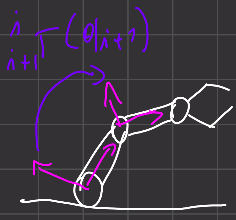
19/02/24

In this course we will mainly focus on manipulators



We saw that we could assign variables to joints,

$$\underline{q}(t) = \begin{bmatrix} q_1(t) \\ \vdots \\ q_n(t) \end{bmatrix}$$

in order to control the robot.

There are two important entities:

$$\begin{cases} {}^{i}_{i+1}R(q_{i+1}) \\ r_{i+1/i}(q_{i+1}) \end{cases}$$

this is what we called forward geometry

In order to control the robot we needed to use the inverse geometric model.

There is always a solution to the forward geometric model.
In the inverse problem we are in trouble if the robot is redundant.
We can have infinite many solutions.

We also have to keep into consideration what happens when time passes.

Typically we have an end effector attached to the end-effector of the robot.

q variables change in time so I might be interested by the trajectory generated by the end effector.

So we have two steps:

- What happens if for a given configuration at a given time instant I assign a set of joint velocities:

$$\dot{q}(t) = \begin{bmatrix} \dot{q}_1(t) \\ \vdots \\ \dot{q}_n(t) \end{bmatrix}$$

If I can compute the velocities then I can compute the angular velocity of the ee wrt any frame of interest.
We can also compute also the linear velocity

Forward kinematic problem

$$\underbrace{\begin{bmatrix} {}^{\phi}\omega_{e/\phi}(t) \\ {}^{\phi}v_{e/\phi}(t) \end{bmatrix}}_{\text{output}} = \underbrace{{}^{\phi}J_{e/\phi}(q)}_{\text{linear transformation}} \overset{\text{input}}{\dot{q}(t)}$$

$\boxed{\text{N.B. These are all function of time}}$

Computing this solves the forward Kinematic Problem

Inverse Kinematic Problem

Here control comes clearly into play.

Now we have to revert the problem.

In this case I want to achieve a particular angular velocity at the EE.

$$\underbrace{x^* = \begin{bmatrix} \omega^*(t) \\ v^*(t) \end{bmatrix}}_{\text{Input}} = \overbrace{J(q)\, \underbrace{\dot{q}(t)}_{\text{output}}}^{\text{this is the forward model of the robot. Not software}}$$

The output is not explicitely expressed.

This expression falls in this type of formulae : $Ax = y$

Now we have to follow rules depending on the shape of J

1) Square → Here you get the solution (excluding sing)
2) Flat (more columns than 6) → Infinite solutions
3) Tall (less columns than 6) → You might not find the solution
   ↳ This is the typical case of mobile robots

In the case of square problems we can compute:

> If square matrix = $\dot{q}(t) = J^{-1} x(t)$

> a flat matrix = We have more solution (the difference between rows and cols number)

The simplest way of solving this is using the least square solution.

$$\dot{q}(t) = \underbrace{J^T (J\, J^T)^{-1}}_{\text{Left pseudoinvers}} x^* \quad \text{(This is software)}$$

There are variations because the motors of some joints may not be as powerfull as required

This is a closed form formula.

The psuedoinverse has some properties: $J^{\#} J = \mathbb{1}$

So for square and flat we can find only one solution.

$\rangle$ In tall configuration you have a best effort solution

$$\dot{q} = J^T (J J^T)^{-1} \dot{x}^*$$

In every case we still have problems with singularities.

In this case you get something close to the movement that you desire.

You go from J to $J^{\#}$ by using SVD

$$A = U \Sigma V^T$$
$$\downarrow$$
$$\begin{bmatrix} \sigma_1 \ddots & & \emptyset \\ & & \sigma_n \end{bmatrix}$$
$$\sigma_1 > \dots > \sigma_n > 0$$

CHI:

$$\chi = \frac{\sigma_1}{\sigma_n}$$

CONDITION NUMBER OF THE MATRIX

IF χ is big then the system will be noisy and the result will be big. This effect gets bigger when you are close to singularities.

In practice you need to adjust the singular values in order to bound the result, and not have jerky motions.

The solution to this problem is to compute $J^{\#}$ like this:

$$J^{\#} = J^T (J J^T + \lambda \mathbb{1})^{-1}$$

λ must be added when this is getting critical. λ Generates a wrong solution but keeps everything bounded.

This is called regularization.

The pseudoinverse of A $\overset{\to \text{ or } J}{\text{is}}$ basically computed as:

$$A^{\#} = U \Sigma^{\#} V^T$$

$\searrow \swarrow$ Orthonormal matrices $\rightarrow \begin{cases} UU^T = \mathbb{1}_n \\ V^T V = \mathbb{1}_m \end{cases}$

$$\Sigma^{\#} = \begin{bmatrix} \frac{1}{\sigma_1} & & & \\ & \ddots & & 0 \\ & & \frac{1}{\sigma_n} & \end{bmatrix}$$

When the critical singular values get critical you can play with lambda.

Every inverse should be done with the pseudoinverse

Inverse geometry problem

I want to find two values:

$* $ means desired.

$^{0}_{e}R^{*}(t)$ $\qquad$ $^{0}_{e}R(q)$

$\underline{r}^{*}_{e/0}$ $\qquad$ $^{0}\underline{r}_{e/0}(q)$

(displacement is a vector)

linear error

$$\underline{e}_L(t) = \underline{r}_{e/0} - \underline{r}^{*}_{e/0}$$

actual position $\qquad$ desired position

misalignment between the two frames

$$\rho = v\,\theta$$

angular error.

My control objective is to $\begin{cases} \rho \to 0 \\ e_L \to 0 \end{cases}$ Here we want to reduce the distance

N.B DISPLACEMENT $\neq$ DISTANCE
(vector) $\qquad$ (scalar)

$$\frac{1}{2}|e_L|^2 = \frac{1}{2} e_L \cdot e_L$$

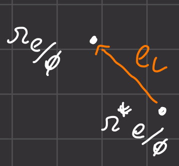
the derivative of this we want negative.

$$\frac{1}{2}\frac{d}{dt}(e_L \cdot e_L) = \frac{1}{2} e_L \cdot \dot{e}_L + \frac{1}{2}\dot{e}_L \cdot e_L = e_L \cdot \dot{e}_L = e_L \cdot \left[v_{e/0} - v^{*} \right]$$

this is the control signal

This is where I define my control structure.

Note that e_L is given



(We want to choose $v_{e/0}$ so that the scalar product is negative)

This process is called control synthesis

We can express:

$$e_L \cdot \underbrace{[v_{e_\phi} - v^*]}_{\boxed{-\gamma e_L}} = \underbrace{-\gamma (e_L \cdot e_L) < 0}$$

If I choose this for the term inside the matrix this will be guaranteed

We want to guarantee that:

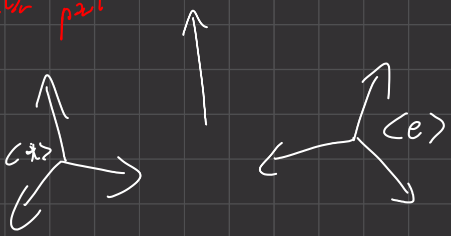
$$\underset{\text{output}}{v_{e_\phi}} - \underset{\text{"input"}}{v^*} = -\underset{\text{control parameter}}{\gamma}\, \underset{\text{input}}{e_L}$$

$$\boxed{v_{e_\phi} = \underset{\text{feedback control term.}}{-\gamma e_L} + v^*}$$

$[\dot{x}^*]$

feed forward term providing velocity over time to keep track of the goal.

Angular part



$$\frac{1}{2}|\rho \cdot \rho|^{\frac{1}{2}} = \frac{1}{2}\theta^2$$

this because v is unit vector

$$\frac{d}{dt}\frac{1}{2}(\rho \cdot \rho) = \theta v \cdot \overbrace{\underbrace{(\omega_{e_\phi} - \omega^*)}}^{-\gamma_\theta v \theta}$$

I want this to be opposite of v because I want to turn the other way round.

$$\boxed{{}^\phi\omega_{e_\phi} = -\gamma_\theta v \theta + \omega^*}$$