Programming Assignment 1:
Dynamic Modeling and State-Space Representation of the RRBot Robotic Arm

Step 1: Generalized coordinates

q = [$\theta 1$ $\theta 2$]

Step2: Generalized input

u = [tau1 tau2]

tau1, tau2 : torques

Step3: Euler Lagrangian Function

Variables:

```
x1 = r1*sin(theta_1)
y1 = r1*cos(theta_1)
x2 = l1*sin(theta_1) + r2*sin(theta_1 + theta_2)
y2 = l1*cos(theta_1) + r2*cos(theta_1 + theta_2)

xd1 = diff(x1,t)
yd1 = diff(y1,t)
xd2 = diff(x2,t)
yd2 = diff(y2,t)


v1 = sqrt(xd1^2 + yd1^2)
v2 = sqrt(xd2^2 + yd2^2)

w1 = jacobian(theta_1,t)
w2 = jacobian(theta_1+theta_2,t)
```

I1,I2 : inertias

h1 = y1

h2 = y2

KE : Kinetic Energy = `m1*(v1^2))/2 + (I1*w1^2)/2 + (m2*(v2^2))/2 + (I2*w2^2)/2)`

`PE : Potential Energy = m1*g*h1 + m2*g*h2`

`L = KE-PE`

Step 4: Euler Lagrangian Equation

Equations of Motion:

```
eq1= ddl_dtheta1_dot_dt - dl_dtheta1 - tau1

eq2 = ddl_dtheta2_dot_dt - dl_dtheta2 - tau2
```

State Space Representation:

X = [theta_1, theta_2,q_dot(1), q_dot(2)]

X’ = [dX(1), dX(2), dX(3), dX(4)]

```
dX(1) = q_dot(1)
```

```
dX(2) = q_dot(2)
```

```
dX(3) = (I2*tau1 - I2*tau2 + m2*r2^2*tau1 - m2*r2^2*tau2 +
g*l1*m2^2*r2^2*sin(theta_1(t)) + I2*g*l1*m2*sin(theta_1(t)) +
I2*g*m1*r1*sin(theta_1(t)) - l1*m2*r2*tau2*cos(theta_2(t)) +
l1*m2^2*r2^3*sin(theta_2(t))*diff(theta_1(t), t)^2 +
l1*m2^2*r2^3*sin(theta_2(t))*diff(theta_2(t), t)^2 + g*m1*m2*r1*r2^2*sin(theta_1(t))
+ 2*l1*m2^2*r2^3*sin(theta_2(t))*diff(theta_1(t), t)*diff(theta_2(t), t) +
I2*l1*m2*r2*sin(theta_2(t))*diff(theta_1(t), t)^2 +
I2*l1*m2*r2*sin(theta_2(t))*diff(theta_2(t), t)^2 +
l1^2*m2^2*r2^2*cos(theta_2(t))*sin(theta_2(t))*diff(theta_1(t), t)^2 -
g*l1*m2^2*r2^2*cos(theta_2(t))*sin(theta_1(t) + theta_2(t)) +
2*I2*l1*m2*r2*sin(theta_2(t))*diff(theta_1(t), t)*diff(theta_2(t), t))/(-
l1^2*m2^2*r2^2*cos(theta_2(t))^2 + l1^2*m2^2*r2^2 + I2*l1^2*m2 + m1*m2*r1^2*r2^2 +
I1*m2*r2^2 + I2*m1*r1^2 + I1*I2)
```

```
dX(4) = -(I2*tau1 - I1*tau2 - I2*tau2 - l1^2*m2*tau2 - m1*r1^2*tau2 + m2*r2^2*tau1 -
m2*r2^2*tau2 + g*l1*m2^2*r2^2*sin(theta_1(t)) + I2*g*l1*m2*sin(theta_1(t)) +
I2*g*m1*r1*sin(theta_1(t)) + l1*m2*r2*tau1*cos(theta_2(t)) -
2*l1*m2*r2*tau2*cos(theta_2(t)) - g*l1^2*m2^2*r2*sin(theta_1(t) + theta_2(t)) -
I1*g*m2*r2*sin(theta_1(t) + theta_2(t)) +
l1*m2^2*r2^3*sin(theta_2(t))*diff(theta_1(t), t)^2 +
l1^3*m2^2*r2*sin(theta_2(t))*diff(theta_1(t), t)^2 +
l1*m2^2*r2^3*sin(theta_2(t))*diff(theta_2(t), t)^2 + g*m1*m2*r1*r2^2*sin(theta_1(t))
+ 2*l1*m2^2*r2^3*sin(theta_2(t))*diff(theta_1(t), t)*diff(theta_2(t), t) +
I1*l1*m2*r2*sin(theta_2(t))*diff(theta_1(t), t)^2 +
I2*l1*m2*r2*sin(theta_2(t))*diff(theta_1(t), t)^2 +
I2*l1*m2*r2*sin(theta_2(t))*diff(theta_2(t), t)^2 +
g*l1^2*m2^2*r2*cos(theta_2(t))*sin(theta_1(t)) - g*m1*m2*r1^2*r2*sin(theta_1(t) +
theta_2(t)) + 2*l1^2*m2^2*r2^2*cos(theta_2(t))*sin(theta_2(t))*diff(theta_1(t), t)^2
+ l1^2*m2^2*r2^2*cos(theta_2(t))*sin(theta_2(t))*diff(theta_2(t), t)^2 -
g*l1*m2^2*r2^2*cos(theta_2(t))*sin(theta_1(t) + theta_2(t)) +
l1*m1*m2*r1^2*r2*sin(theta_2(t))*diff(theta_1(t), t)^2 +
2*I2*l1*m2*r2*sin(theta_2(t))*diff(theta_1(t), t)*diff(theta_2(t), t) +
2*l1^2*m2^2*r2^2*cos(theta_2(t))*sin(theta_2(t))*diff(theta_1(t), t)*diff(theta_2(t),
t) + g*l1*m1*m2*r1*r2*cos(theta_2(t))*sin(theta_1(t)))/(-
l1^2*m2^2*r2^2*cos(theta_2(t))^2 + l1^2*m2^2*r2^2 + I2*l1^2*m2 + m1*m2*r1^2*r2^2 +
I1*m2*r2^2 + I2*m1*r1^2 + I1*I2)
```

Trajectory plots: