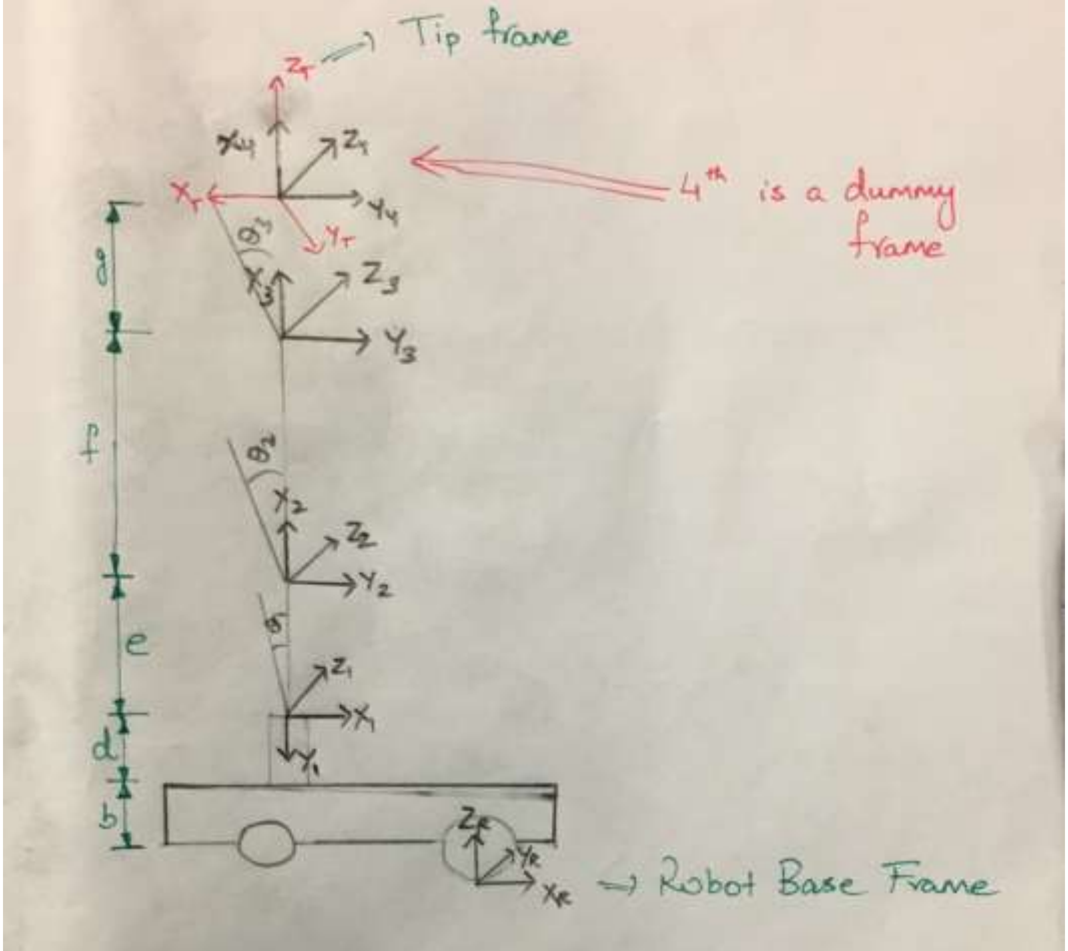
ROBOT DYNAMICS HOMEWORK 3 – ANIMESH NEMA

ANSWER 1



DH parameters

link	θ	d	α	a
R-1	0	b+d	-90	-c
1-2	$\theta_1 - 90$	0	0	e
2-3	θ_2	0	0	f
3-4	θ_3	0	0	g
4-5	-90	0	-90	0

Contents


```
clc; clear all;
syms q1 q2 q3 b c d e f g alp1 beta1 alp2 beta2 alp3 beta3 a wL wr wo ws thet rf ro rs x_o y_o thet_o x_qR y_qR z_qR pi
```

D-H PARAMETERS

```
theta = [0, q1-(pi/2), q2, q3, -(pi/2)];
disp_z =[b+d, 0, 0, 0, 0];
alpha = [-(pi/2), 0, 0, 0, -(pi/2)];
disp_x= [-c, e, f, g, 0];
```

ANSWER 2

```
M = dhparam2matrix(theta, disp_z, alpha, disp_x);
for n = 1:5
    %disp(sprintf('T_%d_%d', n-1 , n))
    vpa(M{n});
end

T_0_5 = M{1}*M{2}*M{3}*M{4}*M{5};
```

THE TRANSFORMATION MATRIX FROM BASE TO TIP FRAME

```
T_0_5 = simplify(T_0_5)
```

```
T_0_5 =

[ -cos(q1 + q2 + q3),  0, sin(q1 + q2 + q3),     f*sin(q1 + q2) - c + e*sin(q1) + g*sin(q1 + q2 + q3)]
[                  0, -1,                 0,                                                     0]
[  sin(q1 + q2 + q3),  0, cos(q1 + q2 + q3), b + d + f*cos(q1 + q2) + e*cos(q1) + g*cos(q1 + q2 + q3)]
[                  0,  0,                 0,                                                     1]
```

ANSWER 3

```
% The position vector for T_0_5
X_0_5 = T_0_5(1:3,4);
```

```
T_0_1 = M{1};
T_0_2 = M{1}*M{2};
T_0_3 = M{1}*M{2}*M{3};

% The rotation matrix
R_0_Z1= T_0_1(1:3,3);
R_0_Z2= T_0_2(1:3,3);
R_0_Z3= T_0_3(1:3,3);

% Differentiating the position with respect to the joint angles
d_X_0_5_1 = diff(X_0_5 , q1);
d_X_0_5_2 = diff(X_0_5 , q2);
d_X_0_5_3 = diff(X_0_5 , q3);
```

JACOBIAN FOR FORWARD VELOCITY KINEMATICS

(x_dot = J(q) * q_dot)

```
J = [d_X_0_5_1 , d_X_0_5_2, d_X_0_5_3 ; R_0_Z1, R_0_Z2, R_0_Z3]

%Reduced Jacobain(Linear y velocity as well as angular x and z are zero)
Reduced_J_matrix= [J(1,:);J(3,:);J(5,:)]
```

```
J =

[   f*cos(q1 + q2) + e*cos(q1) + g*cos(q1 + q2 + q3),   f*cos(q1 + q2) + g*cos(q1 + q2 + q3),  g*cos(q1 + q2 + q3)]
[                                                 0,                                       0,                    0]
[ - f*sin(q1 + q2) - e*sin(q1) - g*sin(q1 + q2 + q3), - f*sin(q1 + q2) - g*sin(q1 + q2 + q3), -g*sin(q1 + q2 + q3)]
[                                                 0,                                       0,                    0]
[                                                 1,                                       1,                    1]
[                                                 0,                                       0,                    0]


Reduced_J_matrix =

[   f*cos(q1 + q2) + e*cos(q1) + g*cos(q1 + q2 + q3),   f*cos(q1 + q2) + g*cos(q1 + q2 + q3),  g*cos(q1 + q2 + q3)]
[ - f*sin(q1 + q2) - e*sin(q1) - g*sin(q1 + q2 + q3), - f*sin(q1 + q2) - g*sin(q1 + q2 + q3), -g*sin(q1 + q2 + q3)]
[                                                 1,                                       1,                    1]
```

ANSWER 4- TRANSPOSE OF JACOBIAN FOR FORCE-TORQUE RELATIONSHIP

Tau = Transpose(J) * Ftip

```
J_T = transpose(J)
```

```
J_T =

[ f*cos(q1 + q2) + e*cos(q1) + g*cos(q1 + q2 + q3), 0, - f*sin(q1 + q2) - e*sin(q1) - g*sin(q1 + q2 + q3), 0, 1, 0]
[              f*cos(q1 + q2) + g*cos(q1 + q2 + q3), 0,              - f*sin(q1 + q2) - g*sin(q1 + q2 + q3), 0, 1, 0]
[                               g*cos(q1 + q2 + q3), 0,                                  -g*sin(q1 + q2 + q3), 0, 1, 0]
```

ANSWER 5-A

Transformation matrix from base to tip frame for the given Configuration

```
T_0_5_ans5a = vpa(simplify(subs(T_0_5, [q1,q2,q3,b,c,d,e,f,g], [(pi/180)*20,(pi/180)*90,(pi/180)*30,.424,.300,.380,.328,.323,.0824])))
```

```
T_0_5_ans5a =

[ 0.766,    0,  0.643, 0.169]
[     0, -1.0,      0,     0]
[ 0.643,    0, -0.766, 0.939]
[     0,    0,      0,   1.0]
```

ANSWER 5-B

Forward Velocity Kinematics for the given configuration

```
q_dot = [(pi/180)*30; (pi/180)*30; (pi/180)*30];
J_ans5b = vpa(simplify(subs(J, [q1,q2,q3,b,c,d,e,f,g], [(pi/180)*20,(pi/180)*90,(pi/180)*30,.424,.300,.380,.328,.323,.0824])));
X_dot = vpa(simplify(J_ans5b*q_dot))
disp('units : m/sec and rad/sec')
```

```
X_dot =

 -0.0535
       0
   -0.46
       0
    1.57
       0

units : m/sec and rad/sec
```

ANSWER 5-C

Calculating Torque through Torque-force relationship.

```
F_tip = [30; 0; 0; 0; 0; 0];
%J_T_ans5b = vpa(simplify(subs(J_T, [q1,q2,q3,b,c,d,e,f,g], [20,90,30,.424,.300,.380,.328,.323,.0824])));
Tau = vpa(simplify(transpose(J_ans5b)*F_tip))
disp('units : N-m ')
```

```
Tau =

  4.04
 -5.21
 -1.89

units : N-m
```

ANSWER 6

INVERSE VELOCITY KINEMATICS

```
x_dot = [ 0; 0; -.1; 0; 0; 0];
Q_dot = vpa(simplify(pinv(J_ans5b)*x_dot))
disp('units :rad/sec ')
```

```
Q_dot =

  0.104
  0.187
 -0.291

units :rad/sec
```

ANSWER 7

The rotation constraint matrix for left fixed wheel

```
FW1_RCM = [sin(alp1 + beta1), -cos(alp1 + beta1) , -(a/2)*cos(beta1)];
% The sliding constraint matrix for left fixed wheel
FW1_SCM = [cos(alp1 + beta1), sin(alp1 + beta1) , -(a/2)*sin(beta1)];
% The rotation constraint matrix for right fixed wheel
FW2_RCM = [sin(alp2 + beta2), -cos(alp2 + beta2) , -(a/2)*cos(beta2)];
% The sliding constraint matrix for right fixed wheel
FW2_SCM = [cos(alp2 + beta2), sin(alp2 + beta2) , -(a/2)*sin(beta2)];
% The rotation constraint matrix for omni wheel
OW3_RCM = [sin(alp3 + beta3), -cos(alp3 + beta3) , -(a/2)*cos(beta3)];
% The sliding constraint matrix for omni wheel
OW3_SCM = [cos(alp3 + beta3), sin(alp3 + beta3) , (a/2)*sin(beta3)];
```

Rotation matrix from robot to world frame

```
R_rob_0 = [cos(thet), sin(thet), 0; -sin(thet), cos(thet), 0; 0, 0, 1];
```

```
R_0_rob = inv(R_rob_0);

c_dot= [x_o ; y_o; thet_o];
```

SIX EQUATIONS FOR THE KINEMATIC CONSTRAINTS

The rolling kinematic constraint equation for left fixed wheel

```
Eq1 = (FW1_RCM) * ((R_rob_0) * (c_dot)) - (rf*wL) == 0
% The sliding kinematic constraint equation for left fixed wheel
Eq2 = (FW1_SCM) * ((R_rob_0) * (c_dot)) == 0
% The rolling kinematic constraint equation for right fixed wheel
Eq3 = (FW2_RCM) * ((R_rob_0) * (c_dot)) - (rf*wr) == 0
% The sliding kinematic constraint equation for right fixed wheel
Eq4 = (FW2_SCM) * ((R_rob_0) * (c_dot)) == 0
% The rolling kinematic constraint equation for omni wheel
Eq5 = (OW3_RCM) * ((R_rob_0) * (c_dot)) - (ro*wo) == 0
% The sliding kinematic constraint equation for omni wheel
Eq6 = (OW3_SCM) * ((R_rob_0) * (c_dot)) -(rs*ws) == 0
```

```
Eq1 =

sin(alp1 + beta1)*(x_o*cos(thet) + y_o*sin(thet)) - cos(alp1 + beta1)*(y_o*cos(thet) - x_o*sin(thet)) - rf*wL - (a*thet_o*cos(beta1))/
2 == 0


Eq2 =

cos(alp1 + beta1)*(x_o*cos(thet) + y_o*sin(thet)) + sin(alp1 + beta1)*(y_o*cos(thet) - x_o*sin(thet)) - (a*thet_o*sin(beta1))/2 == 0


Eq3 =

sin(alp2 + beta2)*(x_o*cos(thet) + y_o*sin(thet)) - cos(alp2 + beta2)*(y_o*cos(thet) - x_o*sin(thet)) - rf*wr - (a*thet_o*cos(beta2))/
2 == 0
```

```
Eq4 =

cos(alp2 + beta2)*(x_o*cos(thet) + y_o*sin(thet)) + sin(alp2 + beta2)*(y_o*cos(thet) - x_o*sin(thet)) - (a*thet_o*sin(beta2))/2 == 0


Eq5 =

sin(alp3 + beta3)*(x_o*cos(thet) + y_o*sin(thet)) - cos(alp3 + beta3)*(y_o*cos(thet) - x_o*sin(thet)) - ro*wo - (a*thet_o*cos(beta3))/
2 == 0


Eq6 =

cos(alp3 + beta3)*(x_o*cos(thet) + y_o*sin(thet)) - rs*ws + sin(alp3 + beta3)*(y_o*cos(thet) - x_o*sin(thet)) + (a*thet_o*sin(beta3))/
2 == 0
```

ANSWER 8

```
% Combined Rolling Constraint Matrix
J_M = [ FW1_RCM ; FW2_RCM ; OW3_RCM]
% Combined Sliding Constraint Matrix
C_M = [ FW1_SCM ; FW2_SCM ; OW3_SCM]

r_J2_M = [ rf*wL; rf*wr ; ro*wo];
r_C2_M = [0;0;rs*ws];

JC = [J_M ; C_M]

% THE COMBINED EQUATION FOR THE KINEMATIC CONSTRAINTS
Equation = (JC) * ((R_rob_0) * (c_dot)) - ([r_J2_M ; r_C2_M]) == 0

% (IGNORING OMNI WHEEL CONSTRAINTS AND ONE OF THE SLIDIING CONSTRAINTS OF FIXED WHEEL)
Reduced_J_M = [ FW1_RCM ; FW2_RCM];
Reduced_C_M = [FW1_SCM];
Reduced_JC = [ Reduced_J_M ; Reduced_C_M];
reduced_r_J2_M = [ rf*wL; rf*wr];
reduced_r_C2_M = [0];

% THE REDUCED EQUATION
```

```
Reduced_Equation = (Reduced_JC) * ((R_rob_0) * (c_dot)) - ([rf*wL; rf*wr; 0]) == 0
```

```
J_M =

[ sin(alp1 + beta1), -cos(alp1 + beta1), -(a*cos(beta1))/2]
[ sin(alp2 + beta2), -cos(alp2 + beta2), -(a*cos(beta2))/2]
[ sin(alp3 + beta3), -cos(alp3 + beta3), -(a*cos(beta3))/2]


C_M =

[ cos(alp1 + beta1), sin(alp1 + beta1), -(a*sin(beta1))/2]
[ cos(alp2 + beta2), sin(alp2 + beta2), -(a*sin(beta2))/2]
[ cos(alp3 + beta3), sin(alp3 + beta3),  (a*sin(beta3))/2]


JC =

[ sin(alp1 + beta1), -cos(alp1 + beta1), -(a*cos(beta1))/2]
[ sin(alp2 + beta2), -cos(alp2 + beta2), -(a*cos(beta2))/2]
[ sin(alp3 + beta3), -cos(alp3 + beta3), -(a*cos(beta3))/2]
[ cos(alp1 + beta1),  sin(alp1 + beta1), -(a*sin(beta1))/2]
[ cos(alp2 + beta2),  sin(alp2 + beta2), -(a*sin(beta2))/2]
[ cos(alp3 + beta3),  sin(alp3 + beta3),  (a*sin(beta3))/2]


Equation =

 sin(alp1 + beta1)*(x_o*cos(thet) + y_o*sin(thet)) - cos(alp1 + beta1)*(y_o*cos(thet) - x_o*sin(thet)) - rf*wL - (a*thet_o*cos(beta1))
/2 == 0
 sin(alp2 + beta2)*(x_o*cos(thet) + y_o*sin(thet)) - cos(alp2 + beta2)*(y_o*cos(thet) - x_o*sin(thet)) - rf*wr - (a*thet_o*cos(beta2))
/2 == 0
 sin(alp3 + beta3)*(x_o*cos(thet) + y_o*sin(thet)) - cos(alp3 + beta3)*(y_o*cos(thet) - x_o*sin(thet)) - ro*wo - (a*thet_o*cos(beta3))
/2 == 0
        cos(alp1 + beta1)*(x_o*cos(thet) + y_o*sin(thet)) + sin(alp1 + beta1)*(y_o*cos(thet) - x_o*sin(thet)) - (a*thet_o*sin(beta1))
/2 == 0
        cos(alp2 + beta2)*(x_o*cos(thet) + y_o*sin(thet)) + sin(alp2 + beta2)*(y_o*cos(thet) - x_o*sin(thet)) - (a*thet_o*sin(beta2))
/2 == 0
 cos(alp3 + beta3)*(x_o*cos(thet) + y_o*sin(thet)) - rs*ws + sin(alp3 + beta3)*(y_o*cos(thet) - x_o*sin(thet)) + (a*thet_o*sin(beta3))
```

```
/2 == 0


Reduced_Equation =

 sin(alp1 + beta1)*(x_o*cos(thet) + y_o*sin(thet)) - cos(alp1 + beta1)*(y_o*cos(thet) - x_o*sin(thet)) - rf*wL - (a*thet_o*cos(beta1))
/2 == 0
 sin(alp2 + beta2)*(x_o*cos(thet) + y_o*sin(thet)) - cos(alp2 + beta2)*(y_o*cos(thet) - x_o*sin(thet)) - rf*wr - (a*thet_o*cos(beta2))
/2 == 0
         cos(alp1 + beta1)*(x_o*cos(thet) + y_o*sin(thet)) + sin(alp1 + beta1)*(y_o*cos(thet) - x_o*sin(thet)) - (a*thet_o*sin(beta1))
/2 == 0
```

ANSWER 9 (VELOCITY KINEMATICS OF MOBILE ROBOT BASE)

```
c_dot = simplify((R_0_rob) * (inv(Reduced_JC)) * ([rf*wL; rf*wr; 0]))
disp('units : m/sec and rad/sec ')
```

```
c_dot =

 -(rf*(wL*sin(alp1 + beta1 + beta2 + thet) - 2*wr*sin(alp1 + 2*beta1 + thet) + wL*sin(alp2 + beta1 + beta2 + thet) + wL*sin(alp1 + bet
a1 - beta2 + thet) - wL*sin(alp2 - beta1 + beta2 + thet)))/(2*(cos(alp1 - alp2 + 2*beta1 - beta2) - cos(beta2)))
  (rf*(wL*cos(alp1 + beta1 + beta2 + thet) - 2*wr*cos(alp1 + 2*beta1 + thet) + wL*cos(alp2 + beta1 + beta2 + thet) + wL*cos(alp1 + bet
a1 - beta2 + thet) - wL*cos(alp2 - beta1 + beta2 + thet)))/(2*(cos(alp1 - alp2 + 2*beta1 - beta2) - cos(beta2)))

         (2*rf*(wr - wL*cos(alp1 - alp2 + beta1 - beta2)))/(a*(cos(alp1 - alp2 + 2*beta1 - beta2) - cos(beta2)))

units : m/sec and rad/sec
```

ANSWER 10 (Numeric Velocity Kinematics for Mobile Robot Base)

```
c_dot_ans10 = vpa(simplify(subs(c_dot, [alp1,beta1,alp2,beta2,a,wL,wr,thet,rf], [(pi/180)*90,0,-(pi/180)*90,pi,.507,pi,2*pi,(pi/180)*3
0,0.143])))
disp('units : m/sec and rad/sec ')
```

```
c_dot_ans10 =

 0.584
 0.337
 0.886

units : m/sec and rad/sec
```

ANSWER 11

The tranformation matrix from world to robot frame

```
T_W_R = [ cos(thet), -sin(thet), 0, 2.5 ; sin(thet), cos(thet), 0, 1.5 ; 0, 0, 1, 0 ; 0, 0, 0, 1];
T_W_R_numeric = vpa(simplify(subs(T_W_R, [thet], [(pi/180)*30])));
% The transformation matrix from World to tip frame (T_W_R * T_R_T)
T_W_T = vpa(simplify((T_W_R) * (T_0_5)))
```

```
T_W_T =

[ -1.0*cos(q1 + q2 + q3)*cos(thet),      sin(thet), sin(q1 + q2 + q3)*cos(thet), cos(thet)*(f*sin(q1 + q2) - 1.0*c + e*sin(q1) + g*sin
(q1 + q2 + q3)) + 2.5]
[ -1.0*cos(q1 + q2 + q3)*sin(thet), -1.0*cos(thet), sin(q1 + q2 + q3)*sin(thet), sin(thet)*(f*sin(q1 + q2) - 1.0*c + e*sin(q1) + g*sin
(q1 + q2 + q3)) + 1.5]
[                sin(q1 + q2 + q3),              0,           cos(q1 + q2 + q3),                   b + d + f*cos(q1 + q2) + e*cos(q1)
+ g*cos(q1 + q2 + q3)]
[                                0,              0,                           0,
                 1.0]
```

ANSWER 12 (Numeric answer for Transformation from world to tip frame).

```
T_W_T_ans12 = vpa(simplify((T_W_R_numeric) * (T_0_5_ans5a)))
T_W_T_tip_Position = [T_W_T_ans12(1:3,4)]
disp('units: m')
```

```
rot_angle = converttoEuler(T_W_T_ans12)
disp('units: degree')
```

```
T_W_T_ans12 =

[ 0.663,    0.5,  0.557,  2.65]
[ 0.383, -0.866,  0.321,  1.58]
[ 0.643,      0, -0.766, 0.939]
[     0,      0,      0,   1.0]


T_W_T_tip_Position =

  2.65
  1.58
 0.939

units: m

rot_angle =

[ 565.0/pi, -126.0/pi, 94.2/pi]

units: degree
```

ANSWER 13 (Combined Velocity Kinematics)

```
R_special = [cos(thet),-sin(thet),0,0,0,0 ;sin(thet),cos(thet),0,0,0,0 ;0,0,1,0,0,0; 0,0,0,cos(thet),-sin(thet),0; 0,0,0,sin(thet),cos
(thet),0; 0,0,0,0,0,1];
q_dot_R = [x_qR; y_qR; z_qR; wL; wr];
new_col = [(rf/2);0;0;0;0;(-rf/a)];
new_col2 = [(rf/2);0;0;0;0;(rf/a)];
J_special = [J , new_col , new_col2];
R_J_special = (R_special * J_special);

X_dot_R = vpa(simplify((R_J_special) * q_dot_R ))
disp('units : m/sec and rad/sec ')
```

```
X_dot_R =

 0.5*cos(thet)*(rf*wL + rf*wr + 2.0*e*x_qR*cos(q1) + 2.0*g*x_qR*cos(q1 + q2 + q3) + 2.0*g*y_qR*cos(q1 + q2 + q3) + 2.0*g*z_qR*cos(q1 +
 q2 + q3) + 2.0*f*x_qR*cos(q1 + q2) + 2.0*f*y_qR*cos(q1 + q2))
 0.5*sin(thet)*(rf*wL + rf*wr + 2.0*e*x_qR*cos(q1) + 2.0*g*x_qR*cos(q1 + q2 + q3) + 2.0*g*y_qR*cos(q1 + q2 + q3) + 2.0*g*z_qR*cos(q1 +
 q2 + q3) + 2.0*f*x_qR*cos(q1 + q2) + 2.0*f*y_qR*cos(q1 + q2))
                                                     - 1.0*x_qR*(f*sin(q1 + q2) + e*sin(q1) + g*sin(q1 + q2 + q3)) - 1.0*y_qR*(f*sin(
q1 + q2) + g*sin(q1 + q2 + q3)) - 1.0*g*z_qR*sin(q1 + q2 + q3)

                           -1.0*sin(thet)*(x_qR + y_qR + z_qR)

                                cos(thet)*(x_qR + y_qR + z_qR)

                                     -(1.0*rf*(wL - 1.0*wr))/a

units : m/sec and rad/sec
```

ANSWER 14 (FORCE PROPAGATION)

```
F_tip_R = [30*cos((pi/180)*30);30*sin((pi/180)*30);0;0;0;0];
J_special_R = vpa(simplify(subs(R_J_special, [q1,q2,q3,b,c,d,e,f,g,wL,wr,rf,a,thet], [(pi/180)*20,(pi/180)*90,(pi/180)*30,.424,.300,.3
80,.328,.323,.0824,pi,2*pi,.143,.507,(pi/180)*30])));

Tau_R = vpa(simplify( transpose(J_special_R) * F_tip_R))
disp('units : N-m ')
Tau_Wheels = [Tau_R(4,1); Tau_R(5,1)]
disp('units : N-m ')

% A function for calculating the Tranformations for answer 3
function M = dhparam2matrix(theta, disp_z, alpha, disp_x)
old = digits(3);
M = cell(5,1);
for i = 1:5
    M{i} = [cos(theta(i)) (-sin(theta(i))*cos(alpha(i))) (sin(theta(i))*sin(alpha(i))) (disp_x(i)*cos(theta(i)));
       sin(theta(i)) cos(theta(i))*cos(alpha(i)) (-cos(theta(i))*sin(alpha(i))) disp_x(i)*sin(theta(i));
       0 (sin(alpha(i))) cos(alpha(i)) disp_z(i);
       0 0 0 1];
end
end
```

```
function rot_angle = converttoEuler(Tfinal)
% Convert rotation matrix to Euler angles
singularity_check = sqrt(Tfinal(1,1)^2 + Tfinal(2,1)^2);
if singularity_check < 10^-6
    PHI = atan2(Tfinal(2,3), Tfinal(2,2)); % corresponds to RX in Robot Studio
    THETA = atan2(-Tfinal(3,1), sqrt(Tfinal(1,1)^2+Tfinal(2,1)^2)); % corresponds to RY in Robot Studio
    PSI = 0; % corresponds to RZ in Robot Studio
else
    THETA = atan2(-Tfinal(3,1),sqrt(Tfinal(1,1)^2+Tfinal(2,1)^2)); % corresponds to RY in Robot Studio
    PHI = atan2(Tfinal(3,2),Tfinal(3,3)); % corresponds to RX in Robot Studio
    PSI = atan2(Tfinal(2,1),Tfinal(1,1)); % corresponds to RZ in Robot Studio
end

rot_angle = [PHI,THETA,PSI]*180/pi;
end
```

```
Tau_R =

  4.04
 -5.21
 -1.89
  2.14
  2.14

units : N-m

Tau_Wheels =

 2.14
 2.14

units : N-m
```