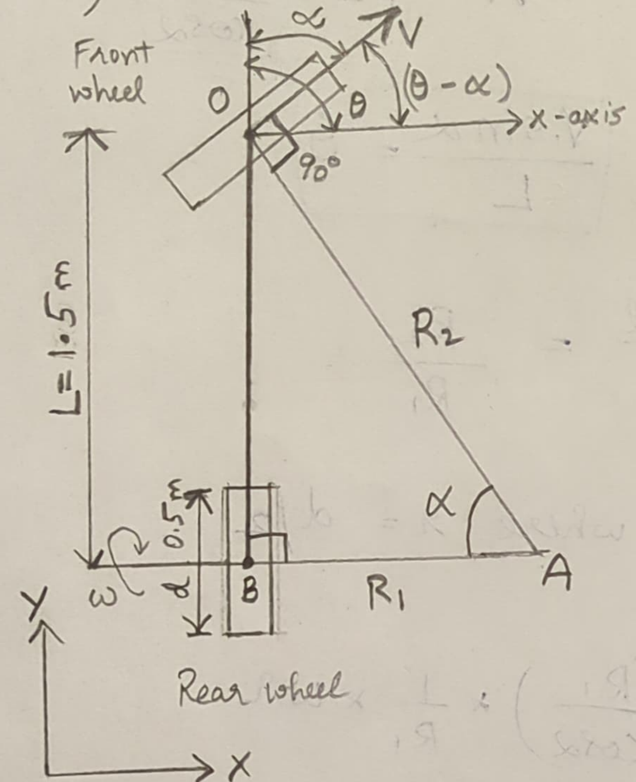
ENPM 662

HOMEWORK - 1 - Report

1.1) Rear Wheel Drive Modelling



where :-

- $\alpha \rightarrow$ steering angle
- $\theta \rightarrow$ Orientation of the Bicycle [Angle between bicycle axis and the x axis]
- $V \rightarrow$ Linear velocity of the front wheel
- $A \rightarrow$ Instantaneous center of rotation of the bicycle
- $L \rightarrow$ Distance between front and rear wheel
- $d \rightarrow$ wheel diameter
- $r \rightarrow$ wheel radius
- $\omega \rightarrow$ angular velocity of the rear wheel
- $V_x = \dot{x} \rightarrow$ Linear velocity of front wheel wrt x-axis
- $V_y = \dot{y} \rightarrow$ Linear velocity of front wheel wrt y-axis
- $\dot{\theta} \rightarrow$ rate of change of orientation of the bicycle

From the figure, At point O

$$\alpha + 90 + \angle BOA = 180^\circ \quad \left[\text{Supplementary Angles}\right]$$

$$\therefore \angle BOA = 180^\circ - 90^\circ - \alpha$$

$$\angle BOA = 90 - \alpha$$

From triangle AOB,

$$\angle BOA + \angle OBA + \angle OAB = 180^\circ$$

$$90 - \alpha + 90 + \angle OAB = 180^\circ$$

$$\therefore \angle OAB = \alpha$$

→ From triangle OAB,

$\sin\alpha = OB/OA = L/R_2$ and $\cos\alpha = \frac{AB}{OA} = \frac{R_1}{R_2}$

$\therefore \boxed{R_2 = \frac{L}{\sin\alpha}}$ $\therefore \boxed{R_2 = \frac{R_1}{\cos\alpha}}$

→ $\dot{\theta} = \frac{V}{R_2} = \frac{V}{L/\sin\alpha} = \boxed{\frac{V.\sin\alpha}{L} = \dot{\theta}}$

→ $\frac{\text{linear velocity of front wheel}}{\text{linear velocity of rear wheel}} = \frac{R_2}{R_1}$

$\Rightarrow \frac{V}{\omega r} = \frac{R_2}{R_1}$ where $r = d/2$

$\Rightarrow V = \frac{R_2}{R_1}\omega r = \left(\frac{R_1}{\cos\alpha}\right) \times \frac{1}{R_1} \times \omega r$

$\Rightarrow \boxed{V = \frac{\omega r}{\cos\alpha}}$

→ From the figure,

$V_x = \dot{x} = V\cos(\theta - \alpha)$

$V_y = \dot{y} = V\sin(\theta - \alpha)$

→ Position equations [Explained in next page]

$x(t+1) = x(t) + \dot{x}\Delta t = x(t) + [V\cos(\theta-\alpha)]\Delta t$

$y(t+1) = y(t) + \dot{y}\Delta t = y(t) + [V\sin(\theta-\alpha)]\Delta t$

$\theta(t+1) = \theta(t) + \dot{\theta}\Delta t = \theta(t) + \left[\frac{V\sin\alpha}{L}\right]\Delta t$

★ Plotting 2D trajectory of point O

given → $L = 1.5\,m$

$d = 0.5\,m$

$\therefore r = d/2 = 0.25\,m$

$\alpha = 0.5 \cdot \sin(\pi t)$

Assumptions → $x_0 = 0$, $y_0 = 0$ } origin

$\theta_0 = \frac{\pi}{2}$

$\omega = 10$ rad/sec

$T = 10$ seconds → Total time considered for the plot

$\Delta t\ \boxed{\text{or } dt} = 0.001\,s = 1\,ms$ → Time interval

⇒ The Final Position equations are derived from formula,

$$\text{velocity} = \frac{\text{Change of distance/angle}}{\text{Change of time}}$$

For example, velocity in x-direction ($\dot{x}$) ⇒

$$\dot{x} = \frac{\Delta x}{\Delta t} = \frac{x(t+1) - x(t)}{\Delta t}$$

$$\therefore \Delta t \cdot \dot{x} = x(t+1) - x(t)$$

$$\therefore x(t+1) = x(t) + \dot{x}\,\Delta t$$

Similarly we can derive for $\dot{y}$ and $\dot{\theta}$

Steps involved in solving the problem

→ Get the values of initial pose, steering angle, velocity of the wheels, and assume a sutaible value for the time period and time interval for plotting the trajectory

→ In the code, calculate the linear velocity of front wheel and establish the relationship between steering angle and time.

→ Calculate the next trajectory point / position from the formulas mentioned, by using the values which are available, assumed and calculated.

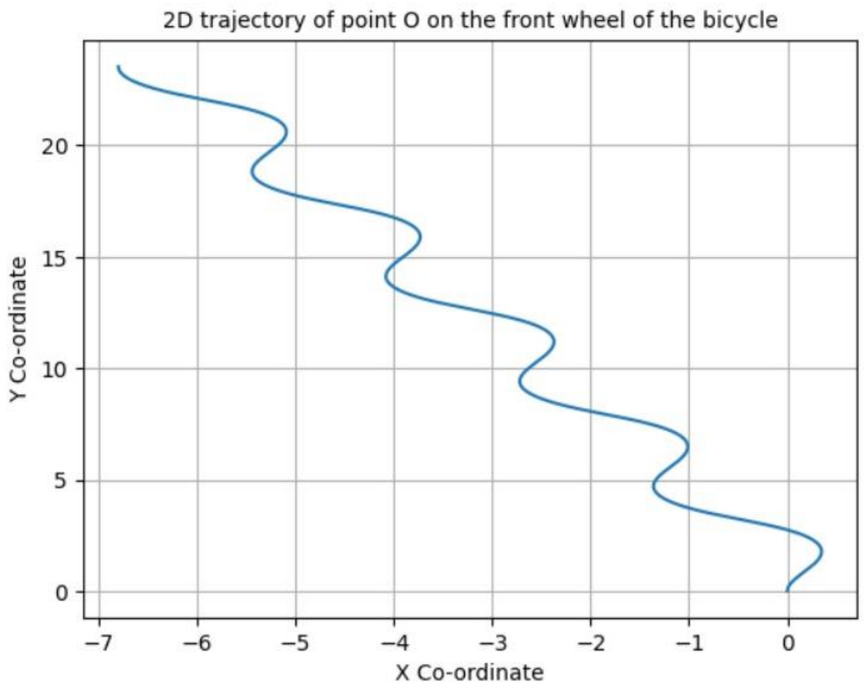
→ Repeat the same steps [above two steps] to get the whole trajectory. ~~Using~~ Use looping statement to perform this task.

→ Plot the trajectory of point O as a graph ~~by~~ [x-coordinate vs y-coordinate]

Snips of the code – Question – 1.1

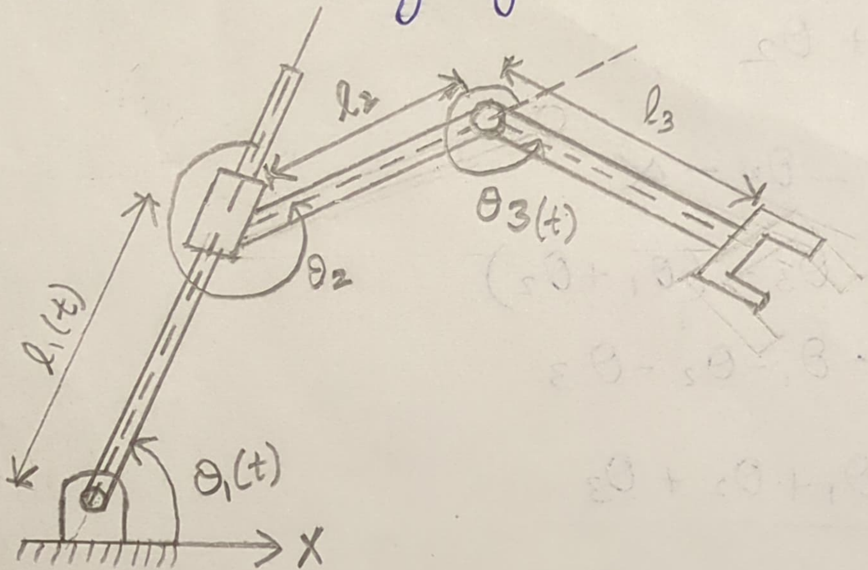
In [1]:
```
# Python program to plot the trajectory of point O on the front wheel of a bicycle

import matplotlib.pyplot as plt
import math

t = 0                   # time
dt = 0.001              # time interval - 1 millisecond
x = 0                   # x coordinate - initial value
y = 0                   # y coordinate - initial value
theta = math.pi/2       # angle between x axis and cycle frame (pose) - initial value
w = 10                  # angular velocity of rear wheel in rad/sec
r = 0.25                # radius of the wheel in m
l = 1.5                 # distance between front and rear wheel
T = 10                  # total time considered
lx = []
ly = []                 # list initialization
la = []

while t <= T :          # loop for getting plot points

    alpha = (0.5) * math.sin(math.pi * t)      #steering angle
    v = ( w * r ) / math.cos(alpha)
    theta = theta + ((( v * math.sin(alpha))/l) * dt)
    x = x + ((( v * math.cos(theta - alpha))) * dt)
    y = y + ((( v * math.sin(theta - alpha))) * dt)

    lx.append(x)
    ly.append(y)

    t = t + dt
```

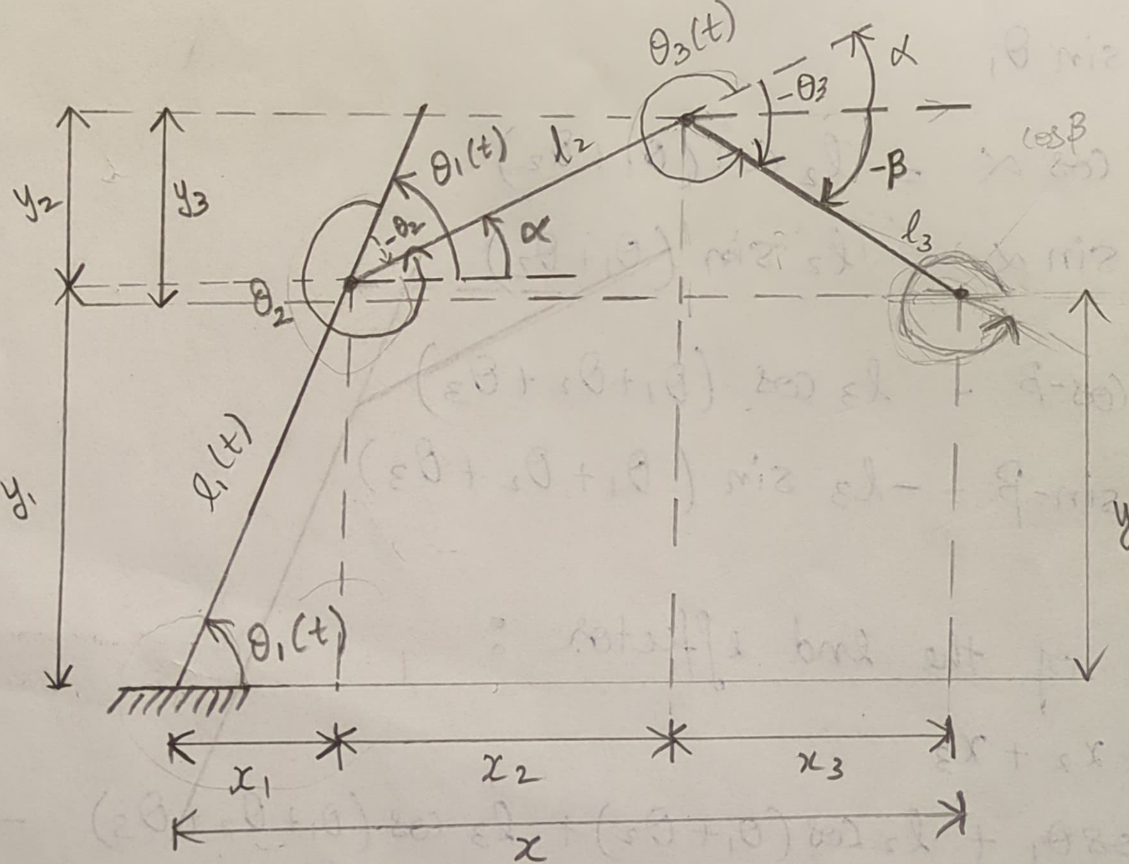
```
plt.plot(lx,ly)
plt.xlabel('X Co-ordinate')
plt.ylabel('Y Co-ordinate')
plt.title('2D trajectory of point O on the front wheel of the bicycle',fontsize=10)
plt.grid(True)
plt.show()
```



1.2) Derive the kinematic equations for a 3-DOF manipulator using geometrical method

1)





Rule used :- Angle measured clockwise is negative and Angle measured anti-clockwise is positive.

From the figure,

$$\alpha = \theta_1 - (-\theta_2)$$

$$\underline{\alpha = \theta_1 + \theta_2}$$

Also
$$-\beta = -\theta_3 - \alpha$$
$$= -\theta_3 - (\theta_1 + \theta_2)$$
$$-\beta = -\theta_1 - \theta_2 - \theta_3$$
$$\therefore \underline{\beta = \theta_1 + \theta_2 + \theta_3}$$

$$x_1 = l_1 \cos \theta_1$$
$$y_1 = l_1 \sin \theta_1$$
$$x_2 = l_2 \cos \alpha = l_2 \cos(\theta_1 + \theta_2)$$
$$y_2 = l_2 \sin \alpha = l_2 \sin(\theta_1 + \theta_2)$$
$$x_3 = l_3 \cos -\beta = l_3 \cos(\theta_1 + \theta_2 + \theta_3)$$
$$y_3 = l_3 \sin -\beta = -l_3 \sin(\theta_1 + \theta_2 + \theta_3)$$

$\therefore$ Position of the end effector:

$$x = x_1 + x_2 + x_3$$
$$x = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \quad \text{—(1)}$$

$$y = y_1 + y_2 + y_3$$
$$= l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \quad \text{—(2)}$$

Orientation of the end effector :

$$\phi = \beta = \theta_1 + \theta_2 + \theta_3 \quad — (3)$$

⇒ In order to find the velocity components of the end effector ($\dot{x}$, $\dot{y}$ and $\dot{\phi}$), differentiate equations (1), (2) and (3) with respect to time.

⇒ After getting the $\dot{x}$, $\dot{y}$ and $\dot{\phi}$ values, extract the coefficients of $\frac{d\, l_1(t)}{dt}$, $\frac{d\, \theta_1(t)}{dt}$ and $\frac{d\, \theta_3(t)}{dt}$ from the equations to construct a Jacobian matrix. Let the Jacobian matrix be of the form:

$$J = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

⇒ The forward kinematics equation is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} \frac{d}{dt} l_1(t) \\ \frac{d}{dt} \theta_1(t) \\ \frac{d}{dt} \theta_3(t) \end{bmatrix}$$

Such that :-

$$\dot{x} = a \cdot \frac{d}{dt} l_1(t) + b \cdot \frac{d}{dt} \theta_1(t) + c \cdot \frac{d}{dt} \theta_3(t)$$

$$\dot{y} = d \cdot \frac{d}{dt} l_1(t) + e \cdot \frac{d}{dt} \theta_1(t) + f \cdot \frac{d}{dt} \theta_3(t)$$

$$\dot{\phi} = g \cdot \frac{d}{dt} l_1(t) + h \cdot \frac{d}{dt} \theta_1(t) + i \cdot \frac{d}{dt} (\theta_3(t))$$

2) For inverse kinematics, take the inverse of the Jacobian matrix and multiply it on both LHS and RHS of the forward Kinematic equation

ie $$J^{-1} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \cancel{J^{-1}} \cdot \cancel{J} \cdot \begin{bmatrix} \dot{l}_1 \\ \dot{\theta}_1 \\ \dot{\theta}_3 \end{bmatrix}$$

$$\therefore \quad J^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{l}_1 \\ \dot{\theta}_1 \\ \dot{\theta}_3 \end{bmatrix}$$ → Inverse Kinematis Equation.

⇒ The derivative of position equations, the Jacobian matrix and its internal values and the inverse of Jacobian matrix are calculated by using Sympy library on Python.

Snips of the code – Question - 1.2

```
In [2]: # Python Program to derive Kinematic Equations for 3 DOF Manipulator

        from sympy import *

        x = symbols("x")              # x-coordinate of end effector
        y = symbols("y")              # y-coordinate of end effector
        phi = symbols("phi")          # Orientation of end effector
        t = symbols("t")              # Time variable
        l1 = symbols("l1")            # length of link 1
        l2 = symbols("l2")            # length of link 2
        l3 = symbols("l3")            # length of link 3
        t1 = symbols("t1")            # Angle of link 1 w.r.t x-axis (base)
        t2 = symbols("t2")            # Angle of link 2 w.r.t link 1
        t3 = symbols("t3")            # Angle of link 3 w.r.t link 2
        l1_dot = symbols("l1_dot")    # linear velocity of prismatic joint between link 1 and 2
        t1_dot = symbols("t1_dot")    # angular velocity of revolute joint between link 1 and base
        t3_dot = symbols("t3_dot")    # angular velocity of revolute joint between link 2 and 3

        # Initialization of variables as a function of time

        l1 = Function('l1')(t)        # initialization of l1 as a function of time
        t1 = Function('t1')(t)        # initialization of theta 1 as a function of time
        t3 = Function('t3')(t)        # initialization of theta 3 as a function of time

        # Velocities of different joints

        l1_dot = diff(l1,t)           # linear velocity of prismatic joint between link 1 and 2
        t1_dot = diff(t1,t)           # angular velocity of revolute joint between link 1 and base
        t3_dot = diff(t3,t)           # angular velocity of revolute joint between link 2 and 3

        # Position and Orientation Equations

        x = (l1 * cos(t1)) + (l2 * cos(t1 + t2)) + (l3 * cos(t1 + t2 + t3))
        y = (l1 * sin(t1)) + (l2 * sin(t1 + t2)) - (l3 * sin(t1 + t2 + t3))
        phi = t1 + t2 + t3  # orientation of the end effector about x-axis
```

```
        # To create a Jacobian Matrix of the form [[a,b,c],[d,e,f],[g,h,i]]

        a=x_dot.coeff(l1_dot)
        b=x_dot.coeff(t1_dot)
        c=x_dot.coeff(t3_dot)
        d=y_dot.coeff(l1_dot)
        e=y_dot.coeff(t1_dot)
        f=y_dot.coeff(t3_dot)
        g=phi_dot.coeff(l1_dot)
        h=phi_dot.coeff(t1_dot)
        i=phi_dot.coeff(t3_dot)

        J1=transpose(Matrix([a,b,c]))            # Row 1 of Jacobian Matrix
        J2=transpose(Matrix([d,e,f]))            # Row 2 of Jacobian Matrix
        J3=transpose(Matrix([g,h,i]))            # Row 3 of Jacobian Matrix

        Jacobian = Matrix.vstack(J1,J2,J3)       # Matrix of the shape 3X3

        Jacobian_inverse = Jacobian.inv()        # Inverse of Jacobian Matrix for Inverse Kinematics
```

Outputs - Question - 1.2

In [3]:
```
x_dot
```

Out[3]:

$$- l_{2} \sin\left(t_{2}\right) \cos\left(t_{1}(t)\right) \frac{d}{dt} t_{1}(t) - l_{2} \sin\left(t_{1}(t)\right) \cos\left(t_{2}\right) \frac{d}{dt} t_{1}(t) + l_{3} \sin\left(t_{2}\right) \sin\left(t_{1}(t)\right) \sin\left(t_{3}(t)\right) \frac{d}{dt} t_{1}(t) + l_{3} \sin\left(t_{2}\right) \sin\left(t_{1}(t)\right) \sin\left(t_{3}(t)\right) \frac{d}{dt} t_{3}(t) - l_{3} \sin\left(t_{2}\right) \cos\left(t_{1}(t)\right) \cos\left(t_{3}(t)\right) \frac{d}{dt} t_{1}(t) - l_{3} \sin\left(t_{2}\right) \cos\left(t_{1}(t)\right) \cos\left(t_{3}(t)\right) \frac{d}{dt} t_{3}(t) - l_{3} \sin\left(t_{1}(t)\right) \cos\left(t_{2}\right) \cos\left(t_{3}(t)\right) \frac{d}{dt} t_{1}(t) - l_{3} \sin\left(t_{1}(t)\right) \cos\left(t_{2}\right) \cos\left(t_{3}(t)\right) \frac{d}{dt} t_{3}(t) - l_{3} \sin\left(t_{3}(t)\right) \cos\left(t_{2}\right) \cos\left(t_{1}(t)\right) \frac{d}{dt} t_{1}(t) - l_{3} \sin\left(t_{3}(t)\right) \cos\left(t_{2}\right) \cos\left(t_{1}(t)\right) \frac{d}{dt} t_{3}(t) - l_{1}(t) \sin\left(t_{1}(t)\right) \frac{d}{dt} t_{1}(t) + \cos\left(t_{1}(t)\right) \frac{d}{dt} l_{1}(t)$$

In [4]:
```
y_dot
```

Out[4]:

$$- l_{2} \sin\left(t_{2}\right) \sin\left(t_{1}(t)\right) \frac{d}{dt} t_{1}(t) + l_{2} \cos\left(t_{2}\right) \cos\left(t_{1}(t)\right) \frac{d}{dt} t_{1}(t) + l_{3} \sin\left(t_{2}\right) \sin\left(t_{1}(t)\right) \cos\left(t_{3}(t)\right) \frac{d}{dt} t_{1}(t) + l_{3} \sin\left(t_{2}\right) \sin\left(t_{1}(t)\right) \cos\left(t_{3}(t)\right) \frac{d}{dt} t_{3}(t) + l_{3} \sin\left(t_{2}\right) \sin\left(t_{3}(t)\right) \cos\left(t_{1}(t)\right) \frac{d}{dt} t_{1}(t) + l_{3} \sin\left(t_{2}\right) \sin\left(t_{3}(t)\right) \cos\left(t_{1}(t)\right) \frac{d}{dt} t_{3}(t) + l_{3} \sin\left(t_{1}(t)\right) \sin\left(t_{3}(t)\right) \cos\left(t_{2}\right) \frac{d}{dt} t_{1}(t) + l_{3} \sin\left(t_{1}(t)\right) \sin\left(t_{3}(t)\right) \cos\left(t_{2}\right) \frac{d}{dt} t_{3}(t) - l_{3} \cos\left(t_{2}\right) \cos\left(t_{1}(t)\right) \cos\left(t_{3}(t)\right) \frac{d}{dt} t_{1}(t) - l_{3} \cos\left(t_{2}\right) \cos\left(t_{1}(t)\right) \cos\left(t_{3}(t)\right) \frac{d}{dt} t_{3}(t) + l_{1}(t) \cos\left(t_{1}(t)\right) \frac{d}{dt} t_{1}(t) + \sin\left(t_{1}(t)\right) \frac{d}{dt} l_{1}(t)$$

In [5]:
```
phi_dot
```

Out[5]:

$$\frac{d}{dt} t_{1}(t) + \frac{d}{dt} t_{3}(t)$$

In [9]:
```
print(Jacobian)
```

```
Matrix([[cos(t1(t)), -l2*sin(t2)*cos(t1(t)) - l2*sin(t1(t))*cos(t2) + l3*sin(t2)*sin(t1(t))*sin(t3(t)) - l3*sin(t2)*cos(t1(t))*
cos(t3(t)) - l3*sin(t1(t))*cos(t2)*cos(t3(t)) - l3*sin(t3(t))*cos(t2)*cos(t1(t)) - l1(t)*sin(t1(t)), l3*sin(t2)*sin(t1(t))*sin
(t3(t)) - l3*sin(t2)*cos(t1(t))*cos(t3(t)) - l3*sin(t1(t))*cos(t2)*cos(t3(t)) - l3*sin(t3(t))*cos(t2)*cos(t1(t))], [sin(t1(t)),
-l2*sin(t2)*sin(t1(t)) + l2*cos(t2)*cos(t1(t)) + l3*sin(t2)*sin(t1(t))*cos(t3(t)) + l3*sin(t2)*sin(t3(t))*cos(t1(t)) + l3*sin(t
1(t))*sin(t3(t))*cos(t2) - l3*cos(t2)*cos(t1(t))*cos(t3(t)) + l1(t)*cos(t1(t)), l3*sin(t2)*sin(t1(t))*cos(t3(t)) + l3*sin(t2)*s
in(t3(t))*cos(t1(t)) + l3*sin(t1(t))*sin(t3(t))*cos(t2) - l3*cos(t2)*cos(t1(t))*cos(t3(t))], [0, 1, 1]])
```

In [11]:
```
print(Jacobian_inverse)
```

```
Matrix([[(-l2*sin(t2)*sin(t1(t)) + l2*cos(t2)*cos(t1(t)) + l1(t)*cos(t1(t)))/(l2*sin(t1(t))**2*cos(t2) + l2*cos(t2)*cos(t1(t))*
*2 + l1(t)*sin(t1(t))**2 + l1(t)*cos(t1(t))**2), (l2*sin(t2)*cos(t1(t)) + l2*sin(t1(t))*cos(t2) + l1(t)*sin(t1(t)))/(l2*sin(t1
(t))**2*cos(t2) + l2*cos(t2)*cos(t1(t))**2 + l1(t)*sin(t1(t))**2 + l1(t)*cos(t1(t))**2), (l2*l3*sin(t2)**2*sin(t1(t))**2*sin(t3
(t)) - 2*l2*l3*sin(t2)**2*sin(t1(t))*cos(t1(t))*cos(t3(t)) - l2*l3*sin(t2)**2*sin(t3(t))*cos(t1(t))**2 - 2*l2*l3*sin(t2)*sin(t1
(t))**2*cos(t2)*cos(t3(t)) - 4*l2*l3*sin(t2)*sin(t1(t))*sin(t3(t))*cos(t2)*cos(t1(t)) + 2*l2*l3*sin(t2)*cos(t2)*cos(t1(t))**2*c
os(t3(t)) - l2*l3*sin(t1(t))**2*sin(t3(t))*cos(t2)**2 + 2*l2*l3*sin(t1(t))*cos(t2)**2*cos(t1(t))*cos(t3(t)) + l2*l3*sin(t3(t))*
cos(t2)**2*cos(t1(t))**2 - l3*l1(t)*sin(t2)*sin(t1(t))**2*cos(t3(t)) - 2*l3*l1(t)*sin(t2)*sin(t1(t))*sin(t3(t))*cos(t1(t)) + l3
*l1(t)*sin(t2)*cos(t1(t))**2*cos(t3(t)) - l3*l1(t)*sin(t1(t))**2*sin(t3(t))*cos(t2) + 2*l3*l1(t)*sin(t1(t))*cos(t2)*cos(t1(t))*
cos(t3(t)) + l3*l1(t)*sin(t3(t))*cos(t2)*cos(t1(t))**2)/(l2*sin(t1(t))**2*cos(t2) + l2*cos(t2)*cos(t1(t))**2 + l1(t)*sin(t1(t))
**2 + l1(t)*cos(t1(t))**2)], [-sin(t1(t))/(l2*sin(t1(t))**2*cos(t2) + l2*cos(t2)*cos(t1(t))**2 + l1(t)*sin(t1(t))**2 + l1(t)*co
s(t1(t))**2), cos(t1(t))/(l2*sin(t1(t))**2*cos(t2) + l2*cos(t2)*cos(t1(t))**2 + l1(t)*sin(t1(t))**2 + l1(t)*cos(t1(t))**2), (l3
*sin(t2)*sin(t1(t))**2*sin(t3(t)) - 2*l3*sin(t2)*sin(t1(t))*cos(t1(t))*cos(t3(t)) - l3*sin(t2)*sin(t3(t))*cos(t1(t))**2 - l3*si
n(t1(t))**2*cos(t2)*cos(t3(t)) - 2*l3*sin(t1(t))*sin(t3(t))*cos(t2)*cos(t1(t)) + l3*cos(t2)*cos(t1(t))**2*cos(t3(t)))/(l2*sin(t
1(t))**2*cos(t2) + l2*cos(t2)*cos(t1(t))**2 + l1(t)*sin(t1(t))**2 + l1(t)*cos(t1(t))**2)], [sin(t1(t))/(l2*sin(t1(t))**2*cos(t
2) + l2*cos(t2)*cos(t1(t))**2 + l1(t)*sin(t1(t))**2 + l1(t)*cos(t1(t))**2), -cos(t1(t))/(l2*sin(t1(t))**2*cos(t2) + l2*cos(t2)*
cos(t1(t))**2 + l1(t)*sin(t1(t))**2 + l1(t)*cos(t1(t))**2), (l2*sin(t1(t))**2*cos(t2) + l2*cos(t2)*cos(t1(t))**2 - l3*sin(t2)*s
in(t1(t))**2*sin(t3(t)) + 2*l3*sin(t2)*sin(t1(t))*cos(t1(t))*cos(t3(t)) + l3*sin(t2)*sin(t3(t))*cos(t1(t))**2 + l3*sin(t1(t))**
2*cos(t2)*cos(t3(t)) + 2*l3*sin(t1(t))*sin(t3(t))*cos(t2)*cos(t1(t)) - l3*cos(t2)*cos(t1(t))**2*cos(t3(t)) + l1(t)*sin(t1(t))**
2 + l1(t)*cos(t1(t))**2)/(l2*sin(t1(t))**2*cos(t2) + l2*cos(t2)*cos(t1(t))**2 + l1(t)*sin(t1(t))**2 + l1(t)*cos(t1(t))**2)]])
```