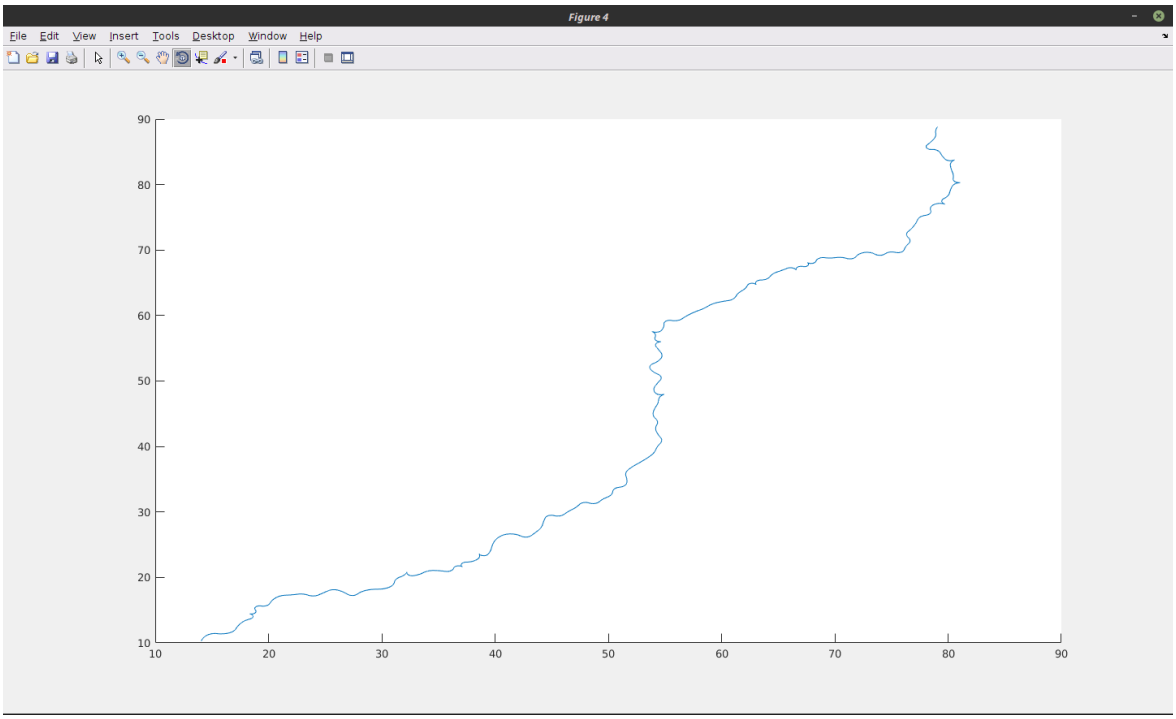
CS685 : Autonomous Robotics Submissions

Brinston Gonsalves

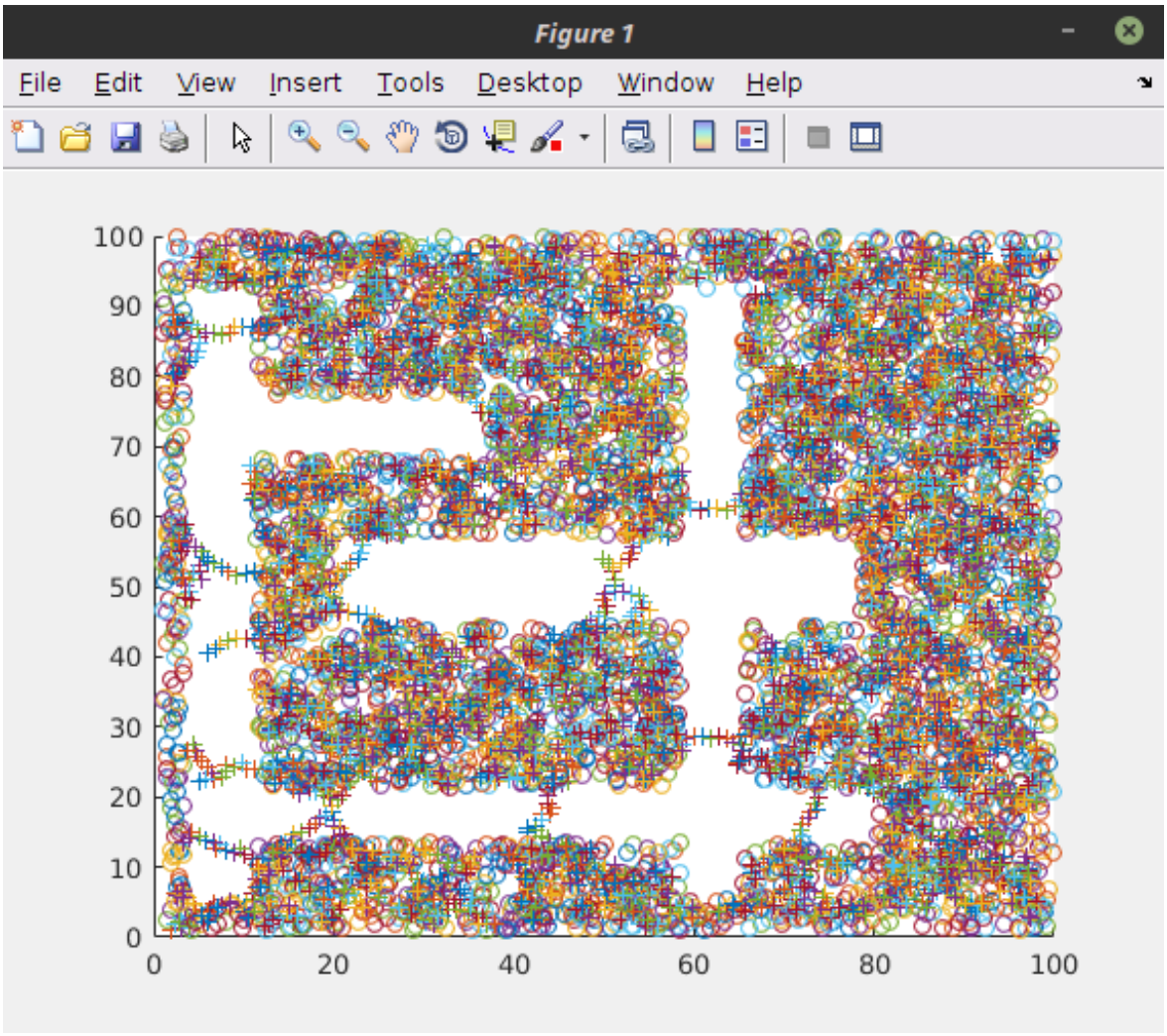
Homework 3

Question 1:

Plots : Click image to view full size.

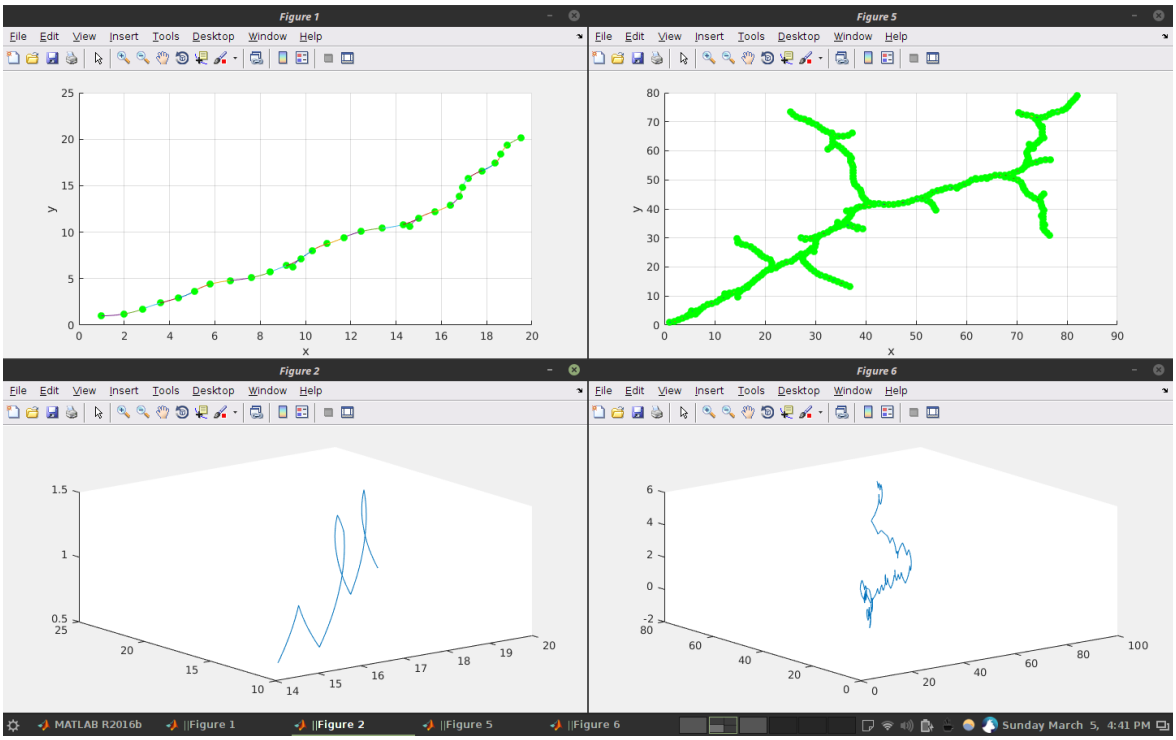


The generated intermediate plot or the given map:

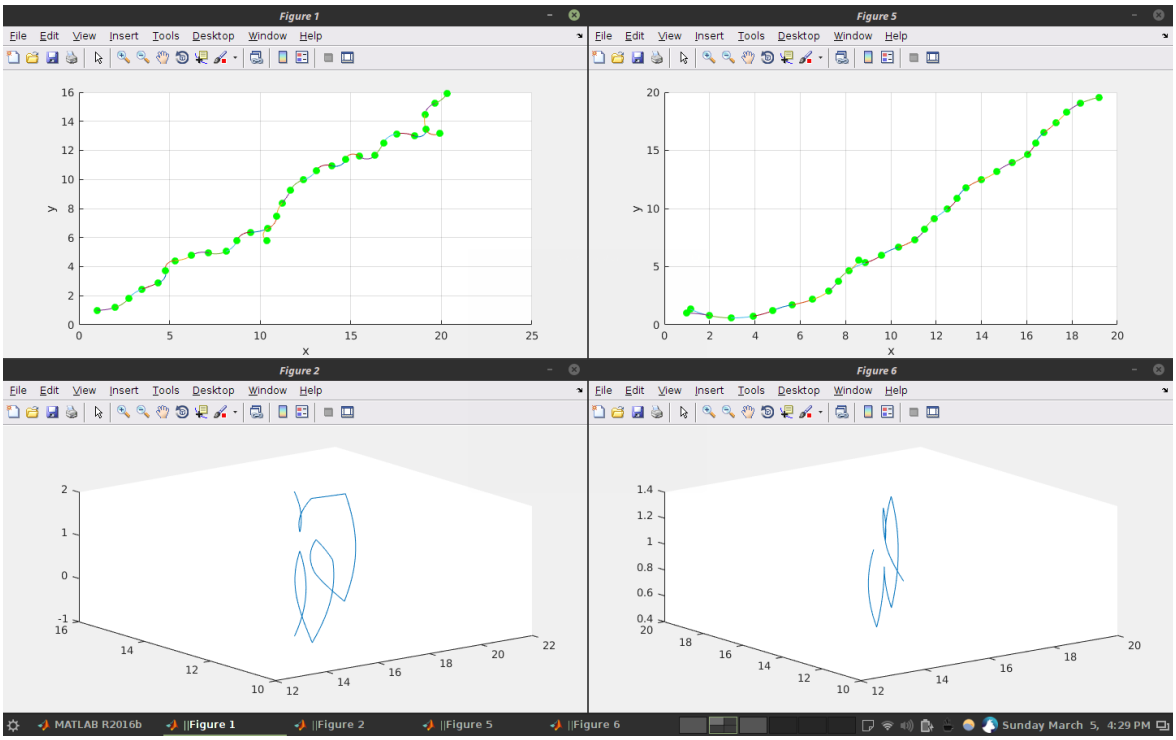


Question 2: Here we see that, as the no of points increases, more areas are being discovered.
We set the no of points using rrt = RRT(map, 'xrange', [1 100], 'yrange', [1 100], 'npoints', 1000); %set to 1000
The plots below are of npoints=30 v/s npoints=300

Plots : Click image to view full size.

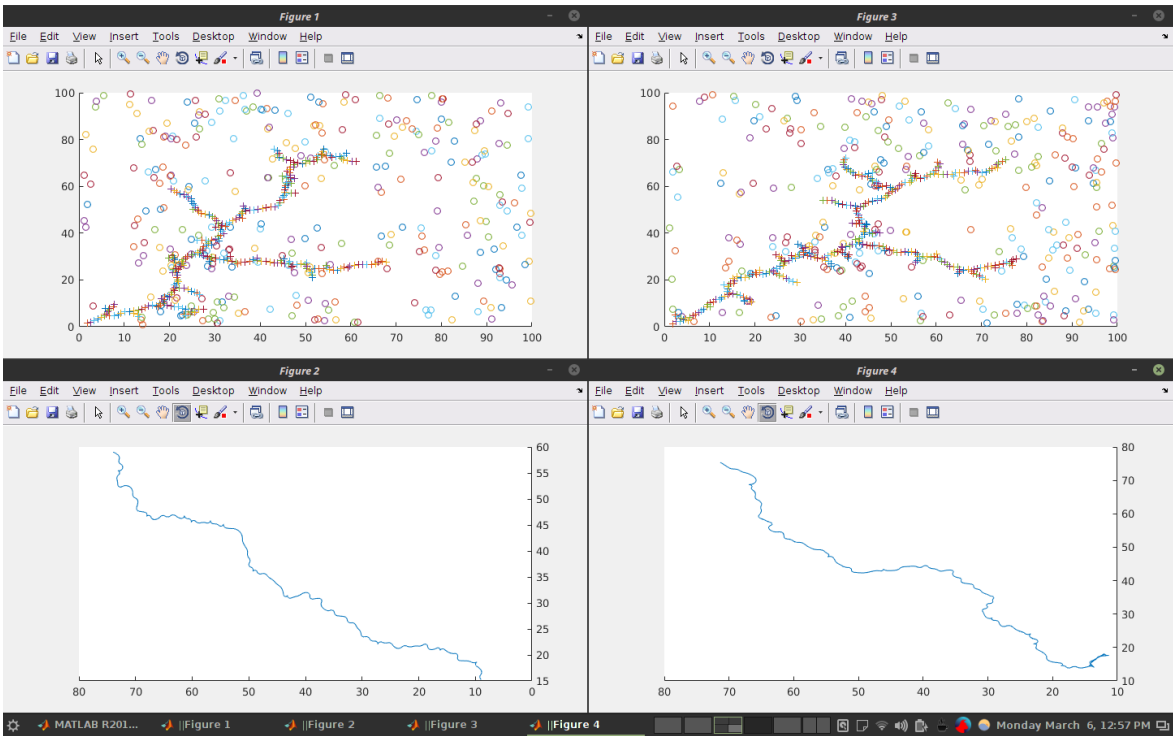


The steering angle limits were hardcoded into the code. On changing it, we see how steep curved path is modified.
Below is the plot of steermax = 1.2pi v/s steermax = 0.2pi

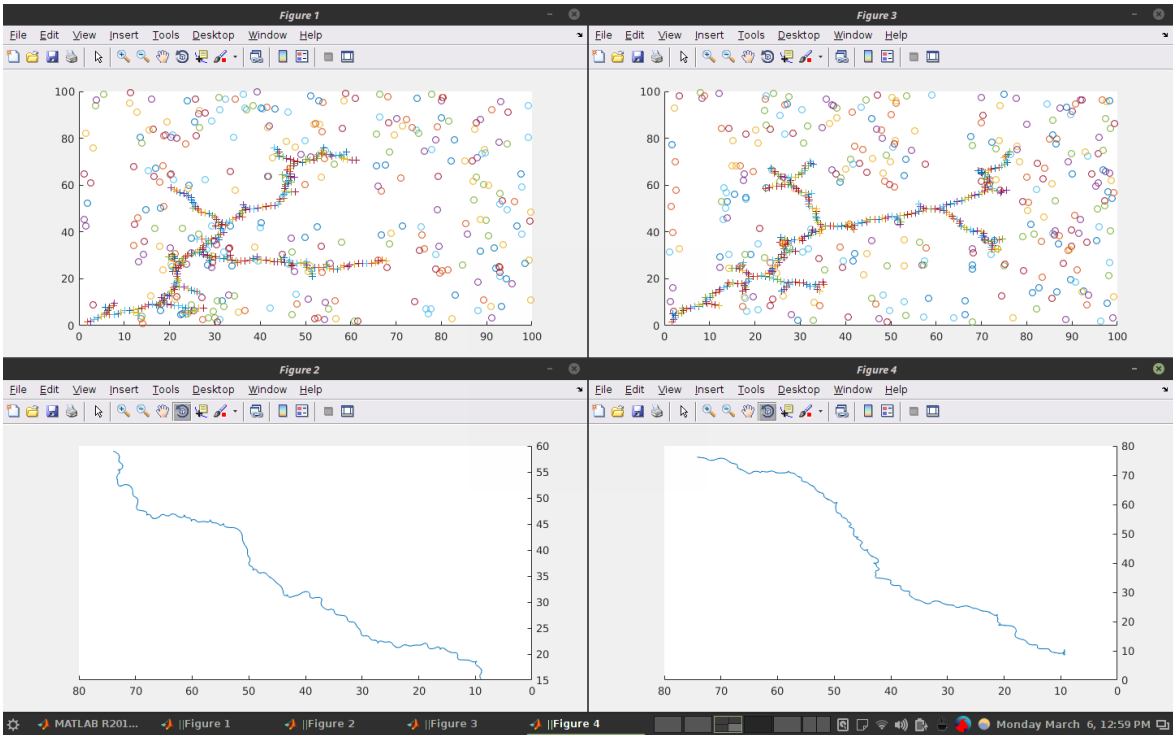


As we change the time value in the code we observe the following plots.

time = 0.5 v/s time = 0.1



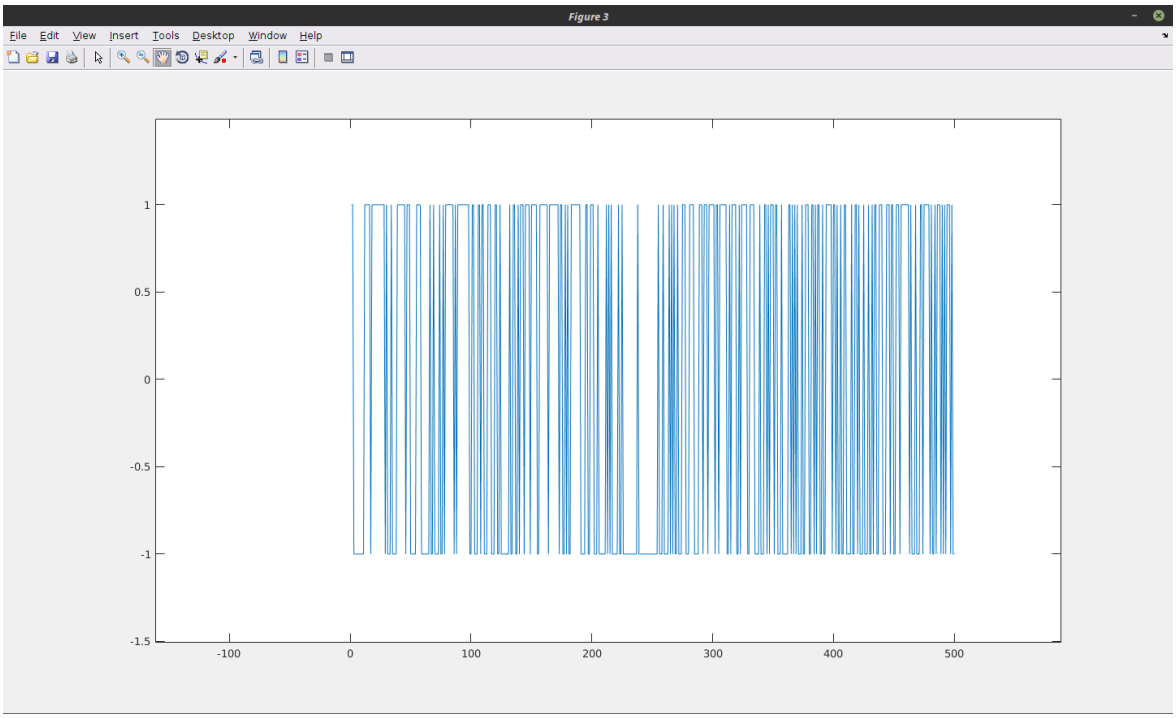
time = 0.5 v/s time = 2



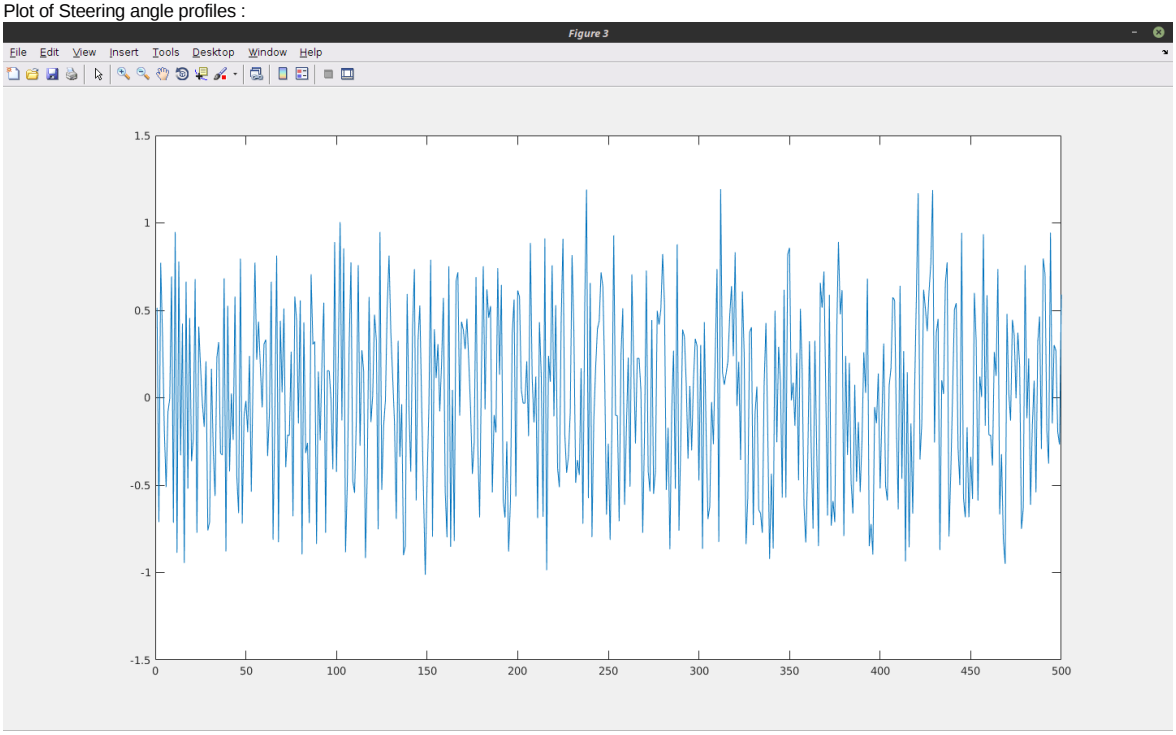
Question 3:

Click image to view full size.

Plot of velocity profiles :



Plot of Steering angle profiles :

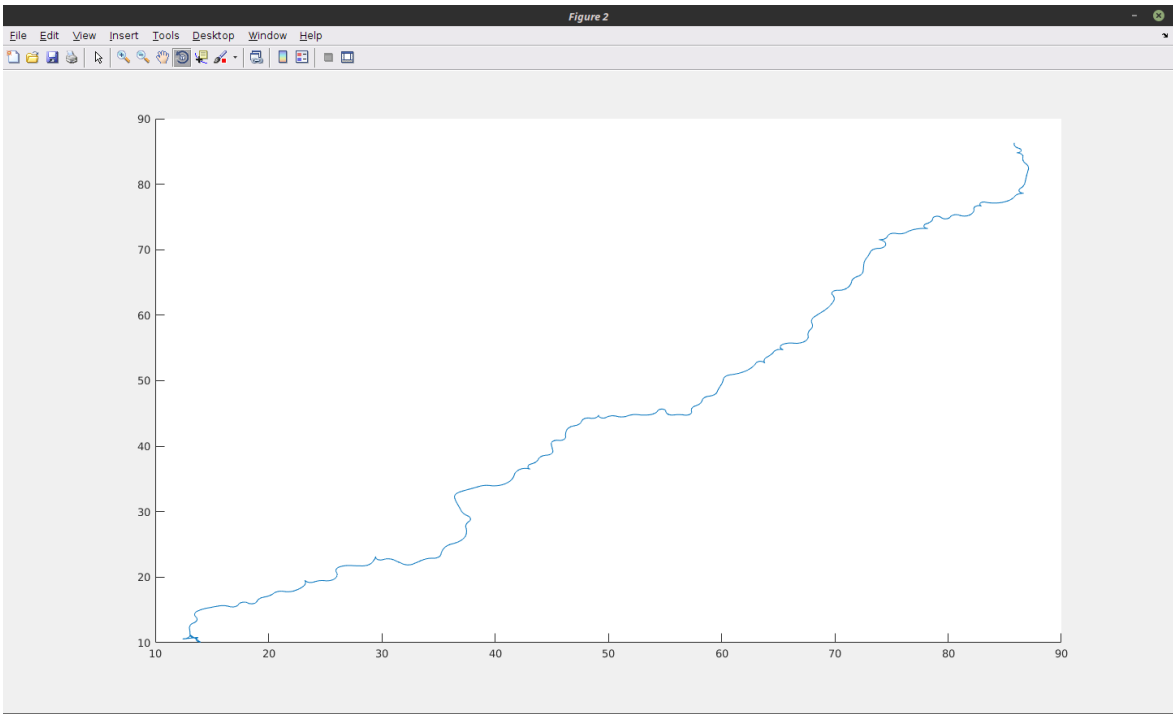


Question 4: In the bestpath function, we calculate the steer angle N times.
Thus, we replace **steer = (2*rand - 1) * steermax;** with the following block
gauss_mean = 0;
gauss_variance = steermax;
steer = gauss_mean + gauss_variance*randn();

Here, the randn function generates values from gaussian, and we shift the value accodingly using the mean and variance values according to our problem.

Click image to view full size.

Plot using values from gaussian distribution:
We observe that the path is more gentle, compared to values from uniform distribution.



Question 5: We add a local planner in the path function of RRT since the path function is responcible for generating the next point.

Here we have access to the generated rrt, xstart(startpoint) and xgoal(endpoint). We generate the closest point to start first by **closestVertex = rrt.graph.closest(xstart); xstart = closestVertex;** Instead, if we want to follow a normal route to the closest Vertex, we may set the immediete goal xgoal temporarily to the calculated closestVertex value as **xgoal = closestVertex;** and then setting the normal goal back when we have reached the calculated closest Vertex. We also could use the next function in RRT which is called to calculate the new node and modify it such that it returns the closest node as the next node. Also, considering that, there are no obstacles present between the current position and the closest vertex, we can simply drive the robot to the closest vertex by using follow the line principle.