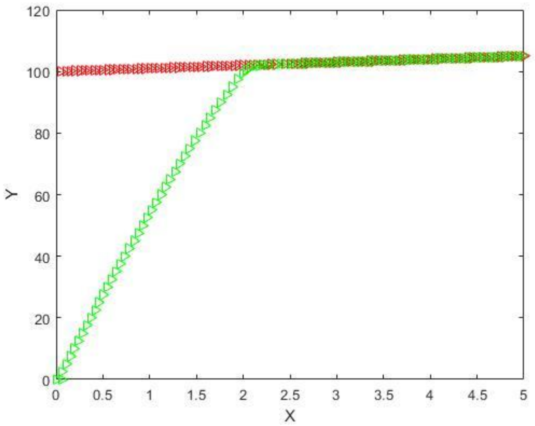
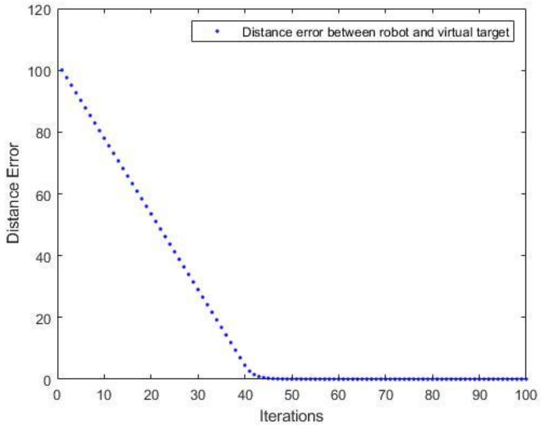
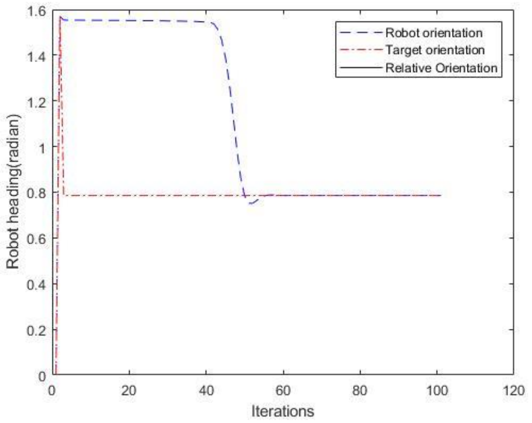
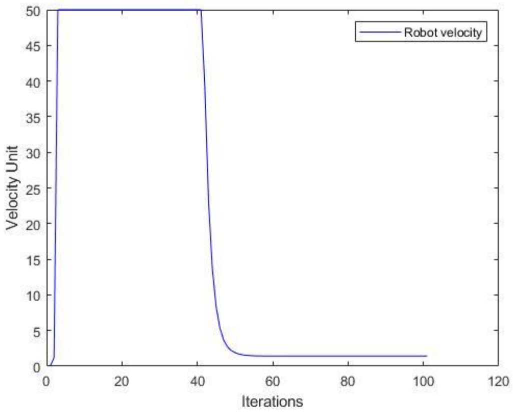
Linear Trajectory

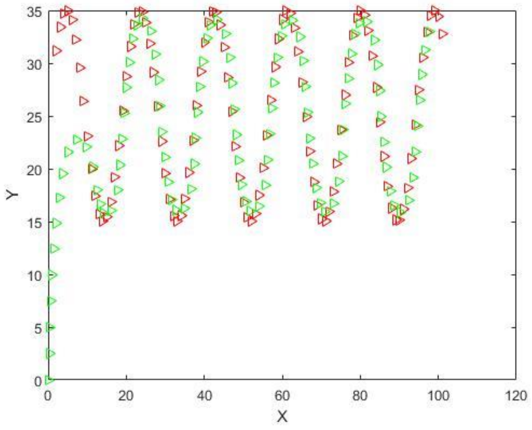


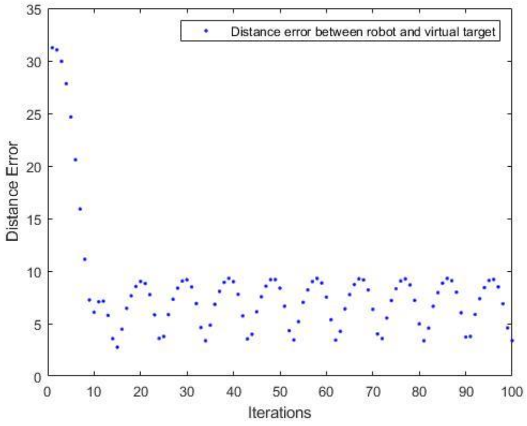


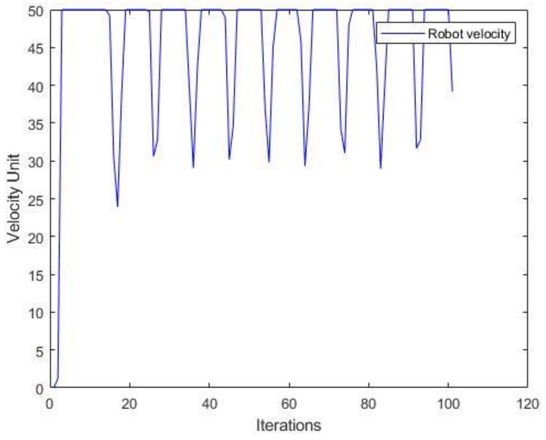


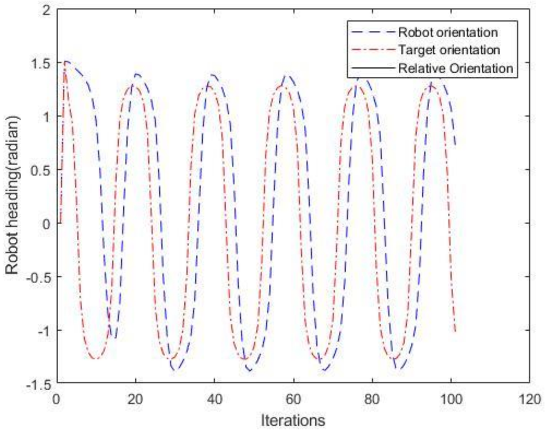


Sine Wave Trajectory

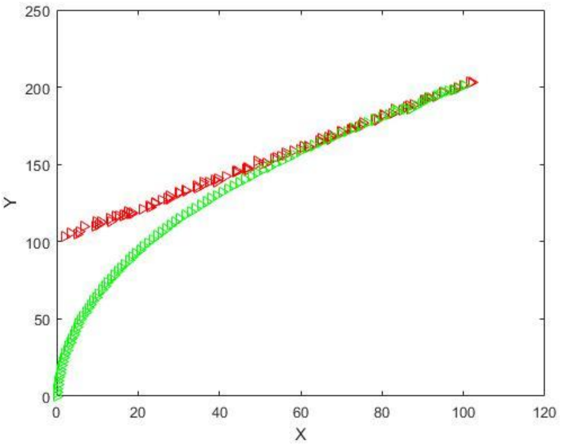


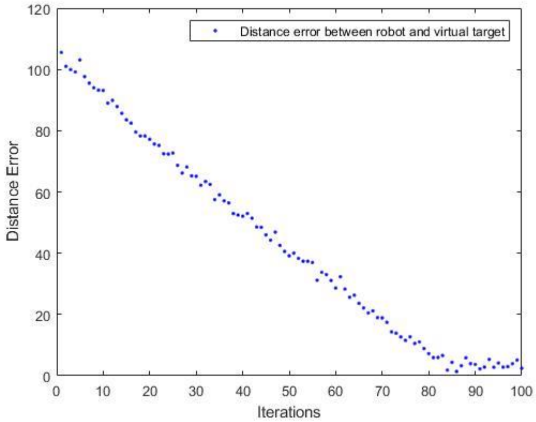


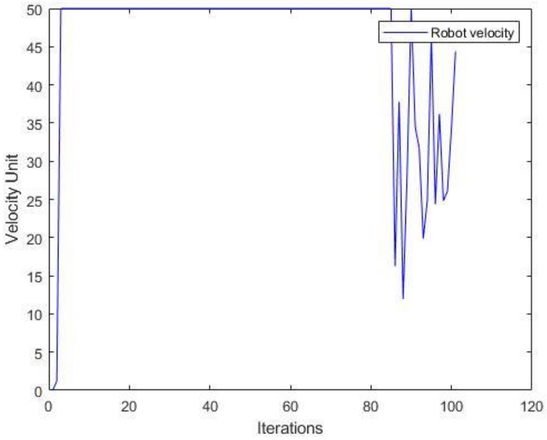


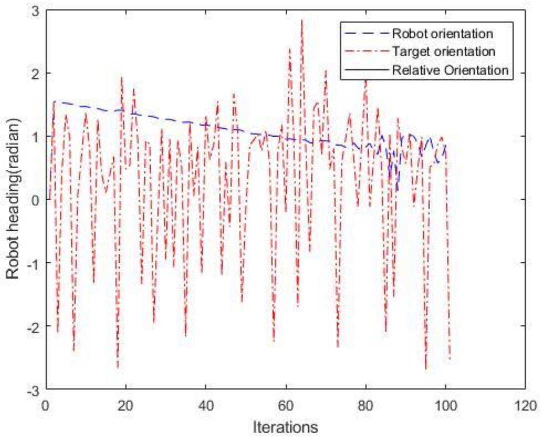


Linear Trajectory with Noise

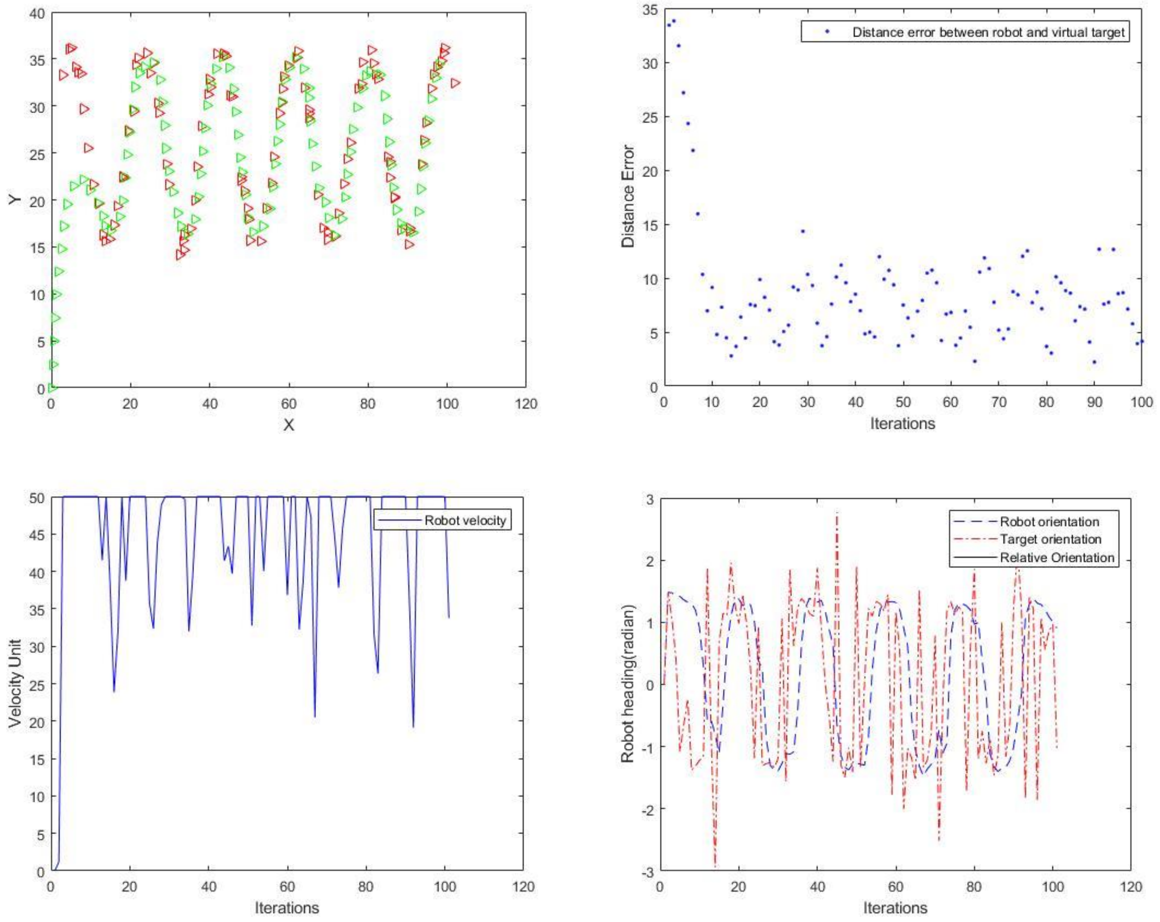








Sine Wave with Noise



Appendix

```
%Russell Keith
%CPE 670
%3/23/2022
%Project 2: Potentioal Field Path Planning

clc,clear
close all
n = 2; % Number of dimensions
delta_t = 0.05; % Set time step
t = 0:delta_t:5;% Set total simulation time 0 0.05 0.1 0.15.....
lambda = 8.5; % Set scaling factor of attractive potential field
vr_max = 50; % Set maximum of robot velocity

%Set Virtual Target
qv = zeros (length(t),n); %Initial positions of virtual target
pv = 1.2; %Set velocity of virtual target
theta_t = zeros (length(t),1); % Initial heading of the virtual target

%Set Robot
qr = zeros (length(t),n); %initial position of robot
vrd =  zeros (length(t),1); %Initial velocity of robot
theta_r = zeros (length(t),1); % Initial heading of the robot


qrv = zeros (length(t),n); %Save relative positions between robot and virtual
target
prv = zeros(length(t),n); %Save relative velocities between robot and virtual
target

qrv(1,:) = qv(1,:) - qr(1,:);%Compute the initial relative position

%Compute the initial relative velocity
prv(1,:) = [pv*cos(theta_t(1))-vrd(1)*cos(theta_r(1)), pv*sin(theta_t(1))-
vrd(1)*sin(theta_r(1))];

%====Set noise mean and standard deviation====
noise_mean = 0.8;
noise_std = 0.8;




for i =2:length(t)
    %++++++++++CIRCULAR TRAJECTORY+++++++++++
      %Set target trajectory moving in CIRCULAR trajectory WITHOUT noise
%     qv_x = 60 - 15*cos(t(i));
%     qv_y = 30 + 15*sin(t(i));
%     qv(i,:) = [qv_x, qv_y]; %compute position of virtual target

     %Set target trajectory moving in CIRCULAR trajectory WITH noise
%    qv_x = 60 - 15*cos(t(i))+ noise_std * randn + noise_mean;
%    qv_y = 30 + 15*sin(t(i)) + noise_std * randn + noise_mean;
%    qv(i,:) = [qv_x, qv_y];  %compute position of target
```

```
    %++++++++++LINEAR TRAJECTORY+++++++++++
      %Set target trajectory moving in Linear trajectory WITHOUT noise
%     qv_x = t(i);
%     qv_y = qv_x + 100;
%     qv(i,:) = [qv_x, qv_y]; %compute position of virtual target

     %Set target trajectory moving in Linear trajectory WITH noise
%    qv_x = i + noise_std * randn + noise_mean;
%    qv_y = qv_x + 100 + noise_std * randn + noise_mean;
%    qv(i,:) = [qv_x, qv_y];  %compute position of target

        %++++++++++SINE WAVE TRAJECTORY+++++++++++
      %Set target trajectory moving in sine trajectory WITHOUT noise
%    qv_x = i;
%    qv_y = 10*sin(1/3*qv_x) + 25;
%    qv(i,:) = [qv_x, qv_y]; %compute position of virtual target

     %Set target trajectory moving in sine trajectory WITH noise
    qv_x = i + noise_std * randn + noise_mean;
    qv_y = 10*sin(1/3*qv_x) + 25 + noise_std * randn + noise_mean;
    qv(i,:) = [qv_x, qv_y];  %compute position of target




    %Compute the target heading
    qt_diff(i,:) =qv(i,:)- qv(i-1,:);
    theta_t(i) = atan2(qt_diff(i,2),qt_diff(i,1));

    %Calculation
    phi=atan2(qrv(i-1,2),qrv(i-1,1));

    vrd(i) = sqrt((norm(pv)^2) + 2*lambda*norm(qrv(i-
1,:))*abs(cos(theta_t(i)- phi)) + (lambda^2)*(norm(qrv(i-1,:))^2));

    if vrd(i)>vr_max
        vrd(i)= vr_max;
    end

    theta_r(i) = phi + asin((norm(pv)*sin(theta_t(i) - phi))/(vrd(i)));







    %======UPDATE position and velocity of robot===========
    qr(i,:) = qr(i-1,:) + vrd(i)*delta_t*[cos(theta_r(i-1)), sin(theta_r(i-
1))];

    qrv(i,:) = qv(i,:) - qr(i,:);
    prv(i,:) = [pv*cos(theta_t(i))-vrd(i)*cos(theta_r(i)),
pv*sin(theta_t(i))-vrd(i)*sin(theta_r(i))];
```

```
    error(i) = norm(qv(i,:) - qr(i,:));


    %plot postions qv of virtual target
    plot(qv(:,1),qv(:,2),'r>')
    xlabel('X');
    ylabel('Y')
    hold on
    %plot postions qv of robot
    plot(qr(:,1),qr(:,2),'g>')
    M = getframe(gca);
    %mov = addframe(mov,M);
end

figure(2), plot(error(2:length(t)), 'b.')
xlabel('Iterations');
ylabel('Distance Error')
legend('Distance error between robot and virtual target')
figure(3), plot(vrd, 'b')
xlabel('Iterations');
ylabel('Velocity Unit')
legend('Robot velocity')
figure(4), plot(theta_r, '--b')
xlabel('Iterations');
ylabel('Robot heading(radian)')
hold on
plot(theta_t, '-.r')
hold on
plot(phi, 'k')
legend('Robot orientation', 'Target orientation', 'Relative Orientation')
```