MEE5114 Advanced Control for Robotics

Lecture 10: Robot Motion Control

Prof. Wei Zhang

SUSTech Insitute of Robotics
Department of Mechanical and Energy Engineering
Southern University of Science and Technology, Shenzhen, China

Outline

- Motion Control Problems
- Motion Control with Velocity/Acceleration as Input
- Motion Control with Torque as Input
- Task-Space Error in SO(3)/SE(3)

Robot Motion Control Problems (1/1)

- Dynamic equation of fully-actuated robot (without external force):

$$\begin{cases} \tau &= M(\theta)\ddot{q} + c(q,\dot{q})\dot{q} + g(q) + J^T F_{ext} \\ y &= h(q) \end{cases} \tag{1}$$

- $\underset{\sim}{q} \in \mathbb{R}^n$: joint positions (generalized coordinate) θ

- $\tau \in \mathbb{R}^n$: joint torque (generalized input)

- $y \in \mathbb{R}^p$: output (variable to be controlled)

- **Motion Control Problems**: Let y track given reference y_d motion planner

Variations in Robot Motion Control

Typically, $q_d(t)$ (i.e. the reference) is given by the planner, represented by polynomial

- Joint-space vs. Task-space control:
  - Joint-space: $y(t) = q(t)$, i.e., want $q(t)$ to track a given $\boxed{q_d(t)}$ joint reference
  - Task-space: $y(t) = T(q(t))$ denotes end-effector pose/configuration, we want $y(t)$ to track $y_d(t)$

forward kinematics

So typically, $\dot{q}_d$, $\ddot{q}_d$ do not have too much error

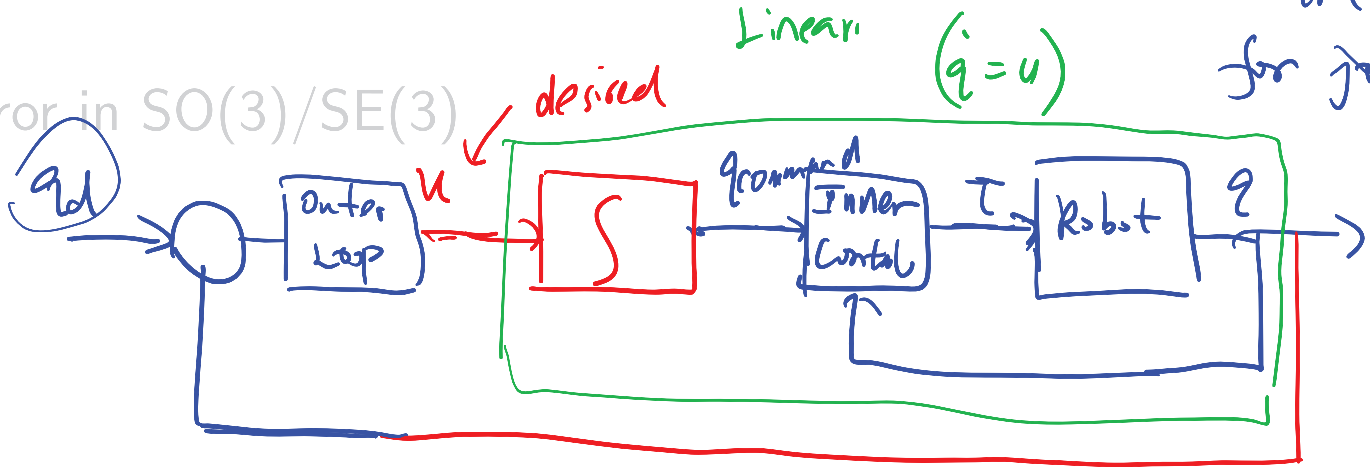
But $\ddot{q}(t)$ measurement data hard on can be noisy.

- Actuation models:
  - Velocity source: $u = \dot{q}$
  - Acceleration sources: $u = \ddot{q}$
  - Torque sources: $u = \tau$

Outline

- Motion Control Problems
- Motion Control with Velocity/Acceleration as Input
- Motion Control with Torque as Input
- Task-Space Error in SO(3)/SE(3)

most robot manipulator: position is common control variable for joints



Velocity-Resolved Control

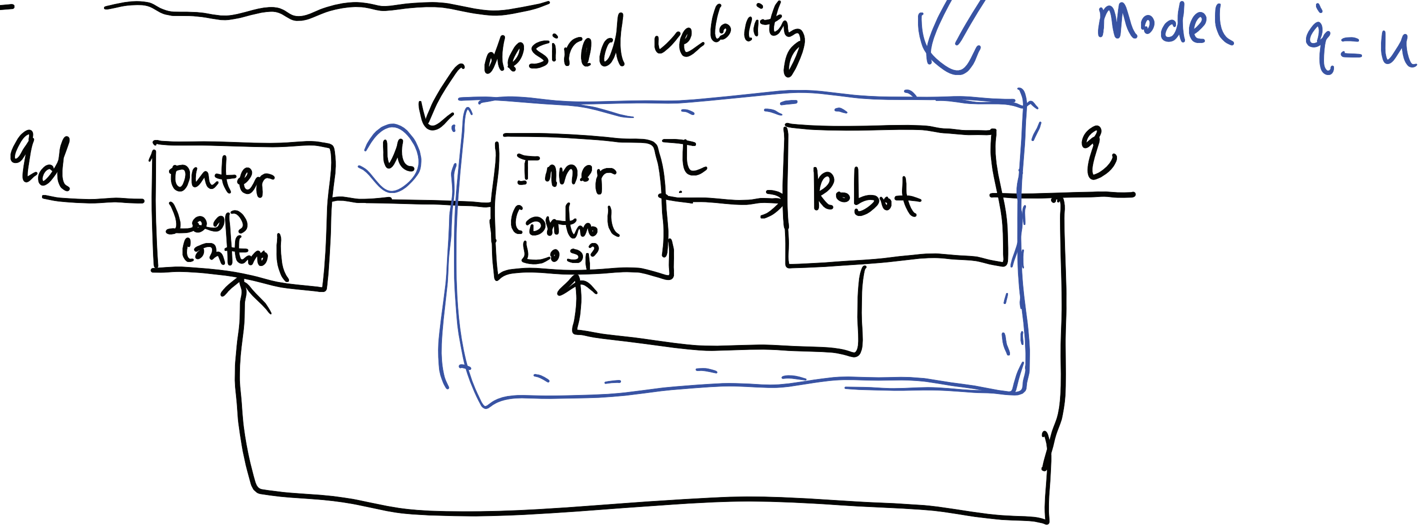
- Each joints' velocity $\dot{q}_i$ can be directly controlled

$$\dot{q} = u \in \mathbb{R}^n \qquad \dot{q}_i = u_i$$

- Good approximation for hydraulic actuators

- Inner/outer loop control diagram

Linear model $\dot{q} = u$



Velocity-Resolved Joint Space Control

- Joint-space "dynamics": single integrator

$$\dot{q} = u \quad \in \mathbb{R}^n$$

- Joint-space tracking becomes standard linear tracking control problem:

$$u = \dot{q}_d + K_0\tilde{q} \quad \Rightarrow \dot{\tilde{q}} + K_0\tilde{q} = 0$$

$K_0 \in \mathbb{R}^{n\times n}$ $\quad \tilde{q} = \text{error} = q_d - q \in \mathbb{R}^n$

where $\tilde{q} = q_d - q$ is the joint position error.

$$\dot{q} = \dot{q}_d + K_0\tilde{q} \Rightarrow \frac{d}{dt}(q_d - q) + K_0\tilde{q} = 0$$

$$\dot{\tilde{q}} + K_0\tilde{q} = 0$$

- The error dynamic is stable if $-K_0$ is Hurwitz

Joint i: $\dot{\tilde{q}}_i = -K_{0,i} \cdot \tilde{q}_i$

$\Rightarrow$ Frequency domain:

$$\dot{\tilde{q}} = -K_0\tilde{q} \Rightarrow \tilde{q}(t) = e^{-K_0 t}\tilde{q}(0)$$

$\text{eig}(-K_0) \in \text{OLHP}$

$\dot{x} = Ax$

Common choice: $K_0 = \begin{bmatrix} K_{0,1} & & & \\ & K_{0,2} & & \\ & & \ddots & \\ & & & K_{0,n}\end{bmatrix}$, where $K_{0,i} > 0$

Velocity-Resolved Task-Space Control (1/3)

- For task space control, $y = T(q)$ needs to track y_d
  - y can be any function of q, in particular, it can represents position and/or the end-effector frame

$y = T(q) = (R(q), p(q))$

- Taking derivatives of y, and letting $u = \dot{q}$, we have

$$\dot{y} = J_a(q)u \tag{2}$$

$\dot{q}$

  - Note that q is function of y through inverse kinematics.

$q = IK(y)$

  - So the above dynamics can be written in terms of y and u only. The detailed form can be quite complex in general

$\dot{y} = J_a(IK(y))\,u$

$\dot{x} = f(x, u)$

Velocity-Resolved Task-Space Control (2/3)

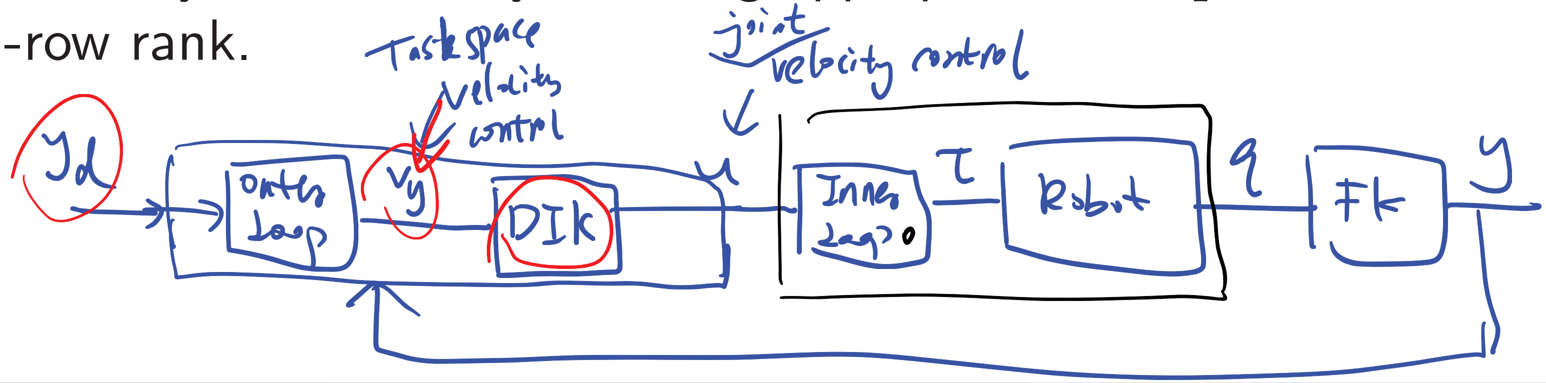
- System (2) is nonlinear system, a common way is to break it into inner-outer loop, where the outer loop directly control velocity of y, and the inner loop tries to find u to generate desired task space velocity

- **Outer loop**: $\boxed{\dot{y} = v_y}$ where control $v_y = \underline{\dot{y}_d + K_0 \tilde{y}}$, resulting in task-space closed-loop error dynamics:

$$\dot{\tilde{y}} + K_0 \tilde{y} = 0$$

$$\dot{y} = v_y$$
$$= \dot{y}_d + k_0 \tilde{y}$$
$$\Rightarrow \dot{y}_d - \dot{y} + k_0 \tilde{y} = 0$$
$$\dot{\tilde{y}} + k_0 \tilde{y} = 0$$

- Above task space tracking relies on a fictitious control v_y, i.e., it assumes $\dot{y}$ can be arbitrarily controlled by selecting appropriate $u = \dot{q}$, which is true if J_a is full-row rank.



Velocity-Resolved Task-Space Control (3/3)

- **Inner loop**: Given v_y from the outer loop, find the joint velocity control by solving

$$\begin{cases} \min_u \|v_y - J_a(q)u\|^2 + \text{regularization term} \\ \text{subj. to:} \quad \text{Constraints on } u \end{cases} \tag{3}$$

$u_{min} \le u \le u_{max}$

joint velocity limits,

- Inner-loop is essentially a differential IK controller

- One can also use the pseudo-inverse control $u = J_a^\dagger v_y$

pseudo-inverse

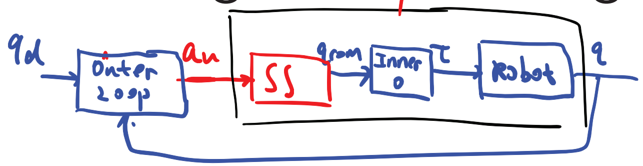
- This can not incorporate constraints on u

- . . . . . regularization term .

Acceleration-Resolved Control in Joint Space

Linear robot model

- Joint acceleration can be directly controlled, resulting in double-integrator dynamics



$$\ddot{q} = u$$

- Joint-space tracking becomes standard linear tracking control problem for double-integrator system:

$$u = \ddot{q}_d + \underset{D}{K_1}\dot{\tilde{q}} + \underset{P}{K_0}\tilde{q} \quad \Rightarrow \ddot{\tilde{q}} + K_1\dot{\tilde{q}} + K_0\tilde{q} = 0$$

where $\tilde{q} = q_d - q$ is the joint position error.

- Stability condition: $\begin{bmatrix} x_1 = \tilde{q} \\ x_2 = \dot{\tilde{q}} \end{bmatrix} \Rightarrow \dot{x} = \underbrace{\begin{bmatrix} 0 & I \\ -K_0 & -K_1 \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

A is Hurwitz

Common choice $K_0 = \begin{bmatrix} k_{0,1} & & 0 \\ & \ddots & \\ 0 & & k_{0,n} \end{bmatrix}$, $K_1 = \begin{bmatrix} k_{1,1} & & 0 \\ & \ddots & \\ 0 & & k_{1,n} \end{bmatrix}$

Acceleration-Resolved Control in Task Space (1/2)

- For task space control, $y = T(q)$ needs to track y_d

We can choose u , $\dot{q} = u$

- Note: $\dot{y} = J_a(q)\dot{q}$ and $\ddot{y} = \dot{J}_a(q)\dot{q} + J_a(q)\ddot{q}$

u

Drake : calcBiasAcceleration

- Following the same inner-outer loop strategy discussed before

- **Outer-loop** dynamics: $\ddot{y} = a_y$ with a_y being the outer-loop control input

$$a_y = \ddot{y}_d + K_1\dot{\tilde{y}} + K_0\tilde{y} \quad \Rightarrow \quad \ddot{\tilde{y}} + K_1\dot{\tilde{y}} + K_0\tilde{y} = 0$$

$$A = \begin{bmatrix} 0 & I \\ -K_0 & -K_1 \end{bmatrix}$$

Acceleration-Resolved Control in Task Space (2/2)

- **Inner-loop:** Given a_y from outer loop, find the "best" joint acceleration:

$$\begin{cases} \min_u \|a_y - \dot{J}_a(q)\dot{q} - J_a(q)u\|^2 + \text{regularization term} \\ \text{subj. to:} \quad \text{Constraints on } u \end{cases} \tag{4}$$

$\ddot{q}_{min} \leq u \leq \ddot{q}_{max}$, $\dot{q}_{min} \leq \dot{q} + u \cdot \Delta t \leq \dot{q}_{max}$

- Mathematically, the above problem is the same as the Differential IK problem

- At any given time, $q, \dot{q}$ can be measured, and then y and $\dot{y}$ can be computed, which allows us to compute outer loop control a_y and inter loop control u

Outline

- Motion Control Problems
- Motion Control with Velocity/Acceleration as Input
- Motion Control with Torque as Input
- Task-Space Error in SO(3)/SE(3)

Recall Properties of Robot Dynamics

For fully actuated robot:

$$\overset{u}{\tau} = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \tag{5}$$

- $M(q) \in \mathbb{R}^{n\times n} \succ 0 \Leftarrow$ positive definite, sym $\in S_n^{++}$
- There are many valid definitions of $C(q,\dot{q})$, typical choice for C include:

$$C_{ij} = \sum_k \frac{1}{2}\left[\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i}\right]$$

- For the above defined C, we have $\dot{M} - 2C$ is skew symmetric
- For all valid C, we have $\dot{q}^T\left[\dot{M} - 2C\right]\dot{q} = 0$
- These properties play important role in designing motion controller

Computed Torque Control (1/2)

- For fully-actuated robot, we have $M(q) \succ 0$ and $\ddot{q}$ can be arbitrarily specified through torque control $u = \tau$

$$\ddot{q} = M^{-1}(q)\left[u - C(q, \dot{q})\dot{q} - g(q)\right]$$

- Thus, for fully-actuated robot, torque controlled case can be reduced to the acceleration-resolved case
- Outer loop: $\ddot{q} = a_q$ with joint acceleration as control input

$$a_q = \ddot{q}_d + K_1\dot{\tilde{q}} + K_0\tilde{q} \quad \Rightarrow \ddot{\tilde{q}} + K_1\dot{\tilde{q}} + K_0\tilde{q} = 0$$

- Inner loop: since $M(q)$ is square and nonsingular, inner loop control u can be found analytically:

$$u = M(q)\left(\underbrace{\ddot{q}_d + K_1\dot{\tilde{q}} + K_0\tilde{q}}_{a_q}\right) + C(q, \dot{q})\dot{q} + g(q) \tag{6}$$

Computed Torque Control (2/2)

- The control law (6) is a function of $q, \dot{q}$ and the reference q_d. It is called *computed-torque control*.

- The control law also relies on system model M, C, g, if these model information are not accurate, the control will not perform well.

- Idea easily extends to task space: $\dot{y} = J_a(q)\dot{q}$ and $\ddot{y} = \dot{J}_a(q)\dot{q} + J_a(q)\ddot{q}$

- Outer loop: $\ddot{y} = a_y$, and $a_y = \ddot{y}_d + K_1\dot{\tilde{y}} + K_0\tilde{y}$

$\ddot{q} = M^{-1}(u - C\dot{q} - g)$

- Inner loop: select torque control $u = \tau$ by

$$\begin{cases} & \min_u \|a_y - \dot{J}_a\dot{q} - J_a M^{-1}(u - C\dot{q} - g)\|^2 \\ \text{subj. to:} & \text{constraints} \end{cases} \tag{7}$$

- If J_a is invertible and we don't impose additional torque constraints, analytical control law can be easily obtained.

Inverse Dynamics Control (1/2)

ID

- The computed-torque controller in (6) is also called *inverse dynamics control*
- Forward dynamics: given τ to compute $\ddot{q}$
- Inverse dynamics: given desired acceleration a_q, we inverted it to find the required control by $u = Ma_q + C\dot{q} + g$
- Task space case can be viewed as inverting the task space dynamics

TSID

- With recent advances in optimization, it is often preferred to do ID with quadratic program

Inverse Dynamics Control (2/2)

- For example, Eq (7) can be viewed as task-space ID. We can incorporate torque contraints explicitly as follows:

$$\begin{cases} & \min_u \|a_y - \dot{J}_a \dot{q} - J_a M^{-1}(u - C\dot{q} - g)\|^2 \\ \text{subj. to:} & u_- \leq u \leq u_+ \end{cases} \tag{8}$$

- This is equivalent to the following more popular form:

$$\begin{cases} \min\limits_{u,\ddot{q}} & \|a_y - \dot{J}_a \dot{q} - J_a \ddot{q}\|^2 \\ \text{subj. to:} & M\ddot{q} + C\dot{q} + g = u \\ & u_- \leq u \leq u_+ \end{cases} \tag{9}$$

Outline

- Motion Control Problems
- Motion Control with Velocity/Acceleration as Input
- Motion Control with Torque as Input
- Task-Space Error in SO(3)/SE(3)

How to Specify Desired a_y in SE(3)?

- Task space desired reference is given in SE(3), i.e.,

$$y_d(t) = (p_d(t), R_d(t))$$

- Current end-effector (body) configuration $y(t) = (p_b(t), R_b(t))$
- Outer-loop control consists of position and angular accelerations:

$$a_y = \begin{bmatrix} a_p \\ a_R \end{bmatrix}$$

- Position vector $p_b(t)$ and $p_d(t)$ lie in Euclidean space, so

$$a_p = \ddot{p}_d + K_D \dot{p}_{db} + K_P p_{db}$$

$p_{db} \triangleq \underbrace{p_d - p_b}_{\text{error}}$

where $p_{db} = p_d - p_b$

- Orientation $R_b(t)$ and $R_d(t)$ lies in $SO(3)$, specifications of angular errors and their derivative are nontrivial.

Euler Angle Errors: Direct Method (1/3)

$R_b(t) \longleftrightarrow \varphi_b$, $R_d(t) \longleftrightarrow \varphi_d$

$R_d^T R_b \longleftrightarrow \varphi_{db}$

exponential coordinate of $R_d^T R_b$

- Euler Angle Error:

current body orientation

$R_d(t) \rightarrow \varphi_d(t) \in \mathbb{R}^3$, $R_b(t) \rightarrow \varphi_b(t) \in \mathbb{R}^3$

(e.g. ZYX-Euler)

Euler Error: $\varphi_{db} = \varphi_d - \varphi_b$

$\omega_d = T(\varphi_d)\dot{\varphi}_d$, $\omega_b = T(\varphi_b)\dot{\varphi}_b$

$\dot{\varphi}_{db} = \dot{\varphi}_d - \dot{\varphi}_b \overset{c}{=} T^{-1}(\varphi_d)\,\omega_d - T^{-1}(\varphi_b)\,\omega_b$

$\dot{\omega}_d = \dot{T}(\varphi_d)\dot{\varphi}_d + T(\varphi_d)\ddot{\varphi}_d$

* Note: - rotation matrix corresponds to orientation

- $R(t) \longleftrightarrow$ Euler Angle $\varphi(t)$

- $\dot{R}(t) = \omega \times R(t)$

$\omega = \Gamma(\varphi(t))\,\dot{\varphi}(t)$

E.g: $\varphi = \begin{bmatrix}\alpha \\ \beta \\ \gamma\end{bmatrix}$, ZYX-Euler Angle

$\Gamma(\varphi(t)) = \begin{bmatrix} 0 & -S_\alpha & C_\beta C_\alpha \\ 0 & C_\alpha & C_\beta S_\alpha \\ 1 & 0 & 0 \end{bmatrix}$

(See Kinematics Lecture)

Euler Angle Errors: Direct Method (2/3)

- $\Rightarrow \ddot{\varphi}_d \overset{c}{=} T^{-1}(\varphi_d)\left[\dot{w}_d - \dot{T}(\varphi_d)\dot{\varphi}_d\right]$

$\Rightarrow$ Our Outer-Loop Control: $a_R = T(\varphi_b)(\ddot{\varphi}_d + k_D\dot{\varphi}_{db} + k_P\varphi_{db}) + \dot{T}(\varphi_b)\dot{\varphi}_b$

$$\ddot{p} = a_p = \ddot{p}_d + k_D\dot{\tilde{p}} + k_P\tilde{p}$$

$$\Rightarrow \dot{w}_b = a_R = T(\varphi_b)(\ddot{\varphi}_d + k_D\dot{\varphi}_{db} + k_P\varphi_{db}) + \dot{T}(\varphi_b)\dot{\varphi}_b$$

$$\overset{c}{\Leftrightarrow} T(\varphi_b)\left[\ddot{\varphi}_{db} + k_D\dot{\varphi}_{db} + k_P\varphi_{db}\right] = 0$$

Note: $\dot{w}_b = \boxed{\dot{T}(\varphi_b)\dot{\varphi}_b + T(\varphi_b)\ddot{\varphi}_b}$ $\quad \overset{c}{\Leftrightarrow} \ddot{\varphi}_{db} + k_D\dot{\varphi}_{db} + k_P\varphi_{db} = 0$

The above derivation require condition (c) holds:

$$T(\varphi_b(t)),\ T(\varphi_d(t)) \text{ nonsingular}$$

Euler Angle Errors: Direct Method (3/3)

-
-

Euler Angle Errors: Relative Orientation (1/3)

- what's difference (error) between R_b, R_d?
  - Previous case: quantify error by $\psi_d - \psi_b$
  - More accurately: $\tilde{R} = R_b^T R_d$ $\quad$ $R_b \tilde{R} = R_d \Rightarrow \tilde{R} = R_b^T R_d$
  - We want $\tilde{R} \to I$ (relative orientation $\to I$) $\quad$ ${}^oR_b^T\, {}^oR_d = {}^bR_o\, {}^oR_d$

Use Euler Angle to represent $\tilde{R}$

$$\tilde{R} \Longleftrightarrow \tilde{\psi} = \psi_{db}$$

$$\dot{\tilde{R}} = \dot{R}_b^T R_d + R_b^T \dot{R}_d = (R_b[{}^bw_b])^T R_d + R_b^T R_d [{}^dw_d]$$

$$= -[{}^bw_b] R_b^T R_d + R_b^T R_d [{}^dR_b\, {}^bw_d]$$

$$({}^dR_b)[{}^bw_d]\, {}^dR_b^T$$

$$[Rw] = R[w]R^T$$

Euler Angle Errors: Relative Orientation (2/3)

- $= -[{}^b\omega_b] R_b^T R_d + \cancel{R_b^T R_d}\, {}^d R_b [{}^b\omega_d]\, {}^b R_d$

$= [{}^b\omega_d]\, {}^b R_d - [{}^b\omega_b]\, {}^b R_d = ({}^b\omega_d - {}^b\omega_b) \times \tilde{R}$

$\dot{\tilde{R}} = ({}^b\omega_d - {}^b\omega_b) \times \tilde{R} \quad \cdots\cdots \quad (*)$

$-{}^b\omega_{db} = \Gamma(\varphi_{db})\, \dot{\varphi}_{db}$ ⇒ take derivative

$${}^b\dot{\omega}_b = \left[{}^b\dot{\omega}_d - \Gamma(\varphi_{db})\, \ddot{\varphi}_{db} - \dot{\Gamma}(\varphi_{db})\, \dot{\varphi}_{db} \right]$$

$\Rightarrow a_R = {}^b\dot{\omega}_d + \Gamma(\varphi_{db}) (K_D \cdot \dot{\varphi}_{db} + K_P \varphi_{db}) - \dot{\Gamma}(\varphi_{db})\, \dot{\varphi}_{db}$

Check:

Euler Angle Errors: Relative Orientation (3/3)

-
- $${}^b\dot{\omega}_b = a_R = {}^b\dot{\omega}_d + \Gamma(\varphi_{db})(\cdots\cdots) - \dot{\Gamma}(\varphi_{db})\dot{\varphi}_{db}$$

$$\Rightarrow \quad {}^b\dot{\omega}_{db} + \left(\Gamma(\varphi_{db})(\cdots\cdots) - \cancel{\dot{\Gamma}(\varphi_{db})}\dot{\varphi}_{db}\right)$$

$$\Gamma(\varphi_{db})\ddot{\varphi}_{db} + \cancel{\dot{\Gamma}(\varphi_{db})\dot{\varphi}_{db}} + \left(\qquad\right)$$

$$\Rightarrow \Gamma(\varphi_{db})\left(\underbrace{\ddot{\varphi}_{db} + k_D\dot{\varphi}_{db} + k_p\varphi_{db}}_{=0}\right) = 0$$

$\therefore \ \varphi_{db} \to 0$, for stable k_D, k_p

– Only need to assume $\Gamma(\varphi_{db})$ nonsingular

Exponential Coordinate Error (1/3)

- Relative orientation: ${}^b R_d$ $\quad \in \mathbb{R}^{3\times 3}$

- Exp coordinate: $\quad S_R = \log({}^b R_d)$

recall: $S_R = \theta \cdot \hat{\omega}$, where ${}^b R_d = e^{[S_R]} = e^{[\hat{\omega}]\theta}$

$$[\hat{\omega}] = \frac{1}{2\sin\theta}\left[{}^b R_d - {}^b R_d^T\right] \quad (\text{trace}({}^b R_d) \neq \pm 1)$$

(side note: $\frac{1}{2}(A+A^T)$: symmetric $\mid$ $\frac{1}{2}(A-A^T)$ skew symmetric)

both S_R and ${}^b R_d$ are func of t:

$$ {}^b\dot{R}_d = \left(e^{[S_R]}\right)' = e^{[S_R]} \cdot [\dot{S}_R] = [\dot{S}_R]\, e^{[S_R]} \quad \cdots\cdots (**)$$

side note: $\left(e^A\right)' = \left(I + A + \frac{A^2}{2!} + \cdots\right)' = 0 + \dot{A} + \frac{2A}{2!}\dot{A} + \cdots$

Exponential Coordinate Error (2/3)

$$= \left(I + A + \frac{A^2}{2!} \cdots\right)\dot{A}$$

- By $(*)$ ${}^b\dot{R}_d = \underbrace{[{}^b\omega_{db}]}\,{}^bR_d$ - - - - - $(*)$

By $(**)$: $\underline{\dot{s}_R} = {}^b\omega_{db} = \underline{{}^b\omega_d - {}^b\omega_b}$

$\{ (*)$

$${}^b\dot{\omega}_b = a_R = \underline{{}^b\dot{\omega}_d} + k_D\,\dot{s}_R + k_P\,s_R$$

$$\Rightarrow \quad \ddot{s}_R + k_D\,\dot{s}_R + k_P\,s_R = 0$$

$$s_R \to 0$$

$$\theta\hat{\omega} \to 0$$

Exponential Coordinate Error (3/3)

- Note: In the UAV Literature:
-

error: $e_R = \frac{1}{2}\left[{}^bR_d - {}^bR_d^T\right]$

$e_\Omega = {}^bR_d\, {}^d\omega_d - {}^b\omega_b$

need Lyapunov to show convergence

More Discussions

-

More Discussions

-

More Discussions

-