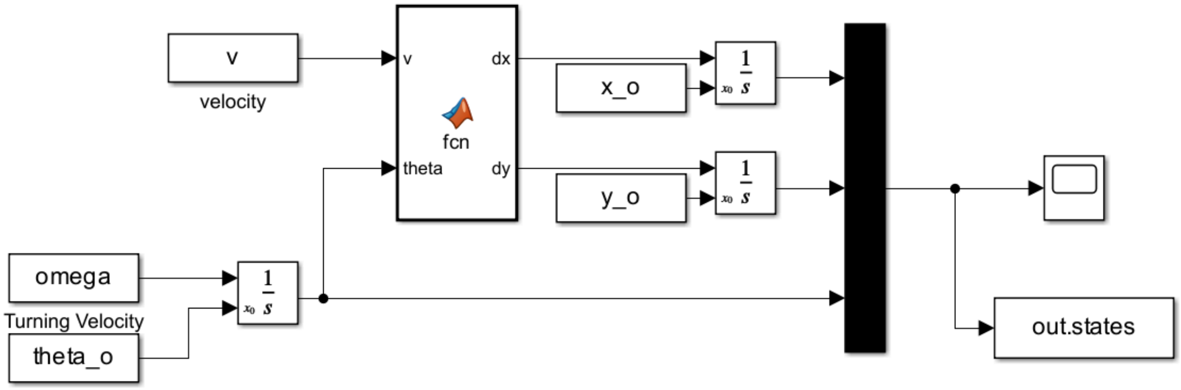
ME145 Robotic Planning and Kinematic: Lab 3

Line Following for Differential-Drive and Unicycle Robots

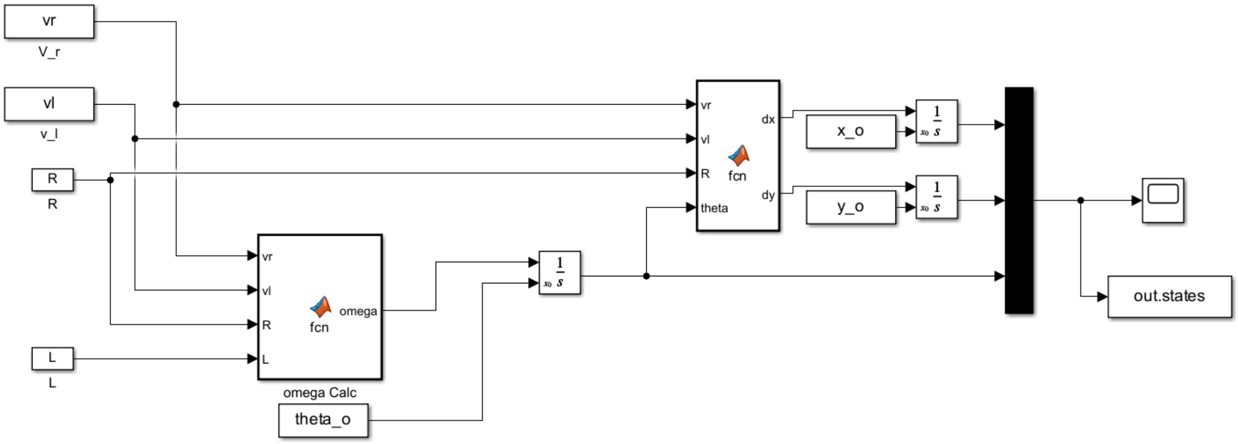
Elijah Perez

Winter 2025, Jan 5

Part 1 (Unicycle Robot Simulink)

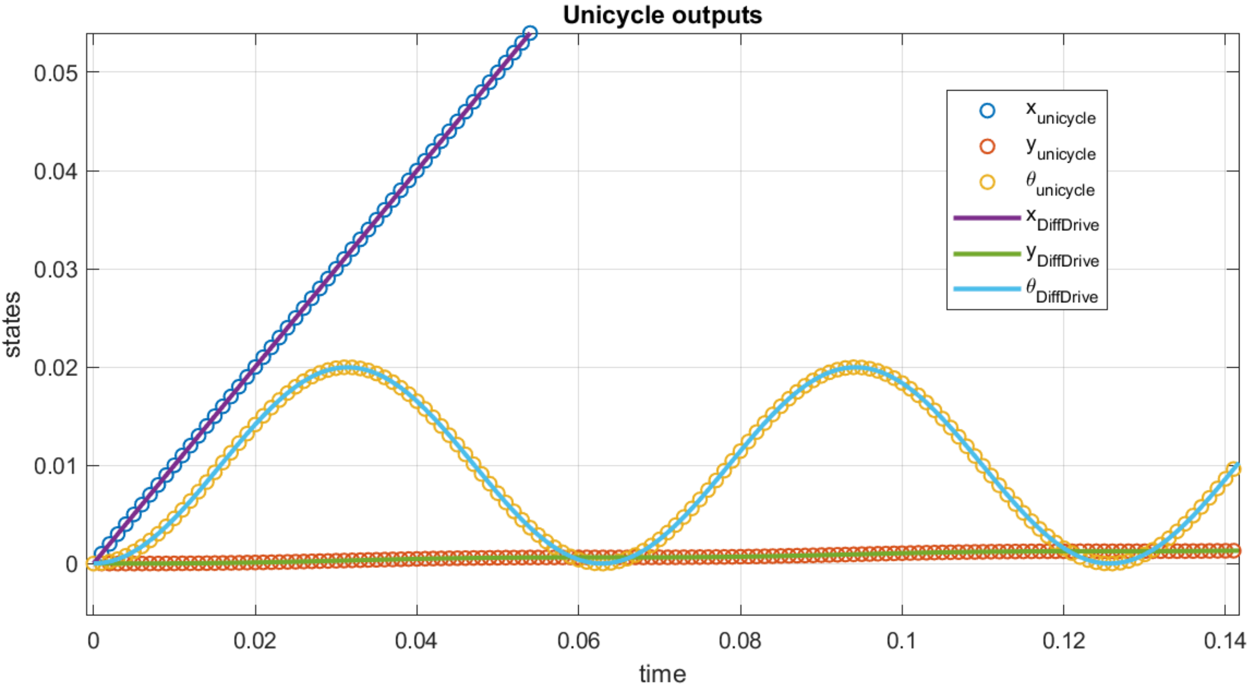


Part 2 (Diff Drive Robot Simulink)



Part 3(Comparing Results)

V = 1, Omega = sin(100t)



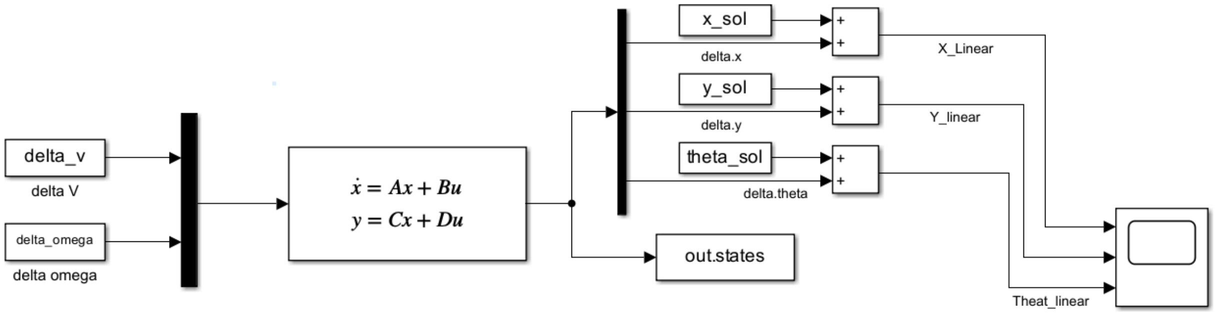
E

a

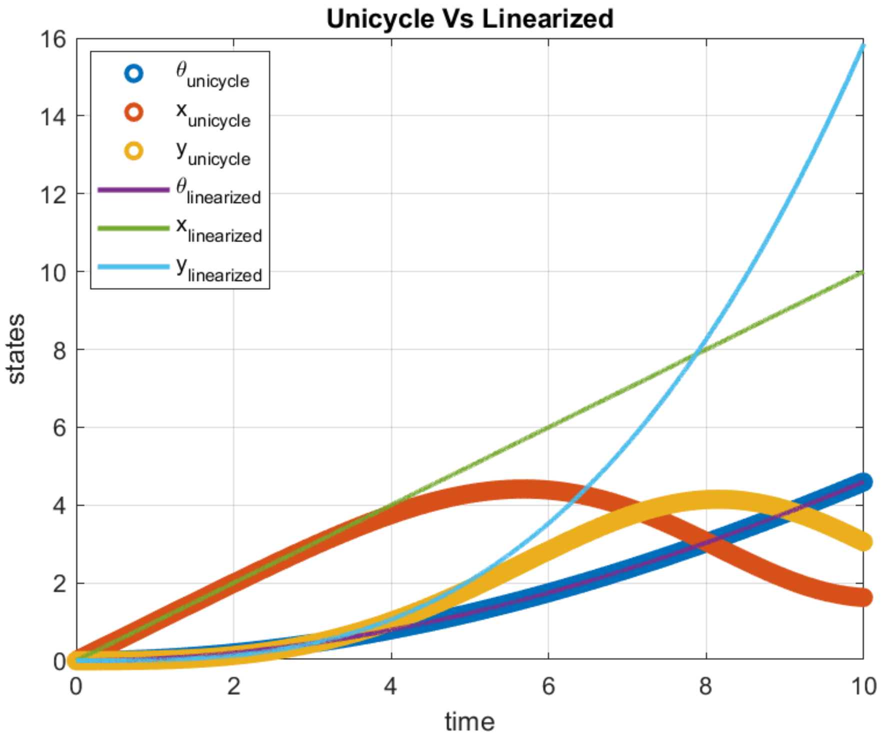
Explanation Part 2:
The Differential Drive Robot takes in input vr and vl. These are the velocity inputs at the right and left wheels. The other inputs are R and L, Where R = radius of the wheel and L = distance between wheels. The Differential drives take these inputs and convert them to a unicycle model using the transformation equations given in the lecture. The output is the states (x,y,theta).

Explanation Part 3: The system was given inputs of V = 1 and Omega = 100t and R = 0.5 and L = 0.1 for the differential. These inputs where used for both the differential drive model and the unicycle model. As we can see the results are approximately the same.

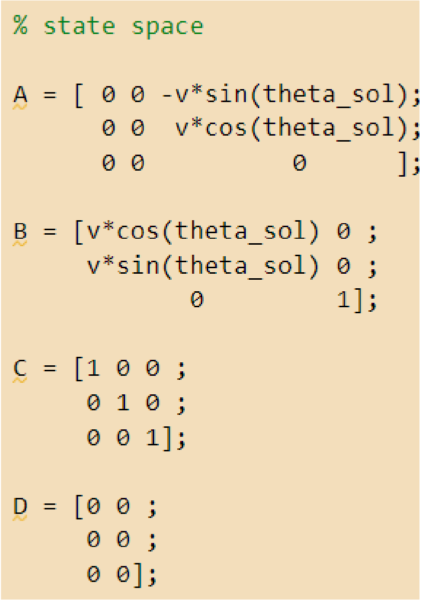
Part 4 (Linearized Unicycle)



Part 5 (Comparison Linearized vs Normal)



Explanation Part 4:
Since the model is Linearized for a straight-line trajectory, I thought it would be easy to model the linear system in state space since its already linearized. The state space values are shown below:



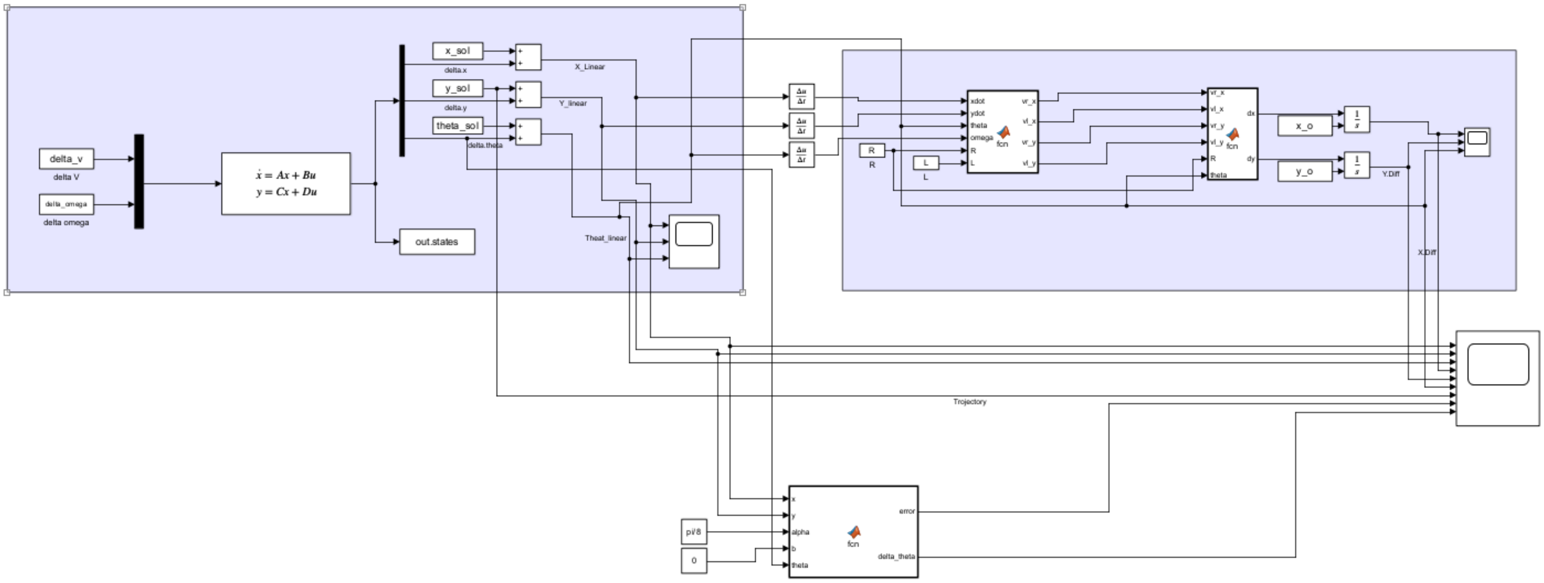
```
% state space

A = [ 0 0 -v*sin(theta_sol);
      0 0  v*cos(theta_sol);
      0 0         0       ];

B = [v*cos(theta_sol) 0 ;
     v*sin(theta_sol) 0 ;
            0         1];

C = [1 0 0 ;
     0 1 0 ;
     0 0 1];

D = [0 0 ;
     0 0 ;
     0 0];
```

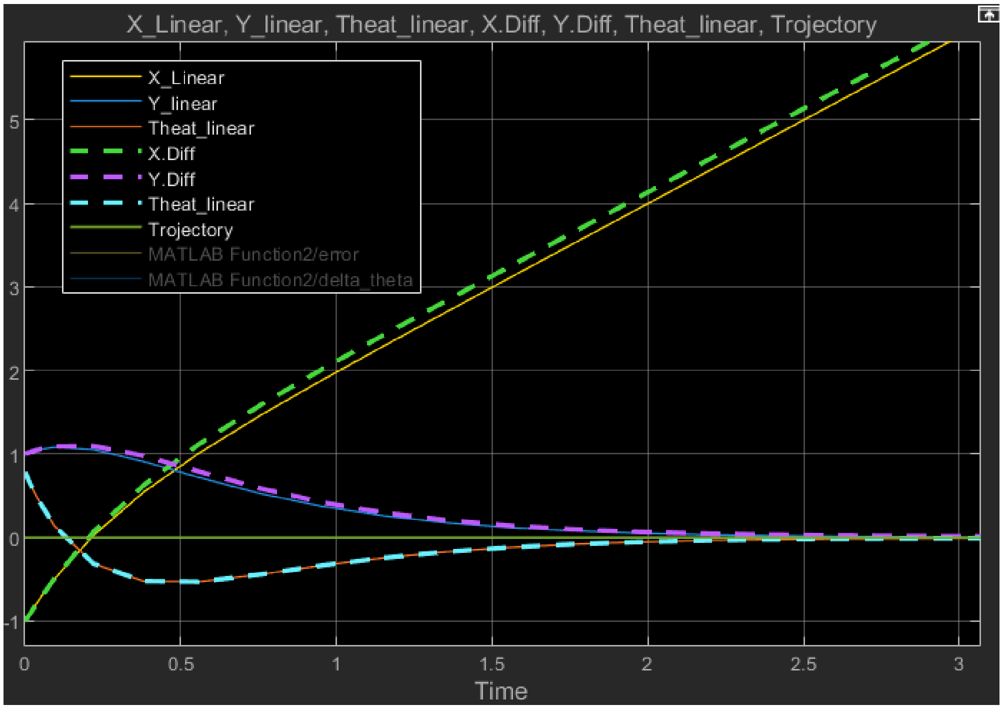
The system still takes inputs such as velocity and omega. Instead, now it is solved using an ODE solver.

Explanation Part 5:
As we can see, for small incremental changes, the linearized and nonlinear unicycle models appear to be the same. But for larger theta values, the positions of x and y start to deviate from the nonlinear model. For this comparison the inputs where: v =1 and theta = 0.1*t.

(Also I know it says plot using scope but since they are two different simulink files I can't overlay the plot and I thought it looks better to overlay them. So I exported the data to the workspace for the plots.)

Part 6 (Control unicycle robot and Diff Drive)

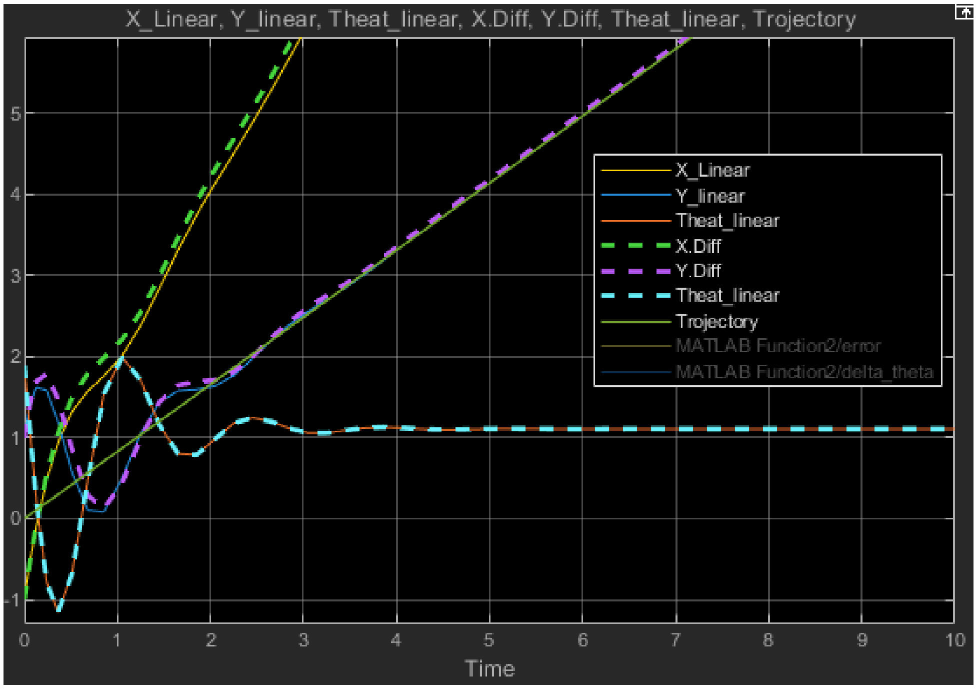


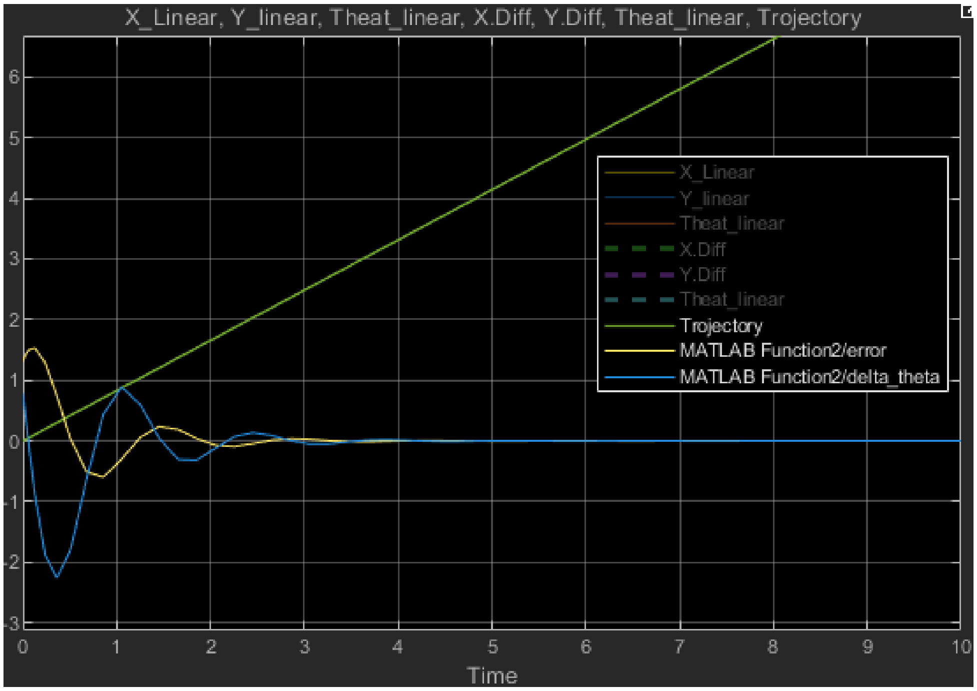


Explanation Part 6:
The system is linearized about a straight line: x = v*t,y = 0, theta = 0. The system received the initial condition of [x_i = -1, y_i = 1, theta_o = pi/4]. Through the linear model was combined with the differential drive model so that we can use the states of the linear model to control the differential drive. The state space model is shown above. The first block is the linearized unicycle model. The second block contains a workspace of the differential drive. In between the blocks are conversion and input calculations for the differential drive.

Looking at the graph the response appears to be the same, but as I looked closer I noticed the theta state from the linearized model does not perfectly match the theta from the differential drive. This can be due to multiple integrations and derivatives. Maybe a piece of information was lost in the calculations. The result of this small deviation can be seen in the limit of the x and y positions. The Theta and Y states approach zero as time goes on. This is what we want since the trajectory is a straight line on the x-axis. The x position is a function of velocity and time so it makes sense that it increases linearly as time goes on.

(Part 7)


X_Linear, Y_linear, Theat_linear, X.Diff, Y.Diff, Theat_linear, Trojectory
X_Linear
Y_linear
Theat_linear
X.Diff
Y.Diff
Theat_linear
Trojectory
MATLAB Function2/error
MATLAB Function2/delta_theta
-1
0
1
2
3
4
5
0
1
2
3
4
5
6
7
8
9
10
Time


X_Linear, Y_linear, Theat_linear, X.Diff, Y.Diff, Theat_linear, Trojectory
X_Linear
Y_linear
Theat_linear
X.Diff
Y.Diff
Theat_linear
Trojectory
MATLAB Function2/error
MATLAB Function2/delta_theta
-3
-2
-1
0
1
2
3
4
5
6
0
1
2
3
4
5
6
7
8
9
10
Time

Explanation Part 7:

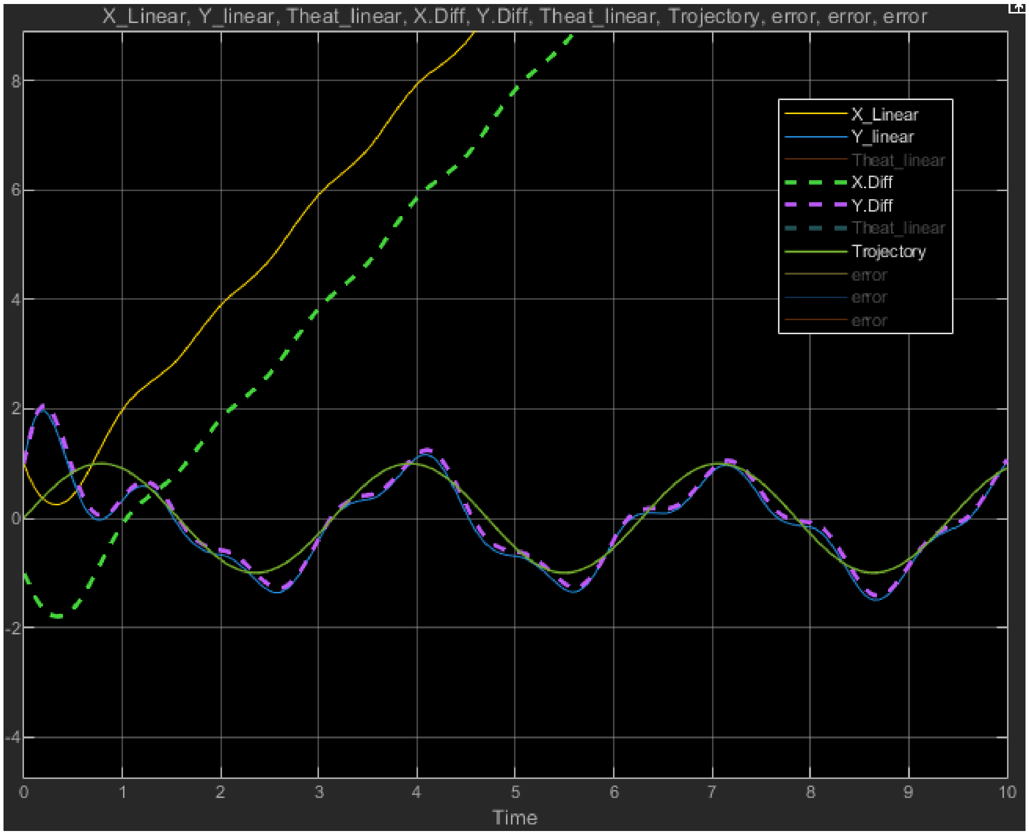
The line Trajectory line I chose is defined by the equation y = tan(alpha) x + b. In my case, I selected alpha to be pi/8 and b = 0 for simplicity. The Trajectory can be seen in green in both screenshots above. Both of the screenshots are the same, just with filtered-out signals for aesthetics. The error was calculated using the equations given: Tracking error = -x sin(alpha)+y cos(alpha)-b cos(alpha), the angle error = theta - alpha.

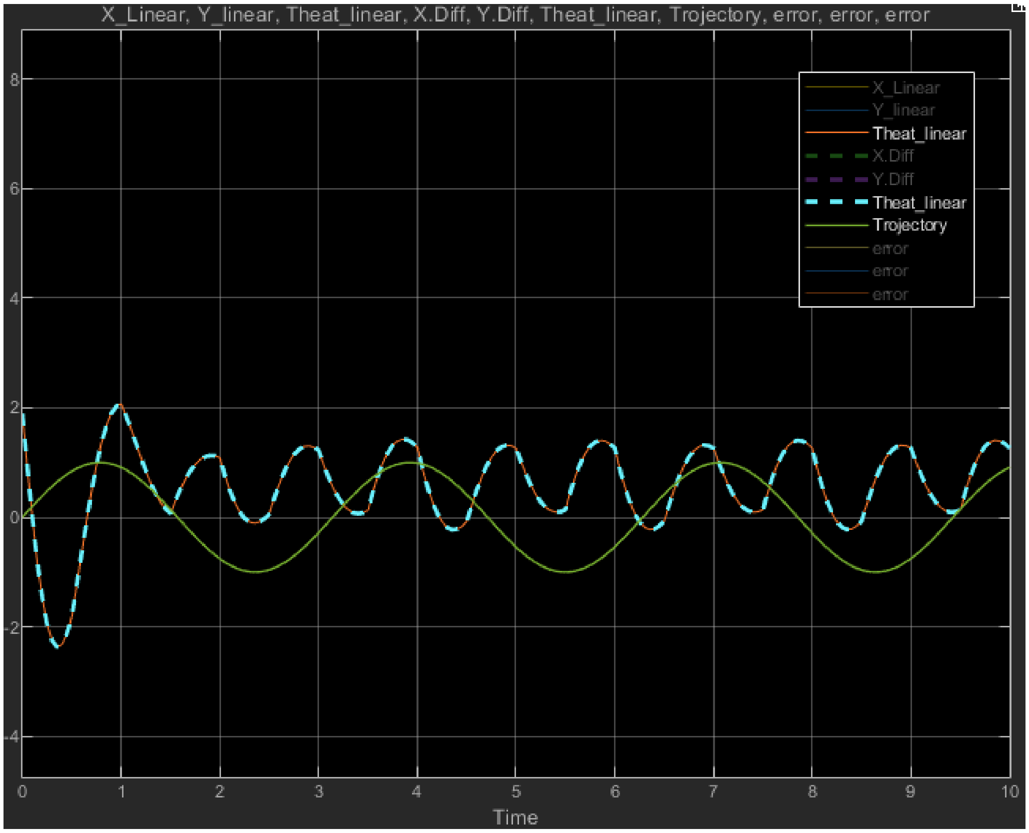
The top graph shows the trajectory of the states from both the linearized unicycle and the differential drive. We can see both theta states find a steady state at about 1 ish as time goes on. This is the calculated angle needed to track the desired path. We Can see x is still following its desired path with a velocity of v = 2 m/s. The y trajectory is of most interest as we can se how the system tries to follow the Trajectory. We see at first the trajectory is unstable but as time goes on it reaches a steady state and tracks the trajectory really well.

The second image shows the tracking error and angle error. We can see when the states are not at steady state the error calculates are very large. And once the system entries start to state the eros decrease to almost zero as time goes son. The IC = [x = -1, y = 1, theta = pi/4].

Also, the Simulink model is the same as from part 6.

(Part 8)


X_Linear, Y_linear, Theat_linear, X.Diff, Y.Diff, Theat_linear, Trojectory, error, error, error
X_Linear
Y_linear
Theat_linear
X.Diff
Y.Diff
Theat_linear
Trojectory
error
error
error
8
6
4
2
0
-2
-4
0
1
2
3
4
5
6
7
8
9
10
Time


X_Linear, Y_linear, Theat_linear, X.Diff, Y.Diff, Theat_linear, Trojectory, error, error, error
X_Linear
Y_linear
Theat_linear
X.Diff
Y.Diff
Theat_linear
Trojectory
error
error
error
8
6
4
2
0
-2
-4
0
1
2
3
4
5
6
7
8
9
10
Time

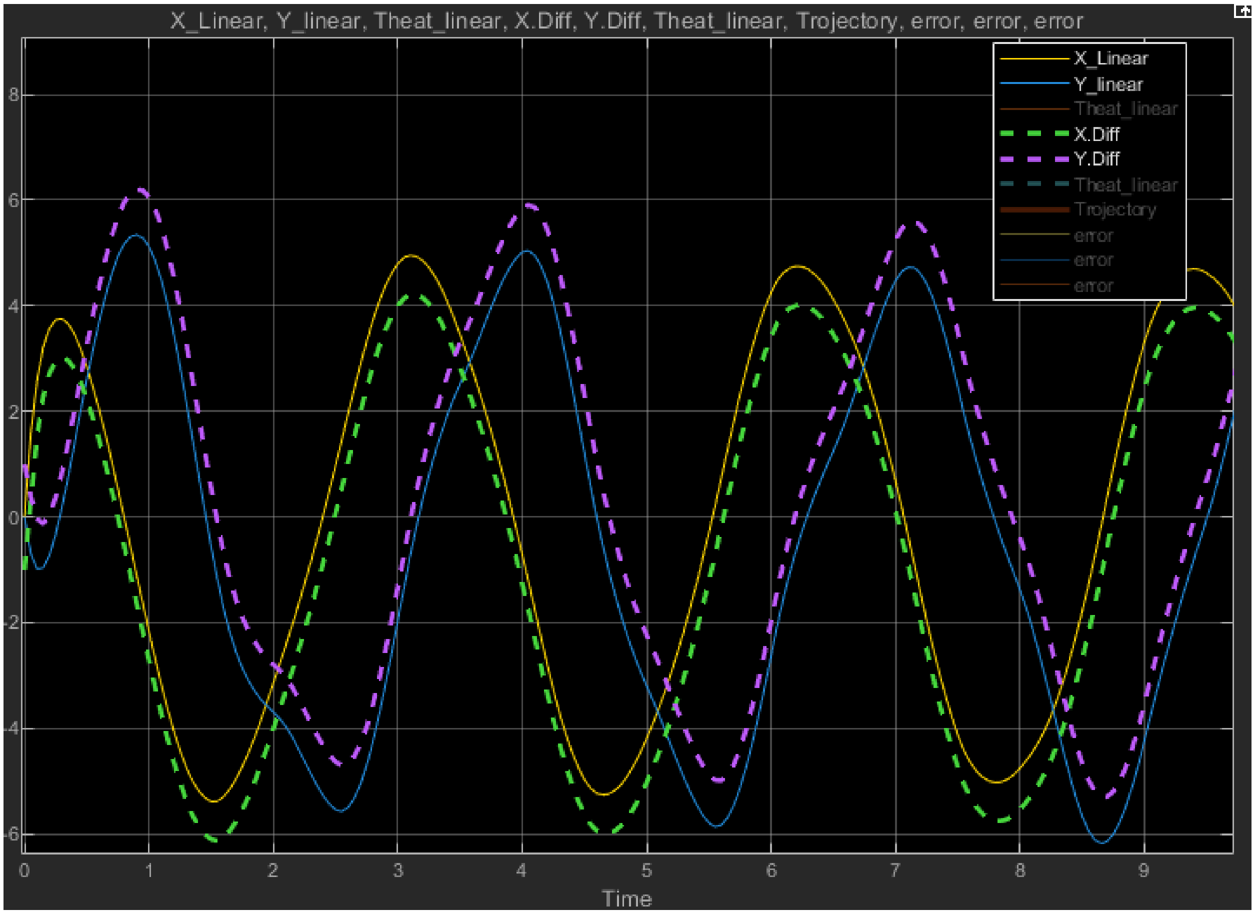
Explanation Part 8:

As we can see the two images are the same figure just with signals cut out for more clarity. The time graph shows the trajectory line and the x and y position of both the diff drive and linearized model. The second graph shows the trajectory line and the theta state of both the linearized model and diff drive. We can see in the first image that the robot tries its best to keep itself aligned with the sin wave, but there are significant errors. This can be primarily due to the controller design chosen. My engine values have imaginary numbers, making the system response have more overshoots. It also is interesting to see that values go crazy trying to stabilize the system's reasons for error.

Also, I am not sure why the trajectory of the x values between the two systems is off exactly by 2.

The Simulink model is a similar model to Part 6.

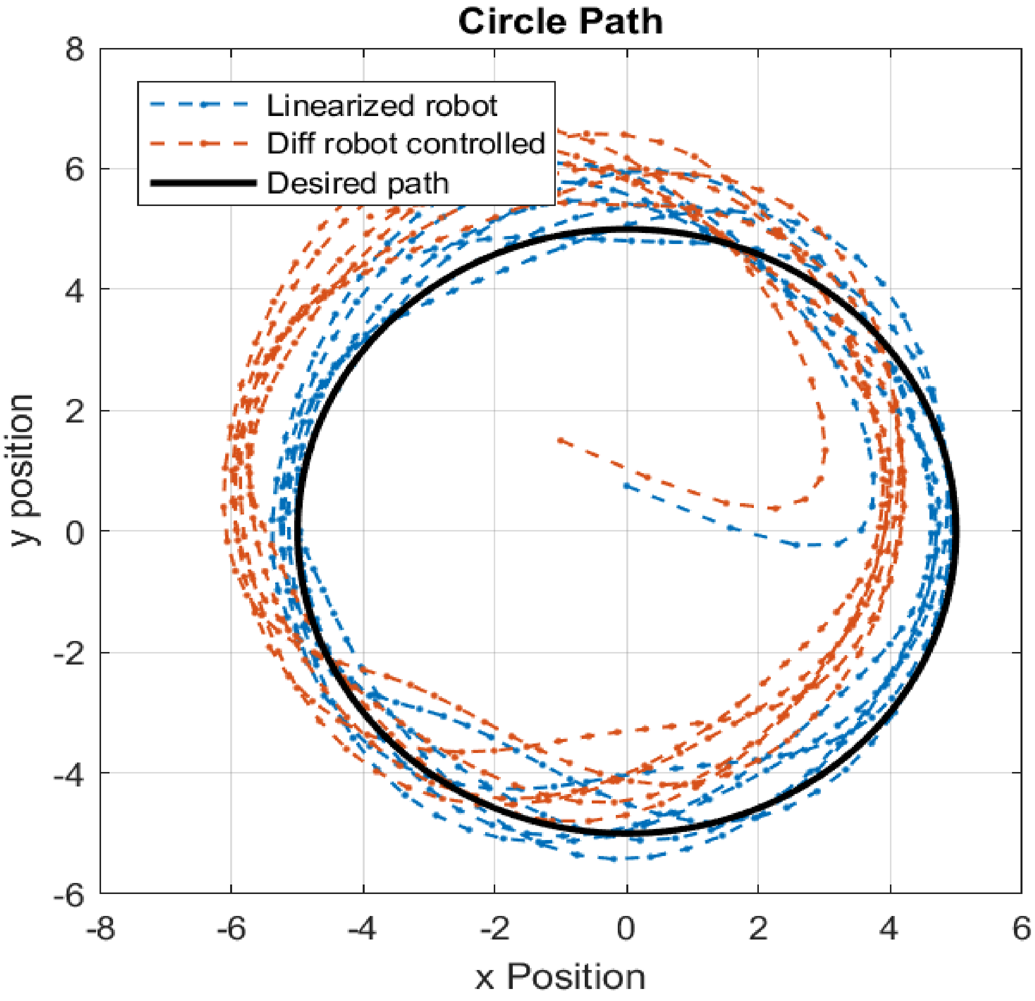
(Part 9)



Explanation Part 9: Modeling a circle will approximately output a sine wave and a cos wave on a position vs time axis. We can see both systems try to follow the circle trajectory but output wonky-looking sin and cos waves as they try to follow the circle. The IC = [x = 0, y= 0, theta = pi/4]. If we want to see the evolution of both x and why position as a function of time. Then we must put time in the z-axis and x and y positions on the planar axis. I exported the data and outputted the results in a Matlab script so we could see how it actually looked.

Looking at the Matlab figure we can see the linearized robot and diff robot actually do follow the circle decently alright. Since the system never reaches a steady state there will always be some error throughout the entire trajectory.

I was having the same issue from last time where the Diff robots x position was shifted by two. I am not sure this


Circle Path
Linearized robot
Diff robot controlled
Desired path
y position
x Position
8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6