```
import numpy as np
import sympy as sym
from sympy.abc import t
%matplotlib inline
import matplotlib.pyplot as plt
import scipy.linalg
import time

#######################
# Custom latex printing
def custom_latex_printer(exp,**options):
    from google.colab.output._publish import javascript
    url = "https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.3/latest.js?config=TeX-AMS_HTML"
    javascript(url=url)
    return sym.printing.latex(exp,**options)
sym.init_printing(use_latex="mathjax",latex_printer=custom_latex_printer)

##################
# Helper Functions
def hat(w,use_sym=True):
    if use_sym:
        what = sym.Matrix([[    0,-w[2], w[1]],
                           [ w[2],    0,-w[0]],
                           [-w[1], w[0],   0]])
    else:
        what = np.array([[    0,-w[2], w[1]],
                         [ w[2],    0,-w[0]],
                         [-w[1], w[0],   0]])
    return what

def unhat(what,use_sym=True):
    if use_sym:
        w = sym.Matrix([what[2,1],what[0,2],what[1,0]])
    else:
        w = np.array([what[2,1],what[0,2],what[1,0]])
    return w

def rot(w,theta,use_sym=True):
    if use_sym:
        rotMat = sym.Matrix(sym.simplify(sym.exp(hat(w,use_sym)*theta)))
        for i in range(rotMat.shape[0]):
            for j in range(rotMat.shape[1]):
                rotMat[i,j] = sym.simplify(rotMat[i,j].rewrite(sym.sin))
    else:
        rotMat = scipy.linalg.expm(hat(w,use_sym)*theta)
    return rotMat

def T(w,th,p,use_sym=True):
    R = rot(w,th,use_sym)
    if use_sym:
        Tmat = sym.Matrix([[R[0,0],R[0,1],R[0,2],p[0]],
                           [R[1,0],R[1,1],R[1,2],p[1]],
                           [R[2,0],R[2,1],R[2,2],p[2]],
                           [     0,     0,     0,   1]])
```

```
                        [     0,     0,     0,   1]])
    else:
        Tmat = np.array([[R[0,0],R[0,1],R[0,2],p[0]],
                         [R[1,0],R[1,1],R[1,2],p[1]],
                         [R[2,0],R[2,1],R[2,2],p[2]],
                         [     0,     0,     0,   1]])
    return Tmat

def pos(T,use_sym=True):
    if use_sym:
        p = sym.Matrix([T[0,3],T[1,3],T[2,3]])
    else:
        p = np.array([T[0,3],T[1,3],T[2,3]])
    return p
```

```
v_x, v_y, omega = sym.symbols(r'v_x v_y \omega')
w = sym.Matrix([0,0,1])
w_hat = hat(w)

R = sym.eye(3) + sym.sin(omega) * w_hat + (1 - sym.cos(omega)) * (w_hat * w_hat)

display(R)
```

$$\begin{bmatrix} \cos(\omega) & -\sin(\omega) & 0 \\ \sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
v = sym.Matrix([v_x,v_y,0]) / omega
# learned: remember to normalize
t = (sym.eye(3) * omega + (1 - sym.cos(omega)) * w_hat + (omega - sym.sin(omega)) * (w_hat * w_hat)) * v
display(t)
```

$$\begin{bmatrix} \frac{v_x}{\omega}\sin(\omega) + \frac{v_y}{\omega}(\cos(\omega) - 1) \\ \frac{v_x}{\omega}(-\cos(\omega) + 1) + \frac{v_y}{\omega}\sin(\omega) \\ 0 \end{bmatrix}$$

```
t.subs({omega: 0.25, v_x: 0.05, v_y: 0}) # example
```

$$\begin{bmatrix} 0.0494807918509046 \\ 0.00621751565787105 \\ 0 \end{bmatrix}$$