Emanuele Costantino

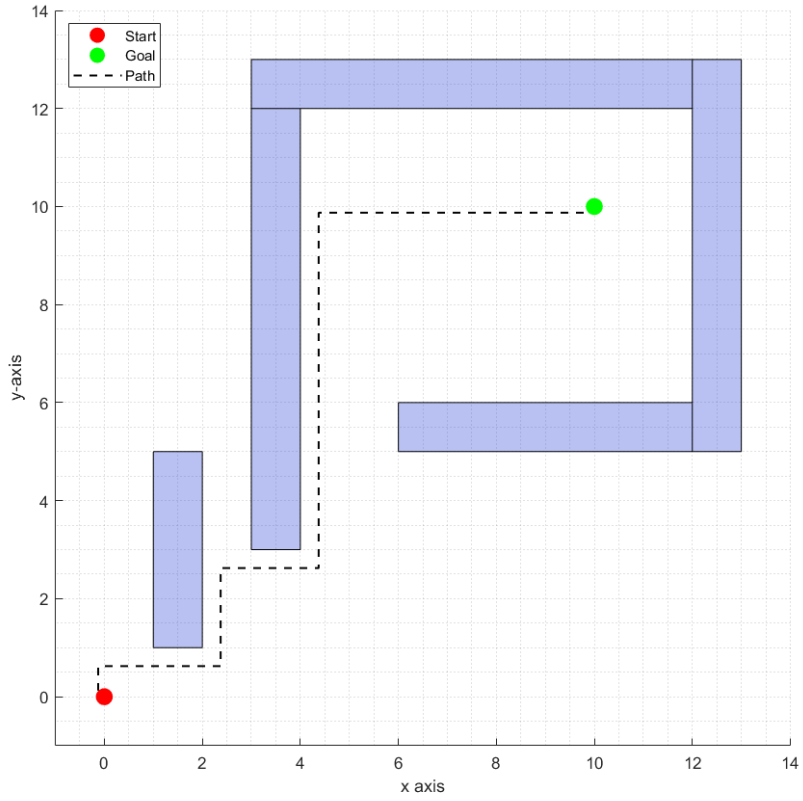
ASEN 5519

Homework #6

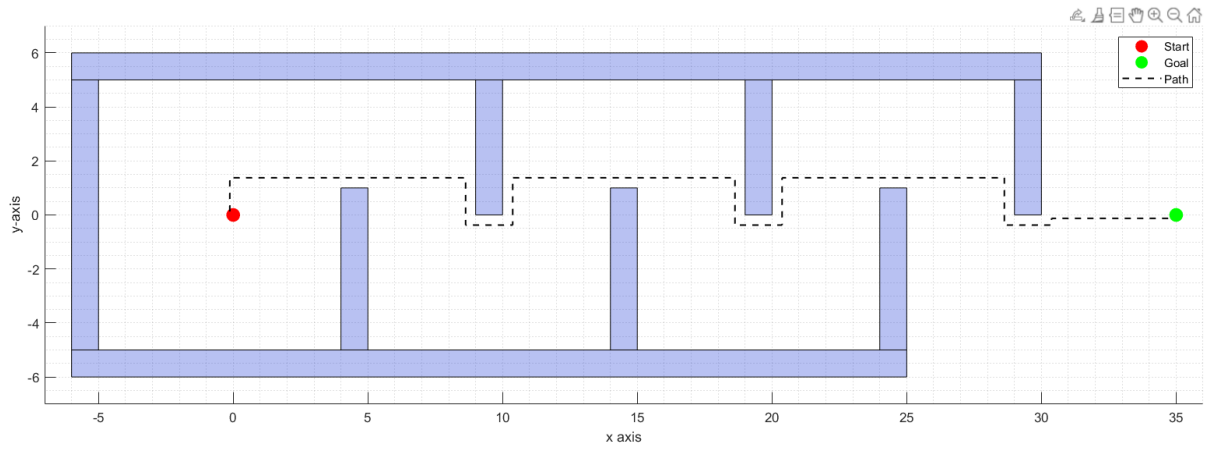
Exercise 1

a) Path plots

Workspace #1



Workspace #2



b)

Path Length Workspace #1: 19.750
Path Length Workspace #2: 45.250

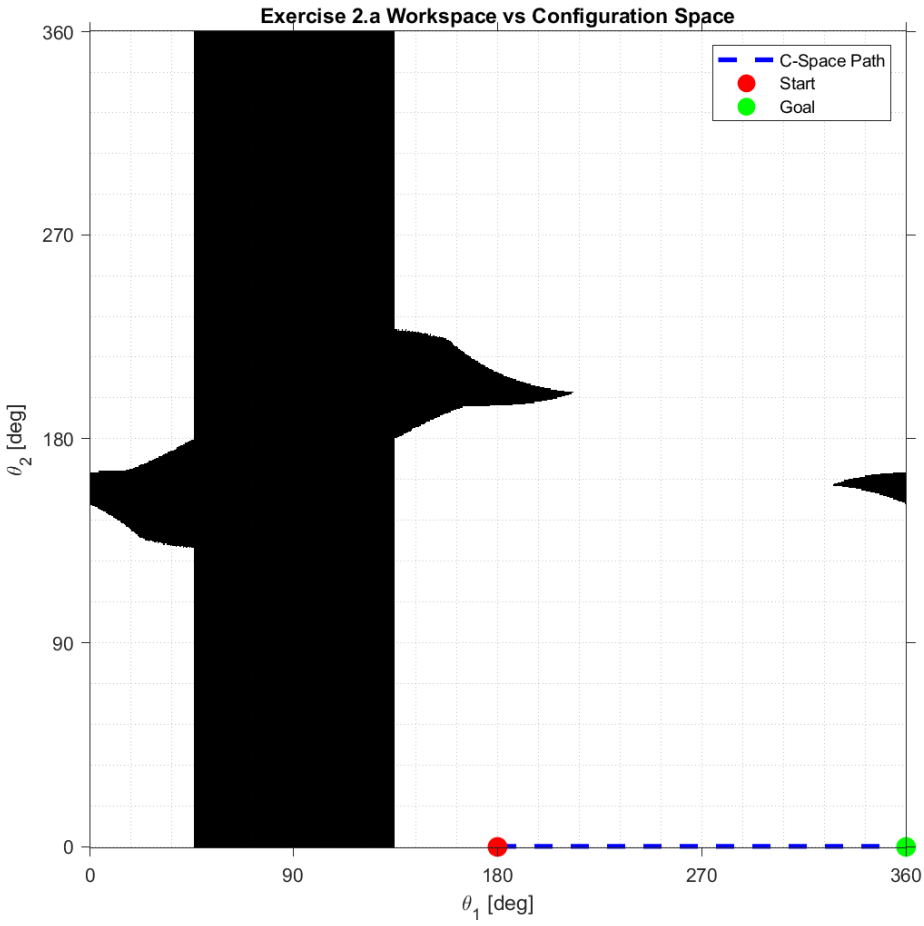
c) I would expect the path length to be smaller for Workspace #2 but NOT Workspace #1. The reason for this is that no matter how small we make our grid size, in Workspace 1 the distance to goal will always be the total Manhattan distance to goal. This is not the case in Workspace # 2 because we have to "wind" around the obstacle and as the grid size decreases, so will the total displacement along the y direction. NOTE: I am ignoring the small changes in distance that may occur at the end of the path due **only** to the change in grid size. Otherwise, I would expect Workspace #1 path to increase in length slightly and Workspace #2 would still decrease in length.

d) The wavefront planner is far more reliable than the gradient descent planner as it is complete with respect to the grid resolutions. I was not able to generate a path to goal for Workspace #2 using the gradient descent planner from HW5, but the wavefront planner was able to find a path easily without having to tune obstacle/goal parameters.

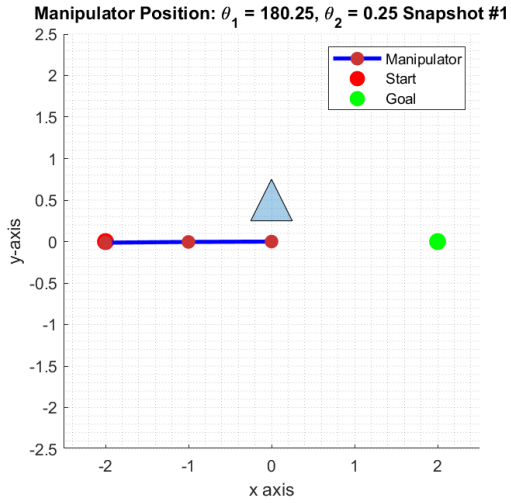
Exercise 2

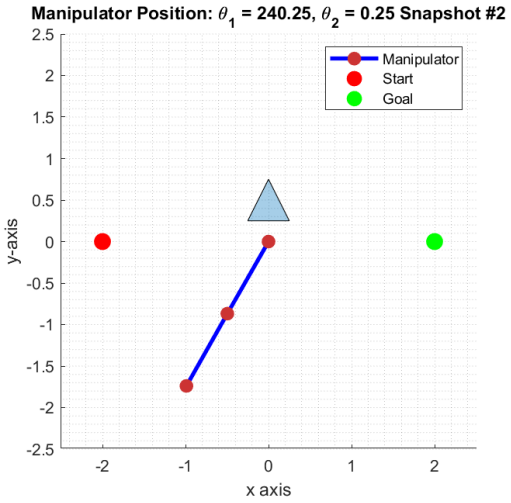
Workspace #1

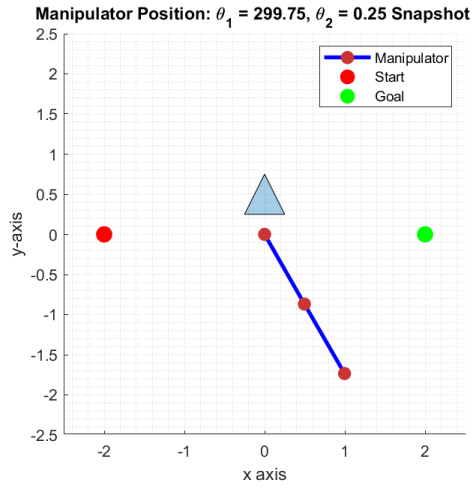
C-space

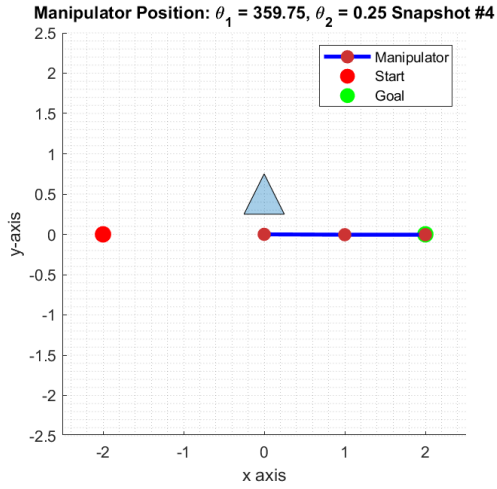

Exercise 2.a Workspace vs Configuration Space
C-Space Path
Start
Goal
360
270
180
90
0
θ_2 [deg]
0
90
180
270
360
θ_1 [deg]

Snapshots



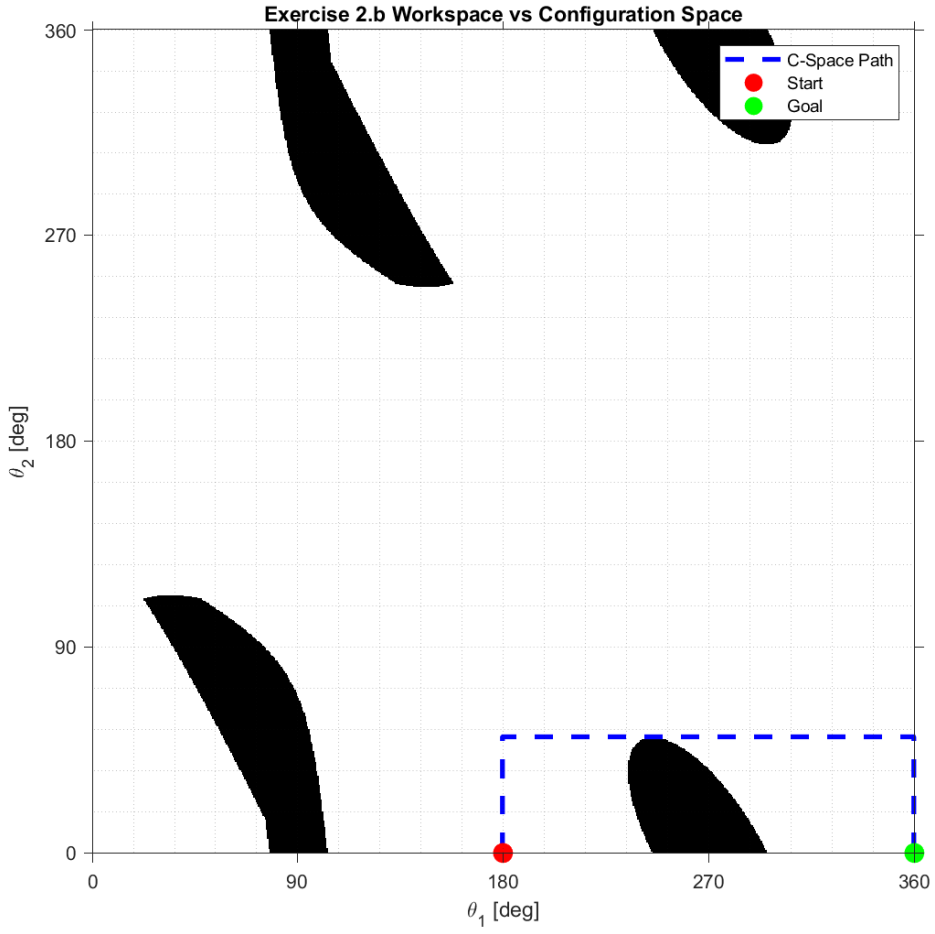




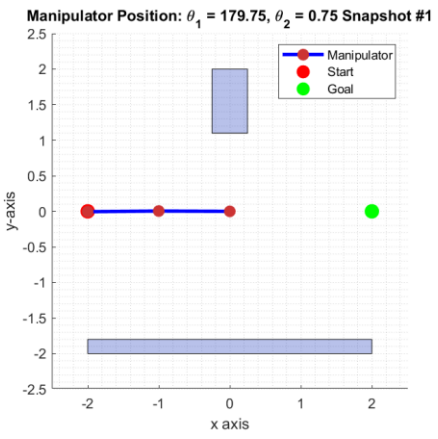


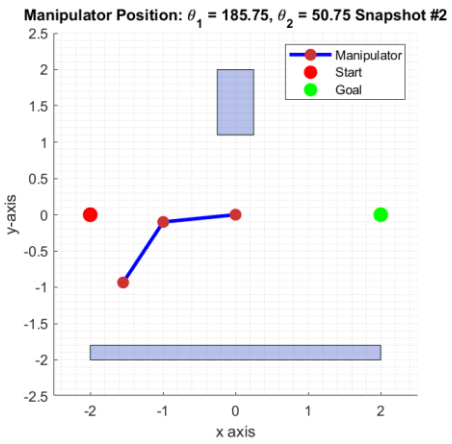
Workspace #2

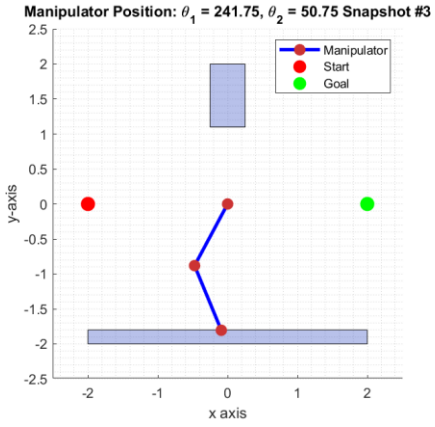
C-space

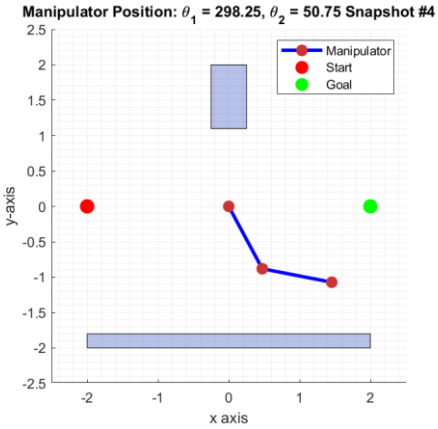

Exercise 2.b Workspace vs Configuration Space
C-Space Path
Start
Goal
360
270
180
90
0
θ_2 [deg]
0
90
180
270
360
θ_1 [deg]

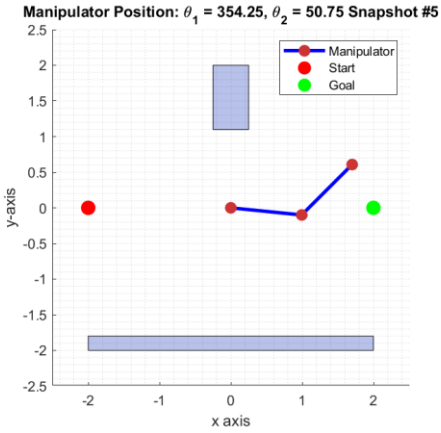
Snapshots

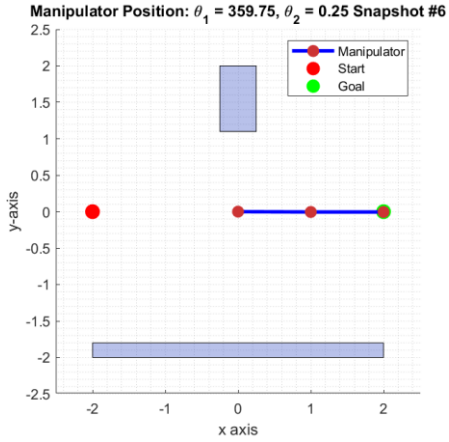

Manipulator Position: θ_1 = 179.75, θ_2 = 0.75 Snapshot #1
Manipulator
Start
Goal
y-axis
x axis


Manipulator Position: θ_1 = 185.75, θ_2 = 50.75 Snapshot #2
Manipulator
Start
Goal
y-axis
x axis



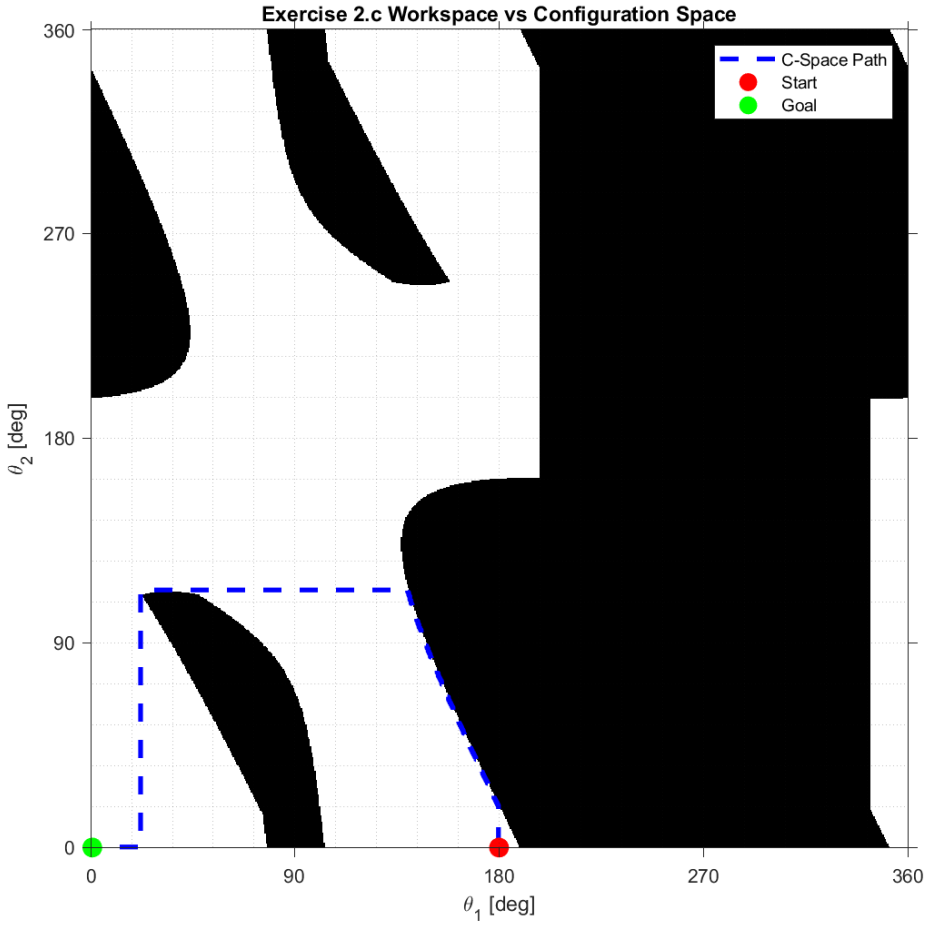




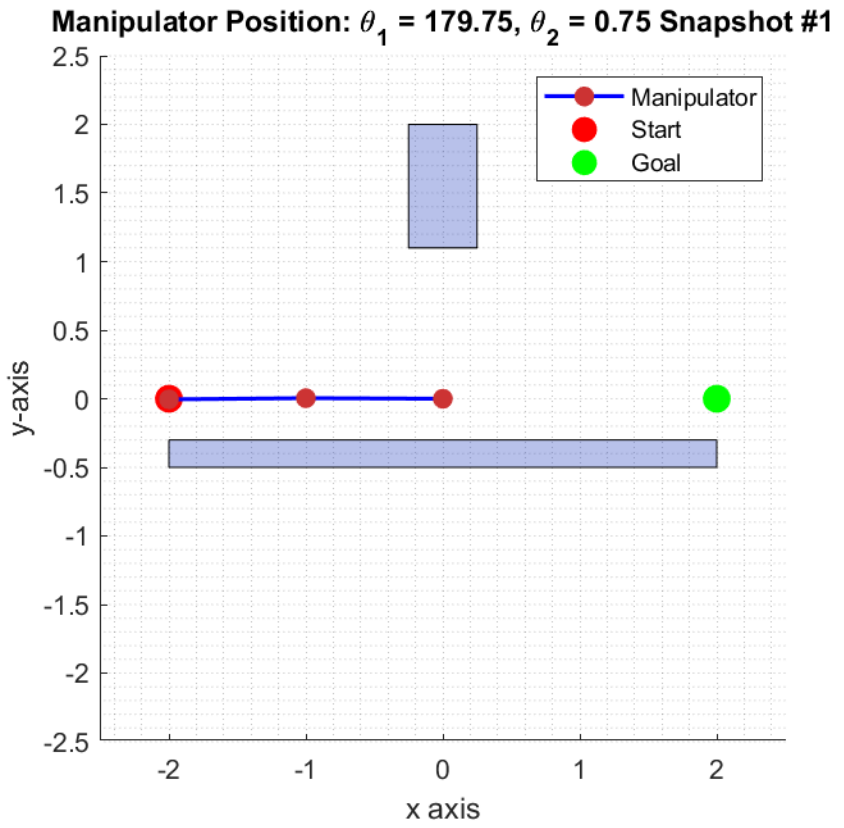


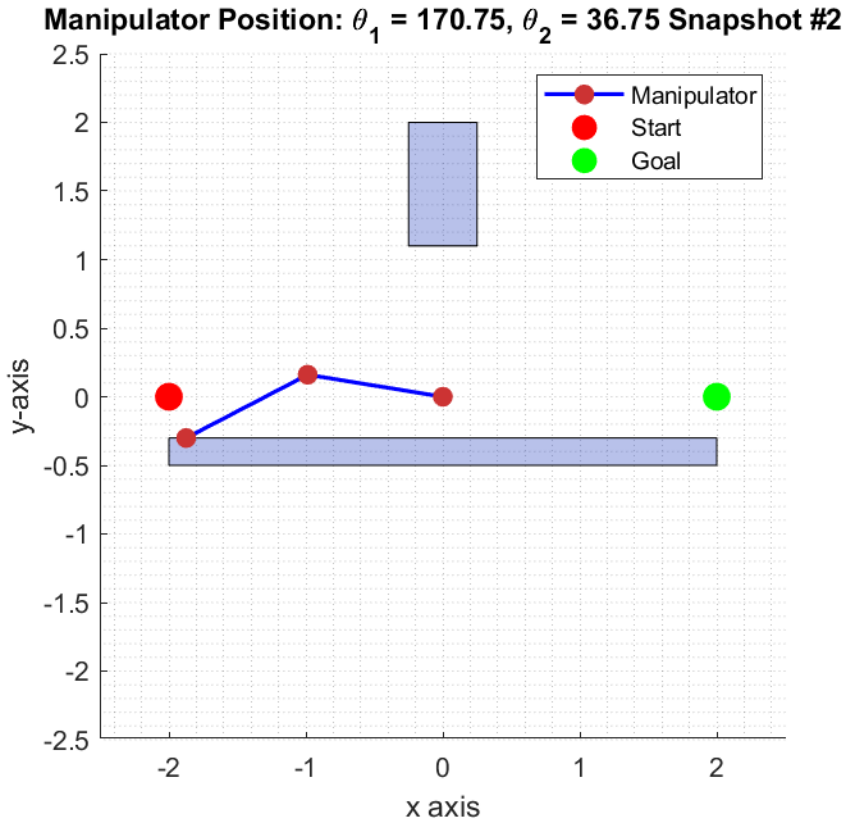
Workspace #3

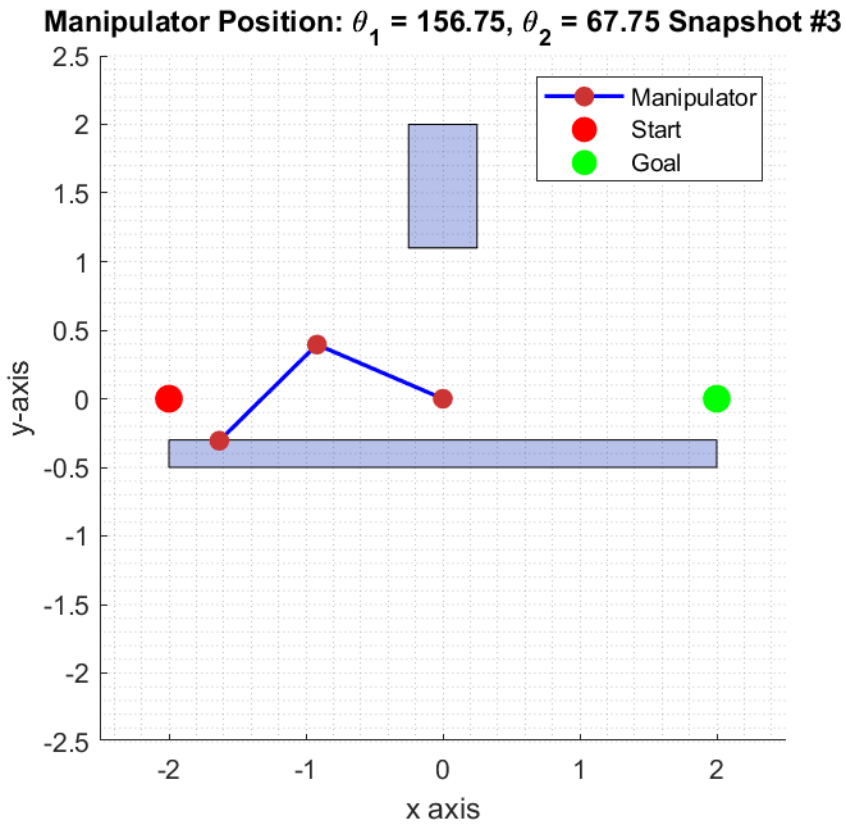
C-space

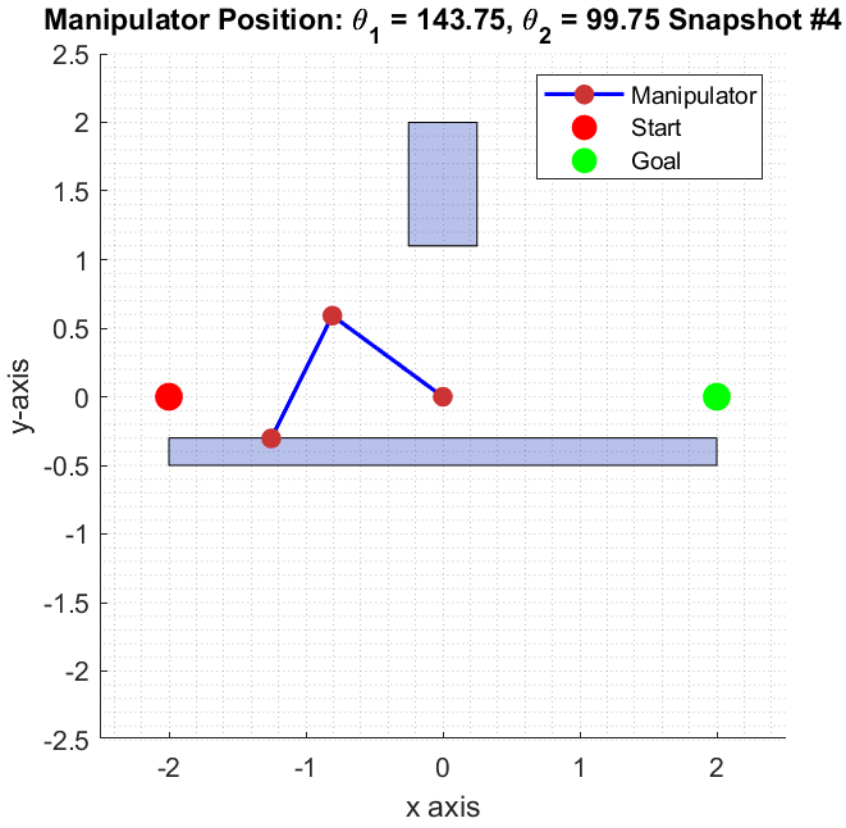

Exercise 2.c Workspace vs Configuration Space
C-Space Path
Start
Goal
360
270
180
90
0
θ_2 [deg]
0
90
180
270
360
θ_1 [deg]

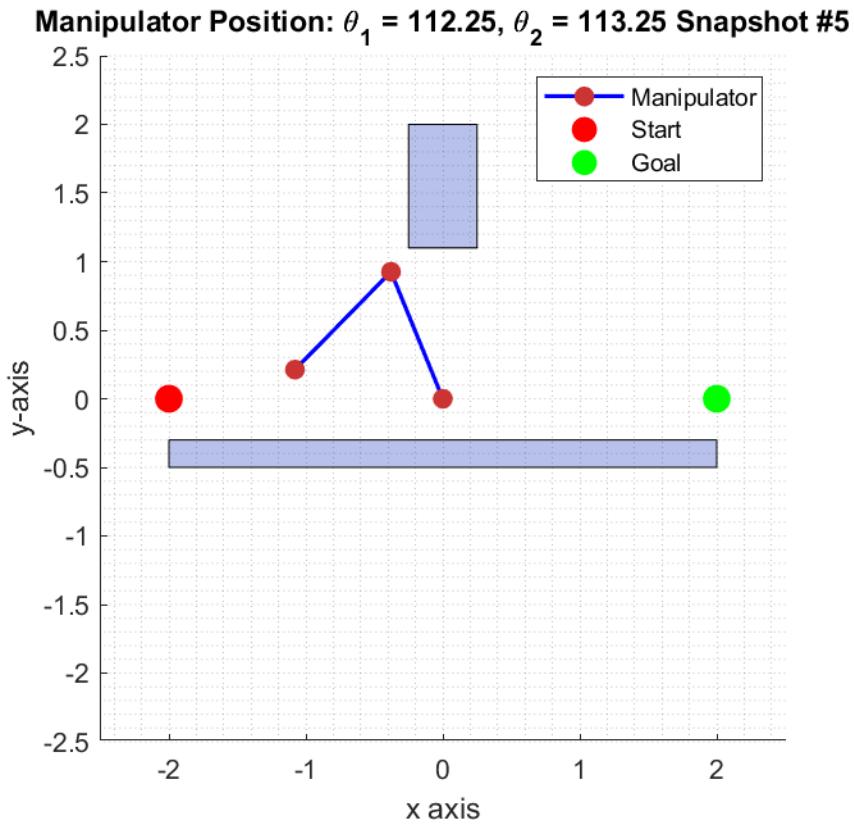
Snapshots

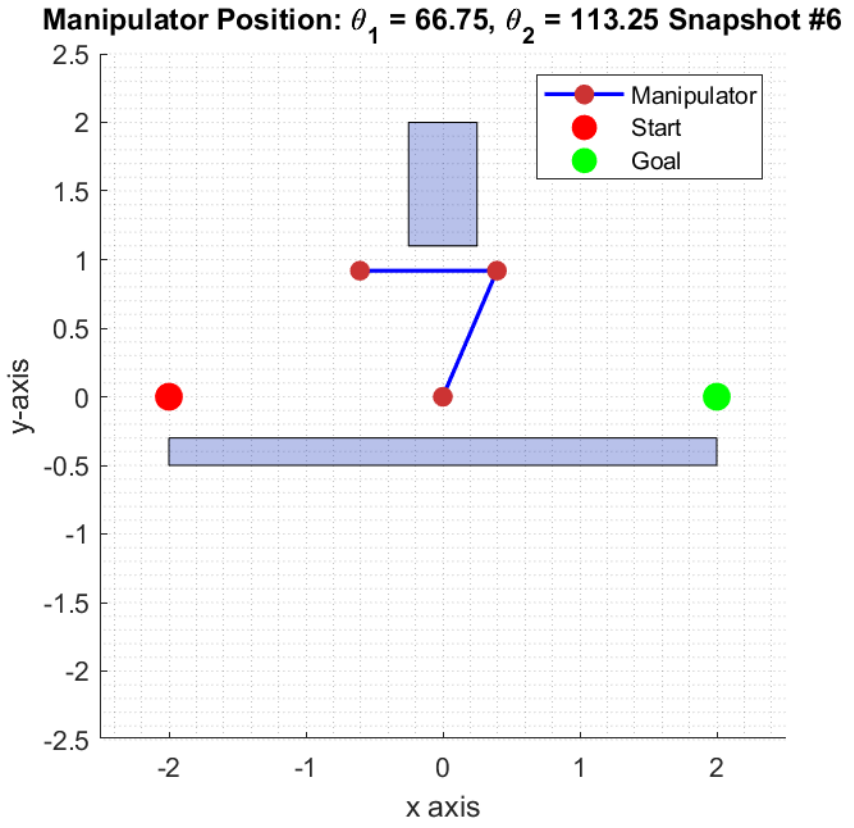

Manipulator Position: θ_1 = 179.75, θ_2 = 0.75 Snapshot #1
Manipulator
Start
Goal
y-axis
x axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2

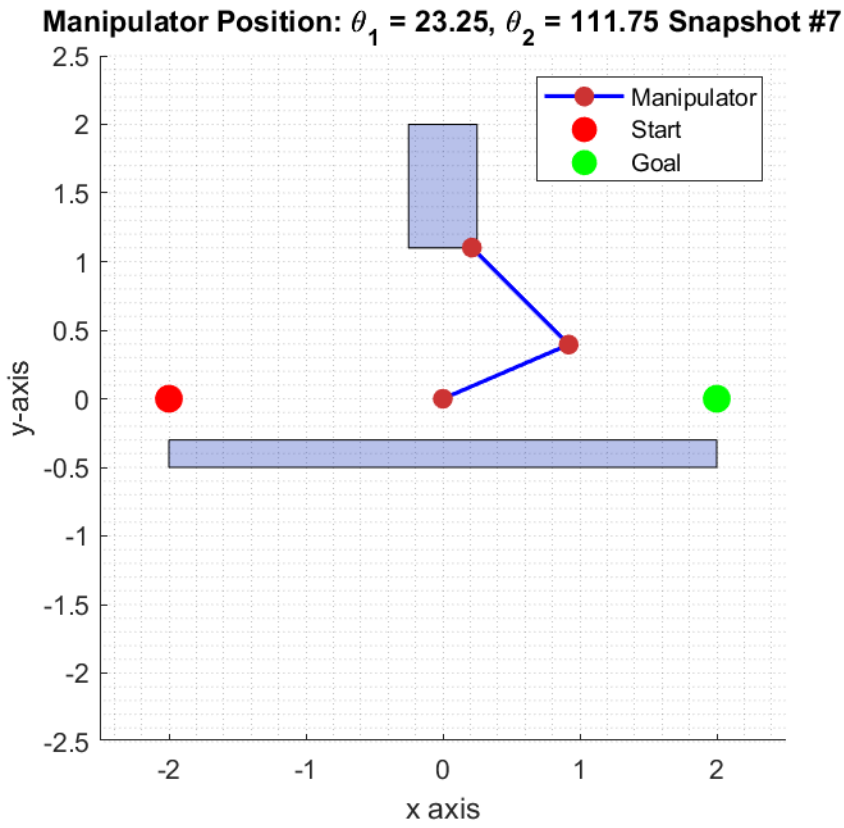

Manipulator Position: θ_1 = 170.75, θ_2 = 36.75 Snapshot #2
Manipulator
Start
Goal
y-axis
x axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2

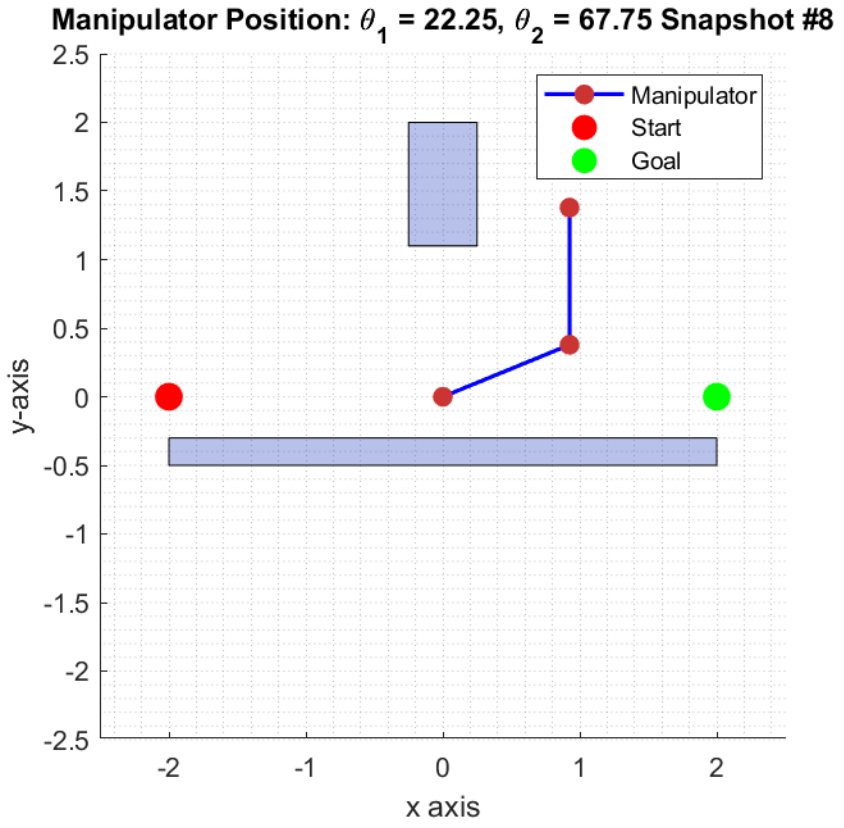

Manipulator Position: θ_1 = 156.75, θ_2 = 67.75 Snapshot #3
Manipulator
Start
Goal
y-axis
x axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2

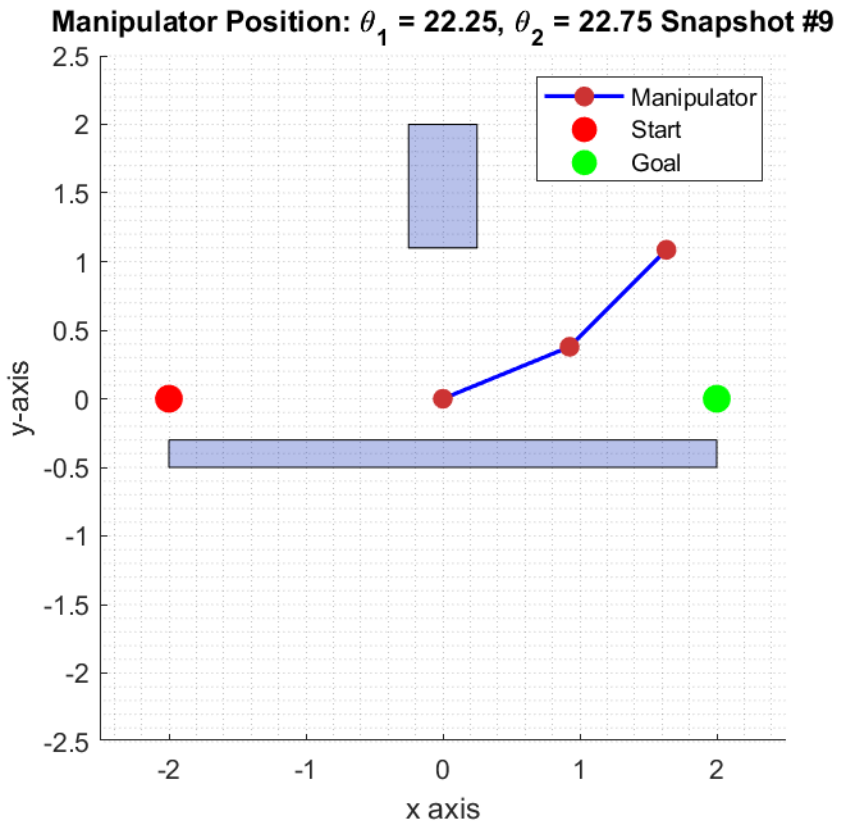

Manipulator Position: θ_1 = 143.75, θ_2 = 99.75 Snapshot #4
Manipulator
Start
Goal
y-axis
x axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2

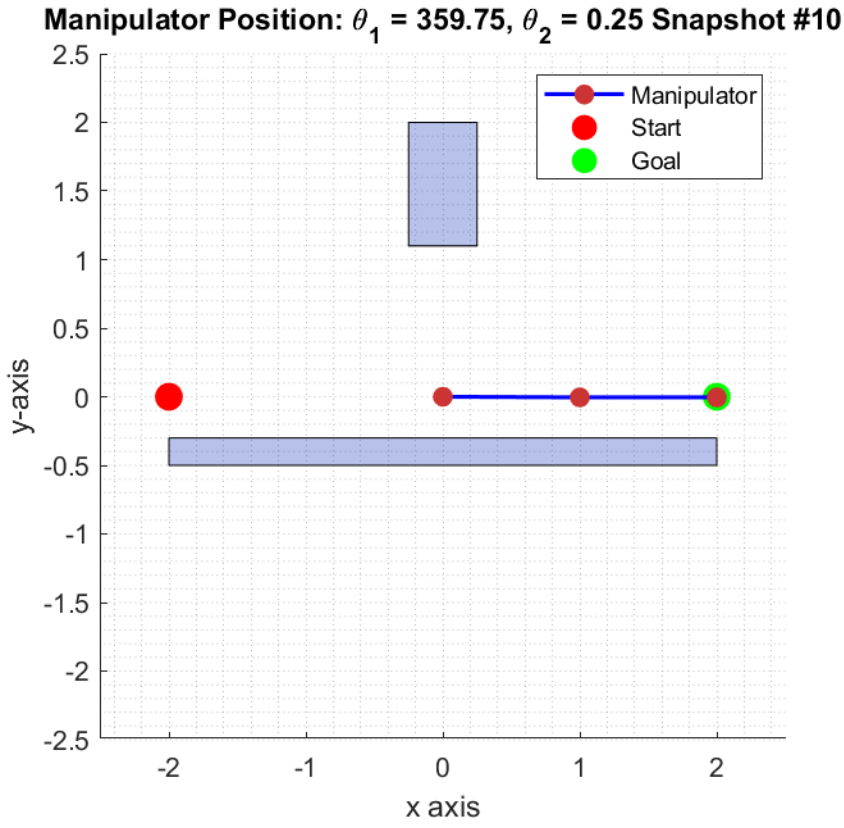

Manipulator Position: θ_1 = 112.25, θ_2 = 113.25 Snapshot #5
Manipulator
Start
Goal
y-axis
x axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2


Manipulator Position: θ_1 = 66.75, θ_2 = 113.25 Snapshot #6
Manipulator
Start
Goal
y-axis
x axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2


Manipulator Position: θ_1 = 23.25, θ_2 = 111.75 Snapshot #7
Manipulator
Start
Goal
y-axis
x axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2


Manipulator Position: θ_1 = 22.25, θ_2 = 67.75 Snapshot #8
Manipulator
Start
Goal
y-axis
x axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2


Manipulator Position: θ_1 = 22.25, θ_2 = 22.75 Snapshot #9
Manipulator
Start
Goal
y-axis
x axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2


Manipulator Position: θ_1 = 359.75, θ_2 = 0.25 Snapshot #10
Manipulator
Start
Goal
y-axis
x axis
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2

Exercise 3

a) A* implementation:
    a. Path: 0→2→9→13
    b. Path Length: 5
    c. Number of Iterations: 9
b) To turn the A* algorithm into Dijkstra's you have to stop using the heuristic and we do not sort the O list. Instead of picking the node with the smallest f (which would now equal g) value in the O list, we just pick the first node of the O list. This is called breadth first search because we look at the subsequent layers of nodes from each node.
c) Dijkstra's implementation:
    a. Path: 0→2→7→13
    b. Path Length: 5
    c. Number of Iterations: 13
d) A* performed better than Dijkstra's in that it found the goal in less iterations. If I was equipped with a good heuristic (admissible) I would choose A* to search a large map from start to goal and from start to any node in V.
e) I used an adjacency matrix to represent my graph. If I were to implement an undirected graph I would make the adjacency matrix *symmetric*. The A* algorithm implemented in this assignment would be able to run over this graph still.