ME314 Dynamic of Machines Final Project

Author: Allen Liu

Project Description

This project is for simulating a jack inside a box with external forces exerted on it. This is the default project for the class.

Configurations

This properties of the project is shown below:

Mass of the box: $M_{\text{box}} = 50.0\ \text{kg}$

Mass of the jack: $m_{\text{jack}} = 1.0\ \text{kg}$

Length of the box: $L_{\text{box}} = 3.0\ \text{m}$

Length of the jack: $L_{\text{jack}} = 1.0\ \text{m}$

Inertia of the box: $\mathcal{I} = 416.67\ \text{kg} \cdot \text{m}^2$

Problem setup

Rigit body transformation

The rigit body transformation is shown in the code output in cell 5.

Calculating Euler-Lagrange Equation

For calculating the euler lagrange equation, I modeled the jack as 4 point masses and the box as a entire box, and the box were modeled as a entire mass. The Inertia of the box were calculated as $\mathcal{I} = \frac{1}{3}ML^2$. And we can calculate the twist $\hat{\mathcal{V}}^b = g^{-1}\dot{g}$ and $\mathcal{V}^b = (g^{-1}\dot{g})^{\check{}}$. So that kinetic energy can be calculated as $\mathcal{K} = \sum_i \mathcal{V}_i^{b^T} \mathcal{I}^{**} \mathcal{V}_i^b$, where $\mathcal{I}^{**}$ is the inertia sensor containing the mass matrix $mI_{3\text{x}3}$ and the inertia matrix $\mathcal{I}$. The potential energy can be calculated as $\mathcal{P} = \sum_i mgy_i$. Then we can can get the lagrangian as $\mathcal{L} = \mathcal{K} - \mathcal{P}$

Calculating the Impact conditions

The impact conditions were modeled as

$$P\Big|_{\tau^-}^{\tau+} = \lambda \nabla \phi$$
$$\mathcal{H}|_{\tau^-}^{\tau+} = 0$$

Where the $\vec{q}\,(\tau^+)$ and $\dot{\vec{q}}\,(\tau^-)$ are same as $\vec{q}\,(t)$ and $\dot{\vec{q}}\,(t)$. So that when we can use that to calculate the $\dot{\vec{q}}\,(\tau^+)$ right after impact.

Explaination

As shown in the video, the jack always bounces back to the opposite direction as impact, which is consistent with the rigit body total-elastic impact. And when no impact, the jack is falling freely, which is consistent with the physical model.

In [ ]:
```
import numpy as np
import sympy as sym
import matplotlib.pyplot as plt

from IPython.display import display, Markdown
```

In [ ]:
```
## Helper Functions
def Trans_Mat(theta, px, py, pz):
    return sym.Matrix([
        [   sym.cos(theta),    -sym.sin(theta),     0.,      px],
        [   sym.sin(theta),     sym.cos(theta),     0.,      py],
        [   0.,                 0.,                 1.,      pz],
        [   0.,                 0.,                 0.,      1.]
    ])

def unhat(V_hat):
    wx = V_hat[2, 1]
    wy = V_hat[0, 2]
    wz = V_hat[1, 0]
```

```python
    vx = V_hat[0, 3]
    vy = V_hat[1, 3]
    vz = V_hat[2, 3]

    return sym.Matrix([vx, vy, vz, wx, wy, wz])

def trans_inv(g):

    R = g[:3, :3]
    p = g[:3,  3]

    R_inv =  R.T
    p_inv = -R_inv * p
    return sym.Matrix([
        [R_inv[0, 0], R_inv[0, 1], R_inv[0, 2], p_inv[0]],
        [R_inv[1, 0], R_inv[1, 1], R_inv[1, 2], p_inv[1]],
        [R_inv[2, 0], R_inv[2, 1], R_inv[2, 2], p_inv[2]],
        [          0,           0,           0,        1]
    ])
```

In [ ]:
```python
## Useful functions
def integrate(f, xt, dt):
    """
    This function takes in an initial condition x(t) and a timestep dt,
    as well as a dynamical system f(x) that outputs a vector of the
    same dimension as x(t). It outputs a vector x(t+dt) at the future
    time step.

    Parameters
    ============
    dyn: Python function
        derivate of the system at a given step x(t),
        it can considered as \dot{x}(t) = func(x(t))
    xt: NumPy array
        current step x(t)
    dt:
        step size for integration

    Return
    ============
    new_xt:
        value of x(t+dt) integrated from x(t)
    """
    k1 = dt * f(xt)
    k2 = dt * f(xt+k1/2.)
    k3 = dt * f(xt+k2/2.)
    k4 = dt * f(xt+k3)
    new_xt = xt + (1/6.) * (k1+2.0*k2+2.0*k3+k4)
    return new_xt
```

In [ ]:
```python
## Definitions

# Constants
m  = 1.
M  = 50.
g  = 9.8
Lj = 1.
Lb = 5.
J  = (M*Lb**2.)/3.
t = sym.symbols(r't')
xb = sym.Function(r'x_b')(t)
xj = sym.Function(r'x_j')(t)
yb = sym.Function(r'y_b')(t)
yj = sym.Function(r'y_j')(t)
thetab = sym.Function(r'\theta_b')(t)
thetaj = sym.Function(r'\theta_j')(t)

lam = sym.Function(r'\lambda')(t)

xbdot     = xb.diff(t)
ybdot     = yb.diff(t)
thetabdot = thetab.diff(t)
xjdot     = xj.diff(t)
yjdot     = yj.diff(t)
thetajdot = thetaj.diff(t)

xbddot     = xbdot.diff(t)
ybddot     = ybdot.diff(t)
thetabddot = thetabdot.diff(t)
xjddot     = xjdot.diff(t)
yjddot     = yjdot.diff(t)
thetajddot = thetajdot.diff(t)

q     = sym.Matrix([xb, yb, thetab, xj, yj, thetaj])
qdot  = q.diff(t)
qddot = qdot.diff(t)
```

In [ ]:
```python
## Define Frames

# Box Frames
gwa = Trans_Mat(0., xb, yb, 0.)
gab = Trans_Mat(thetab, 0., 0., 0.)
gbc = Trans_Mat(0.,  Lb/2.,  Lb/2., 0.)
gbd = Trans_Mat(0., -Lb/2.,  Lb/2., 0.)
gbe = Trans_Mat(0., -Lb/2., -Lb/2., 0.)
gbf = Trans_Mat(0.,  Lb/2., -Lb/2., 0.)

gwb = sym.simplify(gwa*gab)
gwc = sym.simplify(gwb*gbc)
gwd = sym.simplify(gwb*gbd)
gwe = sym.simplify(gwb*gbe)
gwf = sym.simplify(gwb*gbf)

display(Markdown(r'**The Tranformation from world to frame $B$ $g_{WB}$ is:**'))
display(gwb)
display(Markdown(r'**The Tranformation from world to frame $C$ $g_{WC}$ is:**'))
display(gwc)
display(Markdown(r'**The Tranformation from world to frame $D$ $g_{WD}$ is:**'))
display(gwd)
display(Markdown(r'**The Tranformation from world to frame $E$ $g_{WE}$ is:**'))
display(gwe)
display(Markdown(r'**The Tranformation from world to frame $F$ $g_{WF}$ is:**'))
display(gwf)


# Jack Frames
gwg = Trans_Mat(0., xj, yj, 0.)
ggh = Trans_Mat(thetaj, 0., 0., 0.)
ghi = Trans_Mat(0.,  Lj/2.,     0., 0.)
ghj = Trans_Mat(0.,     0.,  Lj/2., 0.)
ghk = Trans_Mat(0., -Lj/2.,     0., 0.)
ghl = Trans_Mat(0.,     0., -Lj/2., 0.)

gwh = gwg*ggh
gwi = gwh*ghi
gwj = gwh*ghj
gwk = gwh*ghk
gwl = gwh*ghl

display(Markdown(r'**The Tranformation from world to frame $H$ $g_{WH}$ is:**'))
display(gwh)
display(Markdown(r'**The Tranformation from world to frame $I$ $g_{WI}$ is:**'))
display(gwi)
display(Markdown(r'**The Tranformation from world to frame $J$ $g_{WJ}$ is:**'))
display(gwj)
display(Markdown(r'**The Tranformation from world to frame $K$ $g_{WK}$ is:**'))
display(gwk)
display(Markdown(r'**The Tranformation from world to frame $L$ $g_{WL}$ is:**'))
display(gwl)
```

The Tranformation from world to frame B g_{WB} is:

$$\begin{bmatrix} \cos(\theta_b(t)) & -\sin(\theta_b(t)) & 0 & 1.0x_b(t) \\ \sin(\theta_b(t)) & \cos(\theta_b(t)) & 0 & 1.0y_b(t) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

The Tranformation from world to frame C g_{WC} is:

$$\begin{bmatrix} \cos(\theta_b(t)) & -\sin(\theta_b(t)) & 0 & 1.0x_b(t) + 2.5\sqrt{2}\cos\left(\theta_b(t) + \frac{\pi}{4}\right) \\ \sin(\theta_b(t)) & \cos(\theta_b(t)) & 0 & 1.0y_b(t) + 2.5\sqrt{2}\sin\left(\theta_b(t) + \frac{\pi}{4}\right) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

The Tranformation from world to frame D g_{WD} is:

$$\begin{bmatrix} \cos(\theta_b(t)) & -\sin(\theta_b(t)) & 0 & 1.0x_b(t) - 2.5\sqrt{2}\sin\left(\theta_b(t) + \frac{\pi}{4}\right) \\ \sin(\theta_b(t)) & \cos(\theta_b(t)) & 0 & 1.0y_b(t) + 2.5\sqrt{2}\cos\left(\theta_b(t) + \frac{\pi}{4}\right) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

The Tranformation from world to frame E g_{WE} is:

$$\begin{bmatrix} \cos(\theta_b(t)) & -\sin(\theta_b(t)) & 0 & 1.0x_b(t) - 2.5\sqrt{2}\cos\left(\theta_b(t) + \frac{\pi}{4}\right) \\ \sin(\theta_b(t)) & \cos(\theta_b(t)) & 0 & 1.0y_b(t) - 2.5\sqrt{2}\sin\left(\theta_b(t) + \frac{\pi}{4}\right) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

The Tranformation from world to frame F g_{WF} is:

$$\begin{bmatrix} \cos(\theta_b(t)) & -\sin(\theta_b(t)) & 0 & 1.0x_b(t) + 2.5\sqrt{2}\sin\left(\theta_b(t) + \frac{\pi}{4}\right) \\ \sin(\theta_b(t)) & \cos(\theta_b(t)) & 0 & 1.0y_b(t) - 2.5\sqrt{2}\cos\left(\theta_b(t) + \frac{\pi}{4}\right) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

The Tranformation from world to frame H g_{WH} is:

$$\begin{bmatrix} \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & 1.0x_j(t) \\ \sin(\theta_j(t)) & \cos(\theta_j(t)) & 0 & 1.0y_j(t) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

The Tranformation from world to frame I g_{WI} is:

$$\begin{bmatrix} \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & 1.0x_j(t) + 0.5\cos(\theta_j(t)) \\ \sin(\theta_j(t)) & \cos(\theta_j(t)) & 0 & 1.0y_j(t) + 0.5\sin(\theta_j(t)) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

The Tranformation from world to frame J g_{WJ} is:

$$\begin{bmatrix} \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & 1.0x_j(t) - 0.5\sin(\theta_j(t)) \\ \sin(\theta_j(t)) & \cos(\theta_j(t)) & 0 & 1.0y_j(t) + 0.5\cos(\theta_j(t)) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

The Tranformation from world to frame K g_{WK} is:

$$\begin{bmatrix} \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & 1.0x_j(t) - 0.5\cos(\theta_j(t)) \\ \sin(\theta_j(t)) & \cos(\theta_j(t)) & 0 & 1.0y_j(t) - 0.5\sin(\theta_j(t)) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

The Tranformation from world to frame L g_{WL} is:

$$\begin{bmatrix} \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & 1.0x_j(t) + 0.5\sin(\theta_j(t)) \\ \sin(\theta_j(t)) & \cos(\theta_j(t)) & 0 & 1.0y_j(t) - 0.5\cos(\theta_j(t)) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

In [ ]:

```
Ib_ten = sym.diag(M, M, M, 0, 0, J)
Ij_ten = sym.diag(m, m, m, 0, 0, 0)

display(Markdown(r'**Inertia tensor of the box ${\cal I}_{box}=$**'))
display(Ib_ten)
display(Markdown(r'**Inertia tensor of the jack ${\cal I}_{jack}=$**'))
display(Ij_ten)

# Box Velocity
Vb_b_hat = sym.simplify(trans_inv(gwb)*gwb.diff(t))
Vb_b     = sym.simplify(unhat(Vb_b_hat))

display(Markdown(r'**Twist of box is ${\cal V}_{box}=$**'))
display(Vb_b)

# Jack Velocities
Vb_i_hat = sym.simplify(trans_inv(gwi)*gwi.diff(t))
Vb_i     = sym.simplify(unhat(Vb_i_hat))

Vb_j_hat = sym.simplify(trans_inv(gwj)*gwj.diff(t))
Vb_j     = sym.simplify(unhat(Vb_j_hat))

Vb_k_hat = sym.simplify(trans_inv(gwk)*gwk.diff(t))
Vb_k     = sym.simplify(unhat(Vb_k_hat))

Vb_l_hat = sym.simplify(trans_inv(gwl)*gwl.diff(t))
Vb_l     = sym.simplify(unhat(Vb_l_hat))

display(Markdown(r'**Twist of first mass on jack ${\cal V}_{jack,1}=$**'))
display(Vb_i)
display(Markdown(r'**Twist of second mass on jack ${\cal V}_{jack,2}=$**'))
display(Vb_j)
display(Markdown(r'**Twist of third mass on jack ${\cal V}_{jack,3}=$**'))
display(Vb_k)
display(Markdown(r'**Twist of fourth mass on jack ${\cal V}_{jack,4}=$**'))
display(Vb_l)
```

Inertia tensor of the box ${\cal I}_{box}=$

$$\begin{bmatrix} 50.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 416.666666666667 \end{bmatrix}$$

Inertia tensor of the jack $\mathcal{I}_{jack} =$

$$\begin{bmatrix} 1.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Twist of box is $\mathcal{V}_{box} =$

$$\begin{bmatrix} 1.0 \sin(\theta_b(t)) \frac{d}{dt} y_b(t) + 1.0 \cos(\theta_b(t)) \frac{d}{dt} x_b(t) \\ -1.0 \sin(\theta_b(t)) \frac{d}{dt} x_b(t) + 1.0 \cos(\theta_b(t)) \frac{d}{dt} y_b(t) \\ 0 \\ 0 \\ 0 \\ \frac{d}{dt} \theta_b(t) \end{bmatrix}$$

Twist of first mass on jack $\mathcal{V}_{jack,1} =$

$$\begin{bmatrix} 1.0 \sin\left(\theta_j(t)\right) \frac{d}{dt} y_j(t) + 1.0 \cos\left(\theta_j(t)\right) \frac{d}{dt} x_j(t) \\ -1.0 \sin\left(\theta_j(t)\right) \frac{d}{dt} x_j(t) + 1.0 \cos\left(\theta_j(t)\right) \frac{d}{dt} y_j(t) + 0.5 \frac{d}{dt} \theta_j(t) \\ 0 \\ 0 \\ 0 \\ \frac{d}{dt} \theta_j(t) \end{bmatrix}$$

Twist of second mass on jack $\mathcal{V}_{jack,2} =$

$$\begin{bmatrix} 1.0 \sin\left(\theta_j(t)\right) \frac{d}{dt} y_j(t) + 1.0 \cos\left(\theta_j(t)\right) \frac{d}{dt} x_j(t) - 0.5 \frac{d}{dt} \theta_j(t) \\ -1.0 \sin\left(\theta_j(t)\right) \frac{d}{dt} x_j(t) + 1.0 \cos\left(\theta_j(t)\right) \frac{d}{dt} y_j(t) \\ 0 \\ 0 \\ 0 \\ \frac{d}{dt} \theta_j(t) \end{bmatrix}$$

Twist of third mass on jack $\mathcal{V}_{jack,3} =$

$$\begin{bmatrix} 1.0 \sin\left(\theta_j(t)\right) \frac{d}{dt} y_j(t) + 1.0 \cos\left(\theta_j(t)\right) \frac{d}{dt} x_j(t) \\ -1.0 \sin\left(\theta_j(t)\right) \frac{d}{dt} x_j(t) + 1.0 \cos\left(\theta_j(t)\right) \frac{d}{dt} y_j(t) - 0.5 \frac{d}{dt} \theta_j(t) \\ 0 \\ 0 \\ 0 \\ \frac{d}{dt} \theta_j(t) \end{bmatrix}$$

Twist of fourth mass on jack $\mathcal{V}_{jack,4} =$

$$\begin{bmatrix} 1.0 \sin\left(\theta_j(t)\right) \frac{d}{dt} y_j(t) + 1.0 \cos\left(\theta_j(t)\right) \frac{d}{dt} x_j(t) + 0.5 \frac{d}{dt} \theta_j(t) \\ -1.0 \sin\left(\theta_j(t)\right) \frac{d}{dt} x_j(t) + 1.0 \cos\left(\theta_j(t)\right) \frac{d}{dt} y_j(t) \\ 0 \\ 0 \\ 0 \\ \frac{d}{dt} \theta_j(t) \end{bmatrix}$$

In [ ]:
```
## Kinetic Energies

# Box
KE_b = sym.simplify((0.5*Vb_b.T @ Ib_ten @ Vb_b)[0])
display(Markdown(r'**Kinetic Energy of the box ${\cal K}_b =$**'))
display(KE_b)

# Jack
KE_1 = sym.simplify((0.5*Vb_i.T @ Ij_ten @ Vb_i)[0])
KE_2 = sym.simplify((0.5*Vb_j.T @ Ij_ten @ Vb_j)[0])
KE_3 = sym.simplify((0.5*Vb_k.T @ Ij_ten @ Vb_k)[0])
KE_4 = sym.simplify((0.5*Vb_l.T @ Ij_ten @ Vb_l)[0])

display(Markdown(r'**Kinetic Energy of the mass 1 ${\cal K}_1 =$**'))
display(KE_1)
display(Markdown(r'**Kinetic Energy of the mass 2 ${\cal K}_2 =$**'))
display(KE_2)
```

```python
display(Markdown(r'**Kinetic Energy of the mass 3 ${\cal K}_3 =$**'))
display(KE_3)
display(Markdown(r'**Kinetic Energy of the mass 4 ${\cal K}_4 =$**'))
display(KE_4)

KE_j = sym.simplify(KE_1 + KE_2 + KE_3 + KE_4)

display(Markdown(r'**Kinetic Energy of the jack ${\cal K}_j =$**'))
display(KE_j)

display(Markdown(r'**Total Kinetic Energy of the system ${\cal K}_{sys} =$**'))
KE = sym.simplify(KE_b + KE_j)
display(KE)
```

Kinetic Energy of the box $\mathcal{K}_b =$

$$208.333333333333\left(\frac{d}{dt}\theta_b(t)\right)^2 + 25.0\left(\frac{d}{dt}x_b(t)\right)^2 + 25.0\left(\frac{d}{dt}y_b(t)\right)^2$$

Kinetic Energy of the mass 1 $\mathcal{K}_1 =$

$$-0.5\sin\left(\theta_j(t)\right)\frac{d}{dt}\theta_j(t)\frac{d}{dt}x_j(t) + 0.5\cos\left(\theta_j(t)\right)\frac{d}{dt}\theta_j(t)\frac{d}{dt}y_j(t) + 0.125\left(\frac{d}{dt}\theta_j(t)\right)^2 + 0.5\left(\frac{d}{dt}x_j(t)\right)^2 + 0.5\left(\frac{d}{dt}y_j(t)\right)^2$$

Kinetic Energy of the mass 2 $\mathcal{K}_2 =$

$$-0.5\sin\left(\theta_j(t)\right)\frac{d}{dt}\theta_j(t)\frac{d}{dt}y_j(t) - 0.5\cos\left(\theta_j(t)\right)\frac{d}{dt}\theta_j(t)\frac{d}{dt}x_j(t) + 0.125\left(\frac{d}{dt}\theta_j(t)\right)^2 + 0.5\left(\frac{d}{dt}x_j(t)\right)^2 + 0.5\left(\frac{d}{dt}y_j(t)\right)^2$$

Kinetic Energy of the mass 3 $\mathcal{K}_3 =$

$$0.5\sin\left(\theta_j(t)\right)\frac{d}{dt}\theta_j(t)\frac{d}{dt}x_j(t) - 0.5\cos\left(\theta_j(t)\right)\frac{d}{dt}\theta_j(t)\frac{d}{dt}y_j(t) + 0.125\left(\frac{d}{dt}\theta_j(t)\right)^2 + 0.5\left(\frac{d}{dt}x_j(t)\right)^2 + 0.5\left(\frac{d}{dt}y_j(t)\right)^2$$

Kinetic Energy of the mass 4 $\mathcal{K}_4 =$

$$0.5\sin\left(\theta_j(t)\right)\frac{d}{dt}\theta_j(t)\frac{d}{dt}y_j(t) + 0.5\cos\left(\theta_j(t)\right)\frac{d}{dt}\theta_j(t)\frac{d}{dt}x_j(t) + 0.125\left(\frac{d}{dt}\theta_j(t)\right)^2 + 0.5\left(\frac{d}{dt}x_j(t)\right)^2 + 0.5\left(\frac{d}{dt}y_j(t)\right)^2$$

Kinetic Energy of the jack $\mathcal{K}_j =$

$$0.5\left(\frac{d}{dt}\theta_j(t)\right)^2 + 2.0\left(\frac{d}{dt}x_j(t)\right)^2 + 2.0\left(\frac{d}{dt}y_j(t)\right)^2$$

Total Kinetic Energy of the system $\mathcal{K}_{sys} =$

$$208.333333333333\left(\frac{d}{dt}\theta_b(t)\right)^2 + 0.5\left(\frac{d}{dt}\theta_j(t)\right)^2 + 25.0\left(\frac{d}{dt}x_b(t)\right)^2 + 2.0\left(\frac{d}{dt}x_j(t)\right)^2 + 25.0\left(\frac{d}{dt}y_b(t)\right)^2 + 2.0\left(\frac{d}{dt}y_j(t)\right)^2$$

In [ ]:

```python
## Potential Energy

# Box
hb = sym.simplify((sym.Matrix([[0, 1, 0, 0]]) * gwb * sym.Matrix([0, 0, 0, 1]))[0])
Vb = M*g*hb

display(Markdown(r'**Potential Energy of the box is <br/>${\cal P}_b=$**'))
display(Vb)

# Jack
h1 = sym.simplify((sym.Matrix([[0, 1, 0, 0]]) * gwi * sym.Matrix([0, 0, 0, 1]))[0])
h2 = sym.simplify((sym.Matrix([[0, 1, 0, 0]]) * gwj * sym.Matrix([0, 0, 0, 1]))[0])
h3 = sym.simplify((sym.Matrix([[0, 1, 0, 0]]) * gwk * sym.Matrix([0, 0, 0, 1]))[0])
h4 = sym.simplify((sym.Matrix([[0, 1, 0, 0]]) * gwl * sym.Matrix([0, 0, 0, 1]))[0])

V1 = m*g*h1
V2 = m*g*h2
V3 = m*g*h3
V4 = m*g*h4

Vj = sym.simplify(V1 + V2 + V3 + V4)
display(Markdown(r'**Potential Energy of the jack is <br/>${\cal P}_b=$**'))
display(Vj)

V = sym.simplify(Vb + Vj)
# V = Vj
display(Markdown(r'**Total Potential Energy of the system is <br/>${\cal P}_{sys}=$**'))
display(V)
```

Potential Energy of the box is
$\mathcal{P}_b =$

$$490.0y_b(t)$$

Potential Energy of the jack is
$\mathcal{P}_b =$

$$39.2y_j(t)$$

Total Potential Energy of the system is
$\mathcal{P}_{sys} =$

$$490.0y_b(t) + 39.2y_j(t)$$

In [ ]:
```
## Lagrangian and Euler Lagrange Equation

# Lagrangian
L = sym.simplify(KE - V)

display(Markdown(r'**Lagrangian of the System is: <br/>${\cal L}_{sys}=$**'))
display(L)

# Euler-Langrange
L_mat      = sym.Matrix([L])
dLdq       = sym.simplify(L_mat.jacobian(q).T)
dLdqdot    = sym.simplify(L_mat.jacobian(qdot)).T
ddtdLdqdot = sym.simplify(dLdqdot.diff(t))

EL = sym.simplify(ddtdLdqdot - dLdq)
display(EL)

xb  = sym.simplify((sym.Matrix([[1, 0, 0, 0]]) * gwa * sym.Matrix([0, 0, 0, 1]))[0])
yb  = sym.simplify((sym.Matrix([[0, 1, 0, 0]]) * gwa * sym.Matrix([0, 0, 0, 1]))[0])
phi = sym.simplify(xb**2 + yb**2)

phi_mat = sym.Matrix([phi])
dphidq  = sym.simplify(phi_mat.jacobian(q).T)
cons    = lam*dphidq
display(cons)

# Force
F_vec = sym.Matrix([0, -10.0*M*ybdot + M*g + 100., 40., 0, 0, 0])

phiddot = sym.simplify(phi.diff(t).diff(t))
# lfs = sym.Matrix([EL, phiddot])
lfs = EL
# rhs = sym.Matrix([cons, sym.Matrix([0])])
# rhs = sym.Matrix([0, 0, 0, 0, 0, 0])
rhs = F_vec

EL_eqn = sym.Eq(lfs, rhs)
display(EL_eqn)
```

Lagrangian of the System is:
$\mathcal{L}_{sys} =$

$$-490.0y_b(t) - 39.2y_j(t) + 208.333333333333\left(\frac{d}{dt}\theta_b(t)\right)^2 + 0.5\left(\frac{d}{dt}\theta_j(t)\right)^2 + 25.0\left(\frac{d}{dt}x_b(t)\right)^2 + 2.0\left(\frac{d}{dt}x_j(t)\right)^2 + 25.0\left(\frac{d}{dt}y_b(t)\right)^2 + 2.0\left(\frac{d}{dt}y_j(t)\right)$$

$$\begin{bmatrix} 50.0\frac{d^2}{dt^2}x_b(t) \\ 50.0\frac{d^2}{dt^2}y_b(t) + 490.0 \\ 416.666666666667\frac{d^2}{dt^2}\theta_b(t) \\ 4.0\frac{d^2}{dt^2}x_j(t) \\ 4.0\frac{d^2}{dt^2}y_j(t) + 39.2 \\ 1.0\frac{d^2}{dt^2}\theta_j(t) \end{bmatrix}$$

$$\begin{bmatrix} 2\lambda(t)x_b(t) \\ 2\lambda(t)y_b(t) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 50.0\frac{d^2}{dt^2}x_b(t) \\ 50.0\frac{d^2}{dt^2}y_b(t) + 490.0 \\ 416.666666666667\frac{d^2}{dt^2}\theta_b(t) \\ 4.0\frac{d^2}{dt^2}x_j(t) \\ 4.0\frac{d^2}{dt^2}y_j(t) + 39.2 \\ 1.0\frac{d^2}{dt^2}\theta_j(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 590.0 - 500.0\frac{d}{dt}y_b(t) \\ 40.0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In [ ]:
```
## Solve Equation

# vars = sym.Matrix([qddot, lam])
vars = qddot
sols = sym.solve(EL_eqn, vars, dict=True)
```

```
funcs = []

for sol in sols:
    for v in vars:
        sol_sim = sym.simplify(sol[v])
        display(sym.Eq(v, sol_sim))
        funcs.append(sym.lambdify(
            [
                xb,
                yb,
                thetab,
                xj,
                yj,
                thetaj,
                xbdot,
                ybdot,
                thetabdot,
                xjdot,
                yjdot,
                thetajdot
            ],
            sol_sim
        ))
```

$$\frac{d^2}{dt^2}x_b(t) = 0.0$$

$$\frac{d^2}{dt^2}y_b(t) = 2.0 - 10.0\frac{d}{dt}y_b(t)$$

$$\frac{d^2}{dt^2}\theta_b(t) = 0.096$$

$$\frac{d^2}{dt^2}x_j(t) = 0.0$$

$$\frac{d^2}{dt^2}y_j(t) = -9.8$$

$$\frac{d^2}{dt^2}\theta_j(t) = 0.0$$

In [ ]:

```
def func_xbddot(s):
    return funcs[0](*s)

def func_ybddot(s):
    return funcs[1](*s)

def func_thetabddot(s):
    return funcs[2](*s)

def func_xjddot(s):
    return funcs[3](*s)

def func_yjddot(s):
    return funcs[4](*s)

def func_thetajddot(s):
    return funcs[5](*s)

def dyn(s):
    xb_v     = s[0]
    yb_v     = s[1]
    thetab_v = s[2]

    xj_v     = s[3]
    yj_v     = s[4]
    thetaj_v = s[5]

    xbdot_v     = s[6]
    ybdot_v     = s[7]
    thetabdot_v = s[8]

    xjdot_v     = s[9]
    yjdot_v     = s[10]
    thetajdot_v = s[11]

    xbddot_v = func_xbddot(s)
    ybddot_v = func_ybddot(s)
    thetabddot_v = func_thetabddot(s)

    xjddot_v = func_xjddot(s)
    yjddot_v = func_yjddot(s)
    thetajddot_v = func_thetajddot(s)

    return np.array([
        xbdot_v,
        ybdot_v,
```

```
        thetabdot_v,
        xjdot_v,
        yjdot_v,
        thetajdot_v,
        xbddot_v,
        ybddot_v,
        thetabddot_v,
        xjddot_v,
        yjddot_v,
        thetajddot_v,
    ])
```

In [ ]:
```
## Get all frame locations
lam = sym.symbols(r'\lambda')

rC = sym.Matrix((gwc @ sym.Matrix([0, 0, 0, 1]))[0:3])
rD = sym.Matrix((gwd @ sym.Matrix([0, 0, 0, 1]))[0:3])
rE = sym.Matrix((gwe @ sym.Matrix([0, 0, 0, 1]))[0:3])
rF = sym.Matrix((gwf @ sym.Matrix([0, 0, 0, 1]))[0:3])

rI = sym.Matrix((gwi @ sym.Matrix([0, 0, 0, 1]))[0:3])
rJ = sym.Matrix((gwj @ sym.Matrix([0, 0, 0, 1]))[0:3])
rK = sym.Matrix((gwk @ sym.Matrix([0, 0, 0, 1]))[0:3])
rL = sym.Matrix((gwl @ sym.Matrix([0, 0, 0, 1]))[0:3])

# C
rCD = sym.simplify(rD - rC)
rCI = sym.simplify(rI - rC)
rCJ = sym.simplify(rJ - rC)
rCK = sym.simplify(rK - rC)
rCL = sym.simplify(rL - rC)

# D
rDE = sym.simplify(rE - rD)
rDI = sym.simplify(rI - rD)
rDJ = sym.simplify(rJ - rD)
rDK = sym.simplify(rK - rD)
rDL = sym.simplify(rL - rD)

# E
rEF = sym.simplify(rF - rE)
rEI = sym.simplify(rI - rE)
rEJ = sym.simplify(rJ - rE)
rEK = sym.simplify(rK - rE)
rEL = sym.simplify(rL - rE)

# F
rFC = sym.simplify(rC - rF)
rFI = sym.simplify(rI - rF)
rFJ = sym.simplify(rJ - rF)
rFK = sym.simplify(rK - rF)
rFL = sym.simplify(rL - rF)
```

In [ ]:
```
## Impact conditions
# I and CD
phi1 = sym.simplify((rCD.cross(rCI))[2])
phi1_mat = sym.Matrix([phi1])
dphi1dq = sym.simplify(phi1_mat.jacobian(q).T)
cons1 = lam*dphi1dq
func_phi1 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi1)

display(Markdown(r'**Impact Constraint 1 is:**'))
display(Markdown(r'$\phi_1=$'))
display(phi1)
display(Markdown(r'$\lambda \nabla \phi_1=$'))
display(cons1)
```

Impact Constraint 1 is:

$\phi_1=$

$$-5.0x_b(t)\sin(\theta_b(t)) + 5.0x_j(t)\sin(\theta_b(t)) + 5.0y_b(t)\cos(\theta_b(t)) - 5.0y_j(t)\cos(\theta_b(t)) + 2.5\sin\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda \nabla \phi_1=$

$$\begin{bmatrix} -5.0\lambda\sin(\theta_b(t)) \\ 5.0\lambda\cos(\theta_b(t)) \\ \lambda\left(-5.0x_b(t)\cos(\theta_b(t)) + 5.0x_j(t)\cos(\theta_b(t)) - 5.0y_b(t)\sin(\theta_b(t)) + 5.0y_j(t)\sin(\theta_b(t)) + 2.5\cos\left(\theta_b(t) - \theta_j(t)\right)\right) \\ 5.0\lambda\sin(\theta_b(t)) \\ -5.0\lambda\cos(\theta_b(t)) \\ -2.5\lambda\cos\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:
```
# I and DE
phi2 = sym.simplify((rDE.cross(rDI))[2])
phi2_mat = sym.Matrix([phi2])
dphi2dq = sym.simplify(phi2_mat.jacobian(q).T)
```

```python
cons2 = lam*dphi2dq
func_phi2 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi2)

display(Markdown(r'**Impact Constraint 2 is:**'))
display(Markdown(r'$\phi_2=$'))
display(phi2)
display(Markdown(r'$\lambda \nabla \phi_2=$'))
display(cons2)
```

Impact Constraint 2 is:

$\phi_2 =$

$$-5.0x_b(t)\cos(\theta_b(t)) + 5.0x_j(t)\cos(\theta_b(t)) - 5.0y_b(t)\sin(\theta_b(t)) + 5.0y_j(t)\sin(\theta_b(t)) + 2.5\cos(\theta_b(t) - \theta_j(t)) + 12.5$$

$\lambda\nabla\phi_2 =$

$$\begin{bmatrix} -5.0\lambda\cos(\theta_b(t)) \\ -5.0\lambda\sin(\theta_b(t)) \\ \lambda\left(5.0x_b(t)\sin(\theta_b(t)) - 5.0x_j(t)\sin(\theta_b(t)) - 5.0y_b(t)\cos(\theta_b(t)) + 5.0y_j(t)\cos(\theta_b(t)) - 2.5\sin(\theta_b(t) - \theta_j(t))\right) \\ 5.0\lambda\cos(\theta_b(t)) \\ 5.0\lambda\sin(\theta_b(t)) \\ 2.5\lambda\sin(\theta_b(t) - \theta_j(t)) \end{bmatrix}$$

In [ ]:
```python
# I and EF
phi3 = sym.simplify((rEF.cross(rEI))[2])
phi3_mat = sym.Matrix([phi3])
dphi3dq = sym.simplify(phi3_mat.jacobian(q).T)
cons3 = lam*dphi3dq
func_phi3 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi3)

display(Markdown(r'**Impact Constraint 3 is:**'))
display(Markdown(r'$\phi_3=$'))
display(phi3)
display(Markdown(r'$\lambda \nabla \phi_3=$'))
display(cons3)
```

Impact Constraint 3 is:

$\phi_3 =$

$$5.0x_b(t)\sin(\theta_b(t)) - 5.0x_j(t)\sin(\theta_b(t)) - 5.0y_b(t)\cos(\theta_b(t)) + 5.0y_j(t)\cos(\theta_b(t)) - 2.5\sin(\theta_b(t) - \theta_j(t)) + 12.5$$

$\lambda\nabla\phi_3 =$

$$\begin{bmatrix} 5.0\lambda\sin(\theta_b(t)) \\ -5.0\lambda\cos(\theta_b(t)) \\ \lambda\left(5.0x_b(t)\cos(\theta_b(t)) - 5.0x_j(t)\cos(\theta_b(t)) + 5.0y_b(t)\sin(\theta_b(t)) - 5.0y_j(t)\sin(\theta_b(t)) - 2.5\cos(\theta_b(t) - \theta_j(t))\right) \\ -5.0\lambda\sin(\theta_b(t)) \\ 5.0\lambda\cos(\theta_b(t)) \\ 2.5\lambda\cos(\theta_b(t) - \theta_j(t)) \end{bmatrix}$$

In [ ]:
```python
# I and FC
phi4 = sym.simplify((rFC.cross(rFI))[2])
phi4_mat = sym.Matrix([phi4])
dphi4dq = sym.simplify(phi4_mat.jacobian(q).T)
cons4 = lam*dphi4dq
func_phi4 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi4)

display(Markdown(r'**Impact Constraint 4 is:**'))
display(Markdown(r'$\phi_4=$'))
display(phi4)
display(Markdown(r'$\lambda \nabla \phi_4=$'))
display(cons4)
```

Impact Constraint 4 is:

$\phi_4 =$

$$5.0x_b(t)\cos(\theta_b(t)) - 5.0x_j(t)\cos(\theta_b(t)) + 5.0y_b(t)\sin(\theta_b(t)) - 5.0y_j(t)\sin(\theta_b(t)) - 2.5\cos(\theta_b(t) - \theta_j(t)) + 12.5$$

$\lambda\nabla\phi_4 =$

$$\begin{bmatrix} 5.0\lambda\cos(\theta_b(t)) \\ 5.0\lambda\sin(\theta_b(t)) \\ \lambda\left(-5.0x_b(t)\sin(\theta_b(t)) + 5.0x_j(t)\sin(\theta_b(t)) + 5.0y_b(t)\cos(\theta_b(t)) - 5.0y_j(t)\cos(\theta_b(t)) + 2.5\sin(\theta_b(t) - \theta_j(t))\right) \\ -5.0\lambda\cos(\theta_b(t)) \\ -5.0\lambda\sin(\theta_b(t)) \\ -2.5\lambda\sin(\theta_b(t) - \theta_j(t)) \end{bmatrix}$$

In [ ]:
```python
# J and CD
phi5 = sym.simplify((rCD.cross(rCJ))[2])
phi5_mat = sym.Matrix([phi5])
dphi5dq = sym.simplify(phi5_mat.jacobian(q).T)
cons5 = lam*dphi5dq
```

```
func_phi5 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi5)

display(Markdown(r'**Impact Constraint 5 is:**'))
display(Markdown(r'$\phi_5=$'))
display(phi5)
display(Markdown(r'$\lambda \nabla \phi_5=$'))
display(cons5)
```

Impact Constraint 5 is:

$\phi_5 =$

$$-5.0x_b(t)\sin(\theta_b(t)) + 5.0x_j(t)\sin(\theta_b(t)) + 5.0y_b(t)\cos(\theta_b(t)) - 5.0y_j(t)\cos(\theta_b(t)) - 2.5\cos\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda\nabla\phi_5 =$

$$\begin{bmatrix} -5.0\lambda\sin(\theta_b(t)) \\ 5.0\lambda\cos(\theta_b(t)) \\ \lambda\left(-5.0x_b(t)\cos(\theta_b(t)) + 5.0x_j(t)\cos(\theta_b(t)) - 5.0y_b(t)\sin(\theta_b(t)) + 5.0y_j(t)\sin(\theta_b(t)) + 2.5\sin\left(\theta_b(t) - \theta_j(t)\right)\right) \\ 5.0\lambda\sin(\theta_b(t)) \\ -5.0\lambda\cos(\theta_b(t)) \\ -2.5\lambda\sin\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:
```
# J and DE
phi6 = sym.simplify((rDE.cross(rDJ))[2])
phi6_mat = sym.Matrix([phi6])
dphi6dq = sym.simplify(phi6_mat.jacobian(q).T)
cons6 = lam*dphi6dq
func_phi6 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi6)

display(Markdown(r'**Impact Constraint 6 is:**'))
display(Markdown(r'$\phi_6=$'))
display(phi6)
display(Markdown(r'$\lambda \nabla \phi_6=$'))
display(cons6)
```

Impact Constraint 6 is:

$\phi_6 =$

$$-5.0x_b(t)\cos(\theta_b(t)) + 5.0x_j(t)\cos(\theta_b(t)) - 5.0y_b(t)\sin(\theta_b(t)) + 5.0y_j(t)\sin(\theta_b(t)) + 2.5\sin\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda\nabla\phi_6 =$

$$\begin{bmatrix} -5.0\lambda\cos(\theta_b(t)) \\ -5.0\lambda\sin(\theta_b(t)) \\ \lambda\left(5.0x_b(t)\sin(\theta_b(t)) - 5.0x_j(t)\sin(\theta_b(t)) - 5.0y_b(t)\cos(\theta_b(t)) + 5.0y_j(t)\cos(\theta_b(t)) + 2.5\cos\left(\theta_b(t) - \theta_j(t)\right)\right) \\ 5.0\lambda\cos(\theta_b(t)) \\ 5.0\lambda\sin(\theta_b(t)) \\ -2.5\lambda\cos\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:
```
# J and EF
phi7 = sym.simplify((rEF.cross(rEJ))[2])
phi7_mat = sym.Matrix([phi7])
dphi7dq = sym.simplify(phi7_mat.jacobian(q).T)
cons7 = lam*dphi7dq
func_phi7 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi7)

display(Markdown(r'**Impact Constraint 7 is:**'))
display(Markdown(r'$\phi_7=$'))
display(phi7)
display(Markdown(r'$\lambda \nabla \phi_7=$'))
display(cons7)
```

Impact Constraint 7 is:

$\phi_7 =$

$$5.0x_b(t)\sin(\theta_b(t)) - 5.0x_j(t)\sin(\theta_b(t)) - 5.0y_b(t)\cos(\theta_b(t)) + 5.0y_j(t)\cos(\theta_b(t)) + 2.5\cos\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda\nabla\phi_7 =$

$$\begin{bmatrix} 5.0\lambda\sin(\theta_b(t)) \\ -5.0\lambda\cos(\theta_b(t)) \\ \lambda\left(5.0x_b(t)\cos(\theta_b(t)) - 5.0x_j(t)\cos(\theta_b(t)) + 5.0y_b(t)\sin(\theta_b(t)) - 5.0y_j(t)\sin(\theta_b(t)) - 2.5\sin\left(\theta_b(t) - \theta_j(t)\right)\right) \\ -5.0\lambda\sin(\theta_b(t)) \\ 5.0\lambda\cos(\theta_b(t)) \\ 2.5\lambda\sin\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:
```
# J and FC
phi8 = sym.simplify((rFC.cross(rFJ))[2])
phi8_mat = sym.Matrix([phi8])
dphi8dq = sym.simplify(phi8_mat.jacobian(q).T)
cons8 = lam*dphi8dq
func_phi8 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi8)
```

```
display(Markdown(r'**Impact Constraint 8 is:**'))
display(Markdown(r'$\phi_8=$'))
display(phi8)
display(Markdown(r'$\lambda \nabla \phi_8=$'))
display(cons8)
```

Impact Constraint 8 is:

$\phi_8 =$

$$5.0x_b(t)\cos(\theta_b(t)) - 5.0x_j(t)\cos(\theta_b(t)) + 5.0y_b(t)\sin(\theta_b(t)) - 5.0y_j(t)\sin(\theta_b(t)) - 2.5\sin\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda\nabla\phi_8 =$

$$\left[\begin{matrix}5.0\lambda\cos(\theta_b(t))\\ 5.0\lambda\sin(\theta_b(t))\\ \lambda\left(-5.0x_b(t)\sin(\theta_b(t)) + 5.0x_j(t)\sin(\theta_b(t)) + 5.0y_b(t)\cos(\theta_b(t)) - 5.0y_j(t)\cos(\theta_b(t)) - 2.5\cos\left(\theta_b(t) - \theta_j(t)\right)\right)\\ -5.0\lambda\cos(\theta_b(t))\\ -5.0\lambda\sin(\theta_b(t))\\ 2.5\lambda\cos\left(\theta_b(t) - \theta_j(t)\right)\end{matrix}\right]$$

In [ ]:
```
# K and CD
phi9 = sym.simplify((rCD.cross(rCK))[2])
phi9_mat = sym.Matrix([phi9])
dphi9dq = sym.simplify(phi9_mat.jacobian(q).T)
cons9 = lam*dphi9dq
func_phi9 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi9)

display(Markdown(r'**Impact Constraint 9 is:**'))
display(Markdown(r'$\phi_9=$'))
display(phi9)
display(Markdown(r'$\lambda \nabla \phi_9=$'))
display(cons9)
```

Impact Constraint 9 is:

$\phi_9 =$

$$-5.0x_b(t)\sin(\theta_b(t)) + 5.0x_j(t)\sin(\theta_b(t)) + 5.0y_b(t)\cos(\theta_b(t)) - 5.0y_j(t)\cos(\theta_b(t)) - 2.5\sin\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda\nabla\phi_9 =$

$$\left[\begin{matrix}-5.0\lambda\sin(\theta_b(t))\\ 5.0\lambda\cos(\theta_b(t))\\ \lambda\left(-5.0x_b(t)\cos(\theta_b(t)) + 5.0x_j(t)\cos(\theta_b(t)) - 5.0y_b(t)\sin(\theta_b(t)) + 5.0y_j(t)\sin(\theta_b(t)) - 2.5\cos\left(\theta_b(t) - \theta_j(t)\right)\right)\\ 5.0\lambda\sin(\theta_b(t))\\ -5.0\lambda\cos(\theta_b(t))\\ 2.5\lambda\cos\left(\theta_b(t) - \theta_j(t)\right)\end{matrix}\right]$$

In [ ]:
```
# K and DE
phi10 = sym.simplify((rDE.cross(rDK))[2])
phi10_mat = sym.Matrix([phi10])
dphi10dq = sym.simplify(phi10_mat.jacobian(q).T)
cons10 = lam*dphi10dq
func_phi10 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi10)

display(Markdown(r'**Impact Constraint 10 is:**'))
display(Markdown(r'$\phi_{10}=$'))
display(phi10)
display(Markdown(r'$\lambda \nabla \phi_{10}=$'))
display(cons10)
```

Impact Constraint 10 is:

$\phi_{10} =$

$$-5.0x_b(t)\cos(\theta_b(t)) + 5.0x_j(t)\cos(\theta_b(t)) - 5.0y_b(t)\sin(\theta_b(t)) + 5.0y_j(t)\sin(\theta_b(t)) - 2.5\cos\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda\nabla\phi_{10} =$

$$\left[\begin{matrix}-5.0\lambda\cos(\theta_b(t))\\ -5.0\lambda\sin(\theta_b(t))\\ \lambda\left(5.0x_b(t)\sin(\theta_b(t)) - 5.0x_j(t)\sin(\theta_b(t)) - 5.0y_b(t)\cos(\theta_b(t)) + 5.0y_j(t)\cos(\theta_b(t)) + 2.5\sin\left(\theta_b(t) - \theta_j(t)\right)\right)\\ 5.0\lambda\cos(\theta_b(t))\\ 5.0\lambda\sin(\theta_b(t))\\ -2.5\lambda\sin\left(\theta_b(t) - \theta_j(t)\right)\end{matrix}\right]$$

In [ ]:
```
# K and EF
phi11 = sym.simplify((rEF.cross(rEK))[2])
phi11_mat = sym.Matrix([phi11])
dphi11dq = sym.simplify(phi11_mat.jacobian(q).T)
cons11 = lam*dphi11dq
func_phi11 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi11)
```

```python
display(Markdown(r'**Impact Constraint 11 is:**'))
display(Markdown(r'$\phi_{11}=$'))
display(phi11)
display(Markdown(r'$\lambda \nabla \phi_{11}=$'))
display(cons11)
```

Impact Constraint 11 is:

$\phi_{11}=$

$$5.0x_b(t)\sin(\theta_b(t)) - 5.0x_j(t)\sin(\theta_b(t)) - 5.0y_b(t)\cos(\theta_b(t)) + 5.0y_j(t)\cos(\theta_b(t)) + 2.5\sin\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda \nabla \phi_{11}=$

$$\begin{bmatrix} 5.0\lambda\sin(\theta_b(t)) \\ -5.0\lambda\cos(\theta_b(t)) \\ \lambda\left(5.0x_b(t)\cos(\theta_b(t)) - 5.0x_j(t)\cos(\theta_b(t)) + 5.0y_b(t)\sin(\theta_b(t)) - 5.0y_j(t)\sin(\theta_b(t)) + 2.5\cos\left(\theta_b(t) - \theta_j(t)\right)\right) \\ -5.0\lambda\sin(\theta_b(t)) \\ 5.0\lambda\cos(\theta_b(t)) \\ -2.5\lambda\cos\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:
```python
# K and FC
phi12 = sym.simplify((rFC.cross(rFK))[2])
phi12_mat = sym.Matrix([phi12])
dphi12dq = sym.simplify(phi12_mat.jacobian(q).T)
cons12 = lam*dphi12dq
func_phi12 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi12)

display(Markdown(r'**Impact Constraint 12 is:**'))
display(Markdown(r'$\phi_{12}=$'))
display(phi12)
display(Markdown(r'$\lambda \nabla \phi_{12}=$'))
display(cons12)
```

Impact Constraint 12 is:

$\phi_{12}=$

$$5.0x_b(t)\cos(\theta_b(t)) - 5.0x_j(t)\cos(\theta_b(t)) + 5.0y_b(t)\sin(\theta_b(t)) - 5.0y_j(t)\sin(\theta_b(t)) + 2.5\cos\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda \nabla \phi_{12}=$

$$\begin{bmatrix} 5.0\lambda\cos(\theta_b(t)) \\ 5.0\lambda\sin(\theta_b(t)) \\ \lambda\left(-5.0x_b(t)\sin(\theta_b(t)) + 5.0x_j(t)\sin(\theta_b(t)) + 5.0y_b(t)\cos(\theta_b(t)) - 5.0y_j(t)\cos(\theta_b(t)) - 2.5\sin\left(\theta_b(t) - \theta_j(t)\right)\right) \\ -5.0\lambda\cos(\theta_b(t)) \\ -5.0\lambda\sin(\theta_b(t)) \\ 2.5\lambda\sin\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:
```python
# L and CD
phi13 = sym.simplify((rCD.cross(rCL))[2])
phi13_mat = sym.Matrix([phi13])
dphi13dq = sym.simplify(phi13_mat.jacobian(q).T)
cons13 = lam*dphi13dq
func_phi13 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi13)

display(Markdown(r'**Impact Constraint 13 is:**'))
display(Markdown(r'$\phi_{13}=$'))
display(phi13)
display(Markdown(r'$\lambda \nabla \phi_{13}=$'))
display(cons13)
```

Impact Constraint 13 is:

$\phi_{13}=$

$$-5.0x_b(t)\sin(\theta_b(t)) + 5.0x_j(t)\sin(\theta_b(t)) + 5.0y_b(t)\cos(\theta_b(t)) - 5.0y_j(t)\cos(\theta_b(t)) + 2.5\cos\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda \nabla \phi_{13}=$

$$\begin{bmatrix} -5.0\lambda\sin(\theta_b(t)) \\ 5.0\lambda\cos(\theta_b(t)) \\ \lambda\left(-5.0x_b(t)\cos(\theta_b(t)) + 5.0x_j(t)\cos(\theta_b(t)) - 5.0y_b(t)\sin(\theta_b(t)) + 5.0y_j(t)\sin(\theta_b(t)) - 2.5\sin\left(\theta_b(t) - \theta_j(t)\right)\right) \\ 5.0\lambda\sin(\theta_b(t)) \\ -5.0\lambda\cos(\theta_b(t)) \\ 2.5\lambda\sin\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:
```python
# L and DE
phi14 = sym.simplify((rDE.cross(rDL))[2])
phi14_mat = sym.Matrix([phi14])
dphi14dq = sym.simplify(phi14_mat.jacobian(q).T)
cons14 = lam*dphi14dq
func_phi14 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi14)

display(Markdown(r'**Impact Constraint 14 is:**'))
```

```python
display(Markdown(r'$\phi_{14}=$'))
display(phi14)
display(Markdown(r'$\lambda \nabla \phi_{14}=$'))
display(cons14)
```

Impact Constraint 14 is:

$\phi_{14} =$

$$-5.0x_b(t)\cos(\theta_b(t)) + 5.0x_j(t)\cos(\theta_b(t)) - 5.0y_b(t)\sin(\theta_b(t)) + 5.0y_j(t)\sin(\theta_b(t)) - 2.5\sin\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda\nabla\phi_{14} =$

$$\begin{bmatrix} -5.0\lambda\cos(\theta_b(t)) \\ -5.0\lambda\sin(\theta_b(t)) \\ \lambda\left(5.0x_b(t)\sin(\theta_b(t)) - 5.0x_j(t)\sin(\theta_b(t)) - 5.0y_b(t)\cos(\theta_b(t)) + 5.0y_j(t)\cos(\theta_b(t)) - 2.5\cos\left(\theta_b(t) - \theta_j(t)\right)\right) \\ 5.0\lambda\cos(\theta_b(t)) \\ 5.0\lambda\sin(\theta_b(t)) \\ 2.5\lambda\cos\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:
```python
# L and EF
phi15 = sym.simplify((rEF.cross(rEL))[2])
phi15_mat = sym.Matrix([phi15])
dphi15dq = sym.simplify(phi15_mat.jacobian(q).T)
cons15 = lam*dphi15dq
func_phi15 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi15)

display(Markdown(r'**Impact Constraint 15 is:**'))
display(Markdown(r'$\phi_15=$'))
display(phi15)
display(Markdown(r'$\lambda \nabla \phi_15=$'))
display(cons15)
```

Impact Constraint 15 is:

$\phi_15 =$

$$5.0x_b(t)\sin(\theta_b(t)) - 5.0x_j(t)\sin(\theta_b(t)) - 5.0y_b(t)\cos(\theta_b(t)) + 5.0y_j(t)\cos(\theta_b(t)) - 2.5\cos\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda\nabla\phi_15 =$

$$\begin{bmatrix} 5.0\lambda\sin(\theta_b(t)) \\ -5.0\lambda\cos(\theta_b(t)) \\ \lambda\left(5.0x_b(t)\cos(\theta_b(t)) - 5.0x_j(t)\cos(\theta_b(t)) + 5.0y_b(t)\sin(\theta_b(t)) - 5.0y_j(t)\sin(\theta_b(t)) + 2.5\sin\left(\theta_b(t) - \theta_j(t)\right)\right) \\ -5.0\lambda\sin(\theta_b(t)) \\ 5.0\lambda\cos(\theta_b(t)) \\ -2.5\lambda\sin\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:
```python
# L and FC
phi16 = sym.simplify((rFC.cross(rFL))[2])
phi16_mat = sym.Matrix([phi16])
dphi16dq = sym.simplify(phi16_mat.jacobian(q).T)
cons16 = lam*dphi16dq
func_phi16 = sym.lambdify([xb, yb, thetab, xj, yj, thetaj], phi16)

display(Markdown(r'**Impact Constraint 16 is:**'))
display(Markdown(r'$\phi_{16}=$'))
display(phi16)
display(Markdown(r'$\lambda \nabla \phi_{16}=$'))
display(cons16)
```

Impact Constraint 16 is:

$\phi_{16} =$

$$5.0x_b(t)\cos(\theta_b(t)) - 5.0x_j(t)\cos(\theta_b(t)) + 5.0y_b(t)\sin(\theta_b(t)) - 5.0y_j(t)\sin(\theta_b(t)) + 2.5\sin\left(\theta_b(t) - \theta_j(t)\right) + 12.5$$

$\lambda\nabla\phi_{16} =$

$$\begin{bmatrix} 5.0\lambda\cos(\theta_b(t)) \\ 5.0\lambda\sin(\theta_b(t)) \\ \lambda\left(-5.0x_b(t)\sin(\theta_b(t)) + 5.0x_j(t)\sin(\theta_b(t)) + 5.0y_b(t)\cos(\theta_b(t)) - 5.0y_j(t)\cos(\theta_b(t)) + 2.5\cos\left(\theta_b(t) - \theta_j(t)\right)\right) \\ -5.0\lambda\cos(\theta_b(t)) \\ -5.0\lambda\sin(\theta_b(t)) \\ -2.5\lambda\cos\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:
```python
## Impact Equations
# Define Variables
xbdot_tau_p = sym.symbols(r'\dot{x}_{b\tau^+}')
ybdot_tau_p = sym.symbols(r'\dot{y}_{b\tau^+}')
thetabdot_tau_p = sym.symbols(r'\dot{\theta}_{b\tau^+}')

xjdot_tau_p = sym.symbols(r'\dot{x}_{j\tau^+}')
yjdot_tau_p = sym.symbols(r'\dot{y}_{j\tau^+}')
thetajdot_tau_p = sym.symbols(r'\dot{\theta}_{j\tau^+}')
```

```
vars_impact = sym.Matrix([
    xbdot_tau_p,
    ybdot_tau_p,
    thetabdot_tau_p,
    xjdot_tau_p,
    yjdot_tau_p,
    thetajdot_tau_p,
    lam
])

vars_output = sym.Matrix([
    xbdot_tau_p,
    ybdot_tau_p,
    thetabdot_tau_p,
    xjdot_tau_p,
    yjdot_tau_p,
    thetajdot_tau_p
])


H = sym.simplify(dLdqdot.dot(qdot) - L)

subs_tp = {
    xbdot:xbdot_tau_p,
    ybdot:ybdot_tau_p,
    thetabdot:thetabdot_tau_p,
    xjdot:xjdot_tau_p,
    yjdot:yjdot_tau_p,
    thetajdot:thetajdot_tau_p,
}

H_tp = H.subs(subs_tp)
display(Markdown(r'**Hamitonian ${\cal H}$ <br/>${\cal H}=$**'))
display(H)
display(Markdown(r'**Hamitonian ${\cal H}(\tau^+)$ <br/>${\cal H}(\tau^+)=$**'))
display(H_tp)

P = sym.simplify(dLdqdot)
P_tp = P.subs(subs_tp)
display(Markdown(r'**Momentun ${\cal P}$ <br/>${\cal P}=$**'))
display(P)
display(Markdown(r'**Momentun ${\cal P}_{\tau^+}$ <br/> ${\cal P}_{\tau^+}=$**'))
display(P_tp)
```

Hamitonian $\mathcal{H}$
$\mathcal{H}=$

$$490.0y_b(t) + 39.2y_j(t) + 208.333333333333\left(\frac{d}{dt}\theta_b(t)\right)^2 + 0.5\left(\frac{d}{dt}\theta_j(t)\right)^2 + 25.0\left(\frac{d}{dt}x_b(t)\right)^2 + 2.0\left(\frac{d}{dt}x_j(t)\right)^2 + 25.0\left(\frac{d}{dt}y_b(t)\right)^2 + 2.0\left(\frac{d}{dt}y_j(t)\right)^2$$

Hamitonian $\mathcal{H}(\tau^+)$
$\mathcal{H}(\tau^+)=$

$$208.333333333333\left(\dot{\theta}_{b\tau^+}\right)^2 + 0.5\left(\dot{\theta}_{j\tau^+}\right)^2 + 25.0(\dot{x}_{b\tau^+})^2 + 2.0(\dot{x}_{j\tau^+})^2 + 25.0(\dot{y}_{b\tau^+})^2 + 2.0\left(\dot{y}_{j\tau^+}\right)^2 + 490.0y_b(t) + 39.2y_j(t)$$

Momentun $\mathcal{P}$
$\mathcal{P}=$

$$\begin{bmatrix} 50.0\frac{d}{dt}x_b(t) \\ 50.0\frac{d}{dt}y_b(t) \\ 416.666666666667\frac{d}{dt}\theta_b(t) \\ 4.0\frac{d}{dt}x_j(t) \\ 4.0\frac{d}{dt}y_j(t) \\ 1.0\frac{d}{dt}\theta_j(t) \end{bmatrix}$$

Momentum $\mathcal{P}_{\tau^+}$
$\mathcal{P}_{\tau^+}=$

$$\begin{bmatrix} 50.0\dot{x}_{b\tau^+} \\ 50.0\dot{y}_{b\tau^+} \\ 416.666666666667\dot{\theta}_{b\tau^+} \\ 4.0\dot{x}_{j\tau^+} \\ 4.0\dot{y}_{j\tau^+} \\ 1.0\dot{\theta}_{j\tau^+} \end{bmatrix}$$

In [ ]:
```
# Impact 1
eq1_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq1_rhs = sym.Matrix([
```

```python
    sym.Matrix([0]),
    cons1
])
eq1 = sym.Eq(lhs=eq1_lfs, rhs=eq1_rhs)
display(Markdown(r'**Impact Equation 1**'))
display(eq1)
```

Impact Equation 1

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^{+}}\right)^{2} + 0.5\left(\dot{\theta}_{j\tau^{+}}\right)^{2} + 25.0(\dot{x}_{b\tau^{+}})^{2} + 2.0\left(\dot{x}_{j\tau^{+}}\right)^{2} + 25.0(\dot{y}_{b\tau^{+}})^{2} + 2.0\left(\dot{y}_{j\tau^{+}}\right)^{2} - 208.333333333333\left(\frac{d}{dt}\theta_{b}(t)\right)^{2} - 0.5\left(\frac{d}{dt}\theta_{j}(t)\right)^{2} - \\ 50.0\dot{x}_{b\tau^{+}} - 50.0\frac{d}{dt}x_{b}(t) \\ 50.0\dot{y}_{b\tau^{+}} - 50.0\frac{d}{dt}y_{b}(t) \\ 416.666666666667\dot{\theta}_{b\tau^{+}} - 416.666666666667\frac{d}{dt}\theta_{b}(t) \\ 4.0\dot{x}_{j\tau^{+}} - 4.0\frac{d}{dt}x_{j}(t) \\ 4.0\dot{y}_{j\tau^{+}} - 4.0\frac{d}{dt}y_{j}(t) \\ 1.0\dot{\theta}_{j\tau^{+}} - 1.0\frac{d}{dt}\theta_{j}(t)\end{matrix}\right] = \left[\begin{matrix}0 \\ -5.0\lambda\sin\left(\theta_{b}(t)\right) \\ 5.0\lambda\cos\left(\theta_{b}(t)\right) \\ \lambda\left(-5.0x_{b}(t)\cos\left(\theta_{b}(t)\right) + 5.0x_{j}(t)\cos\left(\theta_{b}(t)\right) - 5.0y_{b}(t)\sin\left(\theta_{b}(t)\right) + 5.0y_{j}(t)\sin\left(\theta_{b}(t)\right) + 2.5\cos\left(\theta_{b}(t) - \theta_{j}(t)\right)\right) \\ 5.0\lambda\sin\left(\theta_{b}(t)\right) \\ -5.0\lambda\cos\left(\theta_{b}(t)\right) \\ -2.5\lambda\cos\left(\theta_{b}(t) - \theta_{j}(t)\right)\end{matrix}\right]$$

In [ ]:
```python
# Impact 2
eq2_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq2_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons2
])
eq2 = sym.Eq(lhs=eq2_lfs, rhs=eq2_rhs)
display(Markdown(r'**Impact Equation 2**'))
display(eq2)
```

Impact Equation 2

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^{+}}\right)^{2} + 0.5\left(\dot{\theta}_{j\tau^{+}}\right)^{2} + 25.0(\dot{x}_{b\tau^{+}})^{2} + 2.0\left(\dot{x}_{j\tau^{+}}\right)^{2} + 25.0(\dot{y}_{b\tau^{+}})^{2} + 2.0\left(\dot{y}_{j\tau^{+}}\right)^{2} - 208.333333333333\left(\frac{d}{dt}\theta_{b}(t)\right)^{2} - 0.5\left(\frac{d}{dt}\theta_{j}(t)\right)^{2} - \\ 50.0\dot{x}_{b\tau^{+}} - 50.0\frac{d}{dt}x_{b}(t) \\ 50.0\dot{y}_{b\tau^{+}} - 50.0\frac{d}{dt}y_{b}(t) \\ 416.666666666667\dot{\theta}_{b\tau^{+}} - 416.666666666667\frac{d}{dt}\theta_{b}(t) \\ 4.0\dot{x}_{j\tau^{+}} - 4.0\frac{d}{dt}x_{j}(t) \\ 4.0\dot{y}_{j\tau^{+}} - 4.0\frac{d}{dt}y_{j}(t) \\ 1.0\dot{\theta}_{j\tau^{+}} - 1.0\frac{d}{dt}\theta_{j}(t)\end{matrix}\right] = \left[\begin{matrix}0 \\ -5.0\lambda\cos\left(\theta_{b}(t)\right) \\ -5.0\lambda\sin\left(\theta_{b}(t)\right) \\ \lambda\left(5.0x_{b}(t)\sin\left(\theta_{b}(t)\right) - 5.0x_{j}(t)\sin\left(\theta_{b}(t)\right) - 5.0y_{b}(t)\cos\left(\theta_{b}(t)\right) + 5.0y_{j}(t)\cos\left(\theta_{b}(t)\right) - 2.5\sin\left(\theta_{b}(t) - \theta_{j}(t)\right)\right) \\ 5.0\lambda\cos\left(\theta_{b}(t)\right) \\ 5.0\lambda\sin\left(\theta_{b}(t)\right) \\ 2.5\lambda\sin\left(\theta_{b}(t) - \theta_{j}(t)\right)\end{matrix}\right]$$

In [ ]:
```python
# Impact 3
eq3_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq3_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons3
])
eq3 = sym.Eq(lhs=eq3_lfs, rhs=eq3_rhs)
display(Markdown(r'**Impact Equation 3**'))
display(eq3)
```

Impact Equation 3

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^{+}}\right)^{2} + 0.5\left(\dot{\theta}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{x}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{x}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{y}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{y}_{j\tau^{+}}\right)^{2} - 208.333333333333\left(\frac{d}{dt}\theta_{b}(t)\right)^{2} - 0.5\left(\frac{d}{dt}\theta_{j}(t)\right)^{2} - \\ 50.0\dot{x}_{b\tau^{+}} - 50.0\frac{d}{dt}x_{b}(t)\\50.0\dot{y}_{b\tau^{+}} - 50.0\frac{d}{dt}y_{b}(t)\\416.666666666667\dot{\theta}_{b\tau^{+}} - 416.666666666667\frac{d}{dt}\theta_{b}(t)\\4.0\dot{x}_{j\tau^{+}} - 4.0\frac{d}{dt}x_{j}(t)\\4.0\dot{y}_{j\tau^{+}} - 4.0\frac{d}{dt}y_{j}(t)\\1.0\dot{\theta}_{j\tau^{+}} - 1.0\frac{d}{dt}\theta_{j}(t)\end{matrix}\right] = \left[\begin{matrix}0\\5.0\lambda\sin\left(\theta_{b}(t)\right)\\-5.0\lambda\cos\left(\theta_{b}(t)\right)\\\lambda\left(5.0x_{b}(t)\cos\left(\theta_{b}(t)\right) - 5.0x_{j}(t)\cos\left(\theta_{b}(t)\right) + 5.0y_{b}(t)\sin\left(\theta_{b}(t)\right) - 5.0y_{j}(t)\sin\left(\theta_{b}(t)\right) - 2.5\cos\left(\theta_{b}(t) - \theta_{j}(t)\right)\right)\\-5.0\lambda\sin\left(\theta_{b}(t)\right)\\5.0\lambda\cos\left(\theta_{b}(t)\right)\\2.5\lambda\cos\left(\theta_{b}(t) - \theta_{j}(t)\right)\end{matrix}\right]$$

In [ ]:
```
# Impact 4
eq4_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq4_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons4
])
eq4 = sym.Eq(lhs=eq4_lfs, rhs=eq4_rhs)
display(Markdown(r'**Impact Equation 4**'))
display(eq4)
```

Impact Equation 4

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^{+}}\right)^{2} + 0.5\left(\dot{\theta}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{x}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{x}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{y}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{y}_{j\tau^{+}}\right)^{2} - 208.333333333333\left(\frac{d}{dt}\theta_{b}(t)\right)^{2} - 0.5\left(\frac{d}{dt}\theta_{j}(t)\right)^{2} - \\ 50.0\dot{x}_{b\tau^{+}} - 50.0\frac{d}{dt}x_{b}(t)\\50.0\dot{y}_{b\tau^{+}} - 50.0\frac{d}{dt}y_{b}(t)\\416.666666666667\dot{\theta}_{b\tau^{+}} - 416.666666666667\frac{d}{dt}\theta_{b}(t)\\4.0\dot{x}_{j\tau^{+}} - 4.0\frac{d}{dt}x_{j}(t)\\4.0\dot{y}_{j\tau^{+}} - 4.0\frac{d}{dt}y_{j}(t)\\1.0\dot{\theta}_{j\tau^{+}} - 1.0\frac{d}{dt}\theta_{j}(t)\end{matrix}\right] = \left[\begin{matrix}0\\5.0\lambda\cos\left(\theta_{b}(t)\right)\\5.0\lambda\sin\left(\theta_{b}(t)\right)\\\lambda\left(-5.0x_{b}(t)\sin\left(\theta_{b}(t)\right) + 5.0x_{j}(t)\sin\left(\theta_{b}(t)\right) + 5.0y_{b}(t)\cos\left(\theta_{b}(t)\right) - 5.0y_{j}(t)\cos\left(\theta_{b}(t)\right) + 2.5\sin\left(\theta_{b}(t) - \theta_{j}(t)\right)\right)\\-5.0\lambda\cos\left(\theta_{b}(t)\right)\\-5.0\lambda\sin\left(\theta_{b}(t)\right)\\-2.5\lambda\sin\left(\theta_{b}(t) - \theta_{j}(t)\right)\end{matrix}\right]$$

In [ ]:
```
# Impact 5
eq5_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq5_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons5
])
eq5 = sym.Eq(lhs=eq5_lfs, rhs=eq5_rhs)
display(Markdown(r'**Impact Equation 6**'))
display(eq5)
```

Impact Equation 6

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^{+}}\right)^{2} + 0.5\left(\dot{\theta}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{x}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{x}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{y}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{y}_{j\tau^{+}}\right)^{2} - 208.333333333333\left(\frac{d}{dt}\theta_{b}(t)\right)^{2} - 0.5\left(\frac{d}{dt}\theta_{j}(t)\right)^{2} - \\ 50.0\dot{x}_{b\tau^{+}} - 50.0\frac{d}{dt}x_{b}(t) \\ 50.0\dot{y}_{b\tau^{+}} - 50.0\frac{d}{dt}y_{b}(t) \\ 416.666666666667\dot{\theta}_{b\tau^{+}} - 416.666666666667\frac{d}{dt}\theta_{b}(t) \\ 4.0\dot{x}_{j\tau^{+}} - 4.0\frac{d}{dt}x_{j}(t) \\ 4.0\dot{y}_{j\tau^{+}} - 4.0\frac{d}{dt}y_{j}(t) \\ 1.0\dot{\theta}_{j\tau^{+}} - 1.0\frac{d}{dt}\theta_{j}(t)\end{matrix}\right. = \left[\begin{matrix}0 \\ -5.0\lambda\sin\left(\theta_{b}(t)\right) \\ 5.0\lambda\cos\left(\theta_{b}(t)\right) \\ \lambda\left(-5.0x_{b}(t)\cos\left(\theta_{b}(t)\right) + 5.0x_{j}(t)\cos\left(\theta_{b}(t)\right) - 5.0y_{b}(t)\sin\left(\theta_{b}(t)\right) + 5.0y_{j}(t)\sin\left(\theta_{b}(t)\right) + 2.5\sin\left(\theta_{b}(t) - \theta_{j}(t)\right)\right) \\ 5.0\lambda\sin\left(\theta_{b}(t)\right) \\ -5.0\lambda\cos\left(\theta_{b}(t)\right) \\ -2.5\lambda\sin\left(\theta_{b}(t) - \theta_{j}(t)\right)\end{matrix}\right]$$

In [ ]:
```python
# Impact 6
eq6_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq6_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons6
])
eq6 = sym.Eq(lhs=eq6_lfs, rhs=eq6_rhs)
display(Markdown(r'**Impact Equation 7**'))
display(eq6)
```

Impact Equation 7

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^{+}}\right)^{2} + 0.5\left(\dot{\theta}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{x}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{x}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{y}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{y}_{j\tau^{+}}\right)^{2} - 208.333333333333\left(\frac{d}{dt}\theta_{b}(t)\right)^{2} - 0.5\left(\frac{d}{dt}\theta_{j}(t)\right)^{2} - \\ 50.0\dot{x}_{b\tau^{+}} - 50.0\frac{d}{dt}x_{b}(t) \\ 50.0\dot{y}_{b\tau^{+}} - 50.0\frac{d}{dt}y_{b}(t) \\ 416.666666666667\dot{\theta}_{b\tau^{+}} - 416.666666666667\frac{d}{dt}\theta_{b}(t) \\ 4.0\dot{x}_{j\tau^{+}} - 4.0\frac{d}{dt}x_{j}(t) \\ 4.0\dot{y}_{j\tau^{+}} - 4.0\frac{d}{dt}y_{j}(t) \\ 1.0\dot{\theta}_{j\tau^{+}} - 1.0\frac{d}{dt}\theta_{j}(t)\end{matrix}\right. = \left[\begin{matrix}0 \\ -5.0\lambda\cos\left(\theta_{b}(t)\right) \\ -5.0\lambda\sin\left(\theta_{b}(t)\right) \\ \lambda\left(5.0x_{b}(t)\sin\left(\theta_{b}(t)\right) - 5.0x_{j}(t)\sin\left(\theta_{b}(t)\right) - 5.0y_{b}(t)\cos\left(\theta_{b}(t)\right) + 5.0y_{j}(t)\cos\left(\theta_{b}(t)\right) + 2.5\cos\left(\theta_{b}(t) - \theta_{j}(t)\right)\right) \\ 5.0\lambda\cos\left(\theta_{b}(t)\right) \\ 5.0\lambda\sin\left(\theta_{b}(t)\right) \\ -2.5\lambda\cos\left(\theta_{b}(t) - \theta_{j}(t)\right)\end{matrix}\right]$$

In [ ]:
```python
# Impact 7
eq7_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq7_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons7
])
eq7 = sym.Eq(lhs=eq7_lfs, rhs=eq7_rhs)
display(Markdown(r'**Impact Equation 8**'))
display(eq7)
```

Impact Equation 8

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^{+}}\right)^{2} + 0.5\left(\dot{\theta}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{x}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{x}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{y}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{y}_{j\tau^{+}}\right)^{2} - 208.333333333333\left(\frac{d}{dt}\theta_{b}(t)\right)^{2} - 0.5\left(\frac{d}{dt}\theta_{j}(t)\right)^{2} - \\ 50.0\dot{x}_{b\tau^{+}} - 50.0\frac{d}{dt}x_{b}(t) \\ 50.0\dot{y}_{b\tau^{+}} - 50.0\frac{d}{dt}y_{b}(t) \\ 416.666666666667\dot{\theta}_{b\tau^{+}} - 416.666666666667\frac{d}{dt}\theta_{b}(t) \\ 4.0\dot{x}_{j\tau^{+}} - 4.0\frac{d}{dt}x_{j}(t) \\ 4.0\dot{y}_{j\tau^{+}} - 4.0\frac{d}{dt}y_{j}(t) \\ 1.0\dot{\theta}_{j\tau^{+}} - 1.0\frac{d}{dt}\theta_{j}(t)\end{matrix}\right]$$
$$= \left[\begin{matrix}0 \\ 5.0\lambda\sin\left(\theta_{b}(t)\right) \\ -5.0\lambda\cos\left(\theta_{b}(t)\right) \\ \lambda\left(5.0x_{b}(t)\cos\left(\theta_{b}(t)\right) - 5.0x_{j}(t)\cos\left(\theta_{b}(t)\right) + 5.0y_{b}(t)\sin\left(\theta_{b}(t)\right) - 5.0y_{j}(t)\sin\left(\theta_{b}(t)\right) - 2.5\sin\left(\theta_{b}(t) - \theta_{j}(t)\right)\right) \\ -5.0\lambda\sin\left(\theta_{b}(t)\right) \\ 5.0\lambda\cos\left(\theta_{b}(t)\right) \\ 2.5\lambda\sin\left(\theta_{b}(t) - \theta_{j}(t)\right)\end{matrix}\right]$$

In [ ]:
```python
# Impact 8
eq8_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq8_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons8
])
eq8 = sym.Eq(lhs=eq8_lfs, rhs=eq8_rhs)
display(Markdown(r'**Impact Equation 9**'))
display(eq8)
```

Impact Equation 9

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^{+}}\right)^{2} + 0.5\left(\dot{\theta}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{x}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{x}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{y}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{y}_{j\tau^{+}}\right)^{2} - 208.333333333333\left(\frac{d}{dt}\theta_{b}(t)\right)^{2} - 0.5\left(\frac{d}{dt}\theta_{j}(t)\right)^{2} - \\ 50.0\dot{x}_{b\tau^{+}} - 50.0\frac{d}{dt}x_{b}(t) \\ 50.0\dot{y}_{b\tau^{+}} - 50.0\frac{d}{dt}y_{b}(t) \\ 416.666666666667\dot{\theta}_{b\tau^{+}} - 416.666666666667\frac{d}{dt}\theta_{b}(t) \\ 4.0\dot{x}_{j\tau^{+}} - 4.0\frac{d}{dt}x_{j}(t) \\ 4.0\dot{y}_{j\tau^{+}} - 4.0\frac{d}{dt}y_{j}(t) \\ 1.0\dot{\theta}_{j\tau^{+}} - 1.0\frac{d}{dt}\theta_{j}(t)\end{matrix}\right]$$
$$= \left[\begin{matrix}0 \\ 5.0\lambda\cos\left(\theta_{b}(t)\right) \\ 5.0\lambda\sin\left(\theta_{b}(t)\right) \\ \lambda\left(-5.0x_{b}(t)\sin\left(\theta_{b}(t)\right) + 5.0x_{j}(t)\sin\left(\theta_{b}(t)\right) + 5.0y_{b}(t)\cos\left(\theta_{b}(t)\right) - 5.0y_{j}(t)\cos\left(\theta_{b}(t)\right) - 2.5\cos\left(\theta_{b}(t) - \theta_{j}(t)\right)\right) \\ -5.0\lambda\cos\left(\theta_{b}(t)\right) \\ -5.0\lambda\sin\left(\theta_{b}(t)\right) \\ 2.5\lambda\cos\left(\theta_{b}(t) - \theta_{j}(t)\right)\end{matrix}\right]$$

In [ ]:
```python
# Impact 9
eq9_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq9_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons9
])
eq9 = sym.Eq(lhs=eq9_lfs, rhs=eq9_rhs)
display(Markdown(r'**Impact Equation 10**'))
display(eq9)
```

Impact Equation 10

$$\begin{bmatrix} 208.333333333333\left(\dot{\theta}_{b\tau^+}\right)^2 + 0.5\left(\dot{\theta}_{j\tau^+}\right)^2 + 25.0(\dot{x}_{b\tau^+})^2 + 2.0(\dot{x}_{j\tau^+})^2 + 25.0(\dot{y}_{b\tau^+})^2 + 2.0\left(\dot{y}_{j\tau^+}\right)^2 - 208.333333333333\left(\frac{d}{dt}\theta_b(t)\right)^2 - 0.5\left(\frac{d}{dt}\theta_j(t)\right)^2 - \\ 50.0\dot{x}_{b\tau^+} - 50.0\frac{d}{dt}x_b(t) \\ 50.0\dot{y}_{b\tau^+} - 50.0\frac{d}{dt}y_b(t) \\ 416.666666666667\dot{\theta}_{b\tau^+} - 416.666666666667\frac{d}{dt}\theta_b(t) \\ 4.0\dot{x}_{j\tau^+} - 4.0\frac{d}{dt}x_j(t) \\ 4.0\dot{y}_{j\tau^+} - 4.0\frac{d}{dt}y_j(t) \\ 1.0\dot{\theta}_{j\tau^+} - 1.0\frac{d}{dt}\theta_j(t) \end{bmatrix} = \begin{bmatrix} 0 \\ -5.0\lambda\sin(\theta_b(t)) \\ 5.0\lambda\cos(\theta_b(t)) \\ \lambda\left(-5.0x_b(t)\cos(\theta_b(t)) + 5.0x_j(t)\cos(\theta_b(t)) - 5.0y_b(t)\sin(\theta_b(t)) + 5.0y_j(t)\sin(\theta_b(t)) - 2.5\cos\left(\theta_b(t) - \theta_j(t)\right)\right) \\ 5.0\lambda\sin(\theta_b(t)) \\ -5.0\lambda\cos(\theta_b(t)) \\ 2.5\lambda\cos\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:

```python
# Impact 10
eq10_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq10_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons10
])
eq10 = sym.Eq(lhs=eq10_lfs, rhs=eq10_rhs)
display(eq10)
```

$$\begin{bmatrix} 208.333333333333\left(\dot{\theta}_{b\tau^+}\right)^2 + 0.5\left(\dot{\theta}_{j\tau^+}\right)^2 + 25.0(\dot{x}_{b\tau^+})^2 + 2.0(\dot{x}_{j\tau^+})^2 + 25.0(\dot{y}_{b\tau^+})^2 + 2.0\left(\dot{y}_{j\tau^+}\right)^2 - 208.333333333333\left(\frac{d}{dt}\theta_b(t)\right)^2 - 0.5\left(\frac{d}{dt}\theta_j(t)\right)^2 - \\ 50.0\dot{x}_{b\tau^+} - 50.0\frac{d}{dt}x_b(t) \\ 50.0\dot{y}_{b\tau^+} - 50.0\frac{d}{dt}y_b(t) \\ 416.666666666667\dot{\theta}_{b\tau^+} - 416.666666666667\frac{d}{dt}\theta_b(t) \\ 4.0\dot{x}_{j\tau^+} - 4.0\frac{d}{dt}x_j(t) \\ 4.0\dot{y}_{j\tau^+} - 4.0\frac{d}{dt}y_j(t) \\ 1.0\dot{\theta}_{j\tau^+} - 1.0\frac{d}{dt}\theta_j(t) \end{bmatrix} = \begin{bmatrix} 0 \\ -5.0\lambda\cos(\theta_b(t)) \\ -5.0\lambda\sin(\theta_b(t)) \\ \lambda\left(5.0x_b(t)\sin(\theta_b(t)) - 5.0x_j(t)\sin(\theta_b(t)) - 5.0y_b(t)\cos(\theta_b(t)) + 5.0y_j(t)\cos(\theta_b(t)) + 2.5\sin\left(\theta_b(t) - \theta_j(t)\right)\right) \\ 5.0\lambda\cos(\theta_b(t)) \\ 5.0\lambda\sin(\theta_b(t)) \\ -2.5\lambda\sin\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}$$

In [ ]:

```python
# Impact 11
eq11_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq11_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons11
])
eq11 = sym.Eq(lhs=eq11_lfs, rhs=eq11_rhs)
display(Markdown(r'**Impact Equation 11**'))
display(eq11)
```

Impact Equation 11

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^{+}}\right)^{2} + 0.5\left(\dot{\theta}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{x}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{x}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{y}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{y}_{j\tau^{+}}\right)^{2} - 208.333333333333\left(\frac{d}{dt}\theta_{b}(t)\right)^{2} - 0.5\left(\frac{d}{dt}\theta_{j}(t)\right)^{2} - \\ 50.0\dot{x}_{b\tau^{+}} - 50.0\frac{d}{dt}x_{b}(t) \\ 50.0\dot{y}_{b\tau^{+}} - 50.0\frac{d}{dt}y_{b}(t) \\ 416.666666666667\dot{\theta}_{b\tau^{+}} - 416.666666666667\frac{d}{dt}\theta_{b}(t) \\ 4.0\dot{x}_{j\tau^{+}} - 4.0\frac{d}{dt}x_{j}(t) \\ 4.0\dot{y}_{j\tau^{+}} - 4.0\frac{d}{dt}y_{j}(t) \\ 1.0\dot{\theta}_{j\tau^{+}} - 1.0\frac{d}{dt}\theta_{j}(t)\end{matrix}\right]$$

$$= \left[\begin{matrix}0 \\ 5.0\lambda\sin\left(\theta_{b}(t)\right) \\ -5.0\lambda\cos\left(\theta_{b}(t)\right) \\ \lambda\left(5.0x_{b}(t)\cos\left(\theta_{b}(t)\right) - 5.0x_{j}(t)\cos\left(\theta_{b}(t)\right) + 5.0y_{b}(t)\sin\left(\theta_{b}(t)\right) - 5.0y_{j}(t)\sin\left(\theta_{b}(t)\right) + 2.5\cos\left(\theta_{b}(t) - \theta_{j}(t)\right)\right) \\ -5.0\lambda\sin\left(\theta_{b}(t)\right) \\ 5.0\lambda\cos\left(\theta_{b}(t)\right) \\ -2.5\lambda\cos\left(\theta_{b}(t) - \theta_{j}(t)\right)\end{matrix}\right]$$

In [ ]:

```python
# Impact 12
eq12_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq12_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons12
])
eq12 = sym.Eq(lhs=eq12_lfs, rhs=eq12_rhs)
display(Markdown(r'**Impact Equation 12**'))
display(eq12)
```

Impact Equation 12

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^{+}}\right)^{2} + 0.5\left(\dot{\theta}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{x}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{x}_{j\tau^{+}}\right)^{2} + 25.0\left(\dot{y}_{b\tau^{+}}\right)^{2} + 2.0\left(\dot{y}_{j\tau^{+}}\right)^{2} - 208.333333333333\left(\frac{d}{dt}\theta_{b}(t)\right)^{2} - 0.5\left(\frac{d}{dt}\theta_{j}(t)\right)^{2} - \\ 50.0\dot{x}_{b\tau^{+}} - 50.0\frac{d}{dt}x_{b}(t) \\ 50.0\dot{y}_{b\tau^{+}} - 50.0\frac{d}{dt}y_{b}(t) \\ 416.666666666667\dot{\theta}_{b\tau^{+}} - 416.666666666667\frac{d}{dt}\theta_{b}(t) \\ 4.0\dot{x}_{j\tau^{+}} - 4.0\frac{d}{dt}x_{j}(t) \\ 4.0\dot{y}_{j\tau^{+}} - 4.0\frac{d}{dt}y_{j}(t) \\ 1.0\dot{\theta}_{j\tau^{+}} - 1.0\frac{d}{dt}\theta_{j}(t)\end{matrix}\right]$$

$$= \left[\begin{matrix}0 \\ 5.0\lambda\cos\left(\theta_{b}(t)\right) \\ 5.0\lambda\sin\left(\theta_{b}(t)\right) \\ \lambda\left(-5.0x_{b}(t)\sin\left(\theta_{b}(t)\right) + 5.0x_{j}(t)\sin\left(\theta_{b}(t)\right) + 5.0y_{b}(t)\cos\left(\theta_{b}(t)\right) - 5.0y_{j}(t)\cos\left(\theta_{b}(t)\right) - 2.5\sin\left(\theta_{b}(t) - \theta_{j}(t)\right)\right) \\ -5.0\lambda\cos\left(\theta_{b}(t)\right) \\ -5.0\lambda\sin\left(\theta_{b}(t)\right) \\ 2.5\lambda\sin\left(\theta_{b}(t) - \theta_{j}(t)\right)\end{matrix}\right]$$

In [ ]:

```python
# Impact 13
eq13_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq13_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons13
])
eq13 = sym.Eq(lhs=eq13_lfs, rhs=eq13_rhs)
display(Markdown(r'**Impact Equation 13**'))
display(eq13)
```

Impact Equation 13

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^+}\right)^2 + 0.5\left(\dot{\theta}_{j\tau^+}\right)^2 + 25.0(\dot{x}_{b\tau^+})^2 + 2.0\left(\dot{x}_{j\tau^+}\right)^2 + 25.0(\dot{y}_{b\tau^+})^2 + 2.0\left(\dot{y}_{j\tau^+}\right)^2 - 208.333333333333\left(\frac{d}{dt}\theta_b(t)\right)^2 - 0.5\left(\frac{d}{dt}\theta_j(t)\right)^2 - \\ 50.0\dot{x}_{b\tau^+} - 50.0\frac{d}{dt}x_b(t) \\ 50.0\dot{y}_{b\tau^+} - 50.0\frac{d}{dt}y_b(t) \\ 416.666666666667\dot{\theta}_{b\tau^+} - 416.666666666667\frac{d}{dt}\theta_b(t) \\ 4.0\dot{x}_{j\tau^+} - 4.0\frac{d}{dt}x_j(t) \\ 4.0\dot{y}_{j\tau^+} - 4.0\frac{d}{dt}y_j(t) \\ 1.0\dot{\theta}_{j\tau^+} - 1.0\frac{d}{dt}\theta_j(t)\end{matrix}\right.$$

$$= \left[\begin{matrix}0 \\ -5.0\lambda\sin\left(\theta_b(t)\right) \\ 5.0\lambda\cos\left(\theta_b(t)\right) \\ \lambda\left(-5.0x_b(t)\cos\left(\theta_b(t)\right) + 5.0x_j(t)\cos\left(\theta_b(t)\right) - 5.0y_b(t)\sin\left(\theta_b(t)\right) + 5.0y_j(t)\sin\left(\theta_b(t)\right) - 2.5\sin\left(\theta_b(t) - \theta_j(t)\right)\right) \\ 5.0\lambda\sin\left(\theta_b(t)\right) \\ -5.0\lambda\cos\left(\theta_b(t)\right) \\ 2.5\lambda\sin\left(\theta_b(t) - \theta_j(t)\right)\end{matrix}\right]$$

In [ ]:
```python
# Impact 14
eq14_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq14_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons14
])
eq14 = sym.Eq(lhs=eq14_lfs, rhs=eq14_rhs)
display(Markdown(r'**Impact Equation 14**'))
display(eq14)
```

Impact Equation 14

$$\left[\begin{matrix}208.333333333333\left(\dot{\theta}_{b\tau^+}\right)^2 + 0.5\left(\dot{\theta}_{j\tau^+}\right)^2 + 25.0(\dot{x}_{b\tau^+})^2 + 2.0\left(\dot{x}_{j\tau^+}\right)^2 + 25.0(\dot{y}_{b\tau^+})^2 + 2.0\left(\dot{y}_{j\tau^+}\right)^2 - 208.333333333333\left(\frac{d}{dt}\theta_b(t)\right)^2 - 0.5\left(\frac{d}{dt}\theta_j(t)\right)^2 - \\ 50.0\dot{x}_{b\tau^+} - 50.0\frac{d}{dt}x_b(t) \\ 50.0\dot{y}_{b\tau^+} - 50.0\frac{d}{dt}y_b(t) \\ 416.666666666667\dot{\theta}_{b\tau^+} - 416.666666666667\frac{d}{dt}\theta_b(t) \\ 4.0\dot{x}_{j\tau^+} - 4.0\frac{d}{dt}x_j(t) \\ 4.0\dot{y}_{j\tau^+} - 4.0\frac{d}{dt}y_j(t) \\ 1.0\dot{\theta}_{j\tau^+} - 1.0\frac{d}{dt}\theta_j(t)\end{matrix}\right.$$

$$= \left[\begin{matrix}0 \\ -5.0\lambda\cos\left(\theta_b(t)\right) \\ -5.0\lambda\sin\left(\theta_b(t)\right) \\ \lambda\left(5.0x_b(t)\sin\left(\theta_b(t)\right) - 5.0x_j(t)\sin\left(\theta_b(t)\right) - 5.0y_b(t)\cos\left(\theta_b(t)\right) + 5.0y_j(t)\cos\left(\theta_b(t)\right) - 2.5\cos\left(\theta_b(t) - \theta_j(t)\right)\right) \\ 5.0\lambda\cos\left(\theta_b(t)\right) \\ 5.0\lambda\sin\left(\theta_b(t)\right) \\ 2.5\lambda\cos\left(\theta_b(t) - \theta_j(t)\right)\end{matrix}\right]$$

In [ ]:
```python
# Impact 15
eq15_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq15_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons15
])
eq15 = sym.Eq(lhs=eq15_lfs, rhs=eq15_rhs)
display(Markdown(r'**Impact Equation 15**'))
display(eq15)
```

Impact Equation 15

$$
\begin{bmatrix} 208.333333333333\left(\dot{\theta}_{b\tau^+}\right)^2 + 0.5\left(\dot{\theta}_{j\tau^+}\right)^2 + 25.0\left(\dot{x}_{b\tau^+}\right)^2 + 2.0\left(\dot{x}_{j\tau^+}\right)^2 + 25.0\left(\dot{y}_{b\tau^+}\right)^2 + 2.0\left(\dot{y}_{j\tau^+}\right)^2 - 208.333333333333\left(\frac{d}{dt}\theta_b(t)\right)^2 - 0.5\left(\frac{d}{dt}\theta_j(t)\right)^2 - \\ 50.0\dot{x}_{b\tau^+} - 50.0\frac{d}{dt}x_b(t) \\ 50.0\dot{y}_{b\tau^+} - 50.0\frac{d}{dt}y_b(t) \\ 416.666666666667\dot{\theta}_{b\tau^+} - 416.666666666667\frac{d}{dt}\theta_b(t) \\ 4.0\dot{x}_{j\tau^+} - 4.0\frac{d}{dt}x_j(t) \\ 4.0\dot{y}_{j\tau^+} - 4.0\frac{d}{dt}y_j(t) \\ 1.0\dot{\theta}_{j\tau^+} - 1.0\frac{d}{dt}\theta_j(t) \end{bmatrix}
= \begin{bmatrix} 0 \\ 5.0\lambda\sin\left(\theta_b(t)\right) \\ -5.0\lambda\cos\left(\theta_b(t)\right) \\ \lambda\left(5.0x_b(t)\cos\left(\theta_b(t)\right) - 5.0x_j(t)\cos\left(\theta_b(t)\right) + 5.0y_b(t)\sin\left(\theta_b(t)\right) - 5.0y_j(t)\sin\left(\theta_b(t)\right) + 2.5\sin\left(\theta_b(t) - \theta_j(t)\right)\right) \\ -5.0\lambda\sin\left(\theta_b(t)\right) \\ 5.0\lambda\cos\left(\theta_b(t)\right) \\ -2.5\lambda\sin\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}
$$

In [ ]:
```python
# Impact 16
eq16_lfs = sym.Matrix([
    sym.simplify(H_tp - H),
    sym.simplify(P_tp - P)
])
eq16_rhs = sym.Matrix([
    sym.Matrix([0]),
    cons16
])
eq16 = sym.Eq(lhs=eq16_lfs, rhs=eq16_rhs)
display(Markdown(r'**Impact Equation 16**'))
display(eq16)
```

Impact Equation 16

$$
\begin{bmatrix} 208.333333333333\left(\dot{\theta}_{b\tau^+}\right)^2 + 0.5\left(\dot{\theta}_{j\tau^+}\right)^2 + 25.0\left(\dot{x}_{b\tau^+}\right)^2 + 2.0\left(\dot{x}_{j\tau^+}\right)^2 + 25.0\left(\dot{y}_{b\tau^+}\right)^2 + 2.0\left(\dot{y}_{j\tau^+}\right)^2 - 208.333333333333\left(\frac{d}{dt}\theta_b(t)\right)^2 - 0.5\left(\frac{d}{dt}\theta_j(t)\right)^2 - \\ 50.0\dot{x}_{b\tau^+} - 50.0\frac{d}{dt}x_b(t) \\ 50.0\dot{y}_{b\tau^+} - 50.0\frac{d}{dt}y_b(t) \\ 416.666666666667\dot{\theta}_{b\tau^+} - 416.666666666667\frac{d}{dt}\theta_b(t) \\ 4.0\dot{x}_{j\tau^+} - 4.0\frac{d}{dt}x_j(t) \\ 4.0\dot{y}_{j\tau^+} - 4.0\frac{d}{dt}y_j(t) \\ 1.0\dot{\theta}_{j\tau^+} - 1.0\frac{d}{dt}\theta_j(t) \end{bmatrix}
= \begin{bmatrix} 0 \\ 5.0\lambda\cos\left(\theta_b(t)\right) \\ 5.0\lambda\sin\left(\theta_b(t)\right) \\ \lambda\left(-5.0x_b(t)\sin\left(\theta_b(t)\right) + 5.0x_j(t)\sin\left(\theta_b(t)\right) + 5.0y_b(t)\cos\left(\theta_b(t)\right) - 5.0y_j(t)\cos\left(\theta_b(t)\right) + 2.5\cos\left(\theta_b(t) - \theta_j(t)\right)\right) \\ -5.0\lambda\cos\left(\theta_b(t)\right) \\ -5.0\lambda\sin\left(\theta_b(t)\right) \\ -2.5\lambda\cos\left(\theta_b(t) - \theta_j(t)\right) \end{bmatrix}
$$

In [ ]:
```python
def impact_update(s, impact_type):
    xb_v     = s[0]
    yb_v     = s[1]
    thetab_v = s[2]

    xj_v     = s[3]
    yj_v     = s[4]
    thetaj_v = s[5]

    xbdot_v     = s[6]
    ybdot_v     = s[7]
    thetabdot_v = s[8]

    xjdot_v     = s[9]
    yjdot_v     = s[10]
    thetajdot_v = s[11]

    impact_subs = {
        xb: xb_v,
        yb: yb_v,
        thetab: thetab_v,
        xj: xj_v,
        yj: yj_v,
        thetaj: thetaj_v,
        xbdot: xbdot_v,
        ybdot: ybdot_v,
        thetabdot: thetabdot_v,
        xjdot: xjdot_v,
        yjdot: yjdot_v,
```

```python
        thetajdot: thetajdot_v
    }

    output = [xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v]

    if impact_type == 1:

        eq1_subs = eq1.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq1_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 2:

        eq2_subs = eq2.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq2_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 3:

        eq3_subs = eq3.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq3_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 4:

        eq4_subs = eq4.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq4_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 5:

        eq5_subs = eq5.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq5_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 6:

        eq6_subs = eq6.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq6_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
```

```
                output.append(sym.N(sol[v]))

    elif impact_type == 7:

        eq7_subs = eq7.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq7_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 8:

        eq8_subs = eq8.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq8_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 9:

        eq9_subs = eq9.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq9_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 10:

        eq10_subs = eq10.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq10_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 11:

        eq11_subs = eq11.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq11_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 12:

        eq12_subs = eq12.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq12_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 13:
```

```python
        eq13_subs = eq13.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq13_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 14:

        eq14_subs = eq14.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq14_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 15:

        eq15_subs = eq15.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq15_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    elif impact_type == 16:

        eq16_subs = eq16.subs(impact_subs)
        # display(eq1_subs)
        sols = sym.solve(eq16_subs, vars_impact, dict=True)
        # display(sols)

        for sol in sols:
            if abs(sym.N(sol[lam])) < 1e-5:
                continue
            else:
                for v in vars_output:
                    output.append(sym.N(sol[v]))

    else:
        raise RuntimeError('Invalid Impact')

    # print(f'output: {output}')
    return np.array(output)
```

In [ ]:

```python
def impact_condition(s, impact_type, tol=1e-1):
    xb_v     = s[0]
    yb_v     = s[1]
    thetab_v = s[2]

    xj_v     = s[3]
    yj_v     = s[4]
    thetaj_v = s[5]

    # print(xb_v, yb_v, thetab_v)

    if impact_type == 1:
        phi_val = func_phi1(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 2:
        phi_val = func_phi2(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')
```

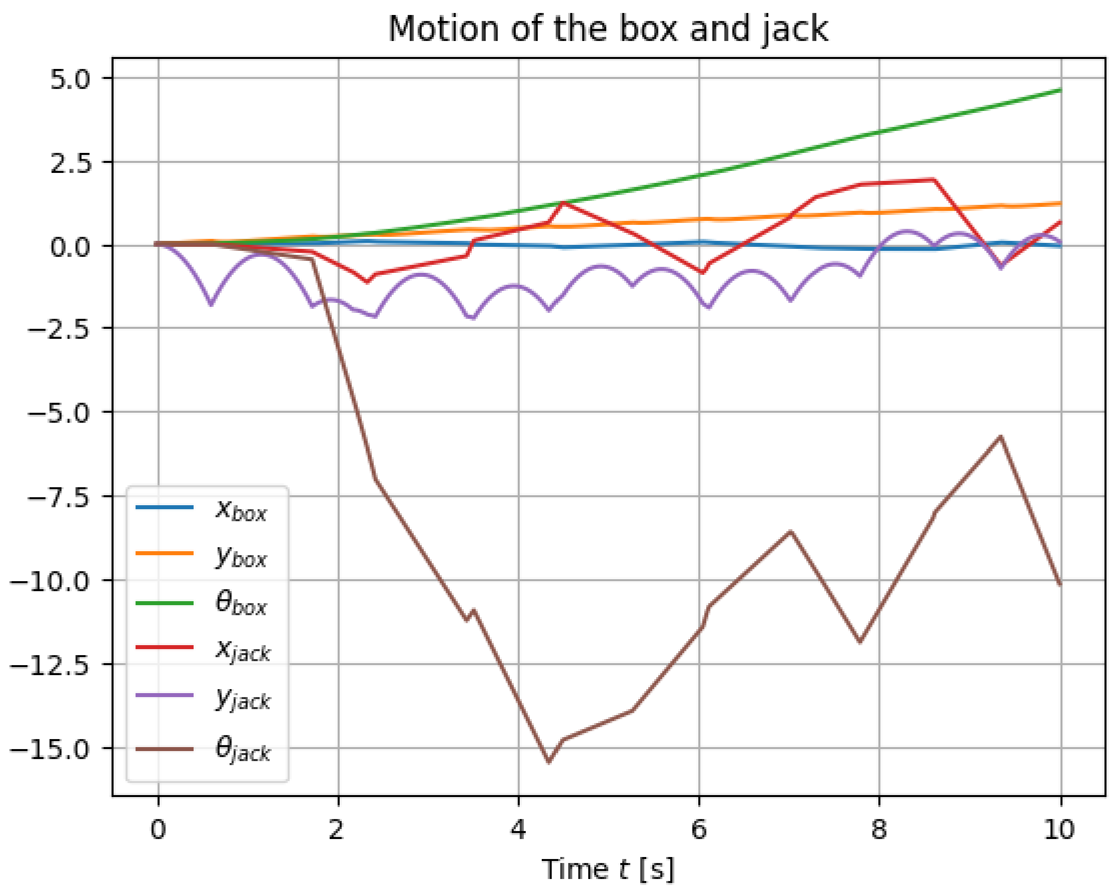
```
        return phi_val < tol and phi_val > -tol

    elif impact_type == 3:
        phi_val = func_phi3(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 4:
        phi_val = func_phi4(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 5:
        phi_val = func_phi5(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 6:
        phi_val = func_phi6(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 7:
        phi_val = func_phi7(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 8:
        phi_val = func_phi8(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 9:
        phi_val = func_phi9(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 10:
        phi_val = func_phi10(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 11:
        phi_val = func_phi11(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 12:
        phi_val = func_phi12(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 13:
        phi_val = func_phi13(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 14:
        phi_val = func_phi14(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 15:
        phi_val = func_phi15(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol

    elif impact_type == 16:
        phi_val = func_phi16(xb_v, yb_v, thetab_v, xj_v, yj_v, thetaj_v)
        # print(f'phi: {phi_val}')

        return phi_val < tol and phi_val > -tol
```

```python
    return False
```

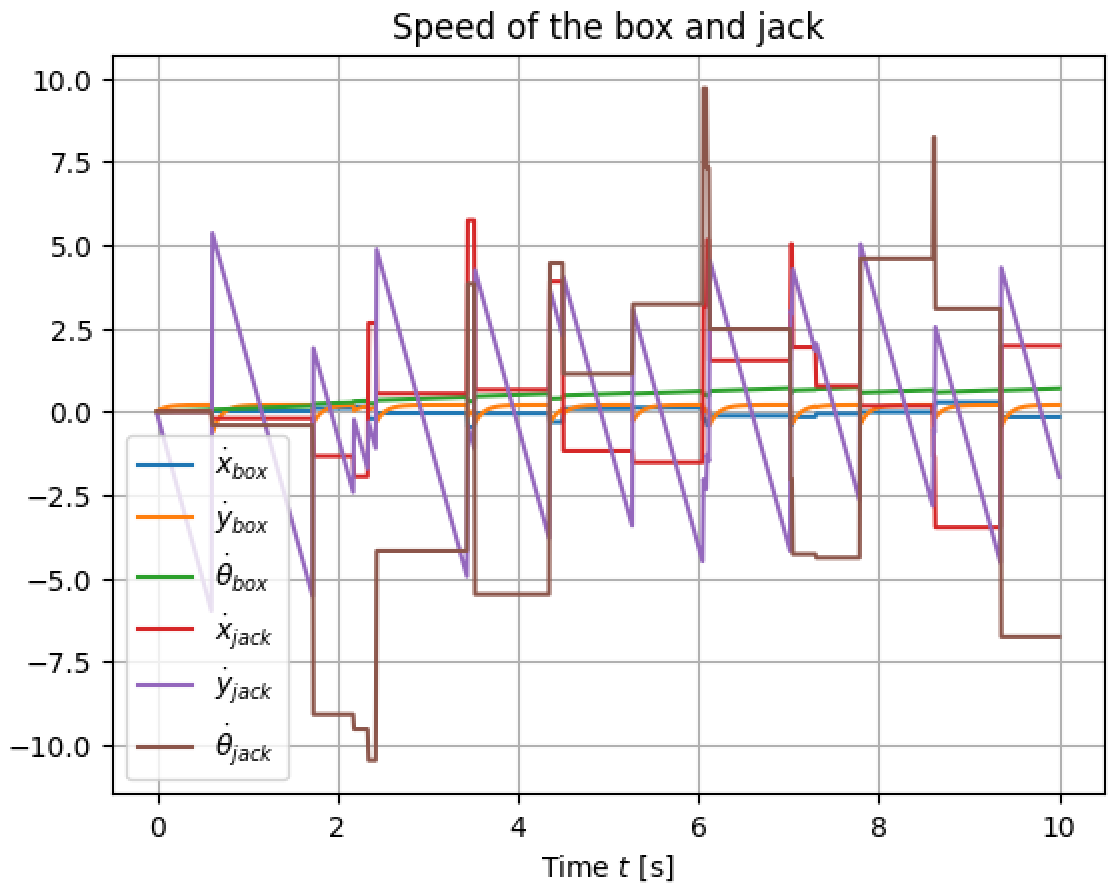
In [ ]:
```python
def simulate(f, x0, tspan, dt, integrate):
    """
    This function takes in an initial condition x0, a timestep dt,
    a time span tspan consisting of a list [min_time, max_time],
    as well as a dynamical system f(x) that outputs a vector of the
    same dimension as x0. It outputs a full trajectory simulated
    over the time span of dimensions (xvec_size, time_vec_size).

    Parameters
    ============
    f: Python function
        derivate of the system at a given step x(t),
        it can considered as \dot{x}(t) = func(x(t))
    x0: NumPy array
        initial conditions
    tspan: Python list
        tspan = [min_time, max_time], it defines the start and end
        time of simulation
    dt:
        time step for numerical integration
    integrate: Python function
        numerical integration method used in this simulation

    Return
    ============
    x_traj:
        simulated trajectory of x(t) from t=0 to tf
    """
    N = int((max(tspan)-min(tspan))/dt)
    x = np.copy(x0)
    tvec = np.linspace(min(tspan),max(tspan),N)
    xtraj = np.zeros((len(x0),N))
    for i in range(N):
        for j in range(16):
            # print(impact_condition(x, j+1))
            if impact_condition(x, impact_type=j+1, tol=5e-1):
                x=impact_update(x, impact_type=j+1)
                # print(x)
                # break
        xtraj[:,i]=integrate(f,x,dt)
        x = np.copy(xtraj[:,i])
    return xtraj
```

In [ ]:
```python
Tf = 10
s0 = np.array([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
traj = simulate(dyn, s0, [0, Tf], 0.01, integrate)
```

In [ ]:
```python
t_plot = np.linspace(0, Tf, Tf*100)
plt.plot(t_plot, traj[0], label=r'$x_{box}$')
plt.plot(t_plot, traj[1], label=r'$y_{box}$')
plt.plot(t_plot, traj[2], label=r'$\theta_{box}$')
plt.plot(t_plot, traj[3], label=r'$x_{jack}$')
plt.plot(t_plot, traj[4], label=r'$y_{jack}$')
plt.plot(t_plot, traj[5], label=r'$\theta_{jack}$')
plt.xlabel(r'Time $t$ [s]')
plt.title(r'Motion of the box and jack')
plt.grid()
plt.legend()
plt.show()
```



In [ ]:
```
plt.plot(t_plot, traj[6], label=r'$\dot{x}_{box}$')
plt.plot(t_plot, traj[7], label=r'$\dot{y}_{box}$')
plt.plot(t_plot, traj[8], label=r'$\dot{\theta}_{box}$')
plt.plot(t_plot, traj[9], label=r'$\dot{x}_{jack}$')
plt.plot(t_plot, traj[10], label=r'$\dot{y}_{jack}$')
plt.plot(t_plot, traj[11], label=r'$\dot{\theta}_{jack}$')
plt.xlabel(r'Time $t$ [s]')
plt.title(r'Speed of the box and jack')
plt.grid()
plt.legend()
plt.show()
```



In [ ]:
```
def get_g_matrix(theta, px, py):
    return np.array([
        [   np.cos(theta), -np.sin(theta),  px],
        [   np.sin(theta),  np.cos(theta),  py],
        [   0,              0,              1 ]
    ])

def animate_system(theta_array,L1=1,L2=1,W=0.2,T=10):
    """
    Function to generate web-based animation of double-pendulum system

    Parameters:
    ================================================
    theta_array:
        trajectory of theta1 and theta2, should be a NumPy array with
        shape of (2,N)
    L1:
        length of the first pendulum
    L2:
        length of the second pendulum
```

```
    T:
        length/seconds of animation duration

    Returns: None
    """

    ################################
    # Imports required for animation.
    from plotly.offline import init_notebook_mode, iplot
    from IPython.display import display, HTML
    import plotly.graph_objects as go

    #######################
    # Browser configuration.
    def configure_plotly_browser_state():
        import IPython
        display(IPython.core.display.HTML('''
            <script src="/static/components/requirejs/require.js"></script>
            <script>
              requirejs.config({
                paths: {
                  base: '/static/base',
                  plotly: 'https://cdn.plot.ly/plotly-1.5.1.min.js?noext',
                },
              });
            </script>
            '''))
    configure_plotly_browser_state()
    init_notebook_mode(connected=False)
    N = len(theta_array[0])

    xxb = theta_array[0]
    yyb = theta_array[1]
    tb  = theta_array[2]
    xxj = theta_array[3]
    yyj = theta_array[4]
    tj  = theta_array[5]

    xb1 = np.zeros(N)
    yb1 = np.zeros(N)
    xb2 = np.zeros(N)
    yb2 = np.zeros(N)
    xb3 = np.zeros(N)
    yb3 = np.zeros(N)
    xb4 = np.zeros(N)
    yb4 = np.zeros(N)

    xj1 = np.zeros(N)
    yj1 = np.zeros(N)
    xj2 = np.zeros(N)
    yj2 = np.zeros(N)
    xj3 = np.zeros(N)
    yj3 = np.zeros(N)
    xj4 = np.zeros(N)
    yj4 = np.zeros(N)

    ###############################################
    # Define frames
    for i in range(N):
        # Box

        twa = get_g_matrix(0, xxb[i], yyb[i])
        tab = get_g_matrix(tb[i], 0, 0)
        tbc = get_g_matrix(0,  L1/2,  L1/2)
        tbd = get_g_matrix(0, -L1/2,  L1/2)
        tbe = get_g_matrix(0, -L1/2, -L1/2)
        tbf = get_g_matrix(0,  L1/2, -L1/2)

        twb = twa @ tab
        twc = twb @ tbc
        twd = twb @ tbd
        twe = twb @ tbe
        twf = twb @ tbf

        xb1[i] = np.array([1, 0, 0]) @ twc @ np.array([0, 0, 1])
        yb1[i] = np.array([0, 1, 0]) @ twc @ np.array([0, 0, 1])
        xb2[i] = np.array([1, 0, 0]) @ twd @ np.array([0, 0, 1])
        yb2[i] = np.array([0, 1, 0]) @ twd @ np.array([0, 0, 1])
        xb3[i] = np.array([1, 0, 0]) @ twe @ np.array([0, 0, 1])
        yb3[i] = np.array([0, 1, 0]) @ twe @ np.array([0, 0, 1])
        xb4[i] = np.array([1, 0, 0]) @ twf @ np.array([0, 0, 1])
        yb4[i] = np.array([0, 1, 0]) @ twf @ np.array([0, 0, 1])

        twg = get_g_matrix(0, xxj[i], yyj[i])
        tgh = get_g_matrix(tj[i], 0, 0)
        thi = get_g_matrix(0, L2/2, 0)
        thj = get_g_matrix(0, 0, L2/2)
```

```
    thk = get_g_matrix(0, -L2/2, 0)
    thl = get_g_matrix(0, 0, -L2/2)

    twh = twg @ tgh
    twi = twh @ thi
    twj = twh @ thj
    twk = twh @ thk
    twl = twh @ thl

    xj1[i] = np.array([1, 0, 0]) @ twi @ np.array([0, 0, 1])
    yj1[i] = np.array([0, 1, 0]) @ twi @ np.array([0, 0, 1])
    xj2[i] = np.array([1, 0, 0]) @ twj @ np.array([0, 0, 1])
    yj2[i] = np.array([0, 1, 0]) @ twj @ np.array([0, 0, 1])
    xj3[i] = np.array([1, 0, 0]) @ twk @ np.array([0, 0, 1])
    yj3[i] = np.array([0, 1, 0]) @ twk @ np.array([0, 0, 1])
    xj4[i] = np.array([1, 0, 0]) @ twl @ np.array([0, 0, 1])
    yj4[i] = np.array([0, 1, 0]) @ twl @ np.array([0, 0, 1])


################################################
# Getting data from pendulum angle trajectories.

# xx1=xx
# yy1=yy
# xx2=xx1+L2*np.sin(t1)
# yy2=yy1-L2*np.cos(t1)
# xx3=xx1+L2*np.sin(t2)
# yy3=yy1-L2*np.cos(t2)

# # Leg 1
# xx4=xx1-W/2*np.cos(t1)
# yy4=yy1-W/2*np.sin(t1)
# xx5=xx1+W/2*np.cos(t1)
# yy5=yy1+W/2*np.sin(t1)
# xx6=xx2+W/2*np.cos(t1)
# yy6=yy2+W/2*np.sin(t1)
# xx7=xx2-W/2*np.cos(t1)
# yy7=yy2-W/2*np.sin(t1)

# # Leg 2
# xx8 =xx1-W/2*np.cos(t2)
# yy8 =yy1-W/2*np.sin(t2)
# xx9 =xx1+W/2*np.cos(t2)
# yy9 =yy1+W/2*np.sin(t2)
# xx10=xx3+W/2*np.cos(t2)
# yy10=yy3+W/2*np.sin(t2)
# xx11=xx3-W/2*np.cos(t2)
# yy11=yy3-W/2*np.sin(t2)
# N = len(theta_array[0]) # Need this for specifying length of simulation

####################################
# Using these to specify axis limits.
# xm=np.min(xx3)-0.5
# xM=np.max(xx2)+0.5
# ym=np.min(yy1)-0.5
# yM=np.max(yy1)+0.5
xm = -3
xM =  3
ym = -4
yM =  4

###########################
# Defining data dictionary.
# Trajectories are here.
data=[
    dict(x=xxb, y=yyb,
           mode='lines', name='Box',
           line=dict(width=2, color='red')
          ),
    dict(x=xxj, y=yyj,
           mode='lines', name='Jack 1',
           line=dict(width=2, color='blue')
          ),
    dict(x=xxj, y=yyj,
           mode='lines', name='Jack 2',
           line=dict(width=2, color='blue')
          ),
      dict(x=xj1, y=yj1,
           mode='markers', name='mass 1',
           line=dict(width=2, color='red')
          ),
      dict(x=xj2, y=yj2,
           mode='markers', name='mass 2',
           line=dict(width=2, color='green')
          ),
      dict(x=xj3, y=yj3,
           mode='markers', name='mass 3',
```

```
           line=dict(width=2, color='yellow')
          ),
      dict(x=xj4, y=yj4,
           mode='markers', name='mass 4',
           line=dict(width=2, color='blue')
          ),
      dict(x=xb1, y=yb1,
           mode='markers', name='box 1',
           line=dict(width=2, color='red')
          ),
      dict(x=xb2, y=yb2,
           mode='markers', name='box 2',
           line=dict(width=2, color='green')
          ),
      dict(x=xb3, y=yb3,
           mode='markers', name='box 3',
           line=dict(width=2, color='yellow')
          ),
      dict(x=xb4, y=yb4,
           mode='markers', name='box 4',
           line=dict(width=2, color='blue')
          ),
      # dict(x=xx2, y=yy2,
      #      mode='markers', name='Mass 1',
      #      line=dict(width=2, color='blue')
      #     ),
    #   dict(x=xx2, y=yy2,
    #        mode='lines', name='Leg 2',
    #        line=dict(width=2, color='blue')
    #       ),
      # dict(x=xx3, y=yy3,display(Markdown(r'$\lambda \nabla \phi_15=$'))
      #      mode='markers', name='Mass 2',
      #      line=dict(width=2, color='blue')
      #     ),
    #   dict(x=xx2, y=yy2,
    #        mode='markers', name='Pendulum 1 Traj',
    #        marker=dict(color="purple", size=2)
    #       ),
    #   dict(x=xx3, y=yy3,
    #        mode='markers', name='Pendulum 2 Traj',
    #        marker=dict(color="green", size=2)
    #       ),
    ]

################################
# Preparing simulation layout.
# Title and axis ranges are here.
layout=dict(xaxis=dict(range=[xm, xM], autorange=False, zeroline=False,dtick=1),
            yaxis=dict(range=[ym, yM], autorange=False, zeroline=False,scaleanchor = "x",dtick=1),
            title='Jack and Box Simulation',
            hovermode='closest',
            updatemenus= [{'type': 'buttons',
                           'buttons': [{'label': 'Play','method': 'animate',
                                        'args': [None, {'frame': {'duration': T, 'redraw': False}}]},
                                       {'args': [[None], {'frame': {'duration': T, 'redraw': False}, 'mode': 'ir
                                        'transition': {'duration': 0}}],'label': 'Pause','method': 'animate'}
                                      ]
                          }]
           )

########################################
# Defining the frames of the simulation.
# This is what draws the lines from
# joint to joint of the pendulum.
frames=[dict(data=[dict(x=[xb1[k],xb2[k],xb3[k],xb4[k],xb1[k]],
                        y=[yb1[k],yb2[k],yb3[k],yb4[k],yb1[k]],
                        mode='lines',
                        line=dict(color='red', width=3)
                        ),
                   dict(x=[xj1[k],xj3[k]],
                        y=[yj1[k],yj3[k]],
                        mode='lines',
                        line=dict(color='blue', width=3)
                        ),
                   dict(x=[xj2[k],xj4[k]],
                        y=[yj2[k],yj4[k]],
                        mode='lines',
                        line=dict(color='blue', width=3)
                        ),
                   go.Scatter(
                        x=[xj1[k]],
                        y=[yj1[k]],
                        mode="markers",
                        marker=dict(color="red", size=12)),
                   go.Scatter(
                        x=[xj2[k]],
                        y=[yj2[k]],
```

```
                            mode="markers",
                            marker=dict(color="green", size=12)),
                       go.Scatter(
                            x=[xj3[k]],
                            y=[yj3[k]],
                            mode="markers",
                            marker=dict(color="yellow", size=12)),
                       go.Scatter(
                            x=[xj4[k]],
                            y=[yj4[k]],
                            mode="markers",
                            marker=dict(color="blue", size=12)),
                       go.Scatter(
                            x=[xb1[k]],
                            y=[yb1[k]],
                            mode="markers",
                            marker=dict(color="red", size=12)),
                       go.Scatter(
                            x=[xb2[k]],
                            y=[yb2[k]],
                            mode="markers",
                            marker=dict(color="green", size=12)),
                       go.Scatter(
                            x=[xb3[k]],
                            y=[yb3[k]],
                            mode="markers",
                            marker=dict(color="yellow", size=12)),
                       go.Scatter(
                            x=[xb4[k]],
                            y=[yb4[k]],
                            mode="markers",
                            marker=dict(color="blue", size=12)),
                      ]) for k in range(N)]

    #######################################
    # Putting it all together and plotting.
    figure1=dict(data=data, layout=layout, frames=frames)
    iplot(figure1)
```

In [ ]:
```
# print(traj)
theta_array = np.array([
    traj[0],
    traj[1],
    traj[2],
    traj[3],
    traj[4],
    traj[5],
])

animate_system(theta_array, L1=Lb, L2=Lj, W=0.2, T=Tf)
```