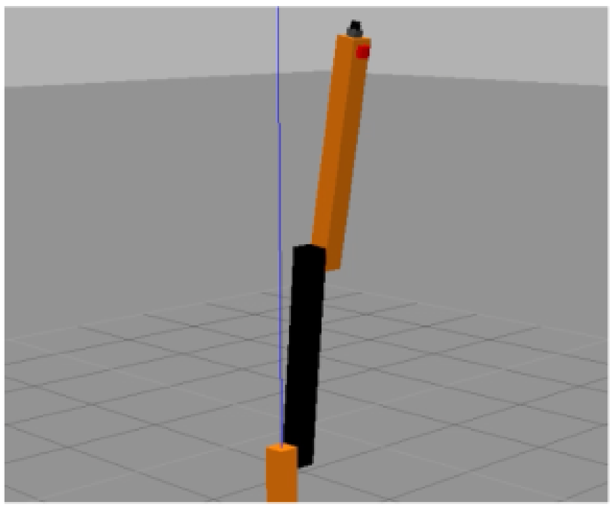
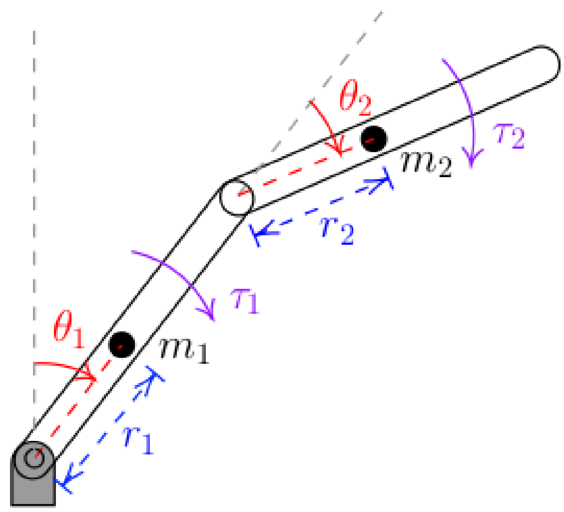
Dynamic modeling of the 2 link Robot





<u>Step 1: Generalized coordinates and input</u>

$q = [\theta_1 \ \theta_2]$
$u = [t_1 \ t_2]$

$t_1, t_2 \rightarrow$ torques

Variables:

$x_1 = r_1 \sin(\theta_1)$
$y_1 = r_1 \cos(\theta_1)$

$x_1' = r_1 \theta_1' \cos(\theta_1)$
$y_1' = - r_1 \theta_1' \sin(\theta_1)$

$x_2 = l_1 \sin(\theta_1) + r_2 \sin(\theta_1+\theta_2)$
$y_2 = l_1 \cos(\theta_1) + r_2 \cos(\theta_1+\theta_2)$

$x_2' = l_1 \theta_1' \cos(\theta_1) + r_2 \cos(\theta_1+\theta_2)(\theta_1' + \theta_2')$
$y_2' = - l_1 \theta_1' \sin(\theta_1) - r_2 \sin(\theta_1+\theta_2)(\theta_1' + \theta_2')$

$i_1, i_2 \rightarrow$ Inertials

<u>Step 2: Kinetic and Potential Energy</u>

Link1

K1 = 0.5 m1 r1^2 θ1'^2 + 0.5 i1 θ1'^2 →Kinetic Energy of link 1
P1 = m1 g r1 cos(θ1) → Potential Energy of link 1

Link2

K2 = 0.5 m2 v2^2 + 0.5 i2 (θ1' + θ2')^2

Where, v2^2 = x2'^2 + y2'^2

K2 = 0.5 m2 { (l1 θ1')^2 + r2 (θ1'+θ2')^2 + 2 l1 θ1' r2 (θ1'+θ2') cos(θ2)} + 0.5 i2 (θ1' + θ2')^2

P2 = m2 g (l1 g r1 cos(θ1) + r2 cos(θ1+θ2))

Total Energy

K = K1 + K2
P = P1 + P2

Step 3: Lagrangian Function

L = K - P

Step 4: Euler-Lagrange

Matlab commands to calculate partial differentiation terms:

∂L
$\partial \theta$**1**
→ **jacobian(L, theta_1)** = *g*m2*(r2*sin(theta_1 + theta_2) + l1*sin(theta_1)) + g*m1*r1*sin(theta_1)*

∂L
$\partial \theta$**2**
→ **jacobian(L, theta_2)** = *g*m2*r2*sin(theta_1 + theta_2) - l1*m2*r2*theta_dot_1*sin(theta_2)*(theta_dot_1 + theta_dot_2)*

∂L
$\partial \theta$**1'**
→ **jacobian(L, theta_dot_1)** = *theta_dot_1*(m1*r1^2 + I1) + (I2*(2*theta_dot_1 + 2*theta_dot_2))/2 + (m2*(r2^2*(2*theta_dot_1 + 2*theta_dot_2) + 2*l1^2*theta_dot_1 + 2*l1*r2*cos(theta_2)*(theta_dot_1 + theta_dot_2) + 2*l1*r2*theta_dot_1*cos(theta_2)))/2*

∂L
$\partial \theta$**2'**
→ **jacobian(L,theta_2_dot)** = (I2*(2*theta_dot_1 + 2*theta_dot_2))/2 + (m2*((2*theta_dot_1 + 2*theta_dot_2)*r2^2 + 2*l1*theta_dot_1*cos(theta_2)*r2))/2

$d \quad \partial L$
$dt\ \partial \theta$**1'**
→jacobian(dl_dtheta1_dot, [theta_1; theta_dot_1])*[theta_dot_1; theta_ddot_1] + jacobian(dl_dtheta1_dot, [theta_2; theta_dot_2])*[theta_dot_2; theta_ddot_2] = theta_ddot_1*(m1*r1^2 + I1 + I2 + (m2*(2*l1^2 + 4*cos(theta_2)*l1*r2 + 2*r2^2))/2) + theta_ddot_2*(I2 + (m2*(2*r2^2 + 2*l1*cos(theta_2)*r2))/2) - (m2*theta_dot_2*(2*l1*r2*sin(theta_2)*(theta_dot_1 + theta_dot_2) + 2*l1*r2*theta_dot_1*sin(theta_2)))/2

$d \quad \partial L$
$dt\ \partial \theta$**2'**
→ **jacobian(dl_dtheta2_dot, [theta_1; theta_dot_1])*[theta_dot_1; theta_ddot_1] + jacobian(dl_dtheta2_dot, [theta_2; theta_dot_2])*[theta_dot_2; theta_ddot_2]** = theta_ddot_2*(m2*r2^2 + I2) + theta_ddot_1*(I2 + (m2*(2*r2^2 + 2*l1*cos(theta_2)*r2))/2) - l1*m2*r2*theta_dot_1*theta_dot_2*sin(theta_2)

Equations of Motion:

Eq1 = ddl_dtheta1_dot_dt - dl_dtheta1 - t1
Eq1 = theta_ddot_1*(m1*r1^2 + I1 + I2 + (m2*(2*l1^2 + 4*cos(theta_2)*l1*r2 + 2*r2^2))/2) - t1 + theta_ddot_2*(I2 + (m2*(2*r2^2 + 2*l1*cos(theta_2)*r2))/2) - (m2*theta_dot_2*(2*l1*r2*sin(theta_2)*(theta_dot_1 + theta_dot_2) + 2*l1*r2*theta_dot_1*sin(theta_2)))/2 - g*m2*(r2*sin(theta_1 + theta_2) + l1*sin(theta_1)) - g*m1*r1*sin(theta_1)

Eq2 = ddl_dtheta2_dot_dt - dl_dtheta2 - t2
Eq2 = theta_ddot_2*(m2*r2^2 + I2) - t2 + theta_ddot_1*(I2 + (m2*(2*r2^2 + 2*l1*cos(theta_2)*r2))/2) - g*m2*r2*sin(theta_1 + theta_2) + l1*m2*r2*theta_dot_1*sin(theta_2)*(theta_dot_1 + theta_dot_2) - l1*m2*r2*theta_dot_1*theta_dot_2*sin(theta_2)

Where t1 and t2 are torques

State Space Representation

X = [theta_1, theta_2,theta_dot_1, theta_dot_2]
X' = [dX(1), dX(2), dX(3), dX(4)]

Where,

dX(1) = theta_dot_1;
dX(2) = theta_dot_2;
dX(3) = (I2*t1 - I2*t2 + m2*r2^2*t1 - m2*r2^2*t2 + l1*m2^2*r2^3*theta_dot_1^2*sin(theta_2) + l1*m2^2*r2^3*theta_dot_2^2*sin(theta_2) + g*l1*m2^2*r2^2*sin(theta_1) + I2*g*l1*m2*sin(theta_1) + I2*g*m1*r1*sin(theta_1) - l1*m2*r2*t2*cos(theta_2) + 2*l1*m2^2*r2^3*theta_dot_1*theta_dot_2*sin(theta_2) + l1^2*m2^2*r2^2*theta_dot_1^2*cos(theta_2)*sin(theta_2) - g*l1*m2^2*r2^2*sin(theta_1 + theta_2)*cos(theta_2) + I2*l1*m2*r2*theta_dot_1^2*sin(theta_2) + I2*l1*m2*r2*theta_dot_2^2*sin(theta_2) + g*m1*m2*r1*r2^2*sin(theta_1) + 2*I2*l1*m2*r2*theta_dot_1*theta_dot_2*sin(theta_2))/(- l1^2*m2^2*r2^2*cos(theta_2)^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*t1 - I1*t2 - I2*t2 - l1^2*m2*t2 - m1*r1^2*t2 + m2*r2^2*t1 - m2*r2^2*t2 + l1*m2^2*r2^3*theta_dot_1^2*sin(theta_2) + l1^3*m2^2*r2*theta_dot_1^2*sin(theta_2) + l1*m2^2*r2^3*theta_dot_2^2*sin(theta_2) - g*l1^2*m2^2*r2*sin(theta_1 + theta_2) - I1*g*m2*r2*sin(theta_1 + theta_2) + g*l1*m2^2*r2^2*sin(theta_1) + I2*g*l1*m2*sin(theta_1) + I2*g*m1*r1*sin(theta_1) + l1*m2*r2*t1*cos(theta_2) - 2*l1*m2*r2*t2*cos(theta_2) + 2*l1*m2^2*r2^3*theta_dot_1*theta_dot_2*sin(theta_2) + 2*l1^2*m2^2*r2^2*theta_dot_1^2*cos(theta_2)*sin(theta_2) + l1^2*m2^2*r2^2*theta_dot_2^2*cos(theta_2)*sin(theta_2) - g*l1*m2^2*r2^2*sin(theta_1 + theta_2)*cos(theta_2) + g*l1^2*m2^2*r2*cos(theta_2)*sin(theta_1) - g*m1*m2*r1^2*r2*sin(theta_1 + theta_2) + I1*l1*m2*r2*theta_dot_1^2*sin(theta_2) + I2*l1*m2*r2*theta_dot_1^2*sin(theta_2) + I2*l1*m2*r2*theta_dot_2^2*sin(theta_2) + g*m1*m2*r1*r2^2*sin(theta_1) + 2*l1^2*m2^2*r2^2*theta_dot_1*theta_dot_2*cos(theta_2)*sin(theta_2) + l1*m1*m2*r1^2*r2*theta_dot_1^2*sin(theta_2) + 2*I2*l1*m2*r2*theta_dot_1*theta_dot_2*sin(theta_2) + g*l1*m1*m2*r1*r2*cos(theta_2)*sin(theta_1))/(- l1^2*m2^2*r2^2*cos(theta_2)^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(3) and dX(4) are calculated using MATLAB solve function:
sol = solve([eq1==0, eq2==0], [theta_ddot_1, theta_ddot_2]);

Trajectory Plots

Plot for the given initial values can be generated using MATLAB function:
[t, y] = ode45(@ode_planner_2link, [0, 10], [(10*pi)/9, (25*pi)/36, 0, 0]);
plot(t, y);