A PD controller with gravity compensation was used for this problem. The block diagram of the controller is shown in Figure 1. The x and y desired positions are converted to joint angles using one of the inverse kinematic solutions. These new desired joint angles are then subtracted from the actual joint angles to get a joint error signal. This error signal is passed through a PD controller. The gains of the controller were Kp = 15000 and Kv = 3000 and were tuned using Ziegler-Nichols. After the signal passes through the PD controller, it is summed with the output of the gravity compensator to give the new joint torques. Each path used a 4th order velocity trajectory profile applied to it where $\Delta = 0.1$ sec.

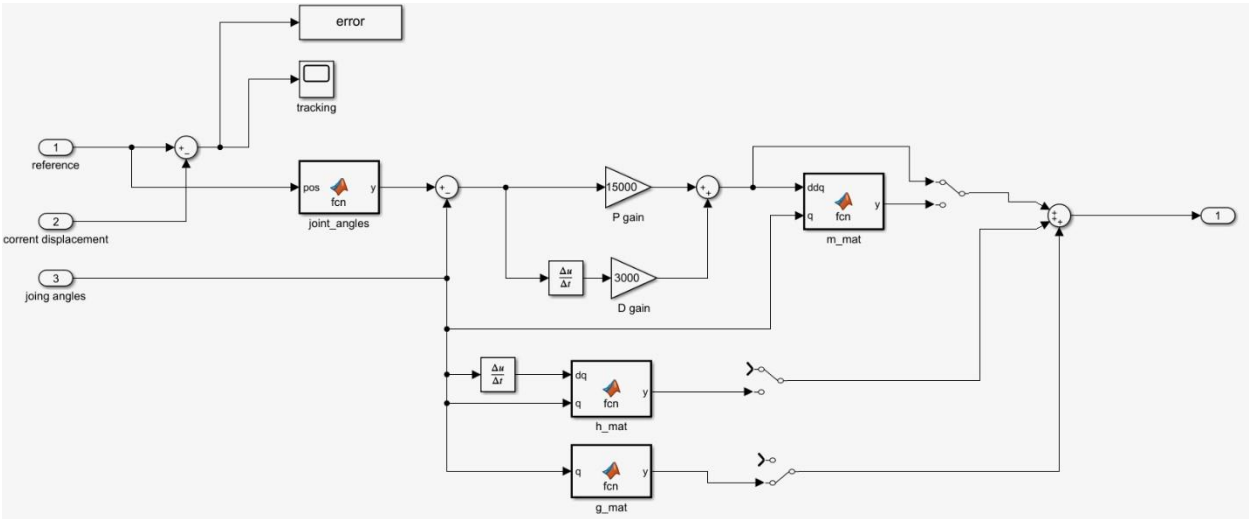

Figure 1. PD Controller with Gravity Compensation

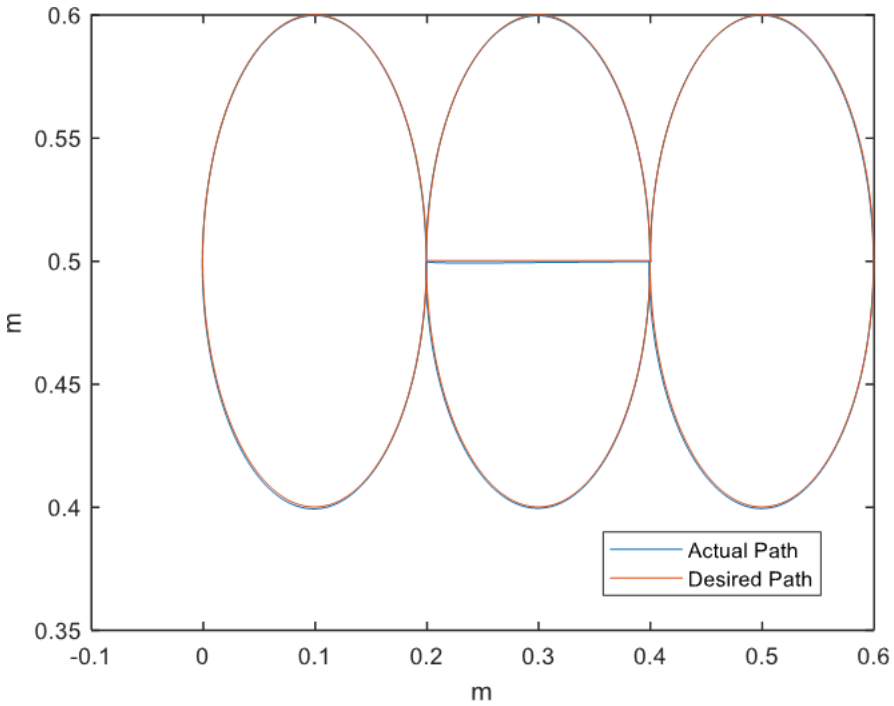

Figure 2. Actual vs. Desired Circles Path

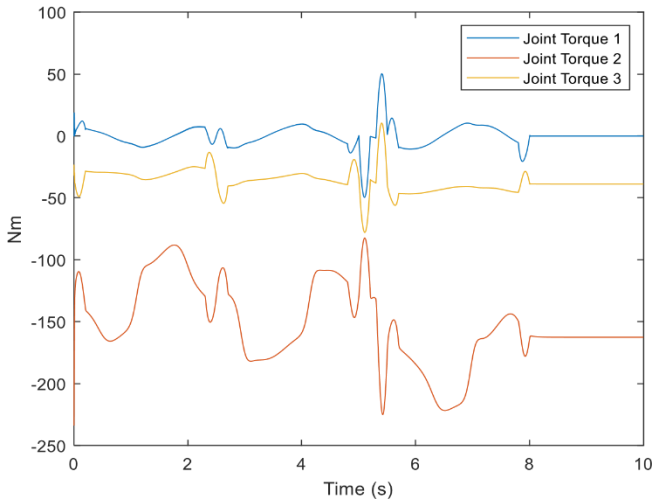

Figure 3. Joint Torques for Circles Path

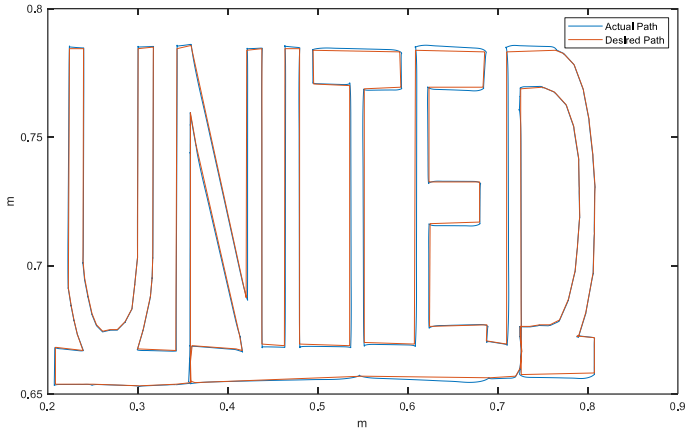

Figure 4. Desired vs. Actual Word Tracing

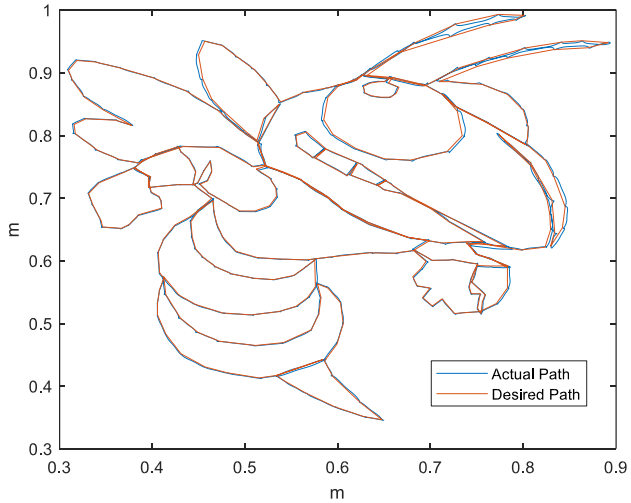

Figure 5. Desired vs. Actual Custom Path

Matlab Code

```
%%
% paths
%external force and moment at the tip (world frame)
external_force=[0;0;-1]; external_moment=[0;1;0];
  tau1=0;
tau2=0;
tau3=0;
la=1;
lb=0.2;
lc=1;
%circle tf
= 10 dt =
0.05; t1 =
2.5;
R = 0.1; %radius in circle
% First Circle
[s, ta] = polynomialTrajectory(0, 1, 0, 0, 0, t1)

xa = R.*cos(2.*pi.*s)+.1;
ya = R.*sin(2.*pi.*s);
%initial angles
[theta1 theta2 theta3] = transposeTest(xa(1), ya(1) + .5, 0)
%path to second circle no path
t2 = t1 + 0; tb = [t1 t2];
%second circle t3 = t2 + t1;
tc = t2 + ta; xb = -xa + 2.*
R.*2; yb = ya;
%  path to third circle  t4
= t3 + .5;
[s, td] = polynomialTrajectory(0, 1, 0, 0, t3, t4)
xc = xb(end) + (xb(1)).*s; yc =
zeros(1,length(td)); % third circle te = t4+ta; xd
= xb + R.*2; yd = yb;   figure x = [xa xb xc xd];
y = [ya yb yc yd] + .5; t = [ta tc td te]; plot(x,
y)
    mask = (diff(t) <=
0); x(mask) = [];
y(mask) = []; t(mask)
= [];
xedyedzedsim = [t' x' y' zeros(length(t),1)];
%% la=1;
lb=0.2;
lc=1;
%external force and moment at the tip (world frame)
external_force=[0;0;-1]; external_moment=[0;1;0];

%joint torque setting
tau1=0; tau2=0;
tau3=0;

tf=10; % simulation end time

N=119; %number of points United
N = 323; %number of points for Buzz
  % Buzz xed = 1-
buzzP(:,1) ./2000; yed = 1-
buzzP(:,2) ./2000; xed =
[xed; xed(1)] - .1; yed =
[yed; yed(1)];

% xed = unitedP(:,1) ./1600;
% yed = .8 - unitedP(:,2) ./1600;
% xed = [xed; xed(1)] + .2;
% yed = [yed; yed(1)];

[theta1 theta2 theta3] = transposeTest(xed(1), yed(1), 0);
```

```
tt=tf/N:tf/N:tf;
[x, y, t] = pathToTrajectory(xed, yed, tt);
%end point data for simscape
  mask = (diff(t) <=
0); x(mask) = [];
y(mask) = []; t(mask)
= [];
xedyedzedsim=[t' x' y' zeros(length(t),1)]; plot(xed,yed);

%% %PATH2 figure
imshow('UNITED.png')
h_poly = drawpolyline;
h_poly.Position


%% % calculations x_a =
actual.Data(:,1); y_a =
actual.Data(:,2); z_a =
actual.Data(:,3); x_t =
trajectory.Data(:,1); y_t =
trajectory.Data(:,2); z_t =
trajectory.Data(:,3);

tau1 = Tau_A.Data(:,1);
tau2 = Tau_A.Data(:,2);
tau3 = Tau_A.Data(:,3);
tt = Tau_A.Time; figure
plot(x_a, y_a); hold on
plot(x_t, y_t);
legend('Actual Path', 'Desired Path');
xlabel('m'); ylabel('m'); figure
plot(tt, tau1); hold on plot(tt,
tau2); plot(tt, tau3);
legend('Joint Torque 1', 'Joint Torque 2', 'Joint Torque 3');
xlabel('Time (s)'); ylabel('Nm');


x_e = error.Data((3:end),1);
y_e = error.Data((3:end),2); z_e
= error.Data((3:end),3);
  figure
t = error.Time((3:end));
plot(t', x_e); hold on
plot(t', y_e); plot(t',
z_e);
legend('X error', 'Y error', 'Z error');
xlabel('Time (s)'); ylabel('m');

errorX_max = max(abs(x_e())) .* 100
errorY_max = max(abs(y_e)) .* 100 errorZ_max
= max(abs(z_e)) .* 100

errorX_rmse = sqrt(mean(x_e.^2)) .* 100 errorY_rmse
= sqrt(mean(y_e.^2)) .* 100 errorZ_rmse =
sqrt(mean(z_e.^2)) .* 100

function [theta1, theta2, theta3] = transposeTest(rx,ry,rz)
la=1; lb=0.2; lc=1;
k = ((rx.^2 + ry.^2 + rz.^2 - lb.^2 + la.^2 + lc.^2).^2 - 2.*((rx.^2 + ry.^2 + rz.^2 - lb.^2).^2
+ la.^4 + lc.^4)).^0.5;
    theta1P = atan2(-rx, ry) + atan2(sqrt(rx.^2 +ry.^2 - lb.^2), lb);
theta1N = atan2(-rx, ry) - atan2(sqrt(rx.^2 +ry.^2 - lb.^2), lb);
    theta2P = atan2(-rz, sqrt(rx.^2 +ry.^2- lb.^2)) + atan2(k, rx.^2+ry.^2 + rz.^2 - lb.^2 +
la.^2 - lc.^2);
    theta2N = atan2(-rz, sqrt(rx.^2 +ry.^2- lb.^2)) - atan2(k, rx.^2+ry.^2 + rz.^2 - lb.^2 +
la.^2 - lc.^2);
```

```
    theta3P = atan2(k, rx.^2 + ry.^2 + rz.^2 - lb.^2 - la.^2 - lc.^2);
theta3N = -theta3P;

    theta3 = theta3P
theta1 = theta1P
    theta21 = atan2(-rz.*(lc.*cos(theta3) + la) - (cos(theta1).*rx +
sin(theta1).*ry).*lc.*sin(theta3), (cos(theta1).*rx + sin(theta1).*ry).*(lc.*cos(theta3) + la) -
rz.*lc.*sin(theta3));

    theta3 = theta3N
theta1 = theta1P
    theta22 = atan2(-rz.*(lc.*cos(theta3) + la) - (cos(theta1).*rx +
sin(theta1).*ry).*lc.*sin(theta3), (cos(theta1).*rx + sin(theta1).*ry).*(lc.*cos(theta3) + la) -
rz.*lc.*sin(theta3));

    theta3 = theta3N
theta1 = theta1N
    theta23 = atan2(-rz.*(lc.*cos(theta3) + la) - (cos(theta1).*rx +
sin(theta1).*ry).*lc.*sin(theta3), (cos(theta1).*rx + sin(theta1).*ry).*(lc.*cos(theta3) + la) -
rz.*lc.*sin(theta3));

    theta3 = theta3P
theta1 = theta1N
    theta24 = atan2(-rz.*(lc.*cos(theta3) + la) - (cos(theta1).*rx +
sin(theta1).*ry).*lc.*sin(theta3), (cos(theta1).*rx + sin(theta1).*ry).*(lc.*cos(theta3) + la) -
rz.*lc.*sin(theta3));

    %theta 2 based on the 4 combos of theta1 and theta3

    %pose 1 correct
    p1 = [theta1P theta22 theta3N];

    %pose 2 correct
    p2 = [theta1P theta21 theta3P];
      theta1 =
p2(1);     theta2 =
p2(2);     theta3 =
p2(3);


function [x, y, t] = pathToTrajectory(xd, yd, td)
  x =
[]; y =
[]; t =
[];
for i = 1:length(xd)-1
    [s, ts] = polynomialTrajectory(0, 1, 0, 0, td(i), td(i+1));
x = [x (xd(i) + (xd(i+1) - xd(i)).*s)];     y = [y (yd(i) +
(yd(i+1) - yd(i)).*s)];     t = [t ts];      end   end


function [s, t] = polynomialTrajectory(x0, xf, v0, vf, t0, tf)
    %note tf is assuming t0 = 0
    %s is a time scaling function from 0 to 1
ta = tf - t0;     dr = 5000     t =
linspace(0,ta,dr);

    dd =.005; %.005 for buzz, .1 for circles
vc = (xf - x0)./(ta - 2.*dd);
        tf1 =
2*dd
    t1 = linspace(0,tf1,dr);

      x01 = x0;
xf1 = vc.*dd;
v01 = v0;     vf1
= vc;     a0 =
```

```
x0;     a1 = v0;
a2 = 0;
    a3 = (1./(2.*tf1.^3)).*(20.*xf1 - 20.*x01 - (8.*vf1 + 12.*v01).*tf1);
a4 = (1./(2.*tf1.^4)).*(30.*x01 - 30.*xf1 + (14.*vf1 + 16.*v01).*tf1);
s1 = a0 + a1.*t1 + a2.*t1.^2 + a3.*t1.^3 + a4.*t1.^4;

    t2 = linspace(2*dd,(ta-2*dd),dr);

    s2 = xf1 + vc.*(t2-2*dd);

    t3 = linspace(t2(end),ta,dr)
x01 = 1 - vc.*dd;     xf1 = 1;
v01 = vc;     vf1 = vf;     a0 =
x01;     a1 = v01;     a2 = 0;
    a3 = (1./(2.*tf1.^3)).*(20.*xf1 - 20.*x01 - (8.*vf1 + 12.*v01).*tf1);
a4 = (1./(2.*tf1.^4)).*(30.*x01 - 30.*xf1 + (14.*vf1 + 16.*v01).*tf1);
s3 = a0 + a1.*t1 + a2.*t1.^2 + a3.*t1.^3 + a4.*t1.^4;
         s = [s1 s2
s3];     t = [t1 t2
t3];     t = t + t0;
end function y =
fcn(pos)     la=1;
lb=0.2;     lc=1;
    rx = pos(1);
ry = pos(2);     rz
= 0;
    k = ((rx.^2 + ry.^2 + rz.^2 - lb.^2 + la.^2 + lc.^2).^2 - 2.*((rx.^2 + ry.^2 + rz.^2 -
lb.^2).^2 + la.^4 + lc.^4)).^0.5;

    theta1P = atan2(-rx, ry) + atan2(sqrt(rx.^2 +ry.^2 - lb.^2), lb);
theta1N = atan2(-rx, ry) - atan2(sqrt(rx.^2 +ry.^2 - lb.^2), lb);

    theta2P = atan2(-rz, sqrt(rx.^2 +ry.^2- lb.^2)) + atan2(k, rx.^2+ry.^2 + rz.^2 - lb.^2 +
la.^2 - lc.^2);
    theta2N = atan2(-rz, sqrt(rx.^2 +ry.^2- lb.^2)) - atan2(k, rx.^2+ry.^2 + rz.^2 - lb.^2 +
la.^2 - lc.^2);
    theta3P = atan2(k, rx.^2 + ry.^2 + rz.^2 - lb.^2 - la.^2 - lc.^2);
theta3N = -theta3P;

    theta3 = theta3P
theta1 = theta1P
    theta21 = atan2(-rz.*(lc.*cos(theta3) + la) - (cos(theta1).*rx +
sin(theta1).*ry).*lc.*sin(theta3), (cos(theta1).*rx + sin(theta1).*ry).*(lc.*cos(theta3) + la) -
rz.*lc.*sin(theta3));

    theta3 = theta3N
theta1 = theta1P
    theta22 = atan2(-rz.*(lc.*cos(theta3) + la) - (cos(theta1).*rx +
sin(theta1).*ry).*lc.*sin(theta3), (cos(theta1).*rx + sin(theta1).*ry).*(lc.*cos(theta3) + la) -
rz.*lc.*sin(theta3));

    theta3 = theta3N
theta1 = theta1N
    theta23 = atan2(-rz.*(lc.*cos(theta3) + la) - (cos(theta1).*rx +
sin(theta1).*ry).*lc.*sin(theta3), (cos(theta1).*rx + sin(theta1).*ry).*(lc.*cos(theta3) + la) -
rz.*lc.*sin(theta3));

    theta3 = theta3P
theta1 = theta1N
    theta24 = atan2(-rz.*(lc.*cos(theta3) + la) - (cos(theta1).*rx +
sin(theta1).*ry).*lc.*sin(theta3), (cos(theta1).*rx + sin(theta1).*ry).*(lc.*cos(theta3) + la) -
rz.*lc.*sin(theta3));

    p2 = [theta1P theta21 theta3P];
```

```
    theta1 = p2(1);
theta2 = p2(2);     theta3
= p2(3);
    y = [theta1; theta2; theta3];

function y = fcn(q) Ta =
q(1) .* (180 ./ pi);
Tb = q(2) .* (180 ./ pi);
Tc = q(3) .* (180 ./ pi);

G = [0;
    16.91482.*cos(Tb) + 18.75.*(cosd(Tb) + 0.5.*cosd(Tb + Tc));
9.375.*cosd(Tb+Tc)]; y = -9.81.*G;
```