Lecture notes: Trajectory tracking control for robot manipulators

Jie Fu

Department of Electrical and Computer Engineering
Robotics Engineering Program
Worcester Polytechnic Institute

RBE502, 2018

Outline

This lecture note is based on

- Chapter 8 in M. Spong **Robot modeling and control.**
- Chapter ~~5:~~ 4.5 Position Control and Trajectory Tracking of Murray et. al. **A Mathematical Introduction to Robotic Manipulation**

Inverse dynamics control

Consider the dynamic model

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q) = \tau.$$

Question: How to design trajectory tracking control?

- Recall: Trajectory tracking control for

$$\dot{x} = Ax + Bu$$

- Consider use it for nonlinear trajectory tracking control: What is the linear system now?

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q) = \tau.$$

Let $\tau = M(q)a_q + C(q,\dot{q})\dot{q} + N(q)$.
Since the inertia matrix is invertible, the system dynamical model reduces to

$$M(q)\ddot{q} + \cancel{C(q,\dot{q})\dot{q}} + \cancel{N(q)} = M(q)a_q + \cancel{C(q,\dot{q})\dot{q}} + \cancel{N(q)}$$

$\because M(q)$ invertible $\Rightarrow M(q)\ddot{q} = M(q)a_q$

$M^{-1}(q)$ exists

$\therefore \ddot{q} = a_q$

Inverse dynamic control

Given

$$\ddot{q} = a_q$$

decoupled joint dynamics.

Write into the state space form:

$$x_1 = q \qquad \Rightarrow \qquad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ a_q \end{bmatrix}$$
$$x_2 = \dot{q}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\text{state}} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \underbrace{a_q}_{\text{input}} \rightarrow u$$

$$\dot{x} = Ax + Bu$$

traj. generation $\leftarrow \quad \dot{x}^d = Ax^d + Bu^d$

Inverse dynamic control

Let $q^d(t)$ be the desired trajectory (cont. differentiable at least twice) the control input a_q should be

$$M(q^d)\ddot{q}^d + C(q^d,\dot{q}^d)\dot{q}^d + N(q^d) = \tau^d$$

$$x^d = \begin{bmatrix} x_1^d \\ x_2^d \end{bmatrix} \qquad x_1^d = q^d \qquad x_2^d = \dot{q}^d \qquad \dot{x}_2^d = \ddot{q}^d$$

$$u^d = a_q^d$$

$$\tau^d = M(q^d)\,a_q^d + C(\,)\dot{q}^d + N(\,)$$

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a_q \; ; \quad \dot{x}^d = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x^d + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a_q^d$$

$$a_q = -K(x - x^d) + a_q^d$$

$$a_q = -\begin{bmatrix} K_1 & K_2 \end{bmatrix}\begin{bmatrix} x_1 - x_1^d \\ x_2 - x_2^d \end{bmatrix} + a_q^d$$

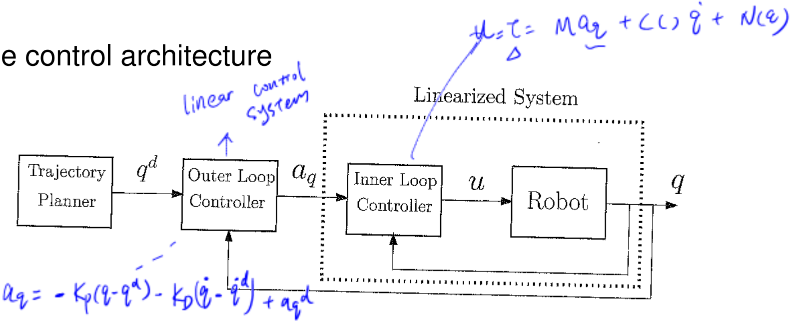
$$= -\begin{bmatrix} K_p & K_D \end{bmatrix}\begin{bmatrix} q - q^d \\ \dot{q} - \dot{q}^d \end{bmatrix} + a_q^d$$

$$a_q = \underbrace{- K_p \underbrace{(q - q^d)}_{\text{position error}} - K_D \underbrace{(\dot{q} - \dot{q}^d)}_{\text{vel. error}}}_{\text{PD control}} + a_q^d$$

$$\tau = M a_q + C(\;)\dot{q} + N(q)$$

Inverse dynamic control

the control architecture



- Outer: input $q^d, \dot{q}^d, q, \dot{q}$, output a_q
- Inner: input $q, \dot{q}, a_q$, and output u (or τ).

Flat output: $z \to x, \dot{x}, \underline{u}$

$q \to q, \dot{q}, a_q = \ddot{q}$

Computed Torque control: idea

Inverse dynamic control is also known as computed torque control, as it can be written as:
Since q and $\dot{q}$ are measured, we modify the input(torques) to be

$$\tau = M(q)\ddot{q}_d + C(q,\dot{q})\dot{q} + N(q,\dot{q})$$

And thus if our model is accurate.

$$M(q)\ddot{q}_d + C(q,\dot{q})\dot{q} + N(q,\dot{q}) = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q,\dot{q})$$

i.e.

$$\ddot{q} = \ddot{q}_d$$

Again, with accurate initial state and model, $q(t) = q_d(t)$ for all time $t \geq 0$.

Computed Torque control

With inaccurate initial state and models, the **mismatch** between our current state q and the desired state q_d is

$$e = q - q_d$$

The control objective is to achieve asymptotical stability: $\lim_{t\to\infty} e(t) = 0$.

$$\dot{e} = \dot{q} - \dot{q}^d$$
$$\ddot{e} = \ddot{q} - \ddot{q}^d$$

$$M\ddot{e} + C(q,\dot{q})\dot{q} + N(q) = \tau - M\ddot{q}^d \quad \ast$$

$$M(\ddot{q} - \ddot{q}^d) + C\dot{q} + N = \tau - M\ddot{q}^d$$

$$\ast: \quad \ddot{e} = M^{-1}(q)(\tau - C(\,)\dot{q} - N(q) - M\ddot{q}^d) \quad \text{-- } ①$$

$$= v$$

$$\boxed{\ddot{e} = v}$$

$$v = -K_P e - K_D \dot{e}$$

using ① = v

$$\tau = \underbrace{M(q)(-K_P e - K_D \dot{e})}_{\text{Feedback}} + \underbrace{M\ddot{q}^d + C(q,\dot{q})\dot{q} + N(q)}_{\text{Feed forward term.}}$$

Computed Torque control

With inaccurate initial state and models, the **mismatch** between our current state q and the desired state q_d is

$$e = q - q_d$$

The control objective is to achieve asymptotical stability: $\lim_{t\to\infty} e(t) = 0$.

- What is the dynamics of the error state?

$$\dot{e} = \dot{q} - \dot{q}_d$$
$$\ddot{e} = \ddot{q} - \ddot{q}_d$$

- What is the relation between input torques and the error states? Since by the dynamic model,

$$M(q)\ddot{q} = \tau - C(q,\dot{q})\dot{q} - N(q,\dot{q})$$

  we have $M(q)\ddot{e} = \tau - C(q,\dot{q})\dot{q} - N(q,\dot{q}) - M(q)\ddot{q}_d$.

Computed Torque control via feedback linearization

$M(q)\ddot{e} = \tau - C(q,\dot{q})\dot{q} - N(q,\dot{q}) - M(q)\ddot{q}_d$.
Consider the input $v = M(q)^{-1}(\tau - C(q,\dot{q})\dot{q} - N(q,\dot{q}) - M(q)\ddot{q}_d)$.
Let $x_1 = e$ and $x_2 = \dot{e}$, reduce to a linear system

Overview

Inverse dynamic control is also known as computed torque control, as it can be written as:

- Computed Torque Control.

if $e = \dot{e} = 0 \Rightarrow \tau \Rightarrow \ddot{q} = \ddot{q}_d$

$$\tau = \underbrace{M(q)(-K_P e - K_D \dot{e})}_{\text{Feedback control}} + \underbrace{C(q,\dot{q})\dot{q} + N(q,\dot{q}) + M(q)\ddot{q}_d}_{\text{Feedforward compensation}}$$

- Inverse dynamic control:

$$\tau = M(q)a_q + C(q,\dot{q})\dot{q} + N(q,\dot{q})$$

$\ddot{q} = a_q^d$

and

$$a_q = -K_P e - K_D \dot{e} + a_q^d$$

and $a_q^d = \ddot{q}^d$

$\tau = M(q)(-K_p e - K_D \dot{e}) + M(q)\, a_q^d + C(\cdot)\dot{q} + N(q)$

Exponential tracking with PD control

- Less demanding PD with gravity compensation realizes asymptotic stability for a setpoint control.

$$\tau = N(q, \dot{q}) - K_D \dot{e} - K_P e.$$

Exponential tracking with PD control

- Less demanding PD with gravity compensation realizes asymptotic stability for a setpoint control.

$$\tau = N(q,\dot{q}) - K_D\dot{e} - K_P e.$$

- Whileas computed Torque control

$$\tau = \underbrace{M(q)(-K_P e - K_D\dot{e})}_{\text{Feedback control}} + \underbrace{C(q,\dot{q})\dot{q} + N(q,\dot{q}) + M(q)\ddot{q}_d}_{\text{Feedforward compensation}}$$

realizes exponential trajectory tracking.

Exponential tracking with PD control

- Less demanding PD with gravity compensation realizes asymptotic stability for a setpoint control.

$$\tau = N(q, \dot{q}) - K_D \dot{e} - K_P e.$$

- Whileas computed Torque control

$$\tau = \underbrace{M(q)(-K_P e - K_D \dot{e})}_{\text{Feedback control}} + \underbrace{C(q, \dot{q})\dot{q} + N(q, \dot{q}) + M(q)\ddot{q}_d}_{\text{Feedforward compensation}}$$

realizes exponential trajectory tracking.

Remove $M(q)$ $\Rightarrow$ $(-K_P e - K_D \dot{e}) + C(\ \ )\dot{q} + N(q) + M(q)\ddot{q}_d$

Augmented PD control law

$$\tau = \underbrace{(-K_P e - K_D \dot{e} + N(q, \dot{q}))}_{\text{PD control w gravity compensation}} + \underbrace{M(q)\ddot{q}_d + C(q, \dot{q})\dot{q}_d}_{\text{Partial feedforward}}$$

Realizes **exponential** trajectory tracking (when $\dot{q}_d = 0$, setpoint stabilization).

Proof

The error dynamics with the control input:

$$\tau = \underbrace{(-K_P e - K_D \dot{e} + N(q, \cancel{\dot{q}}))}_{\text{PD control w gravity compensation}} + \underbrace{M(q)\ddot{q}_d + C(q,\dot{q})\dot{q}_d}_{\text{Partial feedforward}}$$

$$M\ddot{q} + C\dot{q} + N(q) = \tau$$

$$M(q)\ddot{q} + C\dot{q} + \cancel{N(q)} = -K_p e - K_D \dot{e} + \cancel{N(q)} + M\ddot{q}_d + C\dot{q}_d$$

$$M(\ddot{q} - \ddot{q}^d) + C(\dot{q} - \dot{q}^d) = -K_p e - K_D \dot{e}$$

$$M\ddot{e} + C\dot{e} = -K_p e - K_D \dot{e}$$

Asy. stability : $\lim_{t \to \infty} e = 0$

Exponential stability of augmented PD

Proof: Hint:

$$V = \underbrace{\frac{1}{2}\dot{e}^T M(q)\dot{e} + \frac{1}{2}e^T K_P e}_{\text{Set point tracking.}} + \underline{\underline{\varepsilon e^T M(q)\dot{e}}}.$$

$\varepsilon > 0$

① V is lyapunov candidate

② $\dot{V}$ neg. def.

idea: ε is arbitrarily small

$\| M(q) \| < \lambda$

To show that: $V \geq 0$ by a choise of ε and using the upper bound $\|M(q)\|$.

Workspace control

Suppose given a path $x_d(t) \in SE(3)$: desirable end-effector configuration as a function of time.

To apply joint space control (centralized or decentralized)

1. Solve $q_d(t)$ from $x_d(t)$ using inverse kinematics.
2. Design joint space controller.

Workspace control

Suppose given a path $x_d(t) \in SE(3)$: desirable end-effector configuration as a function of time.

To apply joint space control (centralized or decentralized)

1. Solve $q_d(t)$ from $x_d(t)$ using inverse kinematics.
   - time consuming.
   - hard to reason about the trajectory of end-effector with the trajectory of q — may cause undesirable behavior.
2. Design joint space controller.

Workspace control

Suppose given a path $x_d(t) \in SE(3)$: desirable end-effector configuration as a function of time.

To apply joint space control (centralized or decentralized)

1. Solve $q_d(t)$ from $x_d(t)$ using inverse kinematics.
    - time consuming.
    - hard to reason about the trajectory of end-effector with the trajectory of q — may cause undesirable behavior.
2. Design joint space controller.

To overcome these,

Direct workspace control.

Recap

Forward kinematics: A smooth and invertible mapping f from the joint variables to the workspace variables

$$x = f(q)$$

The Jacobian

$$\dot{x} = J(q)\dot{q}, \quad J(q) = \frac{\partial f}{\partial q}$$

Recap

Forward kinematics: A smooth and invertible mapping f from the joint variables to the workspace variables

$$x = f(q)$$

The Jacobian

$$\dot{x} = J(q)\dot{q}, \quad J(q) = \frac{\partial f}{\partial q}$$

Given f is smooth and invertible,

$$\dot{q} = J^{-1}\dot{x}, \quad \ddot{q} = J^{-1}\ddot{x} + \frac{d}{dt}(J^{-1})\dot{x}.$$

The dynamic of robot in the joint space

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q,\dot{q}) = \tau$$

Taskspace control

Suppose given a path $X_d(t) \in \mathbb{R}^6$: desirable end-effector configuration as a function of time using any minimal representation of $SO(3)$.

To apply joint space control (centralized or decentralized)

1. Solve $q^d(t)$ from $X^d(t)$ using inverse kinematics.
   - time consuming.
   - hard to reason about the trajectory of end-effector with the trajectory of q — may cause undesirable behavior.
2. Design joint space controller.

To overcome these, Direct task space control.

From joint space to task space

State variables in

- Joint space: q
- Work space: X.

X is a function of q, we have

$$\dot{X} = J(q)\dot{q}$$

where $J(q)$ is the **analytical** Jacobian. Using inverse dynamics

control:

Summary

- Show Lyapunov stability of decentralized PD control for plannar manipulators.
- PD+Gravity compensation for set-point tracking.
- Inverse dynamics control for trajectory tracking.
- Inverse dynamics control in task space.

Cons:

- Inverse dynamics requires exact knowledge of model dynamics.
- PERFORMANCE is **NOT GUARANTEED** when
  - parameter becomes uncertain.
  - robot picking up an unknown load.

Next: Robust and adaptive control.