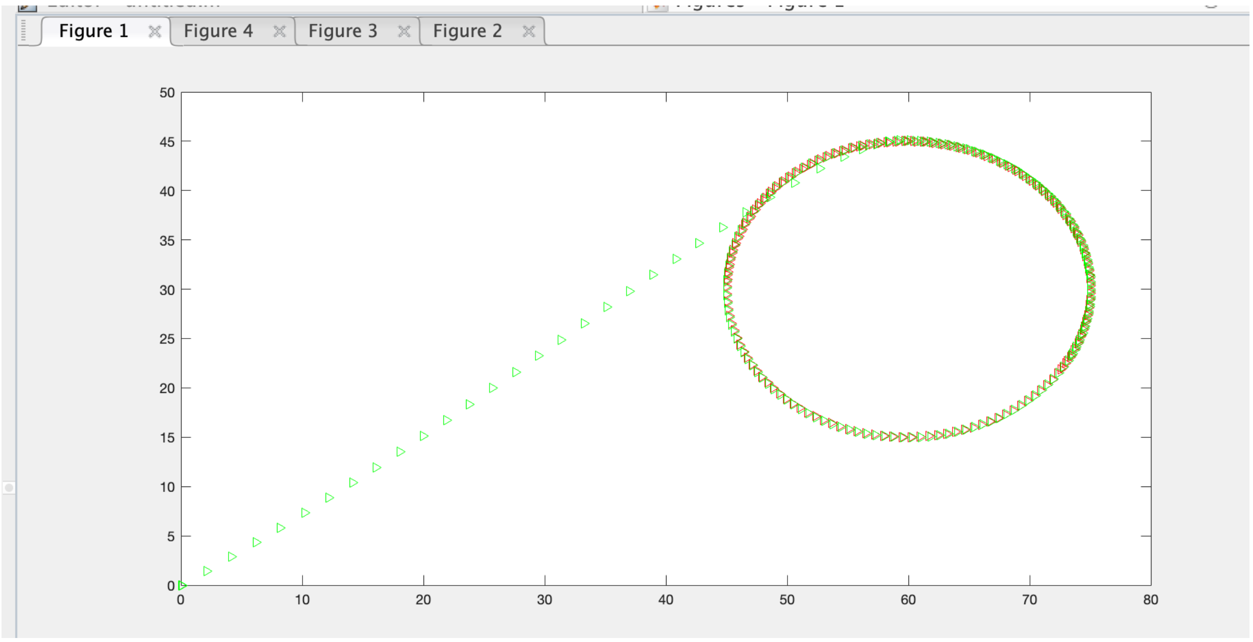
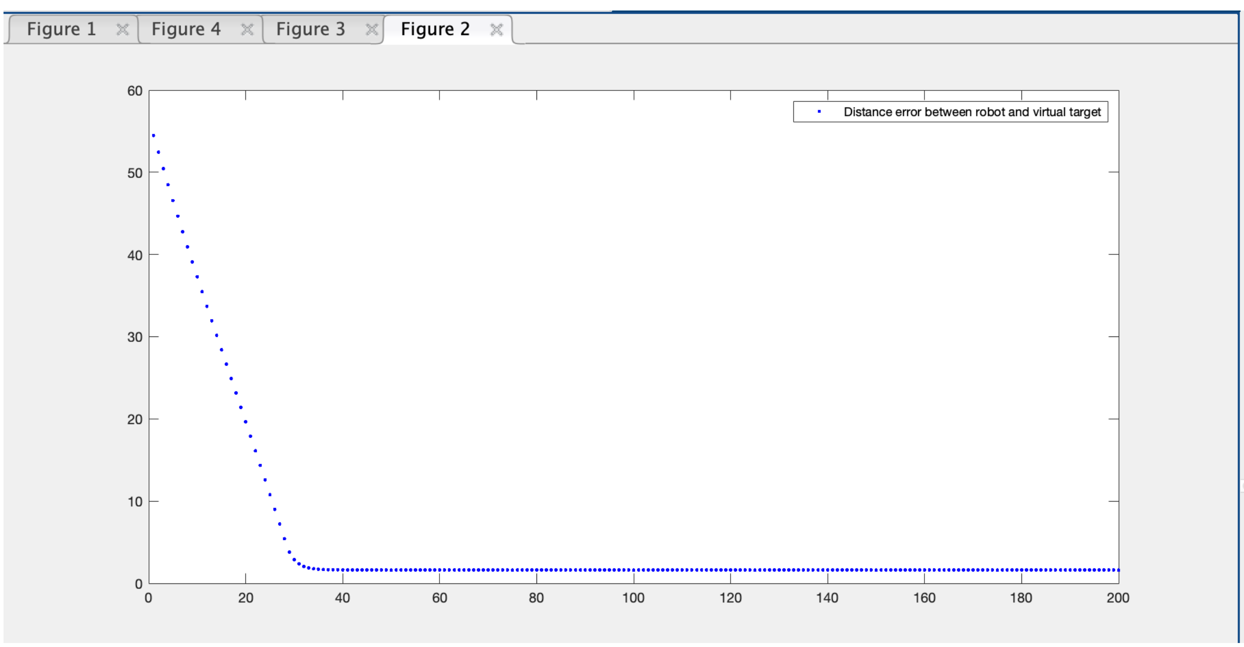
Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

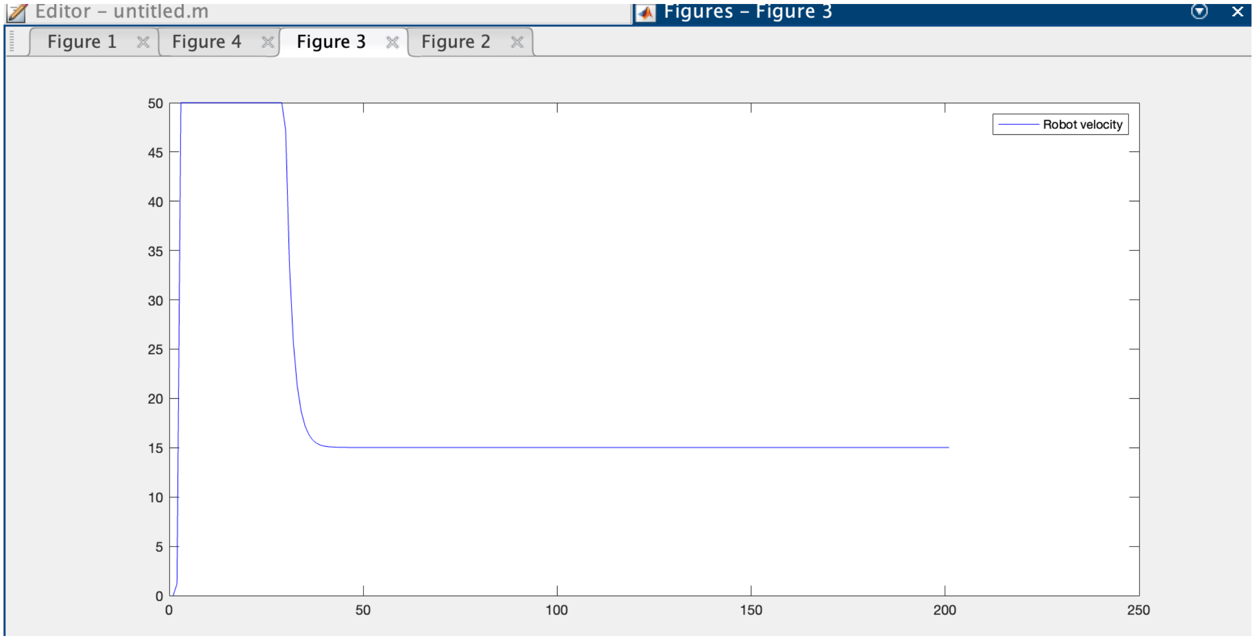
Part 1: Noise Free Environment

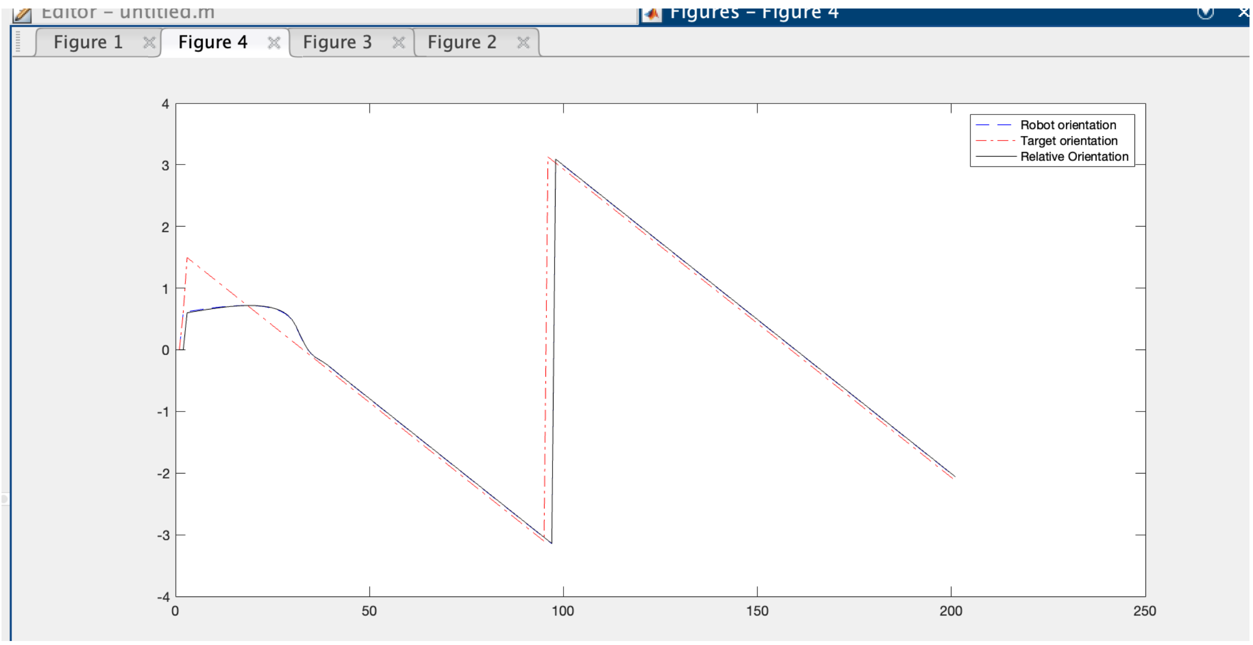
Circular Trajectory:





Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

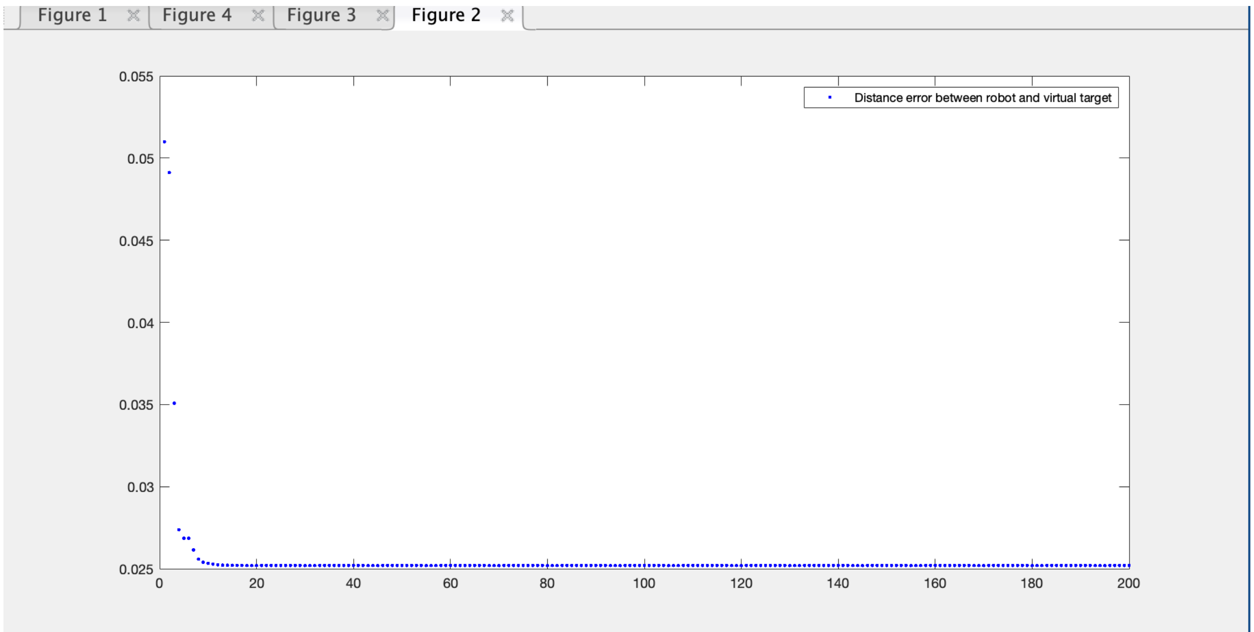


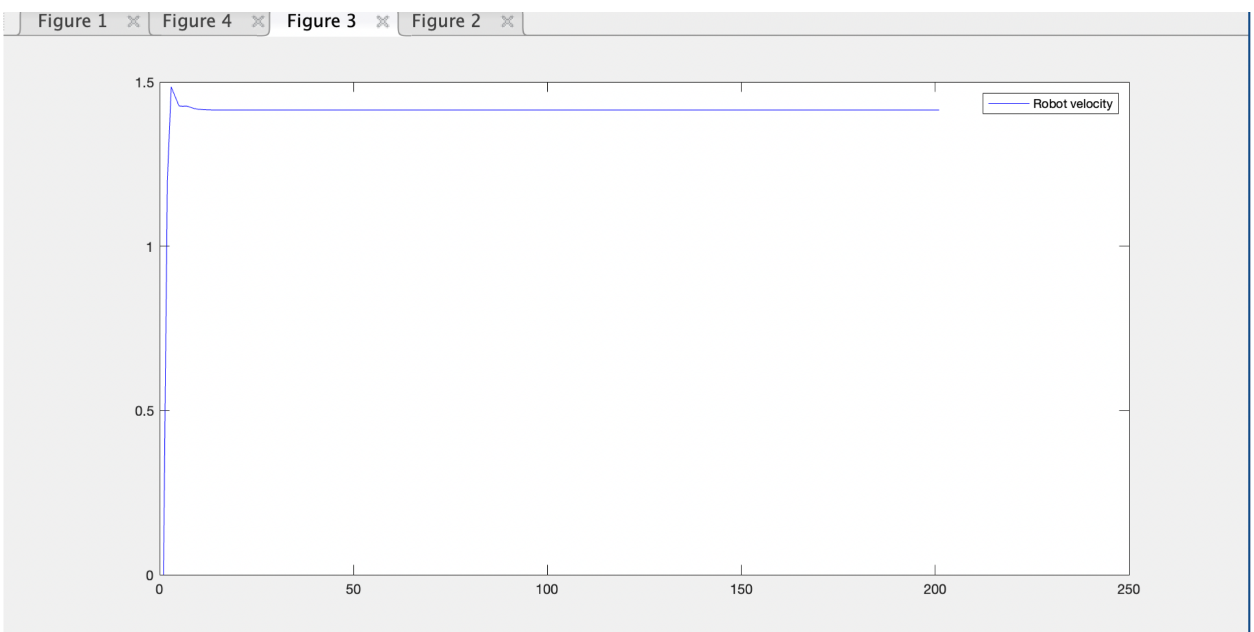


Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

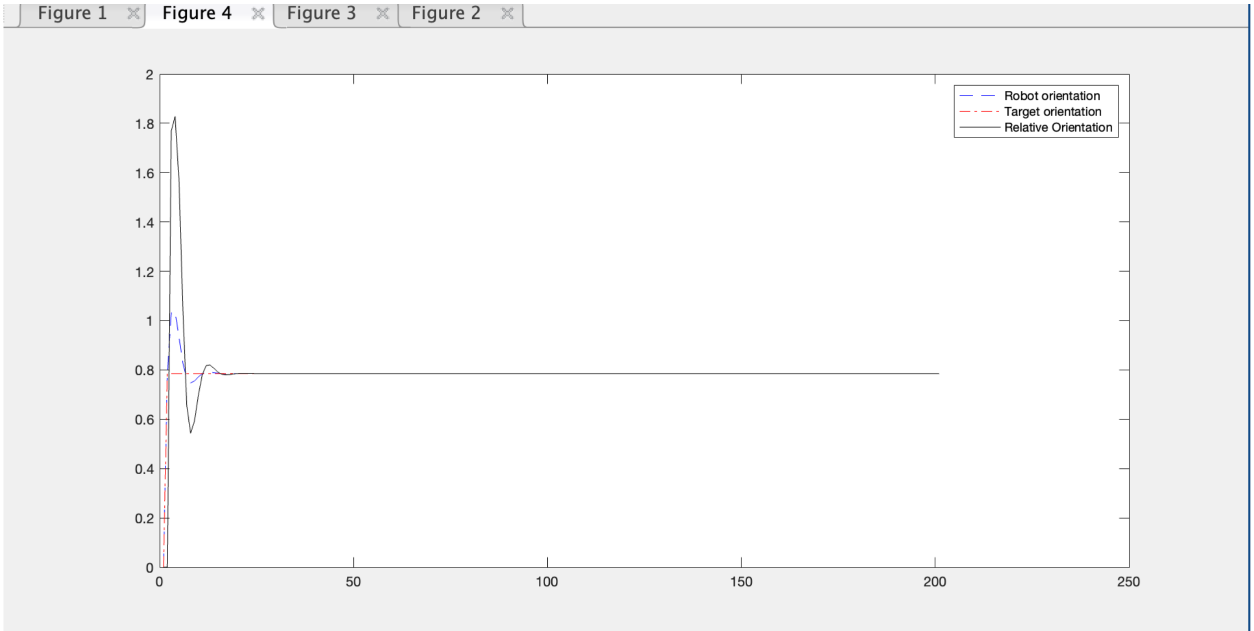
Linear:

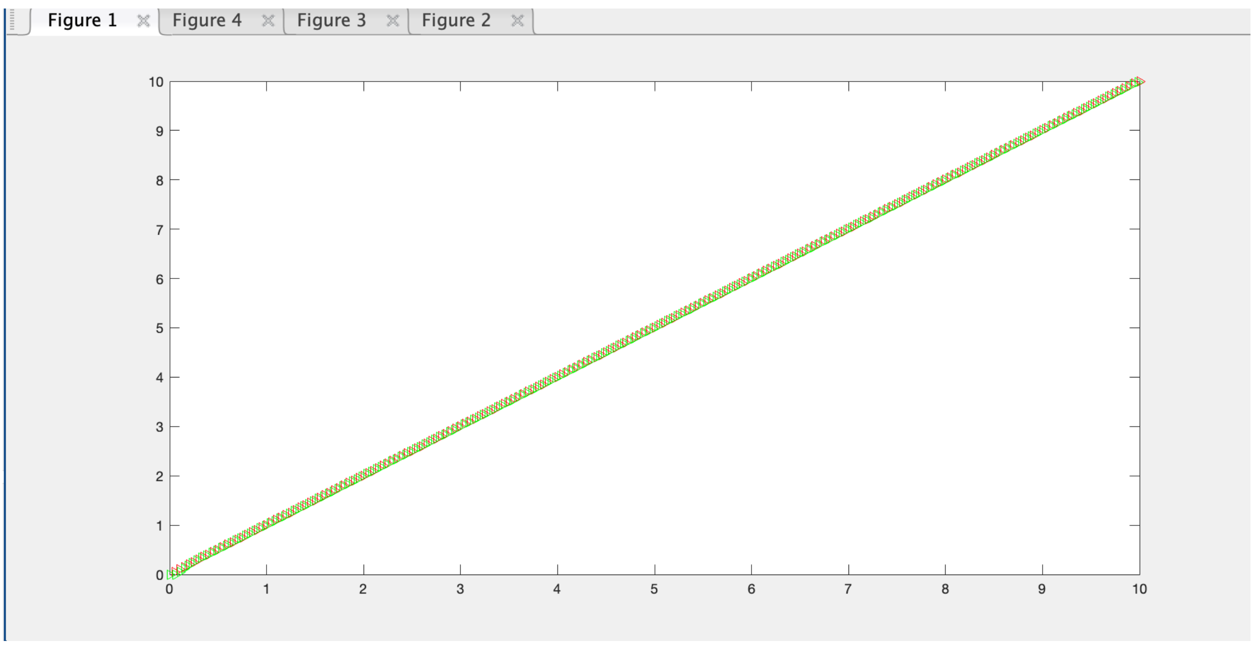
The Figure 1 picture shows how the robot(green) is chasing its target(red) in a linear path. We can see how the distance error becomes closer to zero as the robot is getting closer to the target. The robot is also able to catch up easier because there is no noise.





Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

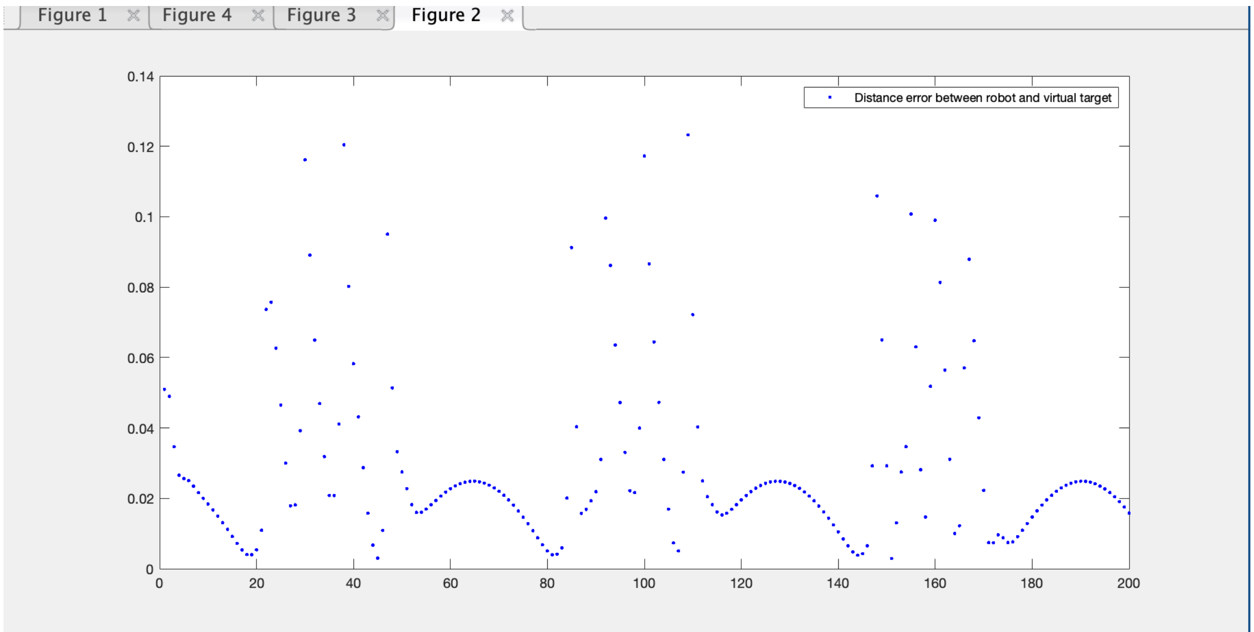


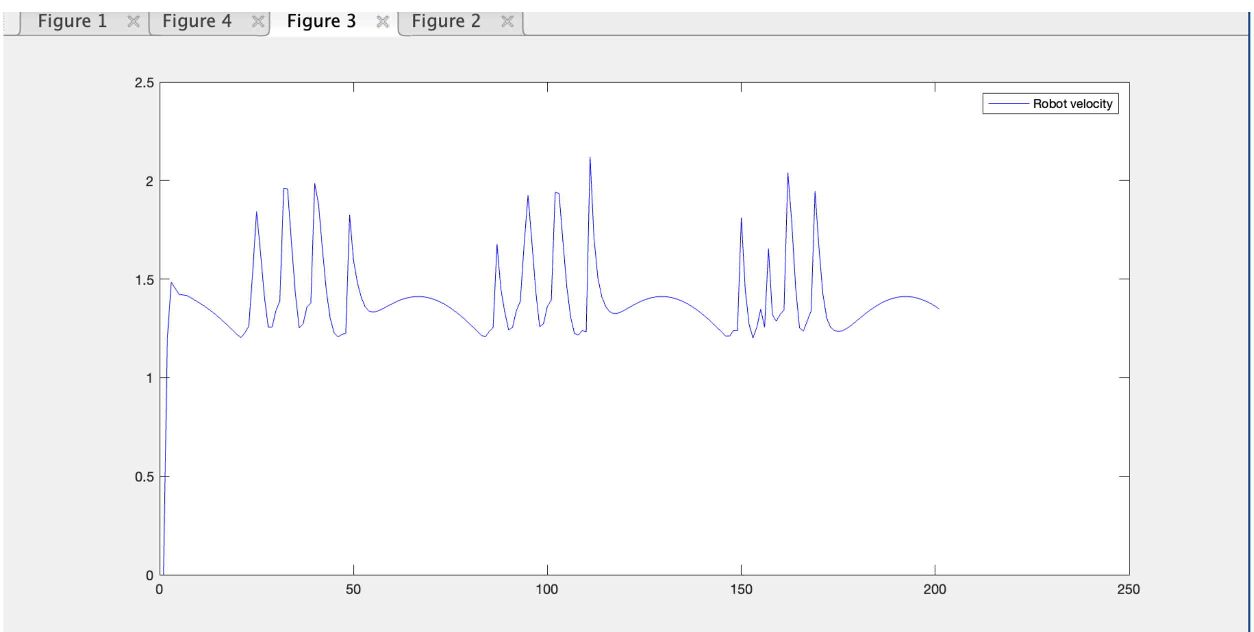


Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

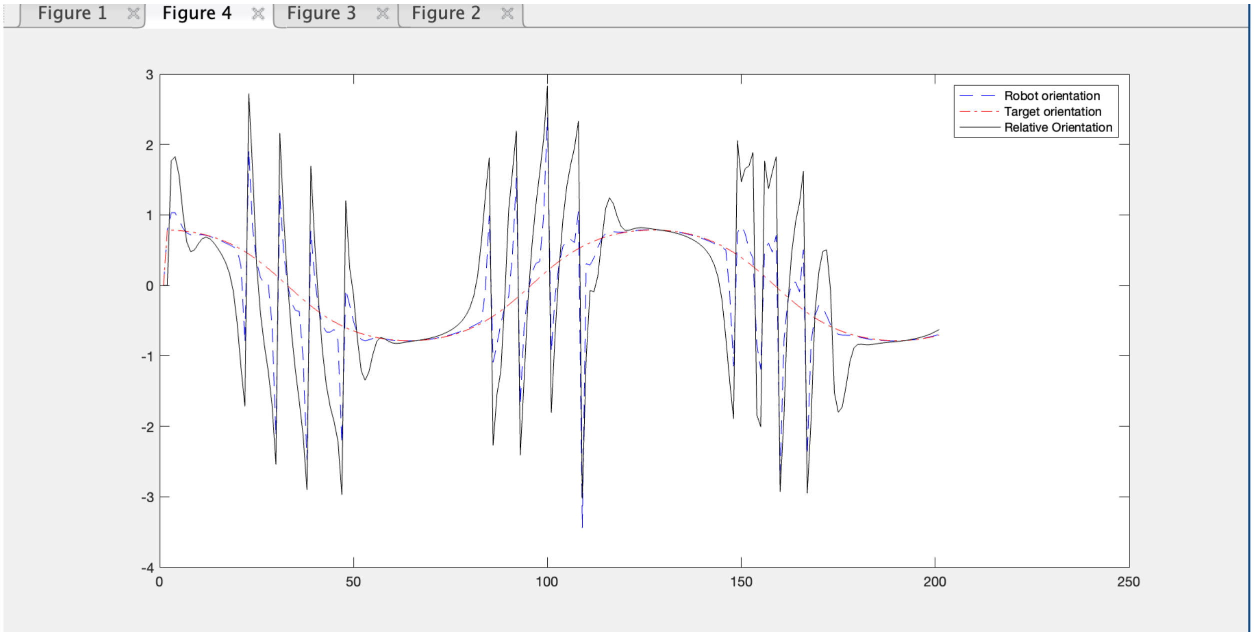
Sine:

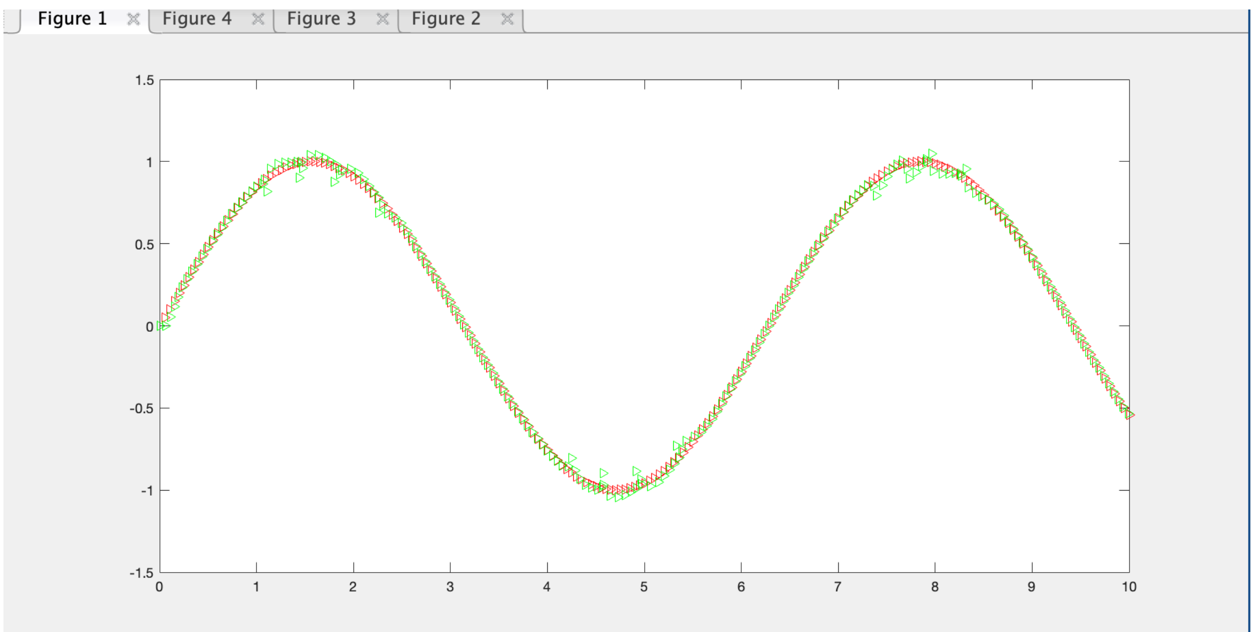
In figure 1 we can see that the robot is able to follow the path of the target with minimal errors. We can see the robot not catching up prominently as the target approaches 1 and -1 in the sine wave. I think this is because the target is turning and for the robot to keep up in the turn, it must also decrease it's speed.





Aaron Ramirez
CPE 470
Project 2
Dr. Jim La



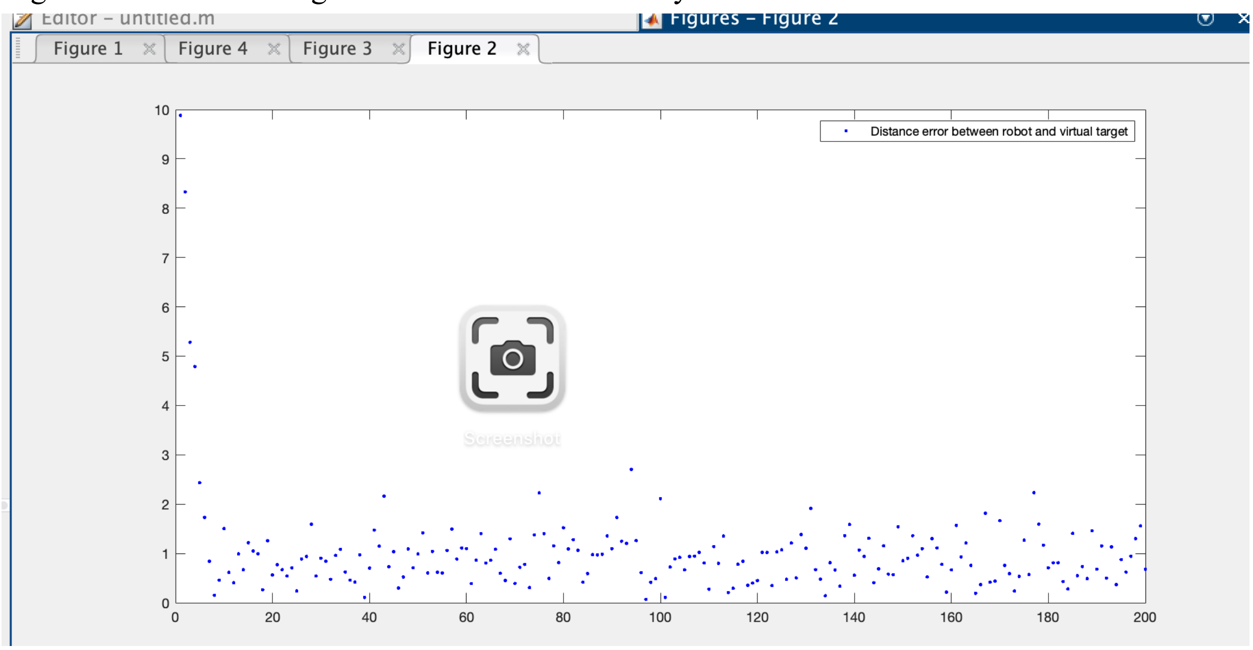


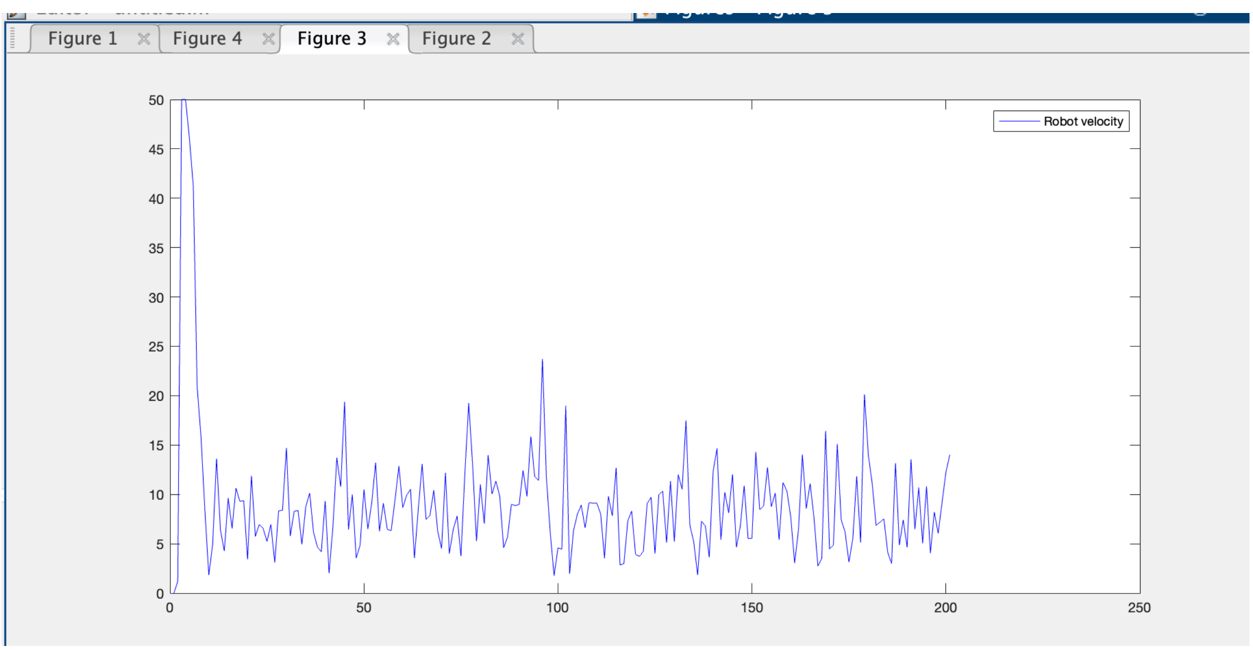
Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

Part 2: Noisy Environment

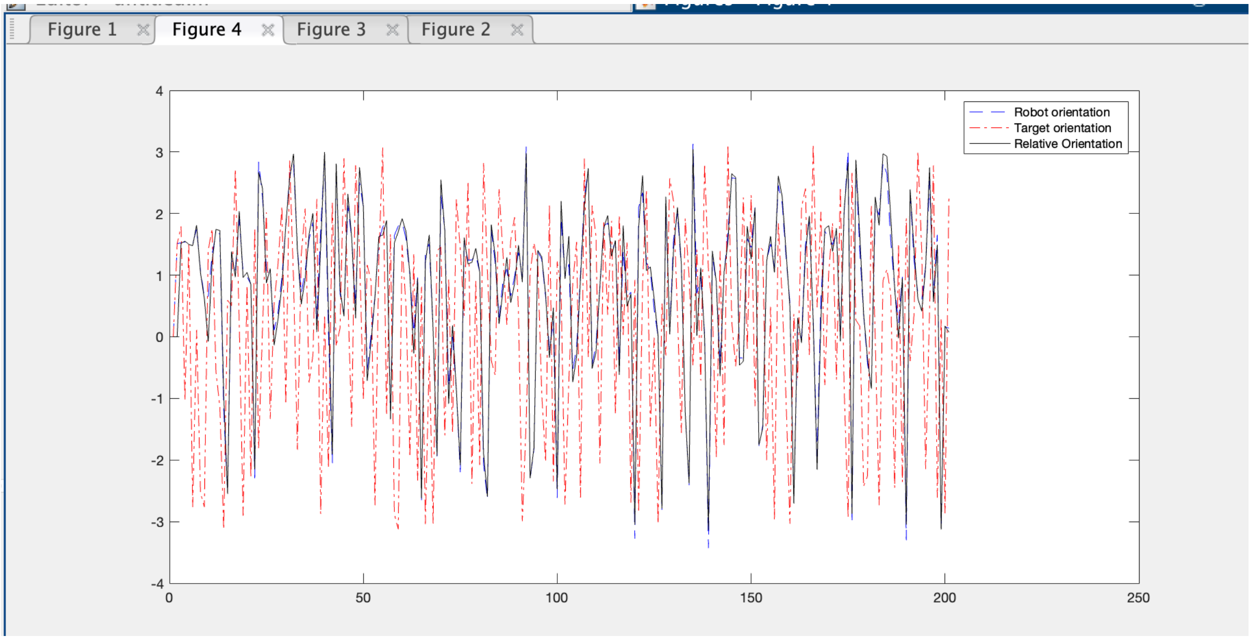
Linear:

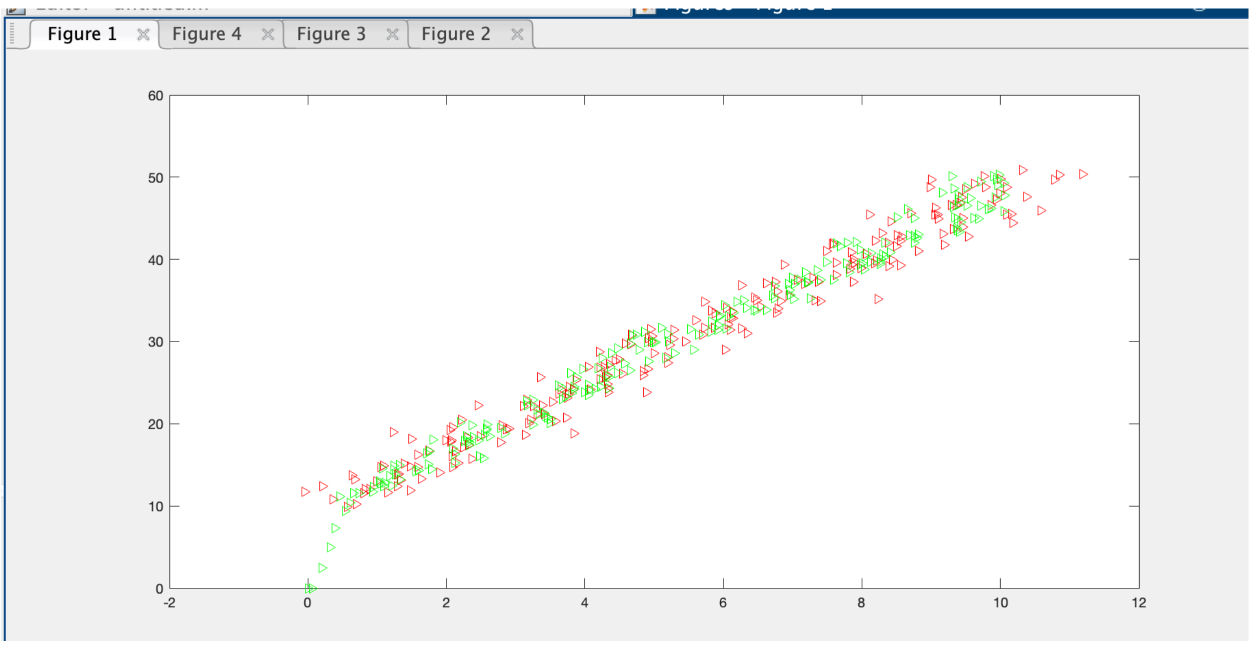
In the noisy environment, the robot has a hard time catching up to the target because of the noise interfering with the calculations of the robot. However, since the path of the target is linear, the robot can head to its general direction and sometimes be able to get very close to the target as we can see in figure 2 where the error is very small.





Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

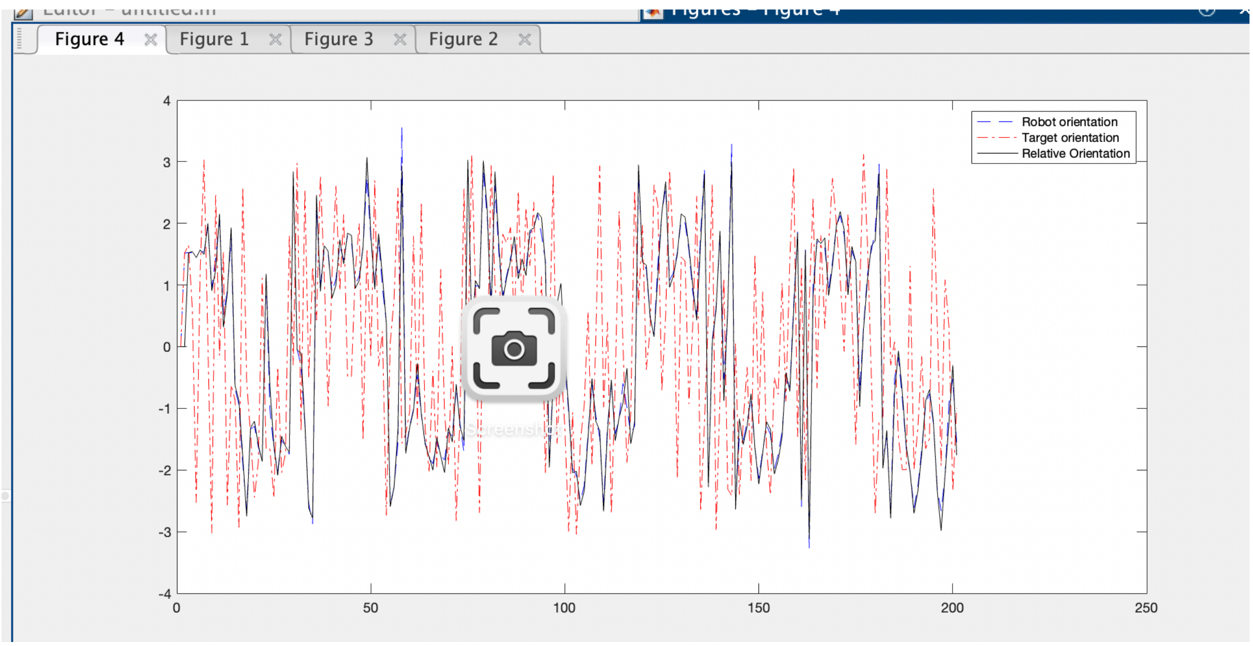


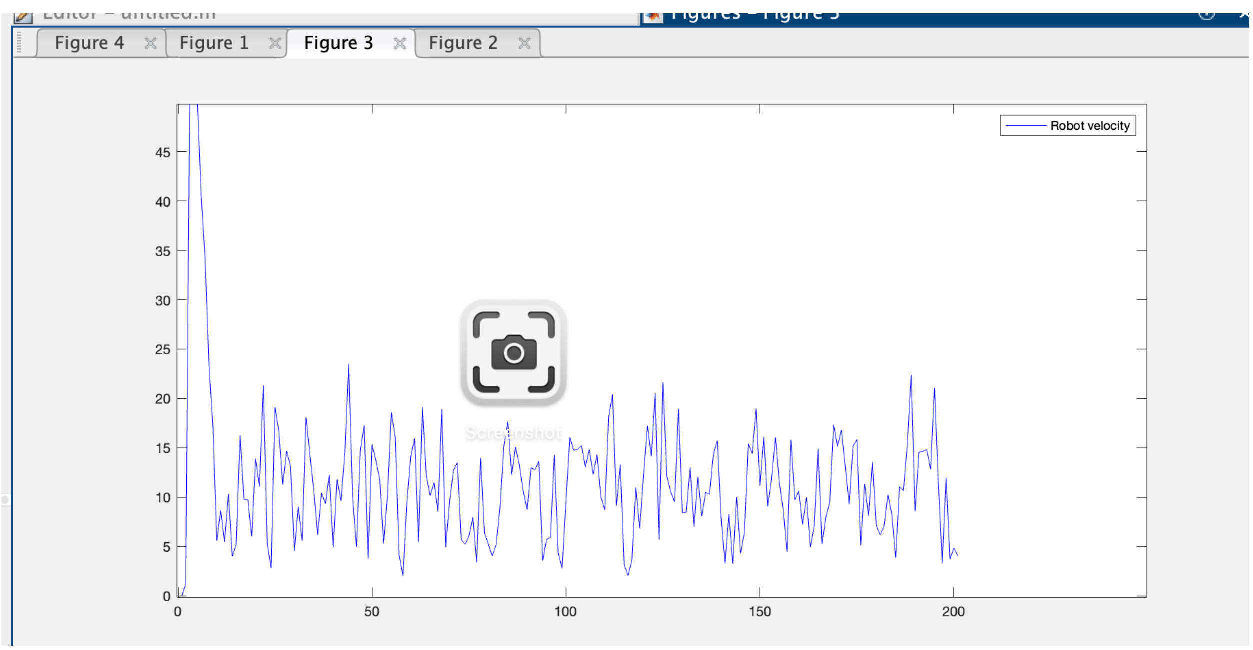


Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

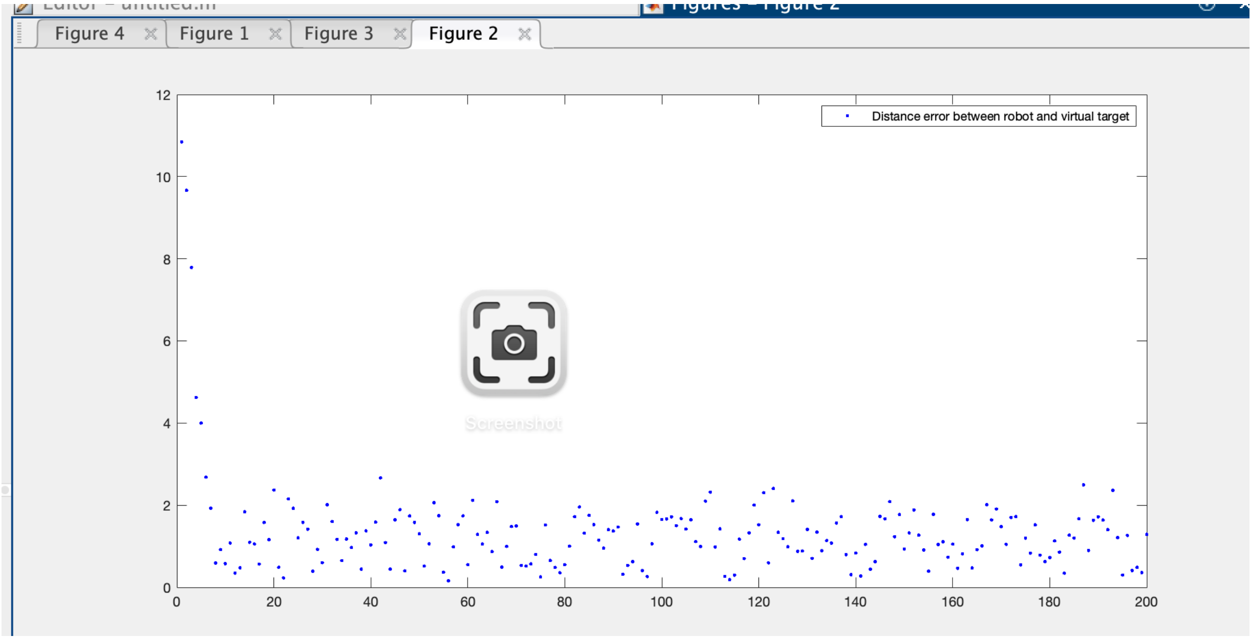
Sine:

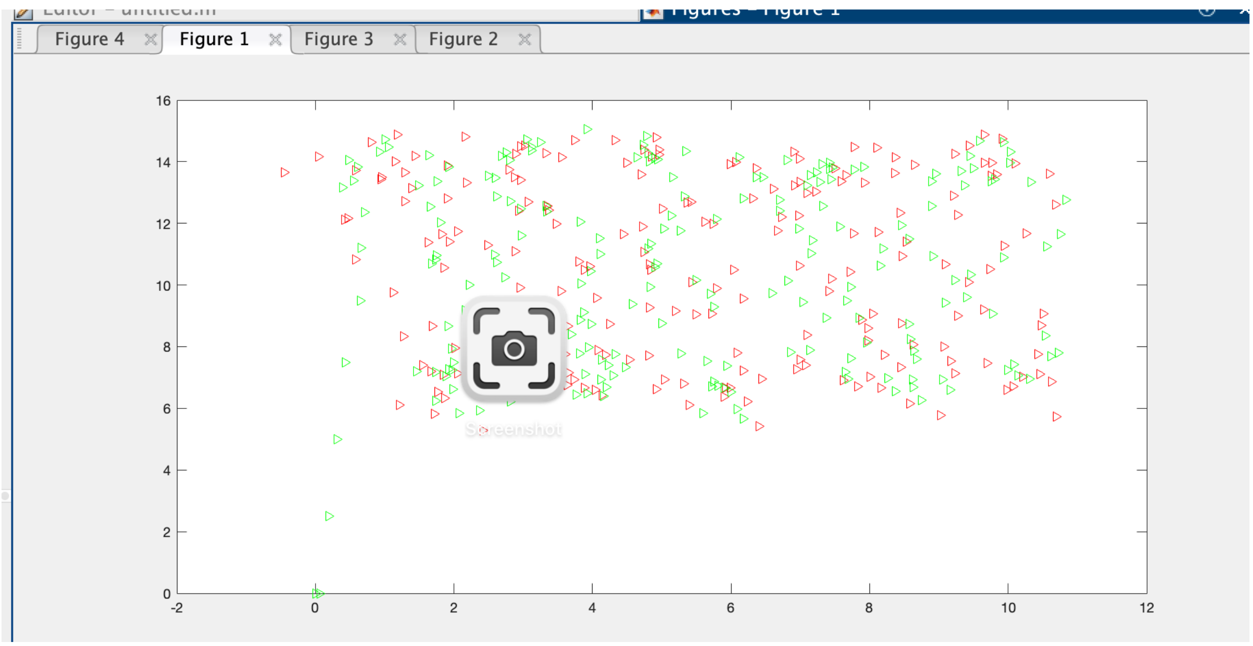
For the noisy Sine environment, the performance of the robot is similar to that of it's linear counterpart. The noise also affects the robot's path and cannot follow the robot as accurately in an environment without noise. It does sometimes get very close to the target as shown in figure 2 where the error is shown to be small.





Aaron Ramirez
CPE 470
Project 2
Dr. Jim La





Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

APPENDIX

- Most of the code is made by Dr. Jim La. The linear and sine path was produced by me as well as the equations for Phi, V_RD and Theta_R

```
% CPE470/670 Project 2: Potential Field Path Planning
% ========Set parameters for simulation============
clc,clear
close all
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
%==================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))];
%====Set noise mean and standard deviation====
noise_mean = 0.5;
noise_std = 0.5; %try 0.2 also
%=========MAIN PROGRAM=================
for i =2:length(t)
   %linear w/o noise
   %qv_x = t(i);
   %qv_y = t(i);
   %sine w/o noise
   %qv_x = t(i);
   %qv_y = sin(t(i));
   %linear noise
```

Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

```
    %qv_x = t(i)+ noise_std * randn + noise_mean;
    %qv_y = 4*t(i) + 10 + noise_std * randn + noise_mean;
    %Sine noisy
    qv_x = t(i) + noise_std * randn + noise_mean;
    qv_y = 4*sin(t(i) * 3) + 10 + noise_std * randn + noise_mean;
    %++++++++++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
      %Compute the target heading
    qv(i,:) = [qv_x, qv_y];
    qt_diff(i,:) =  qv(i,:)- qv(i-1,:);
    theta_t(i) = atan2(qt_diff(i,2),qt_diff(i,1));
...(Your code is here)%======UPDATE position and velocity of robot==========
    Phi(i) = atan2(qrv(i - 1,2), qrv(i - 1,1));
    term1 = pv^2;
    term2 = 2*lambda*norm(qrv(i - 1,:))*pv*abs(cos(theta_t(i)-Phi(i)));
    term3 = (norm(qrv(i - 1,:))*lambda)^2;
    v_rd(i) = sqrt(term1 + term2 + term3);
    if v_rd(i) >= vr_max
        v_rd(i) = vr_max;
    end
    nested = (pv*sin(theta_t(i) - Phi(i))/v_rd(i));
    theta_r(i) = Phi(i) + asin(nested);

    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
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')
```

Aaron Ramirez
CPE 470
Project 2
Dr. Jim La

```
figure(4), plot(theta_r, '--b')
hold on
plot(theta_t, '-.r')
hold on
plot(Phi, 'k')
legend('Robot orientation', 'Target orientation', 'Relative Orientation')
```