РМ втор колоквиум

диференцијални движења и брзини

$\boxed{D = J \cdot Dq}$ — е J-start

→ Јакобијан → (1) е врска меѓу D и Dq или v и $\dot{q}$

во DH модели →
- J = robot.jacobian()
- J_v = robot.linear_jacobian()
- J_w = robot.angular_jacobian()

$J = \begin{bmatrix} J_v \\ J_w \end{bmatrix}_{n \times m}$

n — во колку движења може да се движи; m — бр. зглобови

$$J = \begin{array}{c|c} & \text{зглоб } 1 \ldots n \\ \begin{matrix} V_x \\ V_y \\ \vdots \\ W_z \end{matrix} & \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \end{array}$$

1. зглоб не придонесува за домест.

$D = \begin{bmatrix} dx \\ dy \\ dz \\ \delta x \\ \delta y \\ \delta z \end{bmatrix}$

- dx, dy, dz → диф. движења на извршниот елемент по x, y, z
- $\delta x, \delta y, \delta z$ → диференц. ротации на извршниот елемент околу x, y, z (?)

→ диференцијални движења на изврш. елемент (2)

$Dq = \begin{bmatrix} dq_1 \\ dq_2 \\ dq_3 \\ dq_4 \\ dq_5 \\ dq_6 \end{bmatrix}$ - диф. движ. на зглобој → во случајо 6

- $q_i = \theta_i$ кај ротац.
- $q_i = d_i$ кај призматичен зглоб

$D = J \cdot Dq \quad / : dt \Rightarrow \underline{v} = \underline{J}\, \dot{\underline{q}}$ → брзини

диф. ротација → (околу x)

$$T = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & -\delta x & dy \\ 0 & \delta x & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ (пр)

→ диф. транслација

position = $\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix}$

(1) $\boxed{J_v = \text{position.jacobian ([пром. што фигурираат во pos])}}$

J за линиска брзина

дава редици за x, y, z
↓
ако цела редица е со 0
⇓
не може движ. по таа оска
⇓
може да се скрати таа редица

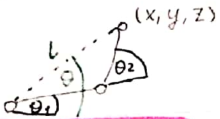
$^{Tn}J_v$ — J_v од гледна точка на n-тиот зглоб

$\boxed{{}^{n}J_v = (R_{0n})^{-1} \cdot J_v}$

$= T_{0n}[:3, :3]$

$=$ get_dh_joint_to_joint (0, n)

(1) алтернативен J



$\boxed{\text{alt\_pos} = \begin{bmatrix} l \\ \theta \end{bmatrix}}$

alt_J = alt_pos.jacobian ([пром во alt_pos])

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \frac{\partial x}{\partial d_3} \\ \frac{\partial y}{\partial \theta_1} & & \\ \frac{\partial z}{\partial \theta_1} & & \frac{\partial z}{\partial d_3} \end{bmatrix}$$ → при 2R1P

* диф трансформација

$\boxed{M\_new = M\_old + \Delta \cdot M\_old}$

$\Delta \cdot M\_old$ → = dM - M после диф. трансф. → Не е ортогонална матрица со единечни базни вектори → ради упростување при пресметки

диференцијален оператор → (опишува по кои оски какво диф. движење има)

- Δ = rt.hdelta3 ($\delta x, \delta y, \delta z$, dx, dy, dz) (2)

$\begin{bmatrix} \delta x \\ \delta y \\ \delta z \\ dx \\ dy \\ dz \end{bmatrix}$ = rt.extract_hdelta3 (delta) (2)

е во однос на реф с-м
⇓
$\boxed{M_\Delta = M^{-1} \Delta M}$
↓
во однос на локалниот М с-м

од $dM = \Delta \cdot M = M \cdot {}^{M}\Delta$
(реф) (локално)

М е M_old сегде

(Jw) $J_w = [\, J_{w1} \;\ldots\; J_{wn} \,]$

$$J_{wn} = \begin{cases} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, & \text{за призматичен зглоб} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, & \text{за ротационен (?)} \end{cases}$$

* $\begin{bmatrix} a \\ b \end{bmatrix}$ = sp.Matrix.vstack(a, b)

* $\theta_1, \theta_2 \ldots$ се во rad

Сингуларитети

* ∃ кога $\text{rang}\, J \leq \min(6, n)$ (?)
  - rang J → зависи од q
  - n → бр на зглобој

* се конфигурации на раката кои прават раката да не може да се движи во некој правец
  ↓
  често се за тр точки од раб. простор
  ⇓
  ако раката не е скроз прецизна ⇒ нема ⇒ прет ↑ v и w да се достигнат

* се определуваат од $\boxed{\det(J) = 0}$

sp.Interval(-sp.pi, sp.pi, left_open=True)

solution = sp.solveset(det_J, (1 пром во det J), interval)

поке пати solveset ако поке пром. фигурираат во det_J

J мора да е кв. матрица

· се отстрануваат редици со 0 → физичко значење на сингуларитети → не може движ/рот. по таа оска

ако редица е цела со 0 ⇒] → откако вредностите за синг. се заменат во J

⇓

ако сакаме да испитаме кога не може движ. по таа оска ⇒ заменуваме со нули

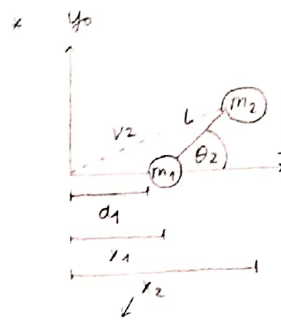
· се користи Jv наместо Jw → (кога точно?)

* скицирање сингуларитети

↓

(за добиените вредности на пром, замени во widget-от)

Динамика

* 

рота од ароботина до моментална

$K = \frac{m_1 v_1^2}{2} + \frac{m_2 v_2^2}{2}$

$v_1 = \dot{d_1}$

$v_{2x} = (d_1 + l\cos\theta_2)'$ → v е од (0,0) до масата

$v_{2y} = (l\sin\theta_2)'$

$v_2^2 = v_{2x}^2 + v_{2y}^2$

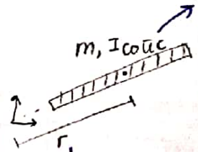
$P = m_1 g y_1 + m_2 g l \sin\theta_2$ (y_1 → 0)

$L = K - P$

rt.Lagrangian(L, [пром. што зависат од t])

се дефинираат ко

d1, theta2 = mechanics.dynamicsymbols('d1, theta2')

* 

I кога прачката ротира околу своја оска

⇓

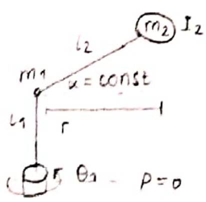
$I_{VK} = I_{COTC} + m r^2$

r → раст. до оска на ротација

· $K = \frac{I_{VK} w^2}{2} = \frac{I_{VK} \dot{\theta}^2}{2}$

· $P = m g r$

r → висина е до тежиш тето, не до крајот на прачката

* 

· m_1 не мрда

· m_2 ротира околу m_1

$I_2 = m_2 r^2 = m_2 (l_2 \cos\alpha)^2$

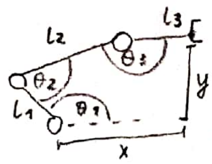
$\Rightarrow K = \frac{I_2 \dot{\theta_1}^2}{2}$ → $K = \frac{I w^2}{2} + \frac{m v^2}{2}$ ако m и ротира и транслира

$P = m_1 g l_1 + m_2 g (l_1 + l_2 \sin\alpha)$

* $\tau = J^T F$

τ → торзиони моменти на зглобој

J → J се наоѓа обично, или



$x = l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) + l_3\cos_{123}$

$y = l_1 s_1 + l_2 s_{12} + l_3 s_{123}$

Jv = sp.Matrix([x, y]).jacobian([theta1, theta2, theta3])

[x, y, θvk] ако ∃ е под агол θvk во однос на реф.

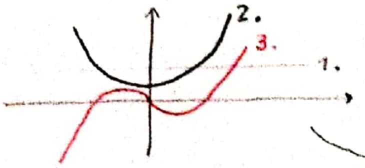
$F = \begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$ → x, y, z се во однос на реф. с-м → а ако не се т.с F_wrt_n ⇒ $\boxed{F = R0n * F\_wrt\_n}$

или на крај на рака

(2) e_Jv.subbed
v_J.subbed

Планирање траекторија

$* \ \theta(t) = a_n t^n + \dots + a_2 t^2 + a_1 t + a_0$ → = ред на полином →



→ (бр. превои = бр. ред − 1)

= позиција на зглоб [°] во зависност од t[s]

1. $\theta = a_1 t + a_0$
   - $\theta(0) = a_0$ → поч позиција
   - $\theta(t_f) = a_1 t_f + a_0$ → крајна
3. $\theta = b_1 t^3 + b_2 t^2 + a_1 t + a_0$ → со поч. и крајна V → $V = \dot\theta$
5. $\theta = a t^5 + \dots$ → со поч и крајно a → $a = \ddot\theta$

```
rt.plot_trajectories([(theta1, t1, t2), (theta2, t2, t3) ...])
```

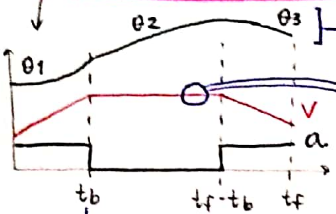
```
theta, solution = rt.trajectory_polynomial_3(ti, tf, xi, xf, vi, vf)
```

или `_5(..., ai, af)`

= коеф. пред $t^0, t^1, t^2, \dots$

ако сакаме параболични сегменти на траект

```
theta1, theta2, theta3, w, tb = rt.trajectory_linear_parabolic_segments(ti, tf, xi, xf, w, tb)
```



→ θ1 и θ3 се параболични, θ2 е линеарен

подолго време се движиме со vmax ∵

tb = време на забрзување / успорување

пр. сакаме max V

реш $V' = 0$

`sp.solveset(V', t)`

=> добиваме t за кое v е max

ако v е линеарно

$V = a_1 t + a_0$

=> $V' = a_1 \neq 0$

⇓

екстремите се на поч. и крај од интервало

∃ ф-ции само за од 3. и 5. ред

⇓

другото сами го правиме

пр.5 13

```
theta1, symbols1 = rt.polynomial(n, char='a')   -ред
v1 = theta1.diff(t)
a1 = theta1.diff(t, 2)
:
ti, tf = 0, 1   (пр)
eq1 = [
    theta1.subs(t, ti) - 20,   (ако θi = 20)
:
eq = [*eq1, *eq2, *eq3]
symbols = [*symbols1, ... *symbols3]
solution = sp.linsolve(eq, symbols)
solution.args[0]
```

ако се добиваат ∞ реш => се заменуваат произволно за парам.

```
пр. a1, a3, c4 = sp.symbols('a1, a3, a4')
    solution.subs(a1, 1).subs(a3, 1).subs(c4, 1)
    for s in zip(symbols, solutions_subbed):
        theta1 = theta1.subs(*s)
        theta2 = ...
```