{q}(t) - [K_v]\dot{\mathbf{q}}(t)$. Note: $\dot{\mathbf{q}}_d(t) = 0$ and $\mathbf{q}_d = \mathbf{0}$[10].
- Consider a candidate Lyapunov function

$$V(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\dot{\mathbf{q}} + \frac{1}{2}\mathbf{q}^T[K_p]\mathbf{q}$$

- Evaluate $\dot{V}(\mathbf{q}, \dot{\mathbf{q}})$ to get

$$\begin{aligned}\dot{V}(\mathbf{q}, \dot{\mathbf{q}}) &= \dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T[\dot{\mathbf{M}}(\mathbf{q})]\dot{\mathbf{q}} + \dot{\mathbf{q}}^T[K_p]\mathbf{q} \\ &= -\dot{\mathbf{q}}^T[K_v]\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T([\dot{\mathbf{M}}(\mathbf{q})] - 2[\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})])\dot{\mathbf{q}}\end{aligned}$$

$[\dot{\mathbf{M}}]$ denotes the derivative of $[\mathbf{M}]$ with respect to time.

[10] Setting $\mathbf{q}_d = \mathbf{0}$ is not a serious issue – perform a linear transformation $\mathbf{q}' = \mathbf{q}_d - \mathbf{q}$ and investigate the stability at $\mathbf{q}' = 0$.

- $([\dot{\mathbf{M}}(\mathbf{q})] - 2[\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})])$ is skew-symmetric → the second quadratic form is zero, and

$$\dot{V}(\mathbf{q},\dot{\mathbf{q}}) = -\dot{\mathbf{q}}^T[K_v]\dot{\mathbf{q}}$$

- Since $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ can be zero even for non-zero $\mathbf{q}$, $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ is negative semi-definite
- By LaSalle's invariance principle (LaSalle and Lefschetz 1961) → equilibrium point $(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{0}$ is asymptotically stable.
- Assumption: $\ddot{\mathbf{q}}_d = \dot{\mathbf{q}}_d = \mathbf{0}$ → PD control scheme is *not* proved asymptotically stable for trajectory following!

- $([\dot{\mathbf{M}}(\mathbf{q})] - 2[\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})])$ is skew-symmetric $\rightarrow$ the second quadratic form is zero, and

$$\dot{V}(\mathbf{q},\dot{\mathbf{q}}) = -\dot{\mathbf{q}}^T [K_v]\dot{\mathbf{q}}$$

- Since $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ can be zero even for non-zero $\mathbf{q}$, $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ is negative semi-definite
- By LaSalle's invariance principle (LaSalle and Lefschetz 1961) $\rightarrow$ equilibrium point $(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{0}$ is asymptotically stable.
- Assumption: $\ddot{\mathbf{q}}_d = \dot{\mathbf{q}}_d = \mathbf{0}$ $\rightarrow$ PD control scheme is *not* proved asymptotically stable for trajectory following!

- $([\dot{\mathbf{M}}(\mathbf{q})] - 2[\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})])$ is skew-symmetric $\rightarrow$ the second quadratic form is zero, and
$$\dot{V}(\mathbf{q},\dot{\mathbf{q}}) = -\dot{\mathbf{q}}^T[K_v]\dot{\mathbf{q}}$$
- Since $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ can be zero even for non-zero $\mathbf{q}$, $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ is negative semi-definite
- By LaSalle's invariance principle (LaSalle and Lefschetz 1961) $\rightarrow$ equilibrium point $(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{0}$ is asymptotically stable.
- Assumption: $\ddot{\mathbf{q}}_d = \dot{\mathbf{q}}_d = \mathbf{0}$ $\rightarrow$ PD control scheme is *not* proved asymptotically stable for trajectory following!

- $([\dot{\mathbf{M}}(\mathbf{q})] - 2[\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})])$ is skew-symmetric $\rightarrow$ the second quadratic form is zero, and
$$\dot{V}(\mathbf{q},\dot{\mathbf{q}}) = -\dot{\mathbf{q}}^T[K_v]\dot{\mathbf{q}}$$
- Since $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ can be zero even for non-zero $\mathbf{q}$, $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ is negative semi-definite
- By LaSalle's invariance principle (LaSalle and Lefschetz 1961) $\rightarrow$ equilibrium point $(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{0}$ is asymptotically stable.
- Assumption: $\ddot{\mathbf{q}}_d = \dot{\mathbf{q}}_d = \mathbf{0}$ $\rightarrow$ PD control scheme is *not* proved asymptotically stable for trajectory following!

- Very little about stability can be proved!
- PD and exact gravity cancellation

$$\tau = -[K_p]\mathbf{q}(t) - [K_v]\dot{\mathbf{q}}(t) + \mathbf{G}(\mathbf{q})$$

  equilibrium point $(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{0}$ is stable!
- Computed torque with *exact cancellation*: Error equation

$$\ddot{e}_i + K_{v_i}\dot{e}_i + K_{p_i}e = 0, \quad i = 1, ..., n$$

  damped second-order linear ODE's → asymptotically stable!
- Stability analysis of non-linear control systems is *unsolved* problem!
- In Module 10, Lecture 1, possibility of *chaotic* motions shown for trajectory following.

- Very little about stability can be proved!
- PD and exact gravity cancellation

$$\tau = -[K_p]\mathbf{q}(t) - [K_v]\dot{\mathbf{q}}(t) + \mathbf{G}(\mathbf{q})$$

  equilibrium point $(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{0}$ is stable!

- Computed torque with *exact cancellation*: Error equation

$$\ddot{e}_i + K_{v_i}\dot{e}_i + K_{p_i}e = 0, \quad i = 1, ..., n$$

  damped second-order linear ODE's $\rightarrow$ asymptotically stable!

- Stability analysis of non-linear control systems is *unsolved* problem!
- In Module 10, Lecture 1, possibility of *chaotic* motions shown for trajectory following.

- Very little about stability can be proved!
- PD and exact gravity cancellation

$$\tau = -[K_p]\mathbf{q}(t) - [K_v]\dot{\mathbf{q}}(t) + \mathbf{G}(\mathbf{q})$$

equilibrium point $(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{0}$ is stable!

- Computed torque with *exact cancellation*: Error equation

$$\ddot{e}_i + K_{v_i}\dot{e}_i + K_{p_i}e = 0, \quad i = 1, ..., n$$

damped second-order linear ODE's → asymptotically stable!

- Stability analysis of non-linear control systems is *unsolved* problem!
- In Module 10, Lecture 1, possibility of *chaotic* motions shown for trajectory following.

- Very little about stability can be proved!
- PD and exact gravity cancellation

$$\tau = -[K_p]\mathbf{q}(t) - [K_v]\dot{\mathbf{q}}(t) + \mathbf{G}(\mathbf{q})$$

equilibrium point $(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{0}$ is stable!
- Computed torque with *exact cancellation*: Error equation

$$\ddot{e}_i + K_{v_i}\dot{e}_i + K_{p_i}e = 0, \quad i = 1, ..., n$$

damped second-order linear ODE's → asymptotically stable!
- Stability analysis of non-linear control systems is *unsolved* problem!
- In Module 10, Lecture 1, possibility of *chaotic* motions shown for trajectory following.

- Very little about stability can be proved!
- PD and exact gravity cancellation

$$\tau = -[K_p]\mathbf{q}(t) - [K_v]\dot{\mathbf{q}}(t) + \mathbf{G}(\mathbf{q})$$

  equilibrium point $(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{0}$ is stable!
- Computed torque with *exact cancellation*: Error equation

$$\ddot{e}_i + K_{v_i}\dot{e}_i + K_{p_i}e = 0, \quad i = 1, ..., n$$

  damped second-order linear ODE's $\rightarrow$ asymptotically stable!
- Stability analysis of non-linear control systems is *unsolved* problem!
- In Module 10, Lecture 1, possibility of *chaotic* motions shown for trajectory following.

Advanced topics in control

- Lack of knowledge of model parameters
  - No "exact" cancellation and difficult to predict evolution of error $e(t)$.
  - Model parameters can be obtained using *adaptive* control schemes (see Craig (1988), Ortega and Spong(1989) for more on adaptive control schemes for robots).
- Mathematical notion of *controllability* of a system.
  - A system $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$ is said to be *controllable* if it is possible to transfer the system from any initial state $\mathbf{X}(0)$ to any desired state $\mathbf{X}(t_f)$ in finite time t_f by the application of the control input $\mathbf{u}(t)$.
  - In a linear system (n state variables and m inputs)

$$\dot{\mathbf{X}} = [F]\mathbf{X} + [G]\mathbf{u}$$

the system is controllable if the controllability matrix

$$[Q_c] = [\ [G] | [F][G] | [F]^2 G | ... | [F]^{n-1}[G]\ ]$$

has rank n.
  - A non-linear robot is *not* controllable at a singularity!

- Lack of knowledge of model parameters
  - No "exact" cancellation and difficult to predict evolution of error $e(t)$.
  - Model parameters can be obtained using *adaptive* control schemes (see Craig (1988), Ortega and Spong(1989) for more on adaptive control schemes for robots).
- Mathematical notion of *controllability* of a system.
  - A system $\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X})$ is said to be *controllable* if it is possible to transfer the system from any initial state $\mathbf{X}(0)$ to any desired state $\mathbf{X}(t_f)$ in finite time t_f by the application of the control input $\mathbf{u}(t)$.
  - In a linear system (n state variables and m inputs)

$$\dot{\mathbf{X}} = [F]\mathbf{X} + [G]\mathbf{u}$$

    the system is controllable if the controllability matrix

$$[Q_c] = [\ [G] | [F][G] | [F]^2 G | \ldots | [F]^{n-1}[G]\ ]$$

    has rank n.
  - A non-linear robot is *not* controllable at a singularity!

Outline

- Exercise Problems
- References & Suggested Reading