function [Transfer_Functions] = get_EOM(DOF)

Contents


```
function [Transfer_Functions] = get_EOM(DOF)
clc

close all

OneDOF = 0;
TwoDOF = 0;
ThreeDOF = 0;
verbose = 1;

if(DOF == 1)
    OneDOF = 1;
end
if(DOF == 2)
    TwoDOF = 1;
end
if(DOF == 3)
    ThreeDOF = 1;
end
signpost(verbose,'Start: get_EOM()')
```

Initialise variables

```
signpost(verbose,'Variable init')

%Symbolic Variables
syms g temp a(t)
syms a1 da1 dda1
syms a2 da2 dda2
syms a3 da3 dda3
```

```
syms A1 A2 A3
syms tf1_a1_T1 tf2_a2_T2 tf3_a3_T3


a_list   = [a1,     a2,     a3];
da_list  = [da1,    da2,    da3];
dda_list = [dda1,   dda2,   dda3];


syms l1 l2 l3
syms L1 L2 L3

simpleMode = 0;
if(simpleMode)
    signpost(verbose,'###Assume center of mass is end of manipulator at next joint')
    L1 = l1;
    L2 = l2;
    L3 = l3;
end

syms m1 m2 m3

syms Ixx1 Ixx2 Ixx3
syms Iyy1 Iyy2 Iyy3
syms Izz1 Izz2 Izz3

syms T1 T2 T3

T_list = [T1;T2;T3];

it1 = 3;
it2 = 3;
it3 = 3;

C = [0*temp 0*temp 0*temp; 0*temp 0*temp 0*temp; 0*temp 0*temp 0*temp];



if(ThreeDOF)
    N = 3;
    signpost(verbose,'Three DOF')
    OneDOF = 0;
    TwoDOF = 0;
end


if OneDOF


    if (OneDOF)
```

```
        N = 1;
        signpost(verbose,'One DOF')
    end

    l2 = 0;
    L2 = 0;
    m2 = 0;

    Ixx2 = 0;
    Iyy2 = 0;
    Izz2 = 0;

    T2 = 0;

    TwoDOF = 1;
end

if TwoDOF || OneDOF

    if (TwoDOF) && ~(OneDOF)
        N = 2;
        signpost(verbose,'Two DOF')
    end

    l3 = 0;
    L3 = 0;
    m3 = 0;

    Ixx3 = 0;
    Iyy3 = 0;
    Izz3 = 0;

    T3 = 0;

end
```

Get P

```
%Angles (relative)
alpha = a1;
beta = alpha + a2;
gamma = beta + a3;

p1_0 = [(l1*cos(alpha));                                (l1*sin(alpha));
0];
p2_0 = [(L1*cos(alpha) + l2*cos(beta));                 (L1*sin(alpha) +
l2*sin(beta));                  0];
p3_0 = [(L1*cos(alpha) + L2*cos(beta) + l3*cos(gamma)); (L1*sin(alpha) + L2*sin(beta)
+ l3*sin(gamma));  0];
```

Finding mn_Jvn_JvnT

```
signpost(verbose,'Finding mn_Jvn_JvnT')

% matrix for Jv1
e11 = diff(p1_0(1), a1);
e12 = diff(p1_0(1), a2);
e13 = diff(p1_0(1), a3);

e21 = diff(p1_0(2), a1);
e22 = diff(p1_0(2), a2);
e23 = diff(p1_0(2), a3);

e31 = diff(p1_0(3), a1);
e32 = diff(p1_0(3), a2);
e33 = diff(p1_0(3), a3);

Jv1 = [e11 e12 e13; e21 e22 e23; e31 e32 e33];

Jv1T = transpose(Jv1);

m1_Jv1_Jv1T = simplify(m1*(Jv1T*Jv1));

% matrix for Jv2
e11 = diff(p2_0(1), a1);
e12 = diff(p2_0(1), a2);
e13 = diff(p2_0(1), a3);

e21 = diff(p2_0(2), a1);
e22 = diff(p2_0(2), a2);
e23 = diff(p2_0(2), a3);

e31 = diff(p2_0(3), a1);
e32 = diff(p2_0(3), a2);
e33 = diff(p2_0(3), a3);

Jv2 = [e11 e12 e13; e21 e22 e23; e31 e32 e33];
Jv2T = transpose(Jv2);

m2_Jv2_Jv2T = simplify(m2*(Jv2T*Jv2));

% matrix for Jv3
e11 = diff(p3_0(1), a1);
e12 = diff(p3_0(1), a2);
e13 = diff(p3_0(1), a3);

e21 = diff(p3_0(2), a1);
e22 = diff(p3_0(2), a2);
e23 = diff(p3_0(2), a3);

e31 = diff(p3_0(3), a1);
e32 = diff(p3_0(3), a2);
```

```
e33 = diff(p3_0(3), a3);

Jv3 = [e11 e12 e13; e21 e22 e23; e31 e32 e33];
Jv3T = transpose(Jv3);

m3_Jv3_Jv3T = simplify(m3*(Jv3T*Jv3));
```

Finding Jwn_In_JwnT

```
signpost(verbose,'Finding Jwn_In_JwnT')

Jw1 = [0 0 0; 0 0 0; 1 0 0];
Jw2 = [0 0 0; 0 0 0; 1 1 0];
Jw3 = [0 0 0; 0 0 0; 1 1 1];

Jw1T = transpose(Jw1);
Jw2T = transpose(Jw2);
Jw3T = transpose(Jw3);

I1 = [Ixx1 0 0; 0 Iyy1 0; 0 0 Izz1];
I2 = [Ixx2 0 0; 0 Iyy2 0; 0 0 Izz2];
I3 = [Ixx3 0 0; 0 Iyy3 0; 0 0 Izz3];

Jw1_I1_Jw1T = Jw1T*I1*Jw1;
Jw2_I2_Jw2T = Jw2T*I2*Jw2;
Jw3_I3_Jw3T = Jw3T*I3*Jw3;
```

Finding Matrix M

```
signpost(verbose,'Finding Matrix M')

M = simplify(Jw1_I1_Jw1T + Jw2_I2_Jw2T + Jw3_I3_Jw3T + m1_Jv1_Jv1T + m2_Jv2_Jv2T +
m3_Jv3_Jv3T);
```

Finding Matrix G

```
signpost(verbose,'Finding Matrix G')

g1 = [0; m1*g; 0];
g2 = [0; m2*g; 0];
g3 = [0; m3*g; 0];

Jv1_g1 = -(Jv1T)*(-g1);
Jv2_g2 = -(Jv2T)*(-g2);
Jv3_g3 = -(Jv3T)*(-g3);

G = simplify((Jv1_g1) + (Jv2_g2) + (Jv3_g3));
```

Finding Matrix B and C

```
signpost(verbose,'Finding Matrix B and C')

for i = 1:it1
    for j = 1:it2
        for k = 1:it3

            Mij = M(i,j);
            Mik = M(i,k);
            Mjk = M(j,k);

            a_i = a_list(i);
            a_j = a_list(j);
            a_k = a_list(k);

            dMijk = diff(Mij, a_k);
            dMikj = diff(Mik, a_j);
            dMjki = diff(Mjk, a_i);

            cijk(i,j,k) = simplify(0.5*(dMjki + dMikj - dMijk));

        end
    end
end

for k = 1:it3
    for j = 1:it2
        for i = 1:it1
            C(k,j) = simplify(C(k,j) + (cijk(i,j,k))*da_list(i));

        end
    end
end
```

Finding EOM

```
signpost(verbose,'Finding EOM')

torque = [0*temp 0*temp; 0*temp 0*temp; 0*temp 0*temp];
for i = 1:N
    torque(i,1) = T_list(i);
    torque(i,2) = simplify(G(i));
    for j = 1:N

        torque(i,2) = torque(i,2) + simplify(M(i,j)*dda_list(j));
        torque(i,2) = torque(i,2) + simplify(C(i,j)*da_list(j));

    end

    torque(i) = simplify(torque(i));
end
```

```
EOM = torque
```

Finding Laplace EOM

pull apart

```
Ts = torque;

eq1 = Ts(1,2);

eq2 = Ts(2,2);

eq3 = Ts(3,2);
```

Equations of Theta/Torque 1

```
signpost(verbose,'Equations of Theta/Torque 1')

EQ = eq1;

% Transform 3rd Order Cosine
wrt = cos(a1 + a2 + a3);

tran = str2sym('A1*( (s*cos(a2 + a3) - sin(a2 + a3)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 2nd Order Cosine
wrt = cos(a1 + a2);

tran = str2sym('A1*( (s*cos(a2) - sin(a2)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

wrt = cos(a1 + a3);

tran = str2sym('A1*( (s*cos(a3) - sin(a3)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 2nd Order Sine
wrt = sin(a1 + a2);

tran = str2sym('A1*( (s*sin(a2) - cos(a2)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

wrt = sin(a1 + a3);
```

```
tran = str2sym('A1*( (s*sin(a3) - cos(a3)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 1st Order Cosine
wrt = cos(a1);

tran = str2sym('A1*(s/(s^2 + 1))');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 1st Order Sine
wrt = sin(a1);

tran = str2sym('A1*(1/(s^2 + 1))');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform dda
wrt = dda1;

tran = str2sym('A1*s^2');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform da
wrt = da1;

tran = str2sym('A1*s');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform a
wrt = a1;

tran = str2sym('A1');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

eq1 = EQ;
```

Equations of Theta/Torque 2

```
signpost(verbose,'Equations of Theta/Torque 2')

EQ = eq2;

% Transform 3rd Order Cosine
wrt = cos(a2 + a1 + a3);
```

```
tran = str2sym('A2*( (s*cos(a1 + a3) - sin(a1 + a3)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 2nd Order Cosine
wrt = cos(a2 + a1);

tran = str2sym('A2*( (s*cos(a1) - sin(a1)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

wrt = cos(a2 + a3);

tran = str2sym('A2*( (s*cos(a3) - sin(a3)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 2nd Order Sine
wrt = sin(a2 + a1);

tran = str2sym('A2*( (s*sin(a1) - cos(a1)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

wrt = sin(a2 + a3);

tran = str2sym('A2*( (s*sin(a3) - cos(a3)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 1st Order Cosine
wrt = cos(a2);

tran = str2sym('A2*(s/(s^2 + 1))');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 1st Order Sine
wrt = sin(a2);

tran = str2sym('A2*(1/(s^2 + 1))');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform dda
wrt = dda2;

tran = str2sym('A2*s^2');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);
```

```
% Transform da
wrt = da2;

tran = str2sym('A2*s');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform a
wrt = a2;

tran = str2sym('A2');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

eq2 = EQ;
```

Equations of Theta/Torque 3

```
signpost(verbose,'Equations of Theta/Torque 3')

EQ = eq3;

% Transform 3rd Order Cosine
wrt = cos(a3 + a2 + a1);

tran = str2sym('A3*( (s*cos(a2 + a1) - sin(a2 + a1)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 2nd Order Cosine
wrt = cos(a3 + a2);

tran = str2sym('A3*( (s*cos(a2) - sin(a2)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

wrt = cos(a3 + a1);

tran = str2sym('A3*( (s*cos(a1) - sin(a1)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 2nd Order Sine
wrt = sin(a3 + a2);

tran = str2sym('A3*( (s*sin(a2) - cos(a2)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

wrt = sin(a3 + a1);
```

```
tran = str2sym('A3*( (s*sin(a1) - cos(a1)) / (s^2+1) )');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 1st Order Cosine
wrt = cos(a3);

tran = str2sym('A3*(s/(s^2 + 1))');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform 1st Order Sine
wrt = sin(a3);

tran = str2sym('A3*(1/(s^2 + 1))');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform dda
wrt = dda3;

tran = str2sym('A3*s^2');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform da
wrt = da3;

tran = str2sym('A3*s');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

% Transform a
wrt = a3;

tran = str2sym('A3');
EQ = subs(EQ,wrt,tran);
EQ = simplify(EQ);

eq3 = EQ;
```

Completing Laplace EOM

```
signpost(verbose,'Completing Laplace EOM')
% put back together
Ts(1,2) = eq1;
Ts(2,2) = eq2;
Ts(3,2) = eq3;
Ts = simplify(Ts);
E(1,1) = (Ts(1,1) == Ts(1,2));
E(2,1) = (Ts(2,1) == Ts(2,2));
```

```
E(3,1) = (Ts(3,1) == Ts(3,2));

Laplace_EOM = E
```

Finding Transfer Functions

```
signpost(verbose,'Finding Transfer Functions')

% tf 1
f = E(1,1);
t = T1;
a = A1;

f = isolate(f, a);
f = 1 == rhs(f);
f = isolate(f, t);
f = 1/f;
f = tf1_a1_T1 == rhs(f);

f = isolate(f, tf1_a1_T1);

Solution(1,1) = f;

% tf 2
f = E(2,1);
t = T2;
a = A2;

if (Ts(2,1) ~= 0)
    f = isolate(f, a);
    f = 1 == rhs(f);
    f = isolate(f, t);
    f = 1/f;
    f = tf2_a2_T2 == rhs(f);

    f = isolate(f, tf2_a2_T2);
end

Solution(2,1) = f;

% tf 3
f = E(3,1);
t = T3;
a = A3;

if (Ts(3,1) ~= 0)
    f = isolate(f, a);
    f = 1 == rhs(f);
    f = isolate(f, t);
    f = 1/f;
    f = tf3_a3_T3 == rhs(f);
```

```
    f = isolate(f, tf3_a3_T3);
end

Solution(3,1) = f;

Transfer_Functions = Solution
```

Tidy Up

```
signpost(verbose,'Done: get_EOM()')
end
```

Published with MATLAB® R2017b