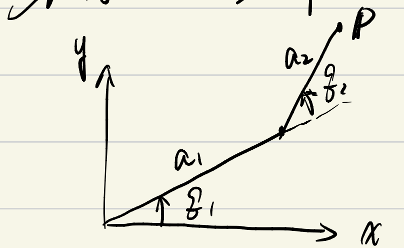
Robotics, Vision, & Mechatronics for Manufacturing

Derivative of Kinematics
(Manipulator Velocity)
(chap 8 of Peter Corke)

Xu Chen

2021 Sp

Intro: 2D planar robot



forward kinematics: $p = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} a_2\cos(q_1+q_2) + a_1\cos q_1 \\ a_2\sin(q_1+q_2) + a_1\sin q_1 \end{bmatrix}$

derivative of p: $\frac{dp}{dt} = \underbrace{\frac{dp}{dq}}\frac{dq}{dt}$ i.e. $\dot{p} = J(q)\dot{q}$

$$\triangleq J(q) = \begin{bmatrix} \frac{dp_1}{dq_1} & \frac{dp_1}{dq_2} \\ \frac{dp_2}{dq_1} & \frac{dp_2}{dq_2} \end{bmatrix}$$

$$= \begin{bmatrix} -a_2\sin(q_1+q_2) - a_1\sin q_1 & -a_2\sin(q_1+q_2) \\ a_2\cos(q_1+q_2) + a_1\cos q_1 & a_2\cos(q_1+q_2) \end{bmatrix}$$

$J(q)$: called the Jacobian matrix.
maps velocity from the joint space to the end-effector's Cartesian space

Generalization to 3D case

Now with rotation & translation

6x6 Jacobian matrix (aka the geometric Jacobian)

$$
{}^{0}\nu = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \frac{dp}{dt} = {}^{0}J(q)\,\dot{q}
$$

defines the instantaneous forward kinematics

spatial velocity of the end-effector in the world coordinate

MATLAB: p560.jacob0(qn)

${}^{0}J(q)$: maps joint velocity to end-effector spatial velocity in the world coordinate frame

Jacobian Condition

in reality, robot joint has velocity control. What joint velocities are needed to achieve an end-effector Cartesian velocity?

$$\nu = J(q)\,\dot{q} \Rightarrow \dot{q} = J(q)^{-1}\nu \quad \text{if invertible}$$

$\Rightarrow$ singularity at $\det J(q) = 0$, such a robot configuration q is called degenerate.

e.g. MATLAB
```
>> J = p560.jacob0(qr)
>> det(J) = 0
```

Manipulability

Joint space $\xrightarrow{\nu = J(q)\dot{q}}$ End-effector Cartesian Space (desired ν)

consider $\dot{q}^T \dot{q} = 1$ — a hypersphere, equally manipulable in joint angles

$$\left(J(q)^{-1}\nu\right)^T\left(J(q)^{-1}\nu\right) = 1$$

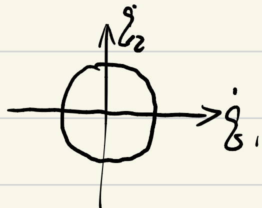
$$\Longleftrightarrow \nu^T (J(q)^{-1})^T J(q)^{-1} \nu = 1$$

$$\Longleftrightarrow \nu^T \underbrace{\left(J(q)J(q)^T\right)^{-1}}\nu = 1$$

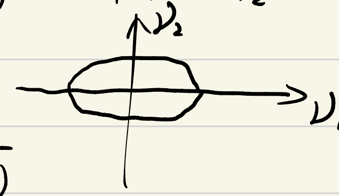
: the shape of an ellipsoid describes how well-conditioned the manipulator is for making certain motions

$$[\ \ ][\ \ ]\cdot[\ ] = 1$$

e.g. $\dot{q}_1^2 + \dot{q}_1^2 = 1$



e.g. $a_1\nu_1^2 + a_2\nu_2^2 = 1$



Yoshikawa's manipulability measure:

$$m \triangleq \sqrt{\det(JJ^T)}$$

Note: The above is the kinematic manipulability.
Mass & inertia also matter. see §9.2.7

Kinematic motion control

Recall if we know desired pose ξ^* w/ homogeneous transformatrix T^*.

$$\xi^* = \mathcal{K}(q^*)$$

we can obtain $q^* = \mathcal{K}^{-1}(\xi^*)$ via inverse kinematics.

desired joint space configuration

Limitations:
- numerical issues when using numerical inverse kinematics
- complexity w/ analytic form

$\exists$ not one-to-one mapping

Motion Control: resolve-rate kinematic control.

Goal: generate straight-line motion in Cartesian space

solution concept: $\nu = J(q)\dot{q} \implies \dot{q} = J(q)^{-1}\nu$

assumption: $J(q)$ is invertible

$$q^*[k+1] = q^*[k] + \Delta t \cdot \dot{q}^*[k] = q^*[k] + \underbrace{\Delta t \cdot J(q^*[k])^{-1}\nu^*}$$

sampling time

desired Cartesian-space velocity gets resolved onto joint-space angles w/o solving explicitly inverse kinematics.

MATLAB: >> mdl_puma560

>> sl_rrmc

limitation: $J(q)$ must be square & nonsingular.

Not going to work for under- & over-actuated robots (e.g. snake robot)

needs very accurate kinematic model

sensitive to parameter variations

using feedback/closed-loop can give robustness

Closed-loop resolve-rate kinematic motion control

$$\dot{q} = J(q)^{-1} \nu$$

$$q[k+1] = q[k] + \Delta t \cdot k_p J(q[k])^{-1} \left(\mathcal{K}(q[k]) - \xi^*[k] \right)$$

target pose

forward kinematics

MATLAB: >> mdl_puma560

>> sl_rrmc2

Relaxing the assumption on an invertible $J(q)$

Method 1: damped inverse : $\dot{q} = (J(q) + \lambda I)^{-1} \nu$ $\Rightarrow$ introduce large feedforward errors, need feedback kinematic control

Method 2: pseudo inverse :

$$J^{+} J = I$$
$$J^{+} = (J^T J)^{-1} J^T$$
$$\dot{q} = J(q)^{+} \nu$$

$\Rightarrow$ minimizes $\| J\dot{q} - \nu \|$

Method 3: manually remove linearly dependent columns

control w/ under-actuated systems §8.4