```
close all
clear all
clc
```

The goal of this script is to derive the dynamic equations

for a 3-link manipulator with only the two first links being

actuated, this means the system has 3 links with mass and

inertia, but only 2 motors with mass, inertia, and gear reductions

```
% Link masses
m_l1 = 1.0;
m_l2 = 1.0;
m_l3 = 1.0;

% Link length
a_1 = 1;
a_2 = 1;
a_3 = 1;

% Motor masses
m_m1 = 0.25;
m_m2 = 0.25;

% Gear reductions
kr_1 = 1;
kr_2 = 1;

% Distance to center of mass
l_1 = a_1/2;
l_2 = a_2/2;
l_3 = a_3/2;

% Link inertia
I_l1 = (1/12)*m_l1*l_1^2;
I_l2 = (1/12)*m_l2*l_2^2;
I_l3 = (1/12)*m_l3*l_3^2;

% Motor ineria
I_m1 = 0.1;
I_m2 = 0.1;

% Enviornment parameters
%g = 9.81; % m/s^2
```

```
syms g real
syms theta_1 real
syms theta_2 real
syms theta_3 real
```

```
syms theta_dot_1 real
syms theta_dot_2 real
syms theta_dot_3 real
syms m_l1 m_l2 m_l3 m_m1 m_m2 real
syms l_1 l_2 l_3 I_l1 I_l2 I_l3 I_m1 I_m2 real
syms a_1 a_2 a_3 real
syms kr_1 kr_2 real
syms g real

%syms m_l1 m_l2 m_l3 m_m1 m_m2 l_1 l_2 l_3 I_l1 I_l2 I_l3 I_m1 I_m2 a_1 a_2 a_3 kr_1 kr

% theta_dot_1 = diff(theta_1,t);
% theta_dot_2 = diff(theta_2,t);
% theta_dot_3 = diff(theta_3,t);

%assume(theta_dot_1,'real');
%assume(theta_dot_2,'real');
%assume(theta_dot_3,'real');

T0_1 = dhTransform(0,0,theta_1,0);
T1_2 = dhTransform(0,a_1,theta_2,0);
T2_3 = dhTransform(0,a_2,theta_3,0);
T3_E = dhTransform(0,a_3,0,0);

T0_E = formula(T0_1*T1_2*T2_3*T3_E);
T0_3 = formula(T0_1*T1_2*T2_3);
T0_2 = formula(T0_1*T1_2);
T0_1
T0_1 = T0_1
R0_1 = T0_1(1:3,1:3);
R0_2 = T0_2(1:3,1:3);
R0_3 = T0_3(1:3,1:3);

% Define Positions along arm
p_l1_s = T0_1*[l_1 0 0 1]'
p_l2_s = T0_2*[l_2 0 0 1]'
p_l3_s = T0_3*[l_3 0 0 1]'

p_m1_s = T0_1*[0 0 0 1]';
p_m2_s = T0_2*[0 0 0 1]';

pl1 = p_l1_s(1:3);
pl2 = p_l2_s(1:3);
pl3 = p_l3_s(1:3);

pm1 = p_m1_s(1:3);
pm2 = p_m2_s(1:3);

% Define Positions to Joints
p0 = [0 0 0]';
p1 = T0_2(1:3,4);
p2 = T0_3(1:3,4);
p3 = T0_E(1:3,4);
```

```
Z0  = [0 0 1]';
Zeros = [0 0 0]';

Jl1_P = [cross(Z0,pl1-p0), Zeros, Zeros];
Jl1_O = [Z0, Zeros, Zeros];
Jl1 = [Jl1_P; Jl1_O]

Jl2_P = [cross(Z0,pl2-p0),cross(Z0,pl2-p1),Zeros];
Jl2_O = [Z0, Z0, Zeros];
Jl2 = [Jl2_P; Jl2_O]

Jl3_P = [cross(Z0,pl3-p0),cross(Z0,pl3-p1),cross(Z0,pl3-p2)];
Jl3_O = [Z0,Z0,Z0];
Jl3 = [Jl3_P; Jl3_O]

Jm1_P = [Zeros, Zeros, Zeros];
Jm1_O = [Z0*kr_1, Zeros, Zeros];
Jm1 = [Jm1_P; Jm1_O]

Jm2_P = [cross(Z0,pm2-p0),Zeros, Zeros];
Jm2_O = [Z0, Z0*kr_2, Zeros];
Jm2 = [Jm2_P; Jm2_O]

q_dot = [theta_dot_1 theta_dot_2 theta_dot_3]';

% Velocities of link center of masses
P_dot_l1 = Jl1_P*q_dot;
P_dot_l2 = Jl2_P*q_dot;
P_dot_l3 = Jl3_P*q_dot;

% Angular Velocities of links
O_dot_l1 = Jl1_O*q_dot;
O_dot_l2 = Jl2_O*q_dot;
O_dot_l3 = Jl3_O*q_dot;

% Velocities of the motors
P_dot_m1 = Jm1_P*q_dot;
P_dot_m2 = Jm2_P*q_dot;

% Angular velocities of the motors
O_dot_m1 = kr_1*theta_dot_1*Z0;
O_dot_m2 = kr_2*theta_dot_2*Z0;

% Kinetic Energy of the links
T_li(1) = 1/2*m_l1*(P_dot_l1')*P_dot_l1+1/2*(O_dot_l1')*R0_1*I_l1*R0_1'*O_dot_l1;
T_li(2) = 1/2*m_l2*(P_dot_l2')*P_dot_l2+1/2*(O_dot_l2')*R0_2*I_l2*R0_2'*O_dot_l2;
T_li(3) = 1/2*m_l3*(P_dot_l3')*P_dot_l3+1/2*(O_dot_l3')*R0_3*I_l3*R0_3'*O_dot_l3;

% Kinetic Energy of the motors
T_mi(1) = 1/2*m_m1*(P_dot_m1')*P_dot_m1+1/2*(O_dot_m1')*I_m1*O_dot_m1;
T_mi(2) = 1/2*m_m2*(P_dot_m2')*P_dot_m2+1/2*(O_dot_m2')*I_m2*O_dot_m2;

Ti(1) = T_li(1)+T_mi(1);
Ti(2) = T_li(2)+T_mi(2);
```

```
Ti(3) = T_li(3);
simplify(Ti');

G = [0 -g 0]';
% Potential Energy of links
Ui(1) = -(m_m1*G'*pm1 + m_l1*G'*pl1);
Ui(2) = -(m_m2*G'*pm2 + m_l2*G'*pl2);
Ui(3) = -(m_l3*G'*pl3);
simplify(Ui');
U = Ui(1) + Ui(2) + Ui(3);
T = Ti(1) + Ti(2) + Ti(3);
```

```
X = {theta_1 theta_dot_1 theta_2 theta_dot_2 theta_3 theta_dot_3}
L=T-U;
syms tau1 tau2 tau3 real
Qe = {tau1 tau2 tau3};
Qi = {0 0 0};
par = {m_l1 m_l2 m_l3 m_m1 m_m2 l_1 l_2 l_3 I_l1 I_l2 I_l3 I_m1 I_m2 a_1 a_2 a_3 kr_1 k
Eqn = EulerLagrange(L,X,Qi,Qe,0,par,'m')
```

```
syms theta_ddot_1 theta_ddot_2 theta_ddot_3 real
tau = [tau1; tau2; tau3];
Eq =[Eqn(2) == theta_ddot_1
     Eqn(4) == theta_ddot_2
     Eqn(6) == theta_ddot_3];
```

```
%[,A] = collect(Eq,{theta_ddot_1 theta_ddot_2 theta_ddot_3})
q_ddot = [theta_ddot_1 theta_ddot_2 theta_ddot_3]';
[W,E] = equationsToMatrix(Eq,[tau1 tau2 tau3])
lhs = simplify(W\E);
rhs = [tau1 tau2 tau3]';

[Mass_matrix,Other] = equationsToMatrix(lhs,[theta_ddot_1 theta_ddot_2 theta_ddot_3])
Mass_matrix
%[C_matrix,Else] = equationsToMatrix(Other,[theta_dot_1 theta_dot_2 theta_dot_3])
N_matrix = simplify(-Other)
N_iso = [N_matrix(1) == 0
         N_matrix(2) == 0
         N_matrix(3) == 0];
%out = equationsToMatrix(N_iso,[theta_dot_1 theta_dot_2 theta_dot_3])

EOM_rough = lhs;
EOM_clean = Mass_matrix*q_ddot + N_matrix ;
isequal(EOM_rough(1),EOM_clean(1))
simplify(EOM_rough,"Steps",100)
simplify(EOM_clean,"Steps",100)
save('Mass_matrix.mat','Mass_matrix')
save('N_term.mat','N_matrix')
```

```
 % Assemble B Matrix
% Shortcut derivation of the mass matrix
```

```
B1 = m_l1*(Jl1_P')*Jl1_P+Jl1_O'*R0_1*I_l1*R0_1'*Jl1_O + m_m1*(Jm1_P')*Jm1_P+Jm1_O'*R0_1
B2 = m_l2*(Jl2_P')*Jl2_P+Jl2_O'*R0_2*I_l2*R0_2'*Jl2_O + m_m2*(Jm2_P')*Jm2_P+Jm2_O'*R0_2
B3 = m_l3*(Jl3_P')*Jl3_P+Jl3_O'*R0_3*I_l3*R0_3'*Jl3_O;
B = B1 + B2 + B3;
```