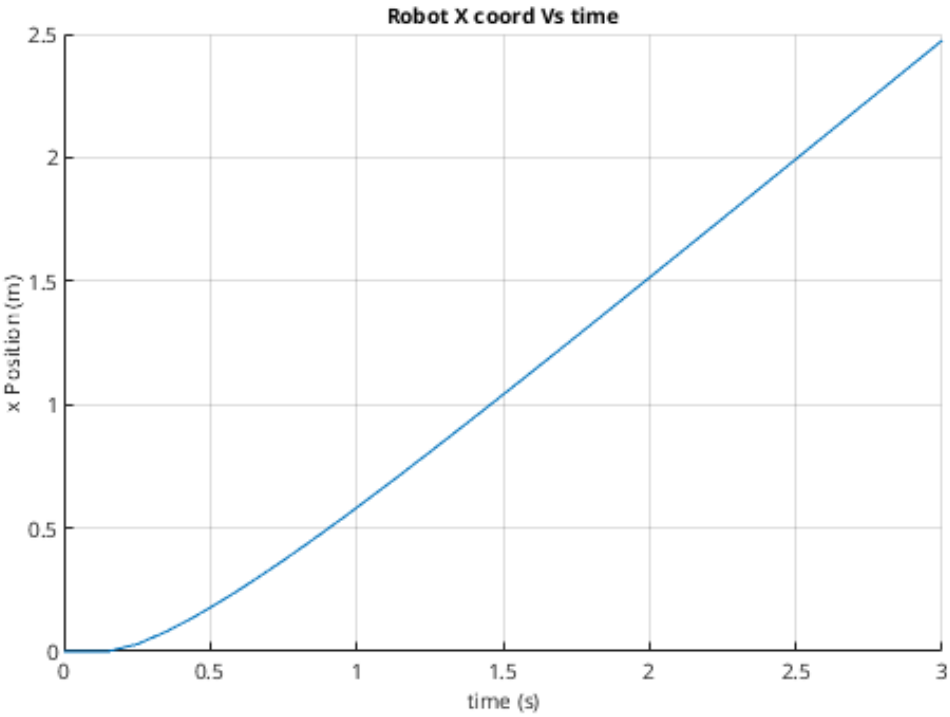
Lab 2 - Introduction to Model – Answer Grid 2

The following should be used as a structure for presenting the requested information. This structure can then be copied into an Appendix of the main assignment report.

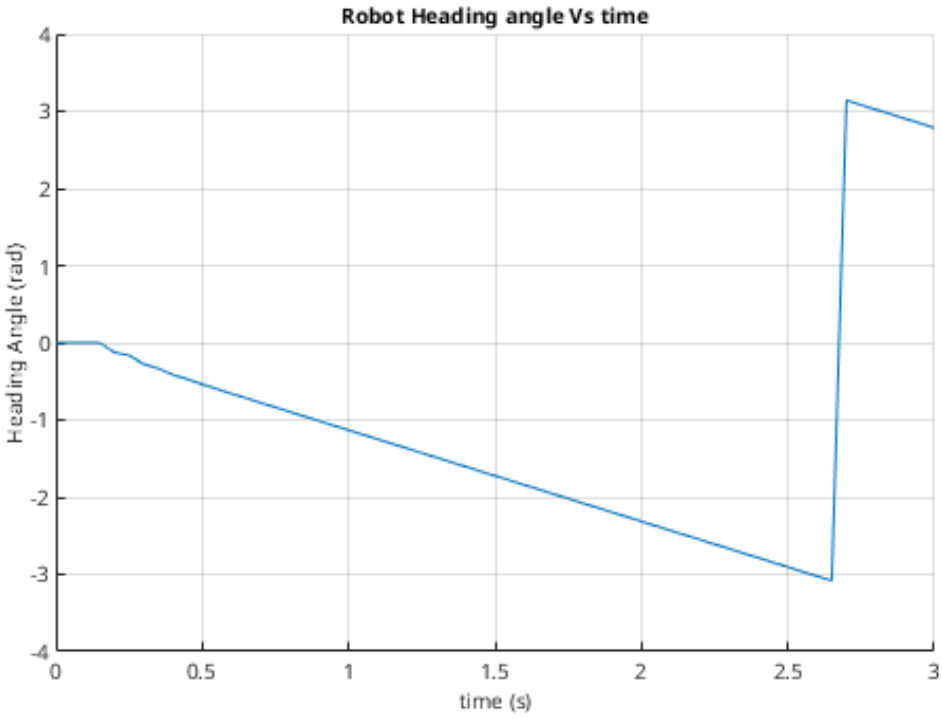
1. *Basic Operation*

The following data shows that you have got the simulation working as expected.

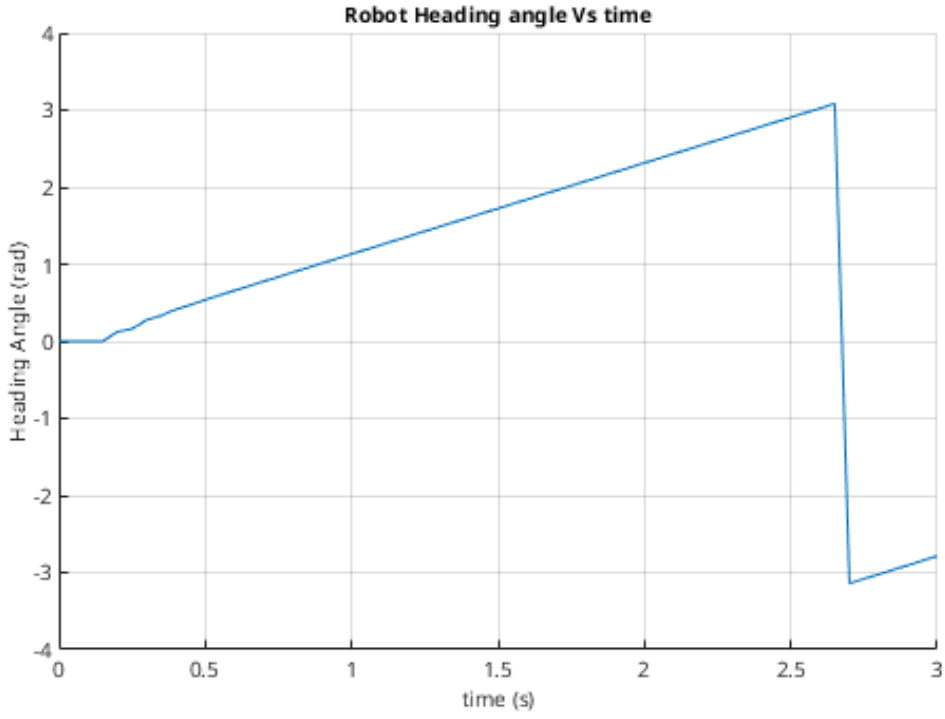
- *Insert a plot for each of the following:*
  - *Straight Line motion*



    - *Insert a plot of the x distance travelled.*
  - *Turn counterclockwise*
    - *Insert a plot of the psi angle.*

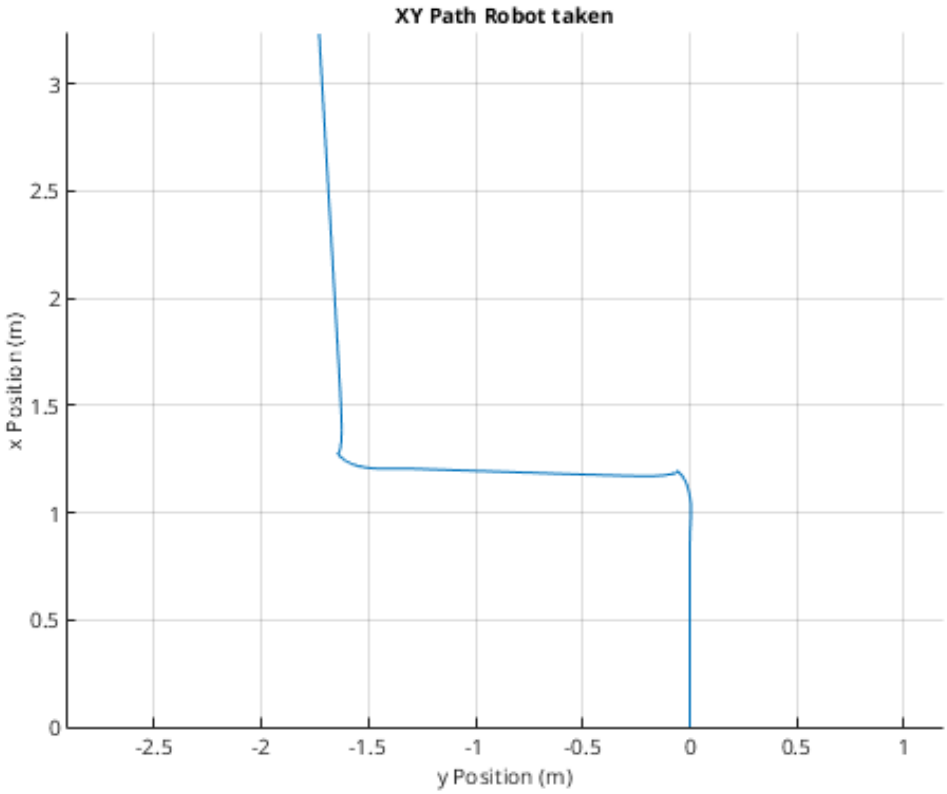


- *Turn Clockwise*
  - *Insert a plot of the psi angle.*



- *Complete a short forward, turn left, forward, turn right*

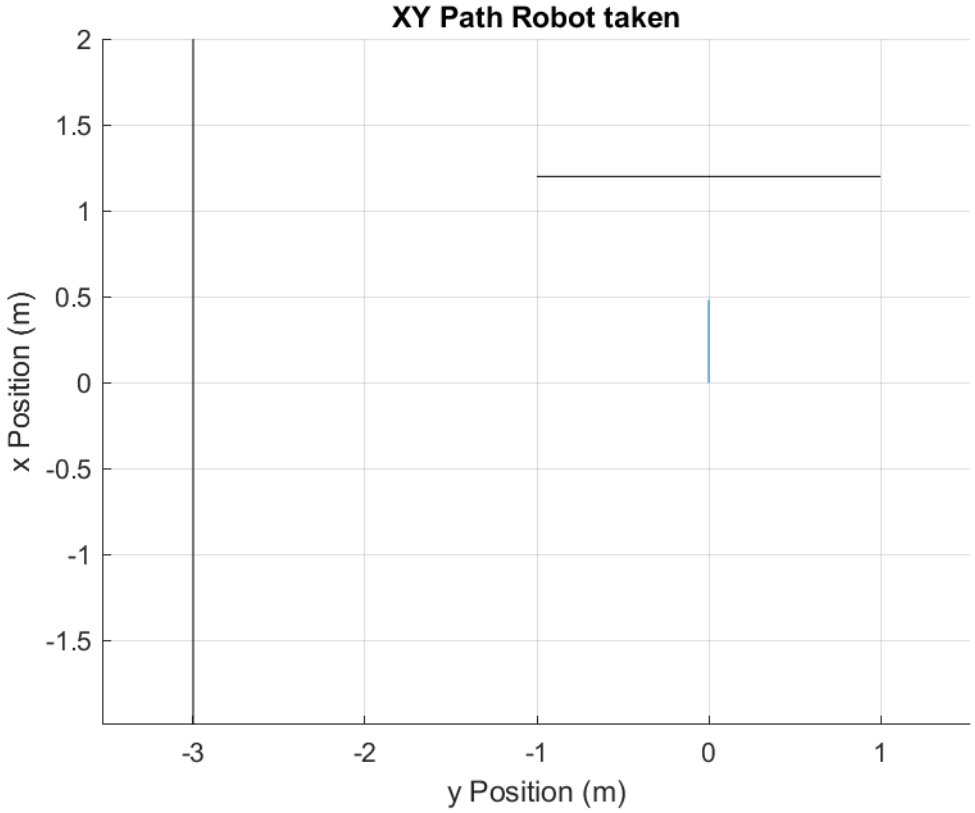
- *Insert a plot of the x/y position*



2. *Running the Model with a Fuzzy Controller*

The following data shows that you have got a basic Fuzzy controller working with the model.

- *Provide a plot of the path of the system using the tutorial example.*



- *Comment on the behaviour of the system*

There were lots of oscillation in the original controller design it would just oscillated in a strait line when it hit a wall from directly front on.

- *Comment on any changes you would make to improve the performance of the system*

To compensate for this osculation I first added a bias term to the left motor membership function as shown below:

colisCont = addMF(colisCont,"distL","trapmf", [-10 -10 0.25+bias 0.45+bias],'Name',"TooClose");

colisCont = addMF(colisCont,"distL","trapmf", [0.2+bias 0.4+bias 0.5+bias 0.7+bias],'Name',"Close");

colisCont = addMF(colisCont,"distL","trapmf", [0.5+bias 0.9+bias 1.1 2],'Name',"Near");

With the bias set to 0.1, this had the effect of the robot preferentially turning left, but the oscillations still remained.

I first tried to remove the oscillation terms by "smoothing out" the transitions between membership functions so they weren't to steep, this did very little to change the behaviour however.

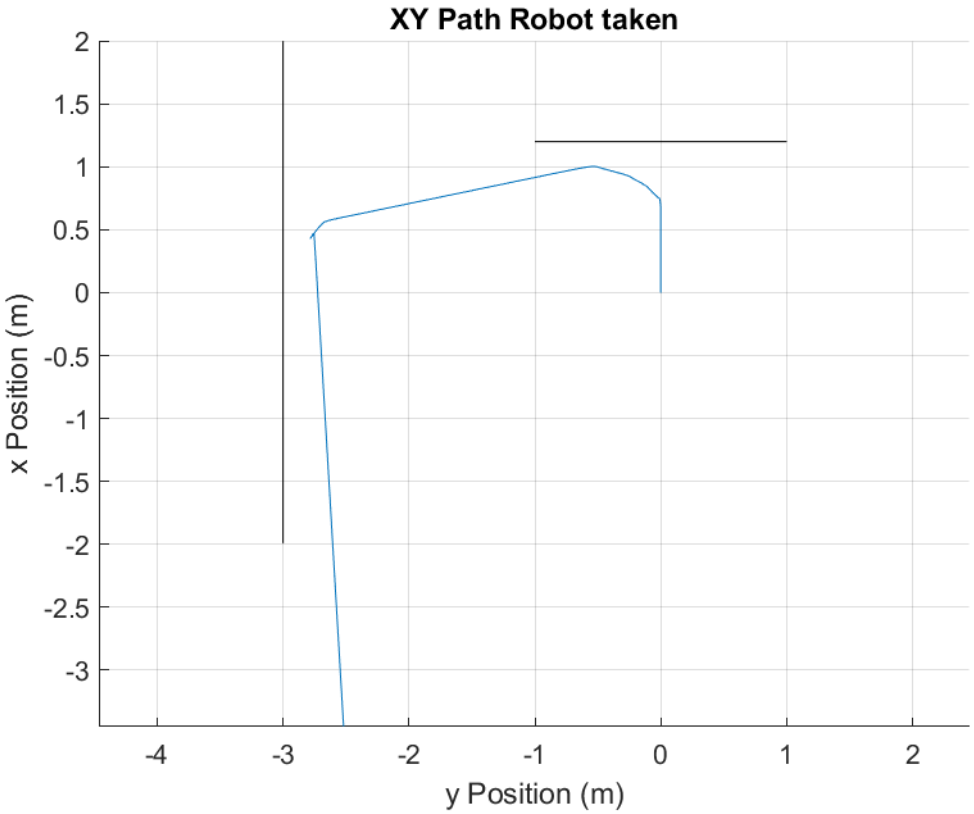
I was suspicious because it seemed that the robot seemed to drive strait until it hit a wall and then immediately started reversing, while I expected it to begin smoothly turning due the bias terms. I dug a little into the rules and found the following suspicious rule:

colisCont = addRule(colisCont,"distL==Close & distR==Close => powerL=Rev, powerR=Rev (1)");

After commenting this out the robot began working without any oscitation as can be seen below:

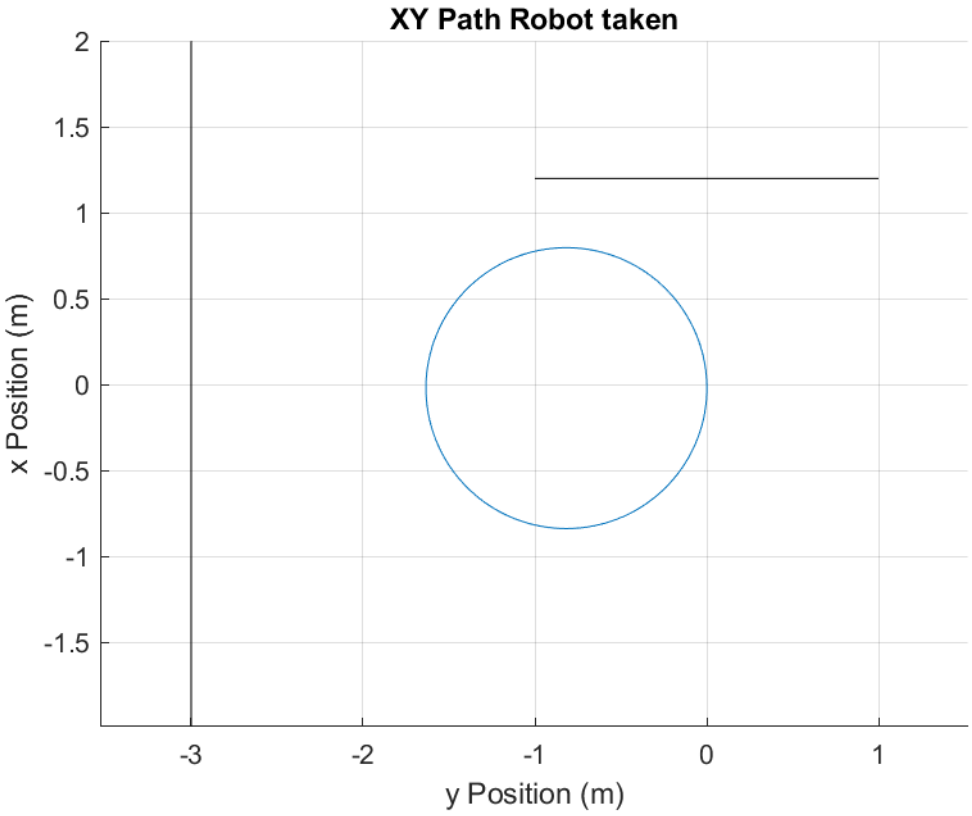
- *Provide a plot of the path of the system with the changes you have implemented*

3.



The following data shows that you have got a basic Neural Network controller working with the model.

- *Provide a plot of the path of the system using the tutorial example.*



- *Briefly comment on the behaviour of the system*

Using the base parameters and the neurons threshold activation function, at max sensor distance (1) the robot still has one wheel going forward and one stationary meaning it turns in a circle.

- *Comment on any changes you would make to improve the performance of the system*

Though not specified, the activation function for the neural net is assumed to be a simple threshold meaning that the output of the net may only be a one or a zero. this means that any motor may only be in one of two states. these could be:

- forward or off

- forward and backward

It is possible to implement a controller for either configuration, however the first option is substantially more challenging this is what i cose to implement and describe here.

*** Design of optimal Gains and Weights

It is worth noting that the limited number of neurons in the network impose fundamental geometric limitations on the logical rules that may be implement. If one considers the inputs to the robot to be a two dimensional state-space, it can be shown that any combination two

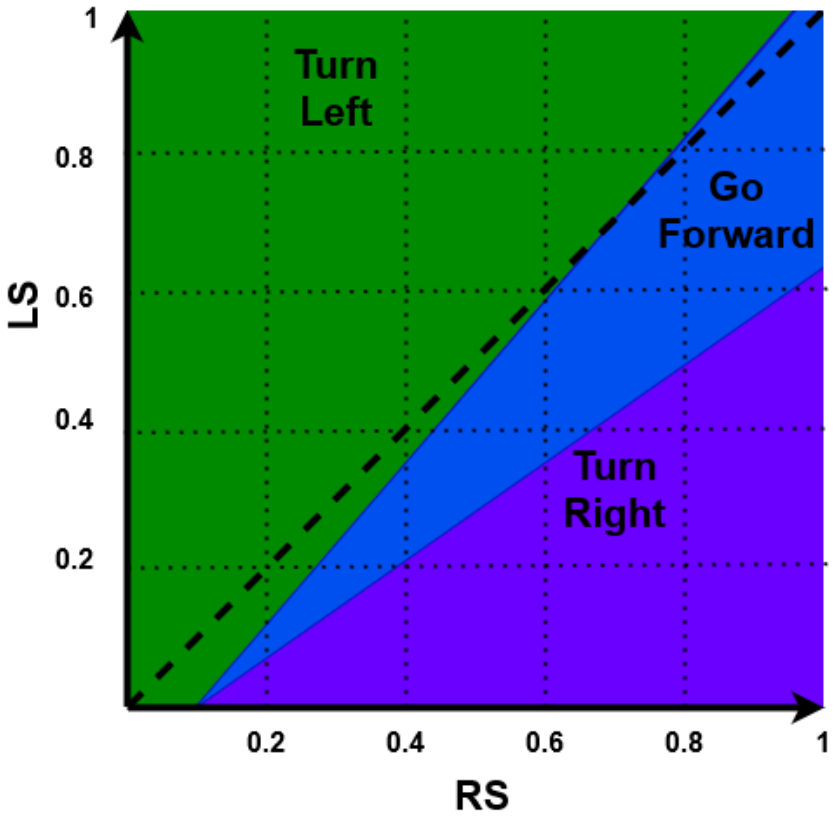
weights and biases must constitute a strait line bisecting the state-space. Depending on the output "wiring" of the neural net to the motors the area enclosed by one of these lines will mean either full forward or reverse/stop for the output of a given wheel as a function of the input states. When the lines for each wheel are drawn on the state-space the intersection of the areas where the wheels are moving forward will constitute forward motion for the robot as a whole, while the areas were both are zero will be either stationary or in full reverse depending on what the output of logical zero from the neural net maps to.

A basic fundamental requirement of the robot be that it should go in a strait line when there are no obstacles in front of it. Since the sensor saturate at distance >1. At minimum, the only thing that must be satisfied in the state-space is that the infinitesimal region around Rs=1 and LS=1 be an intersection of the two regions corresponding to forward motion of each wheel.

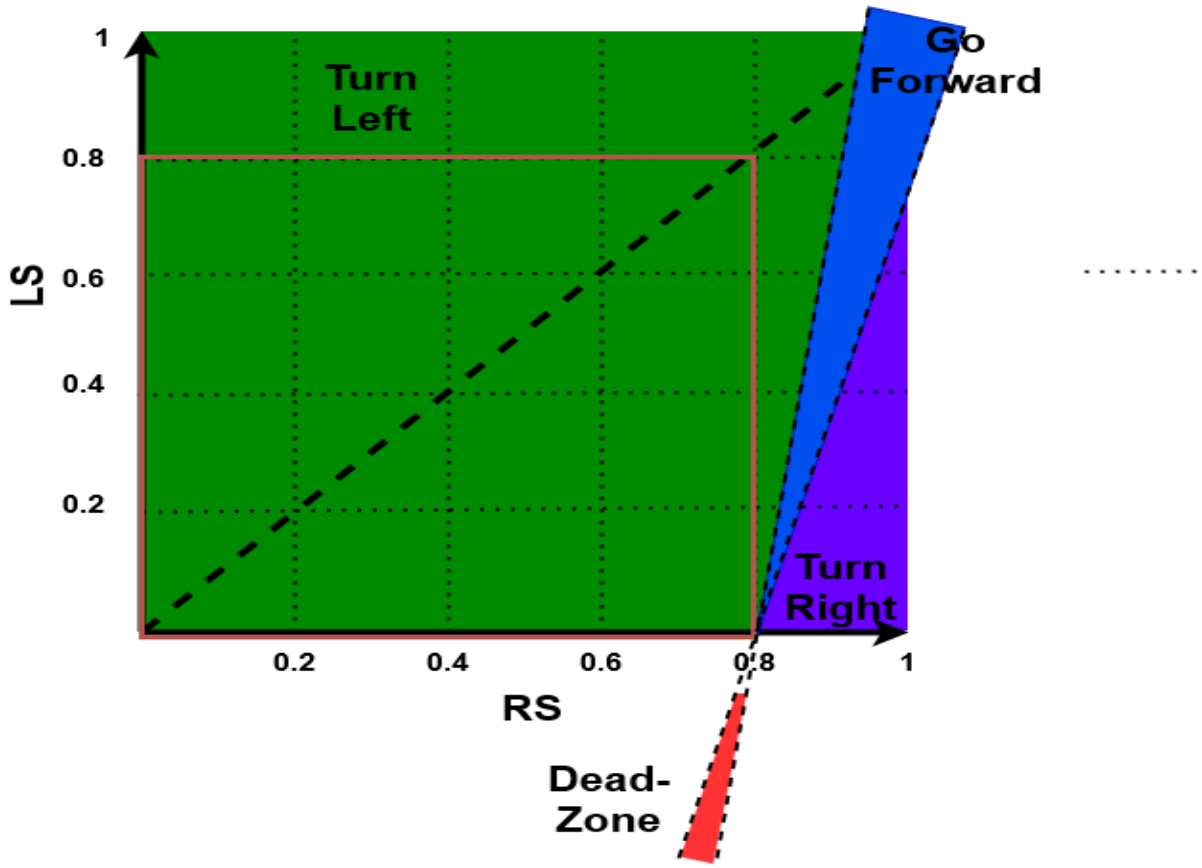
The second constraint relates to the turning circle of the robot, from testing, it was observed that the turning circle with one wheel stationary, is ~0.8, this means that the robot must begin turning almost as soon as the object is in sensory range. This imposes a second geometric constraint, that turning begin out-with the square drawn from the origin to the point LS=0.8,RS=0.8. Stated another way 'going forward' region where LW=1&RW=1 (shown in blue) must not cross into the region, LS=<0.8|RS=<0.8 (in the figure below it does).

A final consideration that in the case of the two selected wheel states being forward and stationary, there exists the possibility for "dead-zones" to exist where both wheels are zero. Once the robot crosses into such a region it is impossible to get out. This is another example where the choice to go with forward and backward wheel activation is superior! One can show that such a region is an inevitable outcome of any overlap in the two regions, just from basic geometry. Restated this means that if we want the robot to move forward, then by definition we must accept a dead-zone. We do however have the freedom to chose where this dead-zone appears. In the figure below the 'go forward' region has been chosen such that the dead-zone is on the negative axis, values which we know the sensor will never take.

0.4



With these geometric constraints in place, and with the understanding that the RW=1,LW=1 region may only be defined by strait lined due to the architecture of the network, the following geometry is proposed:



From the equations of the lines bounding the two regions, it is trivial to derive the corresponding weights and thresholds.

- *Briefly provide a plot of the path of the system with the changes you have implemented*