IN3140 Assignment 3

Vetle H. Olavesen

April 8, 2024

Task 1

a)

First we find the Langrangian, which is given as:

$$L = K - P = \frac{1}{2}m\dot{y}^2 - mgy$$

Put this into the formula for the force working on a rigid body given as:

$$\frac{d}{dt}(\frac{\delta L}{\delta \dot{y}}) - \frac{\delta L}{\delta y} = f$$

$$\frac{d}{dt}\left(\frac{\delta \frac{1}{2}m\dot{y}^2 - mgy}{\delta \dot{y}}\right) - \frac{\delta\left(\frac{1}{2}m\dot{y}^2 - mgy\right)}{\delta y} = f$$

$$\frac{d}{dt}(m\dot{y}) + mg = f$$

$$m\ddot{y} + mg = f \Rightarrow m\ddot{y} = f - mg$$

Which is the result we wanted.

b)

First we find the kinetic energy for mass 1, this mass is modeled so it includes motor 1, link 1 and motor 2, we can then set all values linked to the other links or motors to 0 in the jacobian:

$$v_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$w_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{q_1} \end{bmatrix}$$

$$K_1 = \frac{1}{2}m_1 \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & 0 & \dot{q_1} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_{1,zz} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{q_1} \end{bmatrix}$$

$$K_1 = \frac{1}{2}I_{1,zz} = \frac{1}{2}\frac{m_1 r_1^2}{2}\dot{q_1} = \frac{m_1 r_1^2}{4}\dot{q_1}$$

Find kinetic energy for mass 2, which is a point mass that include link 2 and motor 3, L_3 is not in the model, so i will set its value to 0:

$$v_1 = \begin{bmatrix} -s_1c_2L_2 & -c_1s_2L_2 & 0 \\ c_1c_2L_2 & -s_1s_2L_2 & 0 \\ 0 & L_2c_2 & 0 \end{bmatrix} * \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \begin{bmatrix} -s_1c_2L_2\dot{q_1} - c_1s_2L_2\dot{q_2} \\ c_1c_2L_2\dot{q_1} - s_1s_2L_2\dot{q_2} \\ L_2c_2\dot{q_2} \end{bmatrix}$$

The angular part of the kinetic energy is 0 here:

$$K_2 = \frac{1}{2}m_2 \begin{bmatrix} -s_1c_2L_2\dot{q_1} - c_1s_2L_2\dot{q_2} \\ c_1c_2L_2\dot{q_1} - s_1s_2L_2\dot{q_2} \\ L_2c_2\dot{q_2} \end{bmatrix}^T \begin{bmatrix} -s_1c_2L_2\dot{q_1} - c_1s_2L_2\dot{q_2} \\ c_1c_2L_2\dot{q_1} - s_1s_2L_2\dot{q_2} \\ L_2c_2\dot{q_2} \end{bmatrix}$$

$$= \frac{1}{2}m_2((s_1c_2L_2\dot{q_1})^2 + 2s_1c_2L_2\dot{q_1}c_1s_2L_2\dot{q_2} + (c_1s_2L_2\dot{q_2})^2 + (c_1c_2L_2\dot{q_1})^2 - 2c_1c_2L_2\dot{q_1}s_1s_2L_2\dot{q_2}$$

$$+(s_1s_2L_2\dot{q_2})^2 + (L_2c_2\dot{q_2})^2)$$

$$= \frac{1}{2}m_2((s_1c_2L_2\dot{q_1})^2 + (c_1s_2L_2\dot{q_2})^2 + (c_1c_2L_2\dot{q_1})^2 + (s_1s_2L_2\dot{q_2})^2 + (L_2c_2\dot{q_2})^2)$$

$$= \frac{1}{2}m_2 * (L_2^2\dot{q}_1^2(s_1^2c_2^2 + c_1^2c_2^2) + L_2^2\dot{q}_2^2(c_1^2s_2^2 + s_1^2s_2^2 + c_2^2))$$

$$K_2 = \frac{1}{2}m_2(L_2^2\dot{q}_1^2c_2^2 + L_2^2\dot{q}_2^2) = \frac{1}{2}m_2L_2^2(\dot{q}_1^2c_2^2 + \dot{q}_2^2)$$

This will give us the full kinematic expression:

$$K = \frac{1}{2}m_2L_2^2(\dot{q}_1^2c_2^2 + \dot{q}_2^2) + \frac{m_1r_1^2}{4}\dot{q}_1$$

Now we find the potential energy:

$$h_1 = \frac{L_1}{2}$$

$$s_2 = \frac{h_2 - L_1}{L_2} \Rightarrow h_2 = L_2s_2 + L_1$$

We use this to find the potential energy for each mass:

$$P_1 = m_1g\frac{L_1}{2}$$

$$P_2 = m_2g(L_2s_2 + L_1)$$

$$P = m_1g\frac{L_1}{2} + m_2g(L_2s_2 + L_1)$$

This will give us the Langrangian:

$$L = K - P = \frac{1}{2}m_2L_2^2(\dot{q}_1^2c_2^2 + \dot{q}_2^2) + \frac{m_1r_1^2}{4}\dot{q}_1 - m_1g\frac{L_1}{2} - m_2g(L_2s_2 + L_1)$$

Then i put in the DH-variables instead of $\dot{q}_1$ and $\dot{q}_2$:

$$L = \frac{1}{2}m_2L_2^2(\dot{\theta}_1^2c_2^2 + \dot{\theta}_2^2) + \frac{m_1r_1^2}{4}\dot{\theta}_1 - m_1g\frac{L_1}{2} - m_2g(L_2s_2 + L_1)$$

b)

Now that we have the Langrangian, we can calculate the dynamical model for the dust crawler:

$$\tau_1 = \frac{d}{dt}(\frac{\delta L}{\delta \dot{\theta}_1}) - \frac{\delta L}{\delta \theta_1}$$

$$\tau_2 = \frac{d}{dt}(\frac{\delta L}{\delta \dot{\theta}_2}) - \frac{\delta L}{\delta \theta_2}$$

$$\tau_1 = \frac{d}{dt}\left(m_2 L_2^2 c_2^2 \dot{\theta}_1 + \frac{m_1 r_1^2}{4}\right) = m_2 L_2^2 c_2^2 \ddot{\theta}_1 + \frac{m_1 r_1^2}{4}$$

$$\tau_2 = \frac{d}{dt}\left(m_2 L_2^2 \dot{\theta}_2\right) + m_2 L_2^2 \dot{\theta}_1^2 c_2 s_2 + m_2 g L_2 c_2 = m_2 L_2^2 \ddot{\theta}_2 + m_2 L_2^2 \dot{\theta}_1^2 c_2 s_2 + m_2 g L_2 c_2$$

We can then put these together for the complete dynamic model:

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} m_2 L_2^2 c_2^2 \ddot{\theta}_1 + \frac{m_1 r_1^2}{4} \\ m_2 L_2^2 \ddot{\theta}_2 + m_2 L_2^2 \dot{\theta}_1^2 c_2 s_2 + m_2 g L_2 c_2 \end{bmatrix}$$

We can rewrite this as:

$$\tau = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q)$$

$$\tau = \begin{bmatrix} m_2 L_2^2 cos^2(\theta_2) & 0 \\ 0 & m_2 L_2^2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ m_2 L_2^2 cos(\theta_2) sin(\theta_2) \dot{\theta}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} \frac{m_1 r_1^2}{4} \\ m_2 g L_2 cos(\theta_2) \end{bmatrix}$$

Task 2

a,b,d,e)

Under is the complete code for this task, this is also delivered separately:

```python
import sympy as sp
import numpy as np
from sympy import cos, sin, Symbol, symbols, diff
from sympy.matrices import Matrix
from sympy.physics.vector import dynamicsymbols, ReferenceFrame, time_derivative

sp.init_printing(use_unicode=True)

# Define weight of masses
m1 = 0.3833
m2 = 0.2724
m3 = 0.1406

dot = '\u0307'

# Define symbols for the link lengths and theta
L1 = Symbol("L1")
L2 = Symbol("L2")
L3 = Symbol("L3")
th1 = dynamicsymbols("theta1", real=True)
th2 = dynamicsymbols("theta2", real=True)
th3 = dynamicsymbols("theta3", real=True)

q = Matrix([th1,
            th2,
            th3])
qdot = Matrix([diff(th1,Symbol('t')),
               diff(th2,Symbol('t')),
               diff(th3,Symbol('t'))])

sp.pprint(qdot)
#Column vector containing gravitational constant
```

```
g = Matrix([0,
            0,
            9.81])

# Define the inertia tensors for each mass.
I1x, I1y, I1z, I2x, I2y, I2z, I3x, I3y, I3z = symbols('I1x I1y I1z I2x I2y I2z I3x I3y I3z',
    positive=True)

I1 = Matrix([[I1x, 0, 0],
            [0, I1y, 0],
            [0, 0, I1z]])

I2 = Matrix([[I2x, 0, 0],
            [0, I2y, 0],
            [0, 0, I2z]])

I3 = Matrix([[I3x, 0, 0],
            [0, I3y, 0],
            [0, 0, I3z]])

# Define vectors to height of mass.
rc1 = Matrix([0,
              0,
              L1/2])

rc2 = Matrix([0,
              0,
              (L2*sin(th2) + 2*L1)/2])

rc3 = Matrix([0,
              0,
              (L2*sin(th2) + 2*L1)/2])

J = Matrix([[-sin(th1)*(L2*cos(th2) + L3*cos(th2+th3)), -cos(th1)*(L2*sin(th2) + L3*sin(th2+th3)),
     -cos(th1)*(L3*sin(th2+th3))],
            [cos(th1)*(L2*cos(th2) + L3*cos(th2+th3)), -sin(th1)*(L2*sin(th2) + L3*sin(th2+th3)),
    -sin(th1)*(L3*sin(th2+th3))],
            [0, (L2*cos(th2) + L3*cos(th2+th3)), L3*cos(th2+th3)],
            [0, sin(th1), sin(th1)],
            [0, -cos(th1), -cos(th1)],
            [1, 0, 0]])

#Define the Jvi and Jw matrices
Jv1 = Matrix([[0, 0, 0],
              [0, 0, 0],
              [0, 0, 0]])
Jv2 = Matrix([[-sin(th1)*(L2*cos(th2)), -cos(th1)*(L2*sin(th2)), 0],
              [cos(th1)*(L2*cos(th2)), -sin(th1)*(L2*sin(th2)), 0],
              [0, (L2*cos(th2)), 0]])
Jv3 = Matrix([[-sin(th1)*(L2*cos(th2) + L3*cos(th2+th3)), -cos(th1)*(L2*sin(th2) + L3*sin(th2+th3)
    ), -cos(th1)*(L3*sin(th2+th3))],
            [cos(th1)*(L2*cos(th2) + L3*cos(th2+th3)), -sin(th1)*(L2*sin(th2) + L3*sin(th2+th3)),
    -sin(th1)*(L3*sin(th2+th3))],
            [0, (L2*cos(th2) + L3*cos(th2+th3)), L3*cos(th2+th3)]])

Jw = Matrix([[0, sin(th1), sin(th1)],
              [0, -cos(th1), -cos(th1)],
              [1, 0, 0]])

# Calculate the rotational matrix from base frame to all mass centers.
RM1 = Matrix([[1, 0, 0],
              [0, 1, 0],
              [0, 0, 1]])

RM2 = RM1*Matrix([[cos(th2), -sin(th2), 0],
                  [sin(th2), cos(th2), 0],
                  [0, 0, 1]])

RM3 = RM2*Matrix([[cos(th3), -sin(th3), 0],
                  [sin(th3), cos(th3), 0],
                  [0, 0, 1]])
```

```python

# Calculate the potential energy for the masses.
P1 = m1*g.T*rc1
P2 = m2*g.T*rc2
P3 = m2*g.T*rc3
P = P1+P2+P3

print("The potential energy for the dust crawler: \n")
sp.pprint(sp.simplify(P))

# Kinetic energy terms
K1 = (m1*Jv1.T*Jv1) + (Jw.T*RM1*I1*RM1.T*Jw)
K2 = (m1*Jv2.T*Jv2) + (Jw.T*RM2*I2*RM2.T*Jw)
K3 = (m1*Jv3.T*Jv3) + (Jw.T*RM3*I3*RM3.T*Jw)

Dq = K1+K2+K3

# The finished kinetic energy for the robot.
K = 0.5*qdot.T*Dq*qdot

# Calculate the coriolis/sentripedal matrix
def SumCkj(k,j):
    Ckj = 0
    for i in range(3): # Implementation of  equation (7.60) in book
        Ckj += 0.5*(diff(Dq[k,j],q[j]) + diff(Dq[k,i], q[j]) - diff(Dq[i,j],q[k]))
    return Ckj

C = Matrix([[SumCkj(0,0), SumCkj(0,1), SumCkj(0,2)],
            [SumCkj(1,0), SumCkj(1,1), SumCkj(1,2)],
            [SumCkj(2,0), SumCkj(2,1), SumCkj(2,2)]])



# Calculate the gravitational matrix.
g = Matrix([diff(P,th1),
            diff(P,th2),
            diff(P,th3)])

# The complete dynamic model
tau = Dq + C + g
```

Listing 1: Dynamic model for three link dust crawler implemented in Python.

c)

First we need to expand the elements of the dynamic model, first we extend the Inertia component, the inertia matrix captures the resistance of the system to change its motion due to its mass distribution:

$$D(q)\ddot{q} = \sum_{j=1}^{n} d_{kj}(q)\ddot{q}_j$$

Where dkj is the j-th element of the inertia component. Then we expand the coriolis or centripedal forces matrix, this matrix accounts for the forces that accumulates due to the velocities of the masses of the robot, these forces are depend on both the configuration q and the velocities $\dot{q}$:

$$C(q,\dot{q})\dot{q} = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j$$

The gravitational forces vector represent the gravitational forces acting on the masses on the robot, this component remain unchanged, and are the parts of the dynamic model that doesn't depend on velocities or accelerations, we can then put all these expressions together and end up with our dynamic model.