Grant Davis
CPE 470
03/31/2022

# Project 2 Report: Potential Field Path Planning

This report documents my potential field path planning in Matlab. The Matlab program will simulate a robot following the same trajectory as a moving target. The robot can follow the target using a model, constructed from the target's heading and both relative positions of the robot and target. This model will update the heading and velocity of the robot.

## Noise-free environment

### Target to move in a linear trajectory

For the robot to reach the target the velocity of the robot must be at least the same. With linear trajectory, if the robot and target have the same velocities they will stay an equal distance from each other for the entire traversal period. Fig. 1 shows the robot following the target trajectory. By using the tracking error their velocities are matched. If the tracking error between the robot and the target were to be zero, a collision would occur. To avoid such a collision, when the tracking error decreases the robot's velocity decreases as well, this is shown in Fig. 2 and Fig. 4 respectively. The heading of the robot will match the target's heading. The robot will adjust its heading to match the target, then both will remain constant, shown in Fig. 3.

Fig. 1: Trajectories of the target and robot

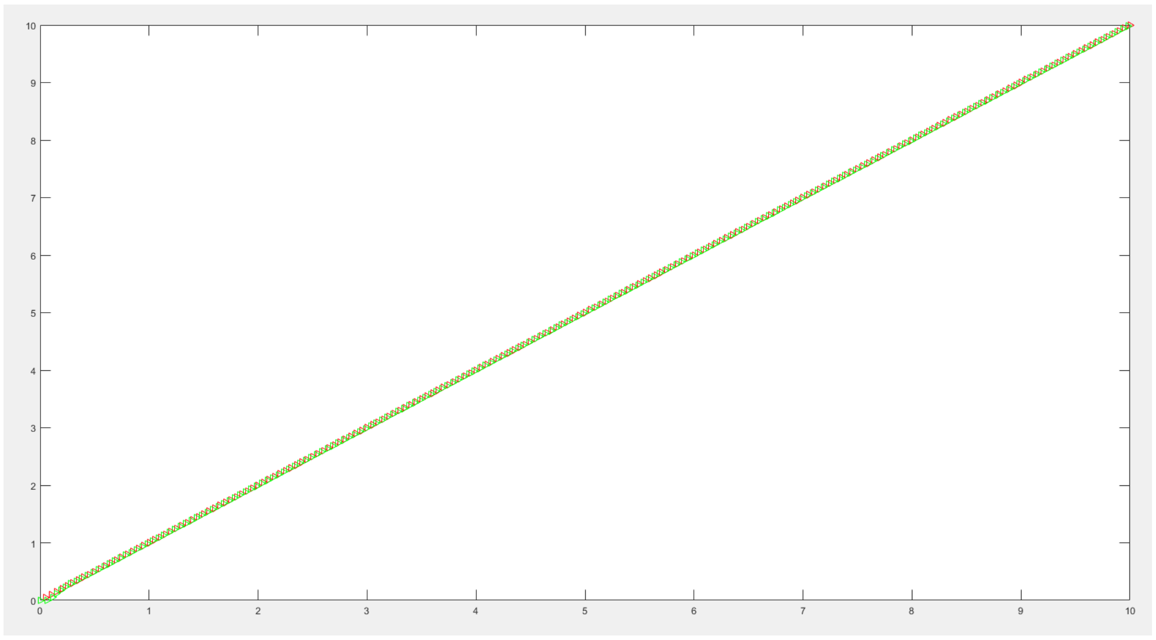


Fig. 2: Tracking error between the target and robot

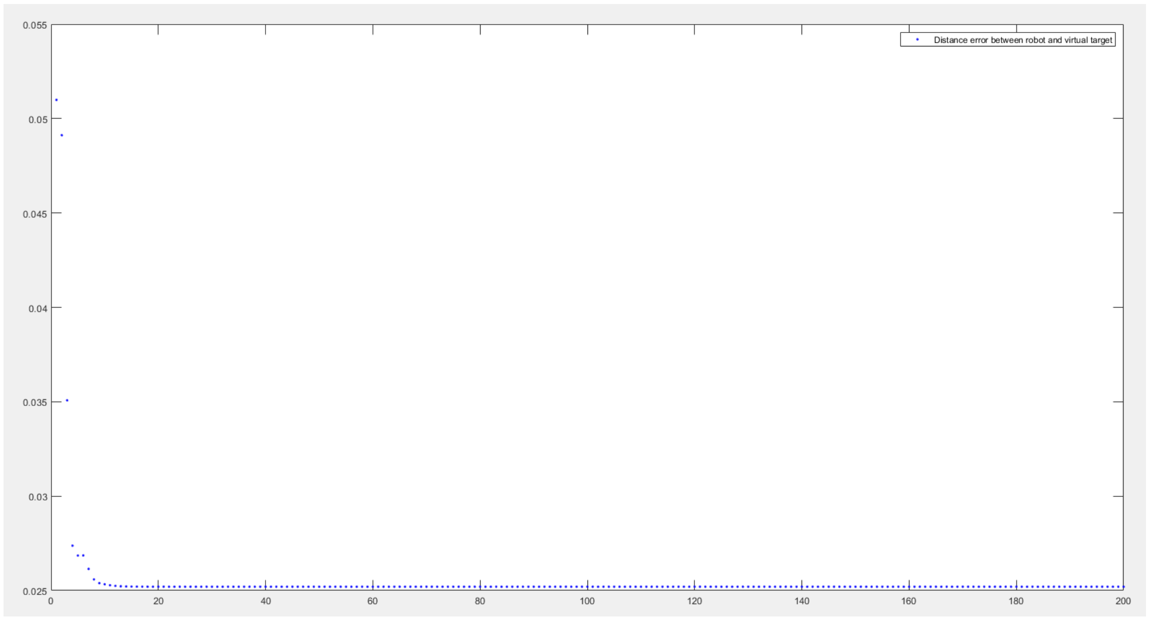


Fig. 3: Robot’s heading

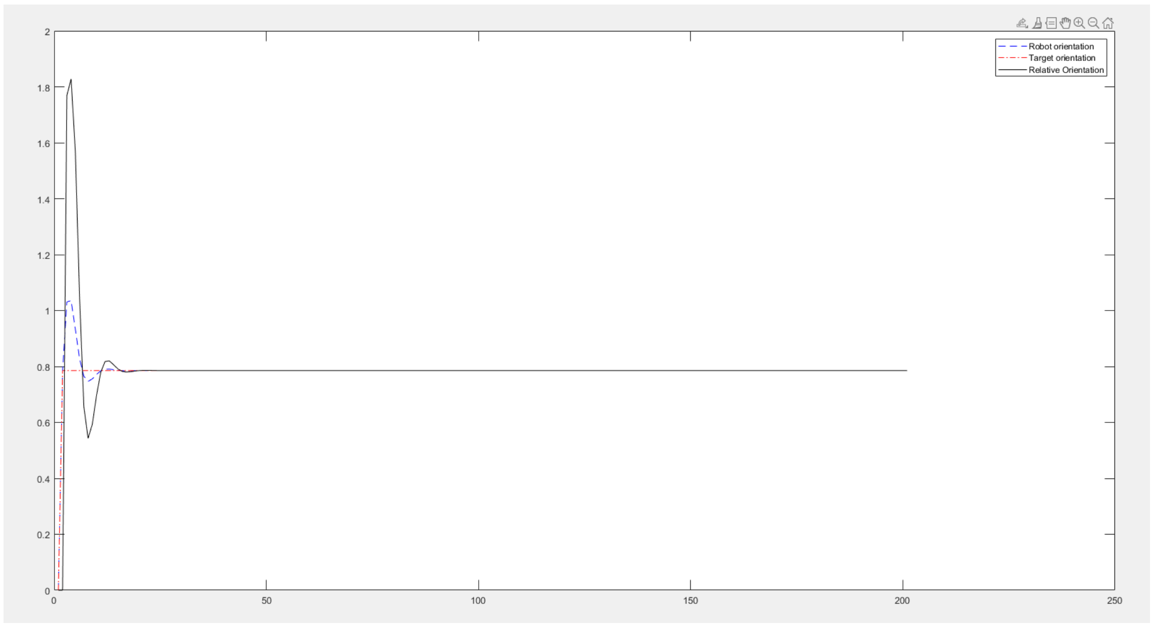
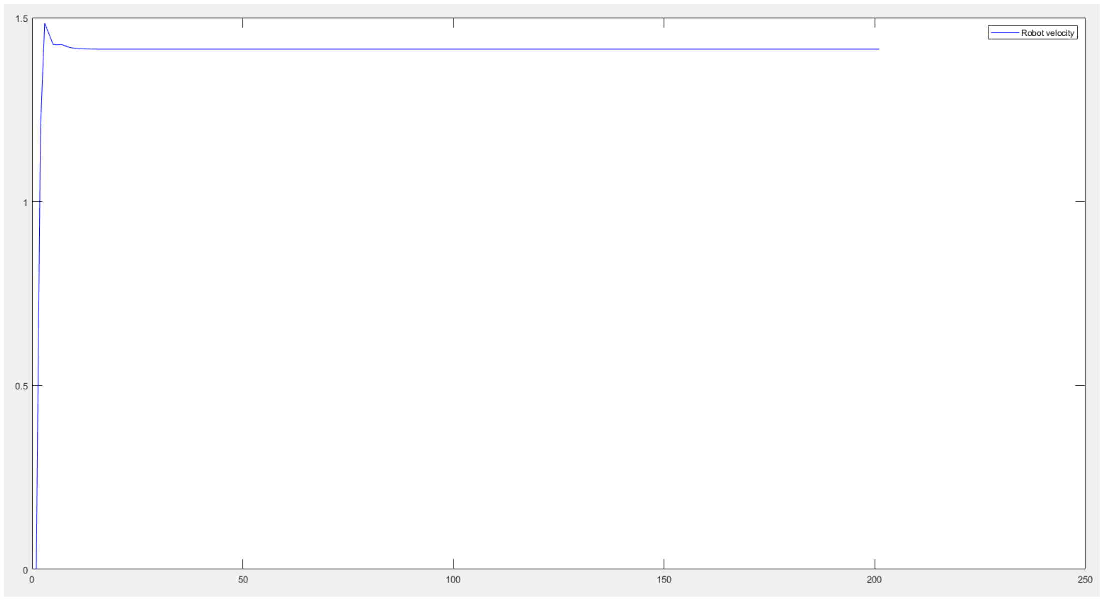


Fig. 4: Robot's velocity



**Target to move in a sin wave trajectory**

In sin wave trajectory the robot is able to catch up to the target when velocities are equal as shown in Fig. 5. This is because each time the target must change direction its velocity must decrease. During each change in direction, the robot could collide with the target as the tracking error reaches closer to zero as shown in Fig. 6. Fig. 7 shows the headings of the target and robot while they follow the sin wave trajectory. To avoid collision with the target the velocity will have a cap as shown in Fig. 8.

Fig. 5: Trajectories of the target and robot

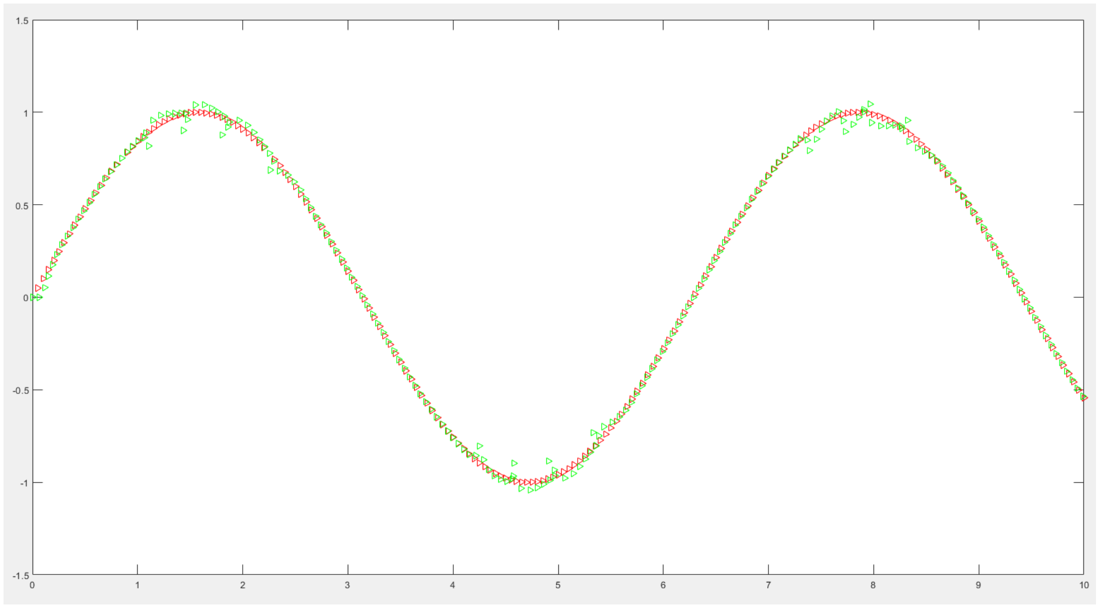


Fig. 6: Tracking error between the target and robot

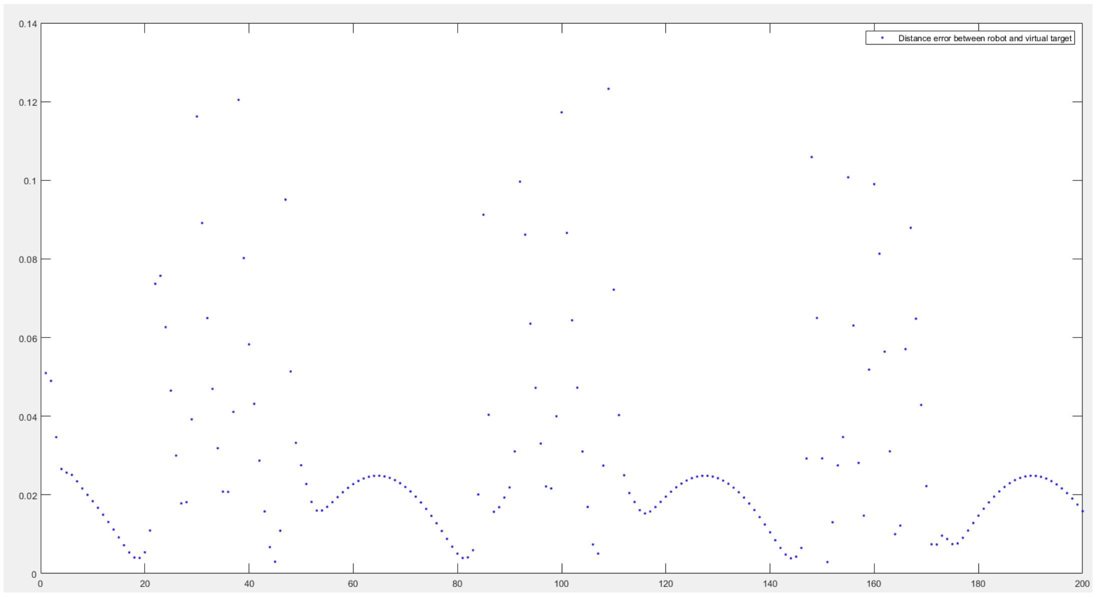


Fig. 7: Robot's heading

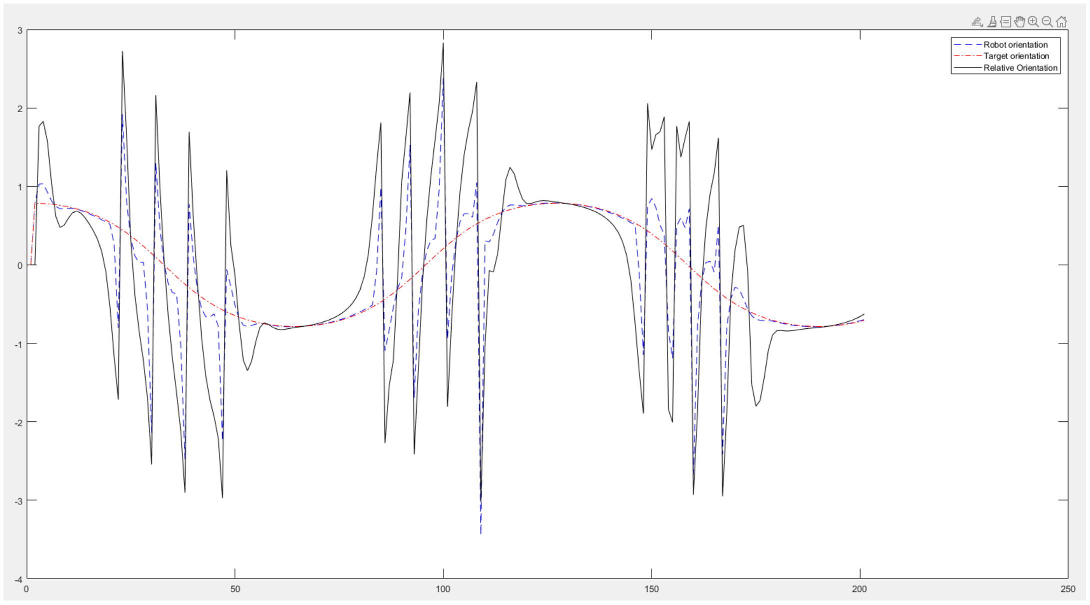
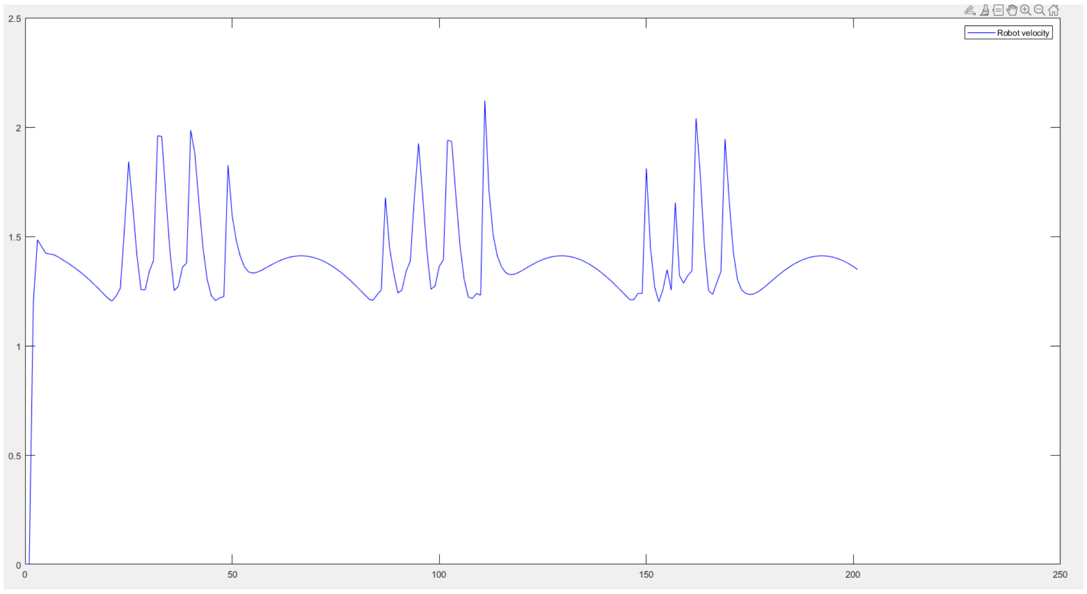


Fig. 8: Robot's velocity



## Noisy environment

### Target to move in a linear trajectory

Noise is being generated for the target using a standard deviation of 0.5 and a mean of 0.5. Fig. 9 shows the result of the robot trying to follow the target with noise. Fig. 10 shows the tracking error, it is difficult for the robot to track the target but the robot does come close to zero at points. The introduction of noise affects the heading of the target drastically. The target heading is not constant and difficult for the robot to determine as shown in Fig. 11. The velocity of the robot is changed constantly as it tries to track and follow the target as shown in Fig. 12.

Fig. 9: Trajectories of the target and robot

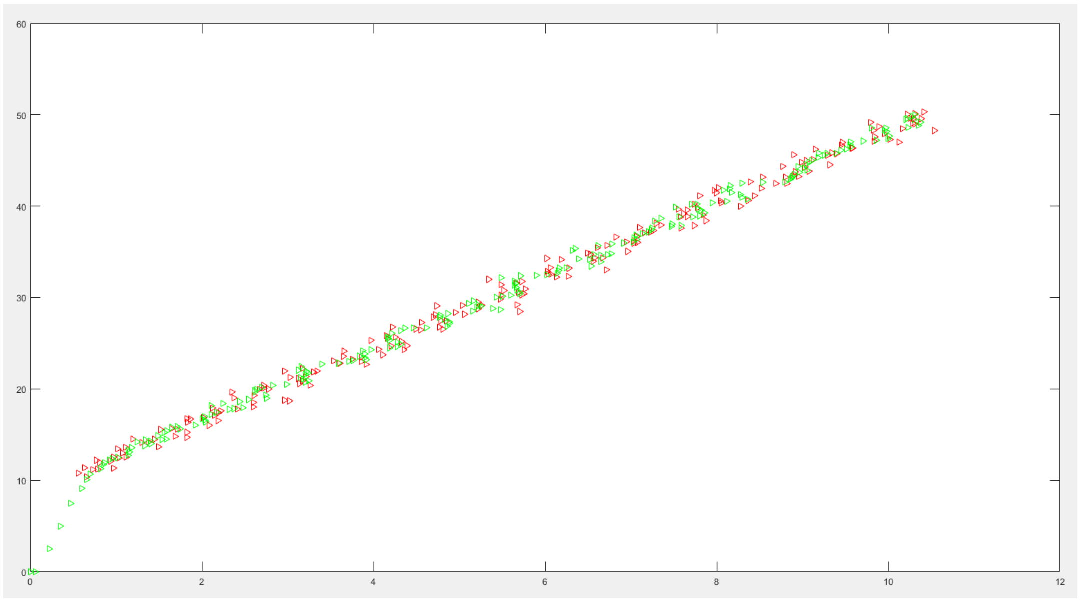


Fig. 10: Tracking error between the target and robot

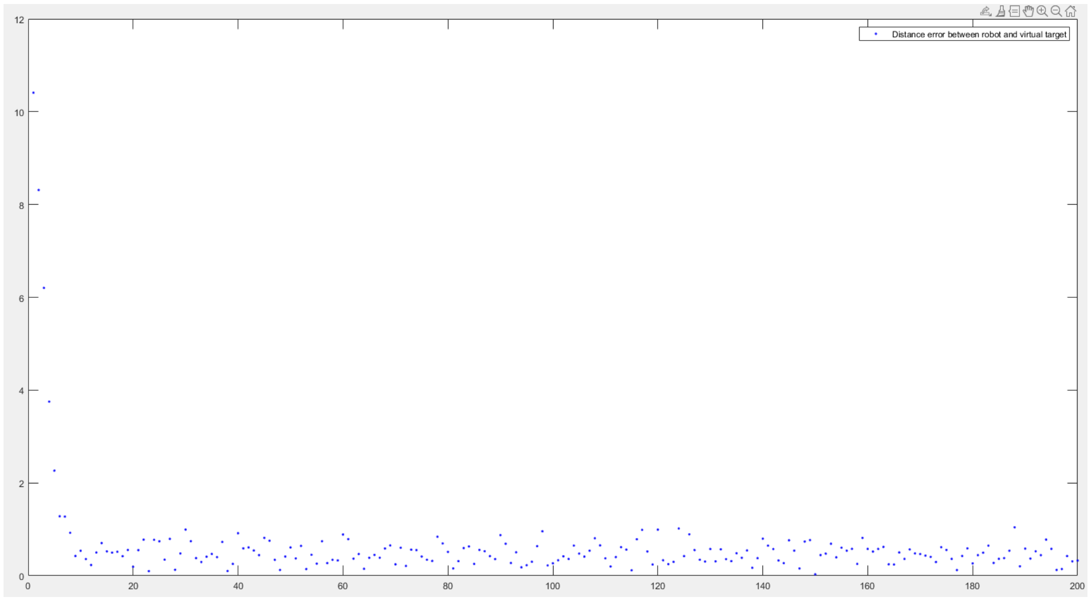


Fig. 11: Robot's heading

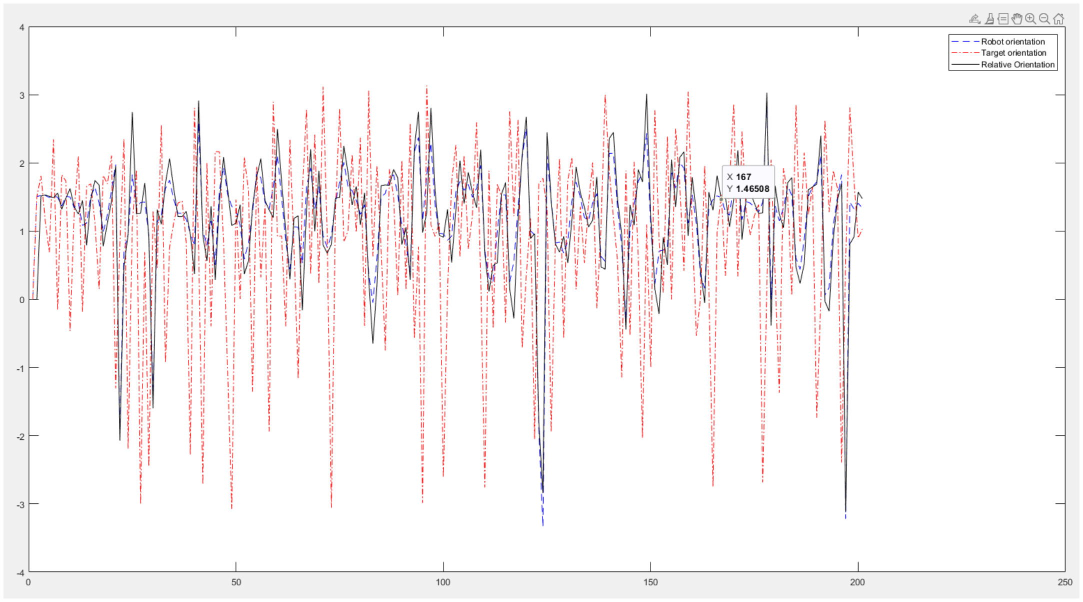
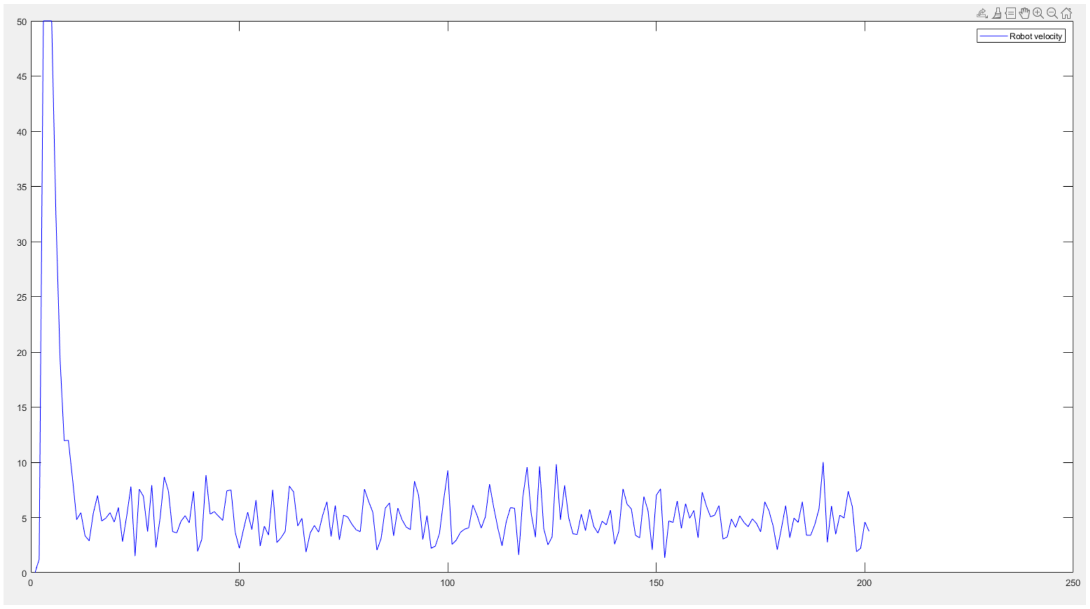


Fig. 12: Robot's velocity



**Target to move in a sin wave trajectory**
Noise is being generated for the target using the same standard deviation and mean. The added noise creates further problems for the sin wave trajectory as shown in Fig. 13. The robot has a less difficult time tracking the target position compared to tracking in a linear trajectory, as shown in Fig. 10 and Fig. 14. While noisy, Fig. 15 shows that the heading of the target is easier for the robot to determine compared to the linear trajectory target heading in Fig. 11. The robot still must slow down and speed up as it follows the target's change in heading as shown in Fig. 16.

Fig. 13: Trajectories of the target and robot

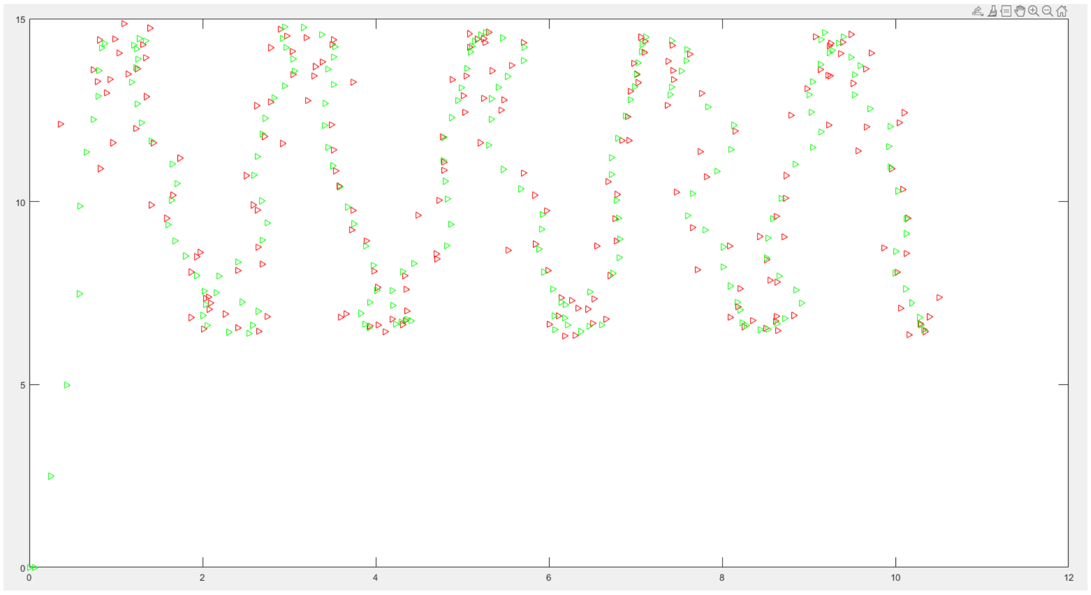


Fig. 14: Tracking error between the target and robot

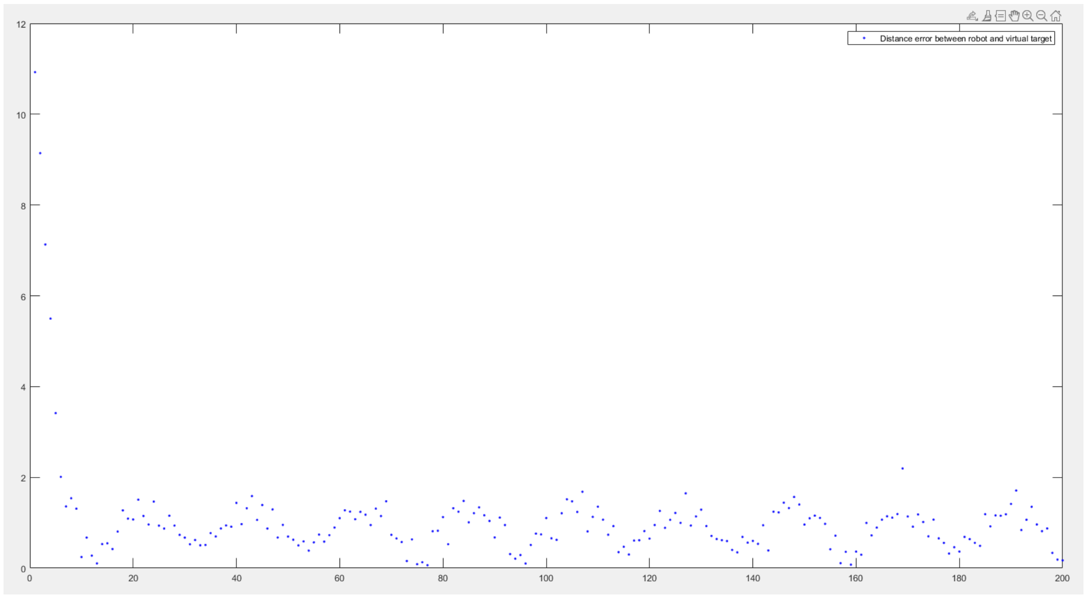


Fig. 15: Robot's heading

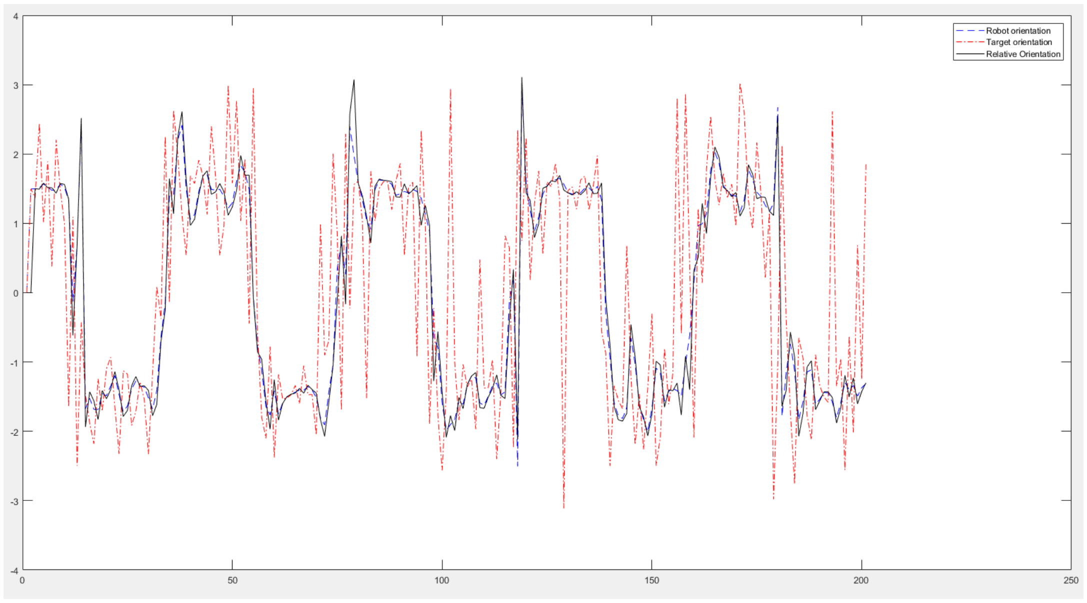
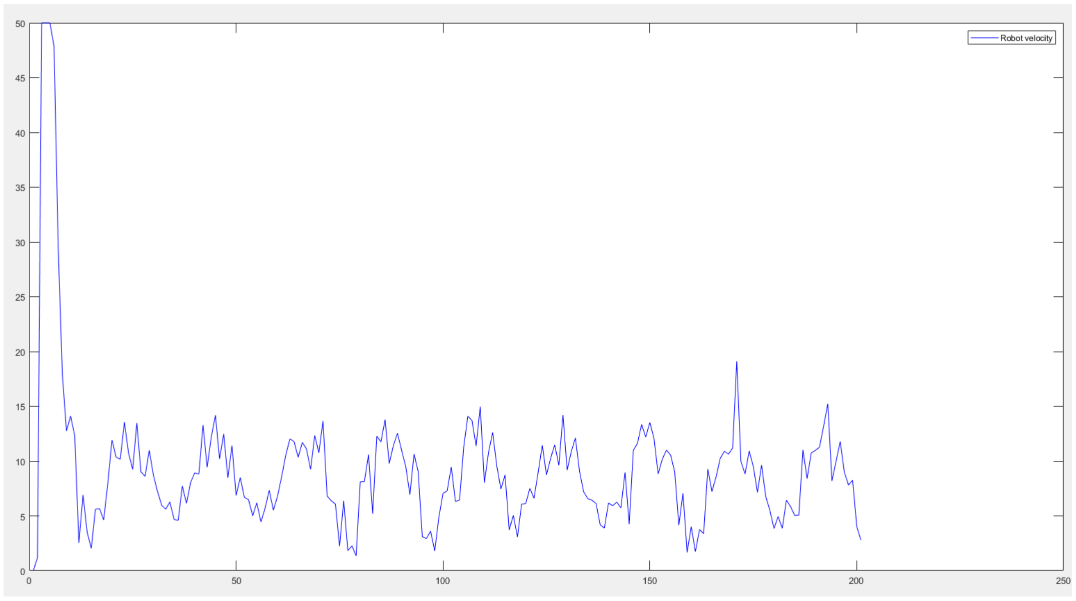


Fig. 16: Robot's velocity



## Appendix: Software Source Code

## Kalman Filter (MATLAB source code)

### 1. main.m

```
%Grant Davis
%CPE470 Project 2: Potential Field Path Planning
%03/31/2022

%Initialize
clc,clear
close all

%Set parameters for simulation
n = 2; % Number of dimensions
delta_t = 0.05; % Set time step
t = 0:delta_t:10;% Set total simulation time
lambda = 8.5; % Set scaling factor of attractive potential field
vr_max = 50; % Set maximum of robot velocity
error = zeros (length(t) - 1, 1); % Set tracking error

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

%Set relative states between robot and VIRTUAL TARGET
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

%Compute initial relative states 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))-v_rd(1)*cos(theta_r(1)),
pv*sin(theta_t(1))-v_rd(1)*sin(theta_r(1))];
qt_diff = zeros(length(t), n);
phi = zeros(length(t), 1);

%Set noise mean and standard deviation
noise_mean = 0.5;
noise_std = 0.5; %try 0.2/0.5
```

```
%=========MAIN PROGRAM==================
for i =2:length(t)
%Trajectory
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Linear Trajectory without noise
qv_x = t(i);
qv_y = t(i);
qv(i,:) = [qv_x, qv_y]; %compute position of target
%Linear Trajectory with noise
% qv_x = t(i)+ noise_std * randn + noise_mean;
% qv_y = 4*t(i) + 10 + noise_std * randn + noise_mean;
% qv(i,:) = [qv_x, qv_y];  %compute position of target
%Sin Wave Trajectory without noise
% qv_x = t(i);
% qv_y = sin(t(i));
% qv(i,:) = [qv_x, qv_y]; %compute position of target
%Sin Wave Trajectory with noise
% qv_x = t(i) + noise_std * randn + noise_mean;
% qv_y = 4*sin(t(i) * 3) + 10 + 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));
phi(i) = atan2(qrv(i - 1,2), qrv(i - 1,1));

%modeling robot velocity
v_rd(i) = sqrt((pv^2) + (2*lambda*norm(qrv(i -
1,:))*pv*abs(cos(theta_t(i)-phi(i)))) + ((norm(qrv(i - 1,:))*lambda)^2));
if v_rd(i) >= vr_max
  v_rd(i) = vr_max;
end

%modeling robot heading
theta_r(i) = phi(i) + asin((pv*sin(theta_t(i) - phi(i))/v_rd(i)));

%UPDATE position and velocity of robot
qr(i,:) = qr(i-1,:) + v_rd(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))-v_rd(i)*cos(theta_r(i)),
pv*sin(theta_t(i))-v_rd(i)*sin(theta_r(i))];
error(i) = norm(qv(i,:)-qr(i,:));

%plot postions qv of virtual target
plot(qv(:,1),qv(:,2),'r>')
hold on
%plot postions qv of robot
plot(qr(:,1),qr(:,2),'g>')
```

```
M = getframe(gca);
%mov = addframe(mov,M);
end
%Plot results
figure(2), plot(error(2:length(t)), 'b.')
legend('Distance error between robot and virtual target')
figure(3), plot(v_rd, 'b')
legend('Robot velocity')
figure(4), plot(theta_r, '--b')
hold on
plot(theta_t, '-.r')
hold on
plot(phi, 'k')
legend('Robot orientation', 'Target orientation', 'Relative Orientation')
```

## Execution Commands

To execute the program using provided source code must use a MATLAB environment with file main.m. Open MATLAB and run main.m, which should display a linear wave trajectory without noise by default. To change to a sine wave or add noise, comment out "%Linear Trajectory without noise" section using ctrl + 'r' and remove comments from the intended trajectory section using ctrl + shift + 'r'.