ME 639: Introduction to Robotics | Assignment - 5

Chaman Modi | 20310015

Task 1 Control tutorial reviewed

In [1]:
```
# Importing libraries and creating basic rotation functions
import numpy as np
import scipy as sci
import sympy as sp
sp.init_printing()
import matplotlib.pyplot as plt
import math
```

Task 2 Stanford dynamics

In [2]:
```
# Creating the joint variables and arm parameters (symbolic)
t = sp.Symbol('t')
q1 = sp.Function('q1')(t)
q2 = sp.Function('q2')(t)
d = sp.Function('d')(t)

l1, l2, m1, m2, m3 = sp.symbols("l1, l2, m1, m2, m3")
I1 = 0
I2 = m2*(l2**2)/12

# Jacobian of  the points at the center  of links i.e. c1,c2,c3
Jvc1 = np.array([[0, 0, 0],
                 [0, 0, 0],
                 [0, 0, 0]])

Jvc2 = np.array([[-0.5*l2*sp.sin(q1)*sp.cos(q2), -0.5*l2*sp.cos(q1)*sp.sin(q2), 0],
                 [0.5*l2*sp.cos(q1)*sp.cos(q2), -0.5*l2*sp.sin(q1)*sp.sin(q2), 0],
                 [0, 0.5*l2*sp.cos(q2), 0]])

Jvc3 = np.array([[-(0.5*d+l2)*sp.sin(q1)*sp.cos(q2), -(0.5*d+l2)*sp.cos(q1)*sp.sin(q2), 0.5*sp.cos(q1)*sp.cos(q2)],
                 [(0.5*d+l2)*sp.cos(q1)*sp.cos(q2), -(d+l2)*sp.sin(q1)*sp.sin(q2), 0.5*sp.sin(q1)*sp.cos(q2)],
                 [0, (l2+0.5*d)*sp.cos(q2), 0.5*sp.sin(q2)]])


# Creating Dq matrix
Dq = m1*np.transpose(Jvc1)@Jvc1  +  m2*np.transpose(Jvc2)@Jvc2  +  m3*np.transpose(Jvc3)@Jvc3 + np.array([[I1+I2, 0, 0],[0, I2, 0],[0, 0, 0]])
print('D(q) = ')
sp.simplify(Dq)
```

Out[2]:
D(q) =

$$\begin{bmatrix} 0.25l_2^2m_2\cos^2(q_2(t)) + 0.0833333333333333l_2^2m_2 + 1.0l_2^2m_3\cos^2(q_2(t)) + 1.0l_2m_3d(t)\cos^2(q_2(t)) + 0.25m_3d^2(t)\cos^2(q_2(t)) & -0.0625m_3\left(l_2 + 0.5d(t)\right)\left(\cos\left(2\,q_1(t) - 2\,q_2(t)\right) - \cos\left(2\,q_1(t) + 2\,q_2(t)\right)\right)\dfrac{d}{dt}d(t) \\ -0.0625m_3\left(l_2 + 0.5d(t)\right)\left(\cos\left(2\,q_1(t) - 2\,q_2(t)\right) - \cos\left(2\,q_1(t) + 2\,q_2(t)\right)\right)d(t) & 0.333333333333333l_2^2m_2 + 1.0l_2^2m_3 + 1.0l_2m_3d(t)\sin^2(q_1(t))\sin^2(q_2(t)) + 1.0l_2m_3d(t) + 0.75m_3d^2(t)\sin^2(q_1(t))\sin^2(q_2(t)) \\ 0 & -0.25m_3d(t)\sin^2(q_1(t))\sin\left(q_2(t)\right)\cos\left(q_2(t)\right) \\ \\ 0 \\ \\ -0.25m_3d(t)\sin^2(q_1(t))\sin\left(q_2(t)\right)\cos\left(q_2(t)\right) \\ \\ 0.333333333333333l_2^2m_2 + 1.0l_2^2m_3 \end{bmatrix}$$

In [3]:
```
# Vector containing joint variables
q = sp.Array([q1,q2,d])

# Joint velocities (q_dot)
q_dot = sp.diff(q,t)

# Joint accelaration (q_dotdot)
q_dotdot = sp.diff(q_dot,t)

# Potential energy expression
g = 9.81
V = m1*g*l1/2 + m2*g*l1 + m3*g*l1

# Intializing list for christoffel symbols
c = [[[0]*3]*3]*3

# Calculating christoffel symbols
for k in range(0,3):
    for i in range(0,3):
        for j in range(0,3):
            c[i][j][k] = 0.5*(sp.diff(Dq[k][j],q[i]) + sp.diff(Dq[k][i],q[j]) - sp.diff(Dq[i][j],q[k]))

phi = sp.zeros(3,1)
tau = sp.zeros(3,1)

# Finally creating the torque vector
for k in range(3):
    phi[k] = sp.diff(V, q[k])
    d_temp = 0
    ct = 0
    for j in range(3):
        d_temp = d_temp + Dq[k][j] * q_dotdot[j]
        for i in range(3):
            ct = ct + c[i][j][k] * q_dot[i] * q_dotdot[j]
    tau[k] = d_temp + ct + phi[k]

print('tau = ')
sp.simplify(tau)
```

Out[3]:
tau =

$$\begin{bmatrix} -0.0625m_3\left(l_2 + 0.5d(t)\right)\left(\cos\left(2\,q_1(t) - 2\,q_2(t)\right) - \cos\left(2\,q_1(t) + 2\,q_2(t)\right)\right)d(t)\dfrac{d^2}{dt^2}q_2(t) + \dfrac{\left(3.0l_2^2m_2\cos^2(q_2(t)) + 1.0l_2^2m_2 + 12.0l_2^2m_3\cos^2(q_2(t)) + 12.0l_2m_3d(t)\cos^2(q_2(t)) + 3.0m_3d^2(t)\cos^2(q_2(t))\right)\dfrac{d}{dt}q_1(t)}{12} \\ -0.0625m_3\left(l_2 + 0.5d(t)\right)\left(\cos\left(2\,q_1(t) - 2\,q_2(t)\right) - \cos\left(2\,q_1(t) + 2\,q_2(t)\right)\right)d(t)\dfrac{d^2}{dt^2}q_1(t) - 0.25m_3d(t)\sin^2(q_1(t))\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d^2}{dt^2}d(t) - 0.25m_3\sin^2(q_1(t))\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d}{dt}d(t)\dfrac{d^2}{dt^2}d(t) - 0.25m_3\sin^2(q_1(t))\sin(q_2(t))\cos\left(q_2(t)\right)\left(\dfrac{d}{dt}q_2(t)\right)^2\dfrac{d^2}{dt^2}d(t) - 0.25m_3\sin^2(q_1(t))\sin(q_2(t))\cos(q_2(t))\dfrac{d^2}{dt^2}q_2(t) - 0.25m_3\sin^2(q_1(t))\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d}{dt}q_1(t)\dfrac{d^2}{dt^2}q_1(t) - 0.25m_3\sin^2(q_1(t))\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d}{dt}q_2(t)\dfrac{d^2}{dt^2}q_2(t) \\ + \dfrac{\left(4.0l_2^2m_2 + 12.0l_2^2m_3\sin^2(q_2(t))\sin^2(q_1(t)) + 12.0l_2m_3d(t) + 9.0m_3d^2(t)\sin^2(q_2(t))\right)\dfrac{d^2}{dt^2}q_2(t)}{12} \\ 0.25m_3\left(-d(t)\sin^2(q_1(t))\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d^2}{dt^2}q_2(t) + \dfrac{d^2}{dt^2}d(t)\right) \end{bmatrix}$$

Task 3 SCARA dynamics

In [4]:
```
# Creating the joint variables and arm parameters (symbolic)
t = sp.Symbol('t')
q1 = sp.Function('q1')(t)
q2 = sp.Function('q2')(t)
d = sp.Function('d')(t)

l1, l2, m1, m2, m3 = sp.symbols("l1, l2, m1, m2, m3")
I1 = m1*(l1**2)/12
I2 = m2*(l2**2)/12

# Jacobian of  the points at the center  of links i.e. c1,c2,c3
Jvc1 = np.array([[-0.5*l1*sp.sin(q1), 0, 0],
                 [0.5*l1*sp.cos(q1), 0, 0],
                 [0, 0, 0]])

Jvc2 = np.array([[-l1*sp.sin(q1)-0.5*l2*sp.sin(q1+q2), -0.5*l2*sp.sin(q1+q2), 0],
                 [l1*sp.cos(q1)+0.5*l2*sp.cos(q1+q2), 0.5*l2*sp.cos(q1+q2), 0],
                 [0, 0, 0]])

Jvc3 = np.array([[-l1*sp.sin(q1)-l2*sp.sin(q1+q2), l2*sp.sin(q1+q2), 0],
                 [l1*sp.cos(q1)+l2*sp.cos(q1+q2), l2*sp.cos(q1+q2), 0],
                 [0, 0, -0.5]])

# Creating Dq matrix
Dq = m1*np.transpose(Jvc1)@Jvc1  +  m2*np.transpose(Jvc2)@Jvc2  +  m3*np.transpose(Jvc3)@Jvc3 + np.array([[I1+I2, I2, 0],[I2, I2, 0],[0, 0, 0]])
print('D(q) = ')
sp.simplify(Dq)
```

Out[4]:
D(q) =

$$\begin{bmatrix} 0.333333333333333l_1^2m_1 + 1.0l_1^2m_2 + 1.0l_1^2m_3 + 1.0l_1l_2m_2\cos\left(q_2(t)\right) + 2.0l_1l_2m_3\cos\left(q_2(t)\right) + 0.333333333333333l_2^2m_2 + 1.0l_2^2m_3 & \frac{l_2\left(l_1m_2 + 6.0m_3\left(l_1\sin\left(q_1(t)\right) + 0.5l_2\sin\left(q_1(t)+q_2(t)\right)\right)\sin\left(q_1(t)+q_2(t)\right) + 6.0m_3l_1\cos\left(q_1(t)+q_2(t)\right) + 3.0l_2m_3\right)}{12} \\ \frac{l_2\left(l_1m_2 + 6.0m_3\left(l_1\sin\left(q_1(t)\right) + 0.5l_2\sin\left(q_1(t)+q_2(t)\right)\right)\sin\left(q_1(t)+q_2(t)\right) + 6.0m_3l_1\cos\left(q_1(t)+q_2(t)\right) + 0.5l_2\cos\left(q_1(t)+q_2(t)\right) + 3.0l_2m_3\right)}{12} & 0 \end{bmatrix}$$

In [5]:
```
# Vector containing joint variables
q = np.array([q1,q2,d])

# Joint velocities (q_dot)
q_dot = sp.diff(q,t)

# Joint accelaration (q_dotdot)
q_dotdot = sp.diff(q_dot,t)

# Potential energy expression
V = m3*9.81*(-d/2)

# Intializing list for christoffel symbols
c = [[[0]*3]*3]*3

# Calculating christoffel symbols
for k in range(0,3):
    for i in range(0,3):
        for j in range(0,3):
            c[i][j][k] = 0.5*(sp.diff(Dq[k][j],q[i]) + sp.diff(Dq[k][i],q[j]) - sp.diff(Dq[i][j],q[k]))

phi = sp.zeros(3,1)
tau = sp.zeros(3,1)

# Finally creating the torque vector
for k in range(3):
    phi[k] = sp.diff(V, q[k])
    d_temp = 0
    ct = 0
    for j in range(3):
        d_temp = d_temp + Dq[k][j] * q_dotdot[j]
        for i in range(3):
            ct = ct + c[i][j][k] * q_dot[i] * q_dotdot[j]
    tau[k] = d_temp + ct + phi[k]

print('tau = ')
sp.simplify(tau)
```

Out[5]:
tau =

$$\begin{bmatrix} \frac{l_2\left(l_1m_2 + 6.0m_3\left(l_1\sin\left(q_1(t)\right) + 0.5l_2\sin\left(q_1(t)+q_2(t)\right)\right)\sin\left(q_1(t)+q_2(t)\right) + 6.0m_3l_1\cos\left(q_1(t)+q_2(t)\right) + 0.5l_2\cos\left(q_1(t)+q_2(t)\right) + 3.0l_2m_3\right)\cos\left(q_1(t)+q_2(t)\right) - 12m_3l_1\dot{q}_1(t)\sin\left(q_1(t)+q_2(t)\right) + l_2\sin\left(q_1(t)+q_2(t)\right)\left(6.0m_3l_1\cos\left(q_1(t)+q_2(t)\right) + 3.0l_2m_3\right)\dot{q}_1(t) + 12m_3l_1\cos\left(q_1(t)+q_2(t)\right) + l_2\cos\left(q_1(t)+q_2(t)\right)\right)\dot{q}_2(t)}{12} \\ + \frac{\left(4.0l_2^2m_2 + 12.0l_2^2m_3 + 12.0l_1l_2m_3\cos\left(q_2(t)\right) + 24.0l_1l_2m_3\cos\left(q_2(t)\right) + 4.0l_1^2m_2 + 12.0l_1^2m_3\right)\dot{q}_2(t)}{12} \\ \frac{l_2\left(l_2\left(4.0m_2 + 12.0m_3\right)\dot{q}_2(t) + \left(l_2m_2 + 6.0m_2\left(l_1\sin\left(q_1(t)\right) + 0.5l_2\sin\left(q_1(t)+q_2(t)\right)\right)\sin\left(q_1(t)+q_2(t)\right) + 6.0m_2l_1\cos\left(q_1(t)+q_2(t)\right) + 0.5l_2\cos\left(q_1(t)+q_2(t)\right) + 3.0l_2m_3\right)\cos\left(q_1(t)+q_2(t)\right) - 12m_2l_1\dot{q}_1(t)\sin\left(q_1(t)+q_2(t)\right) + l_2\sin\left(q_1(t)+q_2(t)\right)\left(6.0m_2l_1\cos\left(q_1(t)+q_2(t)\right) + 3.0l_2m_3\right)\dot{q}_1(t) + 12m_2l_1\cos\left(q_1(t)+q_2(t)\right) + l_2\cos\left(q_1(t)+q_2(t)\right)\right)\dot{q}_1(t)\right)}{12} \\ m_3\left(0.25\dfrac{d^2}{dt^2}d(t) - 4.905\right) \end{bmatrix}$$

Task 4 PUMA dynamics

In [6]:
```
# Creating the joint variables and arm parameters (symbolic)
t = sp.Symbol('t')
q1 = sp.Function('q1')(t)
q2 = sp.Function('q2')(t)
q3 = sp.Function('q3')(t)

l1, l2, l3, m1, m2, m3 = sp.symbols("l1, l2, l3, m1, m2, m3")
I1 = 0
I2 = m2*(l2**2)/12
I3 = m3*(l3**2)/12

# Jacobian of  the points at the center  of links i.e. c1,c2,c3
Jvc1 = np.array([[0, 0, 0],
                 [0, 0, 0],
                 [0, 0, 0]])

Jvc2 = np.array([[-0.5*l2*sp.sin(q1)*sp.cos(q2), -0.5*l2*sp.cos(q1)*sp.sin(q2), 0],
                 [0.5*l2*sp.cos(q1)*sp.cos(q2), -0.5*l2*sp.sin(q1)*sp.sin(q2), 0],
                 [0, 0.5*l2*sp.cos(q2), 0]])

Jvc3 = np.array([[-(0.5*l3*sp.cos(q3)+l2)*sp.sin(q1)*sp.cos(q2), -(0.5*l3*sp.cos(q3)+l2)*sp.cos(q1)*sp.sin(q2), 0.5*l3*sp.cos(q1)*sp.cos(q2)*sp.sin(q3)],
                 [(0.5*l3*sp.cos(q3)+l2)*sp.cos(q1)*sp.cos(q2), -(0.5*l3*sp.cos(q3)+l2)*sp.sin(q1)*sp.sin(q2), 0.5*l3*sp.sin(q1)*sp.cos(q2)*sp.sin(q3)],
                 [0, l2*sp.cos(q2), 0.5*l3*sp.cos(q2)*sp.cos(q3)]])

# Creating Dq matrix
Dq = m1*np.transpose(Jvc1)@Jvc1  +  m2*np.transpose(Jvc2)@Jvc2  +  m3*np.transpose(Jvc3)@Jvc3 + np.array([[I1+I2+I3, 0, 0],[0, I2+I3, I3],[0, I3, I3]])
print('D(q) = ')
sp.simplify(Dq)
```

Out[6]:
D(q) =

$$\begin{bmatrix} 0.25l_2^2m_2\cos^2(q_2(t)) + 0.0833333333333333l_2^2m_2 + 1.0l_2^2m_3\cos^2(q_2(t)) + 1.0l_2l_3m_3\cos^2(q_2(t))\cos(q_3(t)) + 0.25l_3^2m_3\cos^2(q_2(t))\cos^2(q_3(t)) + 0.0833333333333333l_3^2m_3 & 0 & 0 \\ 0 & 0.333333333333333l_2^2m_2 + 1.0l_2^2m_3 + 1.0l_2l_3m_3\sin^2(q_2(t))\cos(q_3(t)) - 0.25l_3^2m_3\sin^2(q_2(t))\cos^2(q_3(t)) + 0.333333333333333l_3^2m_3 & \frac{l_3m_3\left(-1.5l_2\cos\left(2\,q_2(t) - q_3(t)\right) + 1.5l_2\cos\left(2\,q_2(t) + q_3(t)\right) + 3.0l_2\cos\left(2\,q_2(t)\right) - 0.375l_3\cos\left(2\,q_2(t) - 2\,q_3(t)\right) + 0.375l_3\cos\left(2\,q_2(t) + 2\,q_3(t)\right) + 1.0l_3\right)}{12} \\ 0 & \frac{l_3m_3\left(-1.5l_2\cos\left(2\,q_2(t) - q_3(t)\right) + 1.5l_2\cos\left(2\,q_2(t) + q_3(t)\right) + 3.0l_2\cos\left(2\,q_2(t)\right) - 0.375l_3\cos\left(2\,q_2(t) - 2\,q_3(t)\right) + 0.375l_3\cos\left(2\,q_2(t) + 2\,q_3(t)\right) + 1.0l_3\right)}{12} & 0.333333333333333l_3^2m_3 \end{bmatrix}$$

In [7]:
```
# Vector containing joint variables
q = np.array([q1,q2,q3])

# Joint velocities (q_dot)
q_dot = sp.diff(q,t)

# Joint accelaration (q_dotdot)
q_dotdot = sp.diff(q_dot,t)

# Potential energy expression
g = 9.81
V = m1*g*l1/2 + m2*g*(l1+0.5*l2*sp.sin(q2)) + m3*g*(l1+l2*sp.sin(q2)+0.5*l3*sp.sin(q3))

# Intializing list for christoffel symbols
c = [[[0]*3]*3]*3

# Calculating christoffel symbols
for k in range(0,3):
    for i in range(0,3):
        for j in range(0,3):
            c[i][j][k] = 0.5*(sp.diff(Dq[k][j],q[i]) + sp.diff(Dq[k][i],q[j]) - sp.diff(Dq[i][j],q[k]))

phi = sp.zeros(3,1)
tau = sp.zeros(3,1)

# Finally creating the torque vector
for k in range(3):
    phi[k] = sp.diff(V, q[k])
    d_temp = 0
    ct = 0
    for j in range(3):
        d_temp = d_temp + Dq[k][j] * q_dotdot[j]
        for i in range(3):
            ct = ct + c[i][j][k] * q_dot[i] * q_dotdot[j]
    tau[k] = d_temp + ct + phi[k]

print('tau = ')
sp.simplify(tau)
```

Out[7]:
tau =

$$\begin{bmatrix} \frac{\left(3.0l_2^2m_2\cos^2(q_2(t)) + 1.0l_2^2m_2 + 12.0l_2^2m_3\cos^2(q_2(t)) + 12.0l_2l_3m_3\cos^2(q_2(t))\cos(q_3(t)) + 3.0l_3^2m_3\cos^2(q_2(t))\cos^2(q_3(t)) + 1.0l_3^2m_3\right)\dfrac{d}{dt}q_1(t)}{12} \\ 4.905l_2m_2\cos\left(q_2(t)\right) + 9.81l_3m_3\cos\left(q_2(t)+q_3(t)\right) + l_3m_3\left(-0.5l_2\cos\left(q_3(t)\right) + 0.75l_3\sin^2\left(q_3(t)\right)\right)\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d}{dt}q_1(t)\dfrac{d^2}{dt^2}q_1(t) + l_3m_3\left(-0.5l_2\cos\left(q_3(t)\right) + 0.75l_3\sin^2\left(q_3(t)\right)\right)\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d^2}{dt^2}q_1(t)\dfrac{d}{dt}q_2(t) + l_3m_3\left(-0.5l_2\cos\left(q_3(t)\right) + 0.75l_3\sin^2\left(q_3(t)\right)\right)\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d}{dt}q_1(t)\dfrac{d^2}{dt^2}q_2(t) + l_3m_3\left(-0.5l_2\cos\left(q_3(t)\right) + 0.75l_3\sin^2\left(q_3(t)\right)\right)\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d}{dt}q_2(t)\dfrac{d^2}{dt^2}q_1(t) + l_3m_3\left(-0.5l_2\cos\left(q_3(t)\right) + 0.75l_3\sin^2\left(q_3(t)\right)\right)\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d}{dt}q_2(t)\dfrac{d^2}{dt^2}q_2(t) + l_3m_3\left(-0.5l_2\cos\left(q_3(t)\right) + 0.75l_3\sin^2\left(q_3(t)\right)\right)\sin\left(q_2(t)\right)\cos\left(q_2(t)\right)\dfrac{d}{dt}q_2(t)\dfrac{d^2}{dt^2}q_1(t) + \dfrac{l_3m_3\left(-1.5l_2\cos\left(2\,q_2(t) - q_3(t)\right) + 1.5l_2\cos\left(2\,q_2(t) + q_3(t)\right) + 3.0l_2\cos\left(2\,q_2(t)\right) - 0.375l_3\cos\left(2\,q_2(t) - 2\,q_3(t)\right) + 0.375l_3\cos\left(2\,q_2(t) + 2\,q_3(t)\right) + 1.0l_3\right)\dfrac{d^2}{dt^2}q_2(t)}{12} \\ + \dfrac{\left(4.0l_2^2m_2 + 12.0l_2^2m_3 + 12.0l_2l_3m_3\sin^2\left(q_3(t)\right)\cos\left(q_3(t)\right) - 3.0l_2^2m_3\sin^2\left(q_3(t)\right) + 24.0l_2l_3m_3\cos\left(q_3(t)\right) - 3.0l_3^2m_3\sin^2\left(q_3(t)\right) + 1.0l_3^2m_3\right)\dfrac{d^2}{dt^2}q_2(t)}{12} \\ l_3m_3\left(l_3\left(-3.0\cos^2(q_3(t))\cos^2(q_2(t)) + 6.0\cos^2(q_3(t)) + 1.0\right)\dfrac{d}{dt}q_3(t) + 3.0l_3\sin\left(q_3(t)\right)\cos^2\left(q_2(t)\right)\cos\left(q_3(t)\right)\dfrac{d}{dt}q_3(t)\dfrac{d^2}{dt^2}q_3(t) + 3.0l_3\sin\left(q_3(t)\right)\cos^2\left(q_3(t)\right)\cos\left(q_3(t)\right)\dfrac{d}{dt}q_3(t)\dfrac{d^2}{dt^2}q_3(t) + 3.0l_3\sin\left(q_3(t)\right)\cos^2\left(q_2(t)\right)\cos\left(q_3(t)\right)\dfrac{d}{dt}q_2(t)\dfrac{d^2}{dt^2}q_3(t) + 3.0l_3\sin\left(q_3(t)\right)\cos^2\left(q_2(t)\right)\cos\left(q_3(t)\right)\dfrac{d}{dt}q_3(t)\dfrac{d^2}{dt^2}q_2(t) + 3.0l_3\sin\left(q_3(t)\right)\cos^2\left(q_3(t)\right)\cos\left(q_3(t)\right)\dfrac{d}{dt}q_2(t)\dfrac{d^2}{dt^2}q_2(t) + 3.0l_3\sin\left(q_3(t)\right)\cos^2\left(q_2(t)\right)\cos\left(q_3(t)\right)\dfrac{d}{dt}q_2(t)\dfrac{d^2}{dt^2}q_3(t) + 3.0l_3\sin\left(q_3(t)\right)\cos^2\left(q_3(t)\right)\cos\left(q_3(t)\right)\dfrac{d}{dt}q_3(t)\dfrac{d^2}{dt^2}q_3(t) + 3.0l_3\sin\left(q_3(t)\right)\cos^2\left(q_2(t)\right)\cos\left(q_3(t)\right)\dfrac{d}{dt}q_2(t)\dfrac{d^2}{dt^2}q_2(t) + 3.0l_3\sin\left(q_3(t)\right)\cos^2\left(q_3(t)\right)\cos\left(q_3(t)\right)\dfrac{d}{dt}q_3(t)\dfrac{d^2}{dt^2}q_3(t) + \left(-1.5l_2\cos\left(2\,q_2(t) - q_3(t)\right) + 1.5l_2\cos\left(2\,q_2(t) + q_3(t)\right) + 3.0l_2\cos\left(2\,q_2(t)\right) - 0.375l_3\cos\left(2\,q_2(t) - 2\,q_3(t)\right) + 0.375l_3\cos\left(2\,q_2(t) + 2\,q_3(t)\right) + 1.0l_3\right)\dfrac{d^2}{dt^2}q_3(t) + 58.86\cos\left(q_2(t)\right)\right) \\ \hline 12 \end{bmatrix}$$