Jacobian

April 21, 2024

1 Velocities and the manipulator Jacobian

First, we will load the necessary libraries.

[2]:
```
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
```

Define functions for rotational matrices and for translational matrices (in case we have a prismatic joint).

[36]:
```
def rot(u,v,t):
    # returns a 3x3-matrix decribing a rotation around (u,v) by t
    return sp.Matrix([[sp.cos(t),-sp.sin(t),-u*sp.cos(t) + u + v*sp.sin(t)],[sp.
 ↪sin(t),sp.cos(t),-u*sp.sin(t) - v*sp.cos(t) + v],[0,0,1]])
def trns(u,v):
    # returns the 3x3-matrix describing the translation by (u,v)
    return sp.Matrix([[1,0,u],[0,1,v],[0,0,1]])
```

We compute the derivative of a rotation `rot(u,v,p(t)` around a fixed point with time-dependend angle by t.

[37]:
```
u, v = sp.symbols('u v')    # coordinates of the center of rotation
t = sp.symbols('t')         # time instant
p = sp.Function('p')(t)     # a symbolic function depending on t
```

[38]:
```
rot(u,v,p)
```

[38]:
$$\begin{bmatrix} \cos(p(t)) & -\sin(p(t)) & -u\cos(p(t)) + u + v\sin(p(t)) \\ \sin(p(t)) & \cos(p(t)) & -u\sin(p(t)) - v\cos(p(t)) + v \\ 0 & 0 & 1 \end{bmatrix}$$

[35]:
```
sp.diff(rot(u,v,p), t)
```

[35]:
$$\begin{bmatrix} -\sin(p(t))\frac{d}{dt}p(t) & -\cos(p(t))\frac{d}{dt}p(t) & u\sin(p(t))\frac{d}{dt}p(t) + v\cos(p(t))\frac{d}{dt}p(t) \\ \cos(p(t))\frac{d}{dt}p(t) & -\sin(p(t))\frac{d}{dt}p(t) & -u\cos(p(t))\frac{d}{dt}p(t) + v\sin(p(t))\frac{d}{dt}p(t) \\ 0 & 0 & 0 \end{bmatrix}$$

As expected! Next, we compute

$$\left(\frac{d}{dt}\mathrm{rot}(u,v,p(t))\right)\cdot \mathrm{rot}(u,v,-p(t));$$

notice that the latter factor equals the 'inverse' of $\mathrm{rot}(u,v,p(t))$.

[40]:
```
sp.simplify(sp.diff(rot(u,v,p),t)*rot(u,v,-p))
```

[40]:
$$\begin{bmatrix} 0 & -\frac{d}{dt}p(t) & v\frac{d}{dt}p(t) \\ \frac{d}{dt}p(t) & 0 & -u\frac{d}{dt}p(t) \\ 0 & 0 & 0 \end{bmatrix}$$

Notice that

$$\left(\frac{d}{dt}\mathrm{rot}(u,v,p(t))\right)\cdot\mathrm{rot}(u,v,-p(t)) = \dot{p}(t)\cdot\begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ 0 & 0 & 0 \end{bmatrix}$$

is the 'spatial velocity' of a rigid body perfoming a rotation around (u,v) with angular velocity $p(t)$. It is nice to have a function for the latter matrix:

[43]:
```
def angvel(u,v):
    return sp.Matrix([[0,-1,v],[1,0,-u],[0,0,0]])
angvel(u,v)
```

[43]:
$$\begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ 0 & 0 & 0 \end{bmatrix}$$

During the computation of the tool velocity of a serial manipulator, we had to perfom computations of the form

$$\mathrm{rot}(a,b,q(t))\cdot\left(\frac{d}{dt}\mathrm{rot}(u,v,p(t))\right)\cdot\mathrm{rot}(u,v,-p(t))\cdot\mathrm{rot}(a,b,-q(t)).$$

This task can be done by computing

$$\mathrm{rot}(a,b,q(t))\cdot\mathrm{angvel}(u,v)\cdot\mathrm{rot}(a,b,-q(t)).$$

[45]:
```
a, b, q = sp.symbols('a b q')
sp.simplify(rot(a,b,q)*angvel(u,v)*rot(a,b,-q))
```

[45]:
$$\begin{bmatrix} 0 & -1 & -a\sin(q) - b\cos(q) + b + u\sin(q) + v\cos(q) \\ 1 & 0 & a\cos(q) - a - b\sin(q) - u\cos(q) + v\sin(q) \\ 0 & 0 & 0 \end{bmatrix}$$

There is an important observation: a rotation around (a,b) by q will move the point (u,v) to the following point:

[46]:
```
rot(a,b,q)*sp.Matrix([[u],[v],[1]])
```

[46]:
$$\begin{bmatrix} -a\cos(q) + a + b\sin(q) + u\cos(q) - v\sin(q) \\ -a\sin(q) - b\cos(q) + b + u\sin(q) + v\cos(q) \\ 1 \end{bmatrix}$$

Let (u',v') be the coordinates of the resulting point. Then we have that

$$\mathrm{rot}(a,b,q)\cdot\mathrm{angvel}(u,v)\cdot\mathrm{rot}(a,b,-q) = \mathrm{angvel}(u',v').$$

[51]:
```
result = sp.simplify(rot(a,b,q)*angvel(u,v)*rot(a,b,-q))
result - angvel(-result[1,2],result[0,2])
```

[51]:
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Identifying

$$\text{angvel}(u,v) = \begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 \\ u \\ v \end{bmatrix},$$

we learned that

$$\text{rot}(a,b,q) \cdot \text{angvel}(u,v) \cdot \text{rot}(a,b,-q) \widehat{=} \text{rot}(a,b,q) \cdot \begin{bmatrix} 1 \\ u \\ v \end{bmatrix}.$$

The very same formula remains true if we replace $\text{rot}(a,b,q)$ by $\text{trns}(a,b)$, i.e., we found that

$$g \cdot \text{angvel}(u,v) \cdot g^{-1} \widehat{=} g \cdot \begin{bmatrix} 1 \\ u \\ v \end{bmatrix}$$

is valid for all planar motions g. Summing up, we proved that

$$\begin{bmatrix} \dot{\alpha} \\ \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & x' & x'' \\ 0 & y' & y'' \end{bmatrix} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

describes the tool velocity of a planar 3R-manipulator, where

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \text{rot}(0,0,\theta_1) \cdot \begin{bmatrix} l_1 \\ 0 \\ 1 \end{bmatrix} \quad \text{and where} \quad \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \text{rot}(0,0,\theta_1) \cdot \text{rot}(l_1,0,\theta_2) \cdot \begin{bmatrix} l_1 + l_2 \\ 0 \\ 1 \end{bmatrix}.$$

We will do this in detail in the lectures; here, we only compute the latter two points.

[55]:
```
l1, l2, t1, t2, t3 = sp.symbols('l_1 l_2 t_1 t_2 t_3')
elbow = sp.Matrix([[l1],[0],[1]])
wrist = sp.Matrix([[l1+l2],[0],[1]])
```

[61]:
```
newelbow = rot(0,0,t1)*elbow
newwrist = sp.simplify(rot(0,0,t1)*rot(l1,0,t2)*wrist)
```

[62]:
```
newelbow
```

[62]:
$$\begin{bmatrix} l_1 \cos(t_1) \\ l_1 \sin(t_1) \\ 1 \end{bmatrix}$$

[63]:
```
newwrist
```

[63]:
$$\begin{bmatrix} l_1 \cos(t_1) + l_2 \cos(t_1 + t_2) \\ l_1 \sin(t_1) + l_2 \sin(t_1 + t_2) \\ 1 \end{bmatrix}$$

[67]:
```
Jacobian = sp.
↪Matrix([[1,1,1],[0,newelbow[0,0],newwrist[0,0]],[0,newelbow[1,0],newwrist[1,0]]])
Jacobian
```

[67]:
$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & l_1 \cos(t_1) & l_1 \cos(t_1) + l_2 \cos(t_1 + t_2) \\ 0 & l_1 \sin(t_1) & l_1 \sin(t_1) + l_2 \sin(t_1 + t_2) \end{bmatrix}$$

Finally, we prove that we really have the correct answer: the tool velocity is given by $\dot{g}g^{-1}$, where g denotes the position matrix of the tool at instant t.

[72]:
```
p1 = sp.Function('p1')(t)
p2 = sp.Function('p2')(t)
p3 = sp.Function('p3')(t)
position = sp.simplify(rot(0,0,p1)*rot(l1,0,p2)*rot(l1+l2,0,p3))
position
```

[72]:
$$\begin{bmatrix} \cos(p_1(t) + p_2(t) + p_3(t)) & -\sin(p_1(t) + p_2(t) + p_3(t)) & -l_1 \cos(p_1(t) + p_2(t) + p_3(t)) + l_1 \cos(p_1(t)) + l_2 \cos(p_1(\\ \sin(p_1(t) + p_2(t) + p_3(t)) & \cos(p_1(t) + p_2(t) + p_3(t)) & -l_1 \sin(p_1(t) + p_2(t) + p_3(t)) + l_1 \sin(p_1(t)) + l_2 \sin(p_1(\\ 0 & 0 & 1 \end{bmatrix}$$

Recall how to derive the inverse of the position matrix – computing the matrix will lead to very ugly expressions:

[78]:
```
inverse_position = sp.simplify(rot(l1+l2,0,-p3)*rot(l1,0,-p2)*rot(0,0,-p1))
inverse_position
```

[78]:
$$\begin{bmatrix} \cos(p_1(t) + p_2(t) + p_3(t)) & \sin(p_1(t) + p_2(t) + p_3(t)) & -l_1 \cos(p_2(t) + p_3(t)) + l_1 - l_2 \cos(p_3(t)) + l_2 \\ -\sin(p_1(t) + p_2(t) + p_3(t)) & \cos(p_1(t) + p_2(t) + p_3(t)) & l_1 \sin(p_2(t) + p_3(t)) + l_2 \sin(p_3(t)) \\ 0 & 0 & 1 \end{bmatrix}$$

[80]:
```
velocity = sp.simplify( sp.diff(position,t) * inverse_position )
velocity
```

[80]:
$$\begin{bmatrix} 0 & -\frac{d}{dt}p_1(t) - \frac{d}{dt}p_2(t) - \frac{d}{dt}p_3(t) & l_1 \sin(p_1(t)) \frac{d}{dt}p_2(t) + l_1 \sin(p_1(t)) \frac{d}{dt}p_3(t) + l_2 \sin(p_1(t \\ \frac{d}{dt}p_1(t) + \frac{d}{dt}p_2(t) + \frac{d}{dt}p_3(t) & 0 & -l_1 \cos(p_1(t)) \frac{d}{dt}p_2(t) - l_1 \cos(p_1(t)) \frac{d}{dt}p_3(t) - l_2 \cos(p_1 \\ 0 & 0 & 0 \end{bmatrix}$$

Recall that we identified

$$\dot{p} \cdot \begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \dot{p} \cdot \begin{bmatrix} 1 \\ u \\ v \end{bmatrix}$$

and write a function for that identification.

[82]:
```
def same(m):
    return sp.Matrix([[0,-m[0,0],m[2,0]],[m[0,0],0,-m[1,0]],[0,0,0]])
```

[86]:
```
joint_velocity = sp.Matrix([[sp.diff(p1)],[sp.diff(p2,t)],[sp.diff(p3,t)]])
joint_velocity
```

[86]:

$$\begin{bmatrix} \frac{d}{dt}p_1(t) \\ \frac{d}{dt}p_2(t) \\ \frac{d}{dt}p_3(t) \end{bmatrix}$$

[96]:
```
velocity2 = sp.simplify(Jacobian.subs([(t1,p1),(t2,p2),(t3,p3)])*joint_velocity)
# Notice that we have to repace t_1 = p_1(t) etc. in the Jacobian
velocity2
```

[96]:
$$\begin{bmatrix} \frac{d}{dt}p_1(t) + \frac{d}{dt}p_2(t) + \frac{d}{dt}p_3(t) \\ l_1 \cos\left(p_1(t)\right)\frac{d}{dt}p_2(t) + \left(l_1 \cos\left(p_1(t)\right) + l_2 \cos\left(p_1(t) + p_2(t)\right)\right)\frac{d}{dt}p_3(t) \\ l_1 \sin\left(p_1(t)\right)\frac{d}{dt}p_2(t) + \left(l_1 \sin\left(p_1(t)\right) + l_2 \sin\left(p_1(t) + p_2(t)\right)\right)\frac{d}{dt}p_3(t) \end{bmatrix}$$

[97]:
```
sp.expand(velocity-same(velocity2))
```

[97]:
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Since the two velocities are the same, we proved the assertion.

[ ]: