Non-Holonomic Constraint on Car like Robot

AKSHAY BHOLA 14060

Content

- Introduction
- System Dynamics
- Runge-Kutta Method
- Algorithm
- Specification
- Implementation

Introduction

- **Holonomic Constraint**

  For a constraint to be holonomic it must be expressible in the form

  $$f(x_1, x_2, x_3, \dots, x_N, t) = 0$$

  In holonomic system, the state depends doesn't depend on the path taken to reach the state.

- **Non - Holonomic Constraint**

  These are constraints on system that can't be expressed as function shown above.

  $$\sum_{1}^{n} a_i \delta q_i = 0$$

  The system is path dependent

System Dynamics

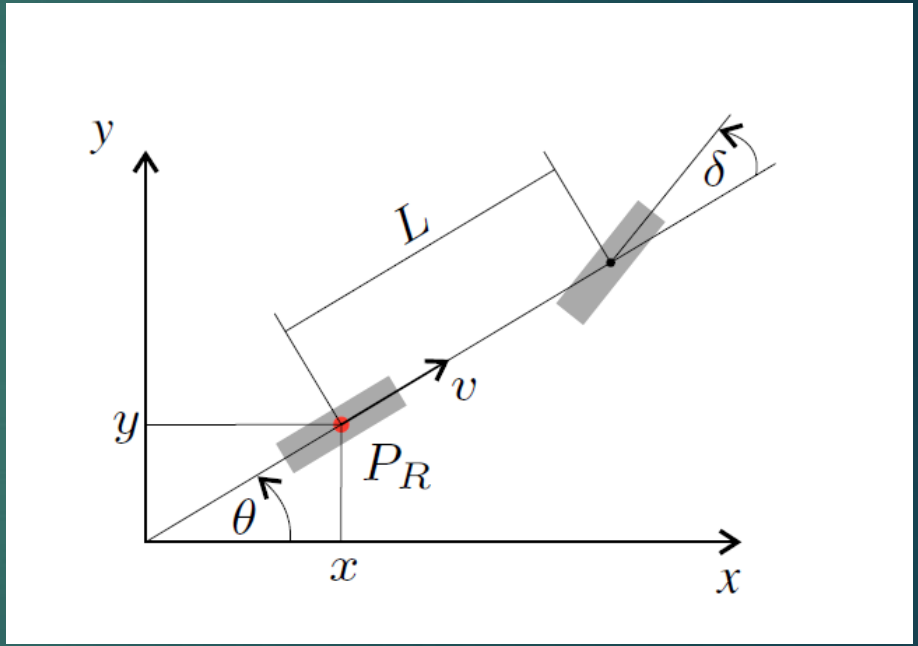
State of car (coordinates), $q = [x, y, \theta]^T$

x,y are coordinates of one of reference point

δ : steering angle

The non-holonomic kinematic differential equations in these generalised coordinates

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v}{L}\tan(\delta) \end{pmatrix} = \mathbf{f}_t(\mathbf{q}, \mathbf{u}_A).$$



System Dynamics
Path Velocity Decomposition

Control Input to car, $u_A = [v, \delta]$

Path Velocity Decomposition

Decoupling geometric path planning from kinematic velocity planning

$$v = D\frac{ds}{dt}$$

s: path position

D: vehicle Direction $\quad D \epsilon \{-1,1\}$

$$u_l = \frac{tan\delta}{L}$$

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} D\cos(\theta) \\ D\sin(\theta) \\ Du_l \end{pmatrix} = \mathbf{f}_s(\mathbf{q}, u_l, D),$$

Runge-Kutta Method

- The method is widely known to solve the equation:

$$\frac{dy}{dt} = f(t, y)$$

Initial condition: $y(t_o) = y_0$

- The method is an efficient way to solve the initial value problem and the desired accuracy can be obtained by selecting suitable order of method
- In our case second order Runge-Kutta Method used

Runge-Kutta Method

- The algorithm for second order Runge Kutta Method is as follows:

$$k_{1_i} = \eta_i f_s(q_i, u_{l_i})$$

$$k_{2_i} = \eta_i f_s\left(q_i + \frac{k_{1_i}}{2}, u_{l_i}\right)$$

$$q_{i+1} = q_i + k_{2_i} + O(\eta_i^3)$$

η_i: step size

- Applying the above equations to the desired system

$$\mathbf{q}_{i+1} = \begin{pmatrix} x_{i+1} \\ y_{i+1} \\ \theta_{i+1} \end{pmatrix} = \begin{pmatrix} x_i + D\eta_i \cos\left(\theta_i + D\frac{\eta_i u_{l_i}}{2}\right) \\ y_i + D\eta_i \sin\left(\theta_i + D\frac{\eta_i u_{l_i}}{2}\right) \\ \theta_i + D\eta_i u_{l_i} \end{pmatrix}$$
$$= \mathbf{f}(\mathbf{q}_i, \mathbf{u}_i, D),$$

Algorithm

- Cost Function

$$l_{O_i}(q_{i+1}) = r_\theta e^2_{\theta_{(i+1)}} + e^T_{P_{i+1}} R e_{P_{i+1}}$$

where r_θ, R are cost parameter

$$e_{P_i} = [x_i - x_s, y_i - y_s]^T$$

where s correspond to goal position

$$e_{\theta_i} = \theta_i - \theta_O$$

θ_O is the target angel

- Aim is to find path such that cost function would be minimized

Algorithm

- At every step cost of function is calculated for n different positions corresponding to the steering of

$$\delta = \delta_{min} + idx * \frac{(\delta_{max} - \delta_{min})}{n}$$

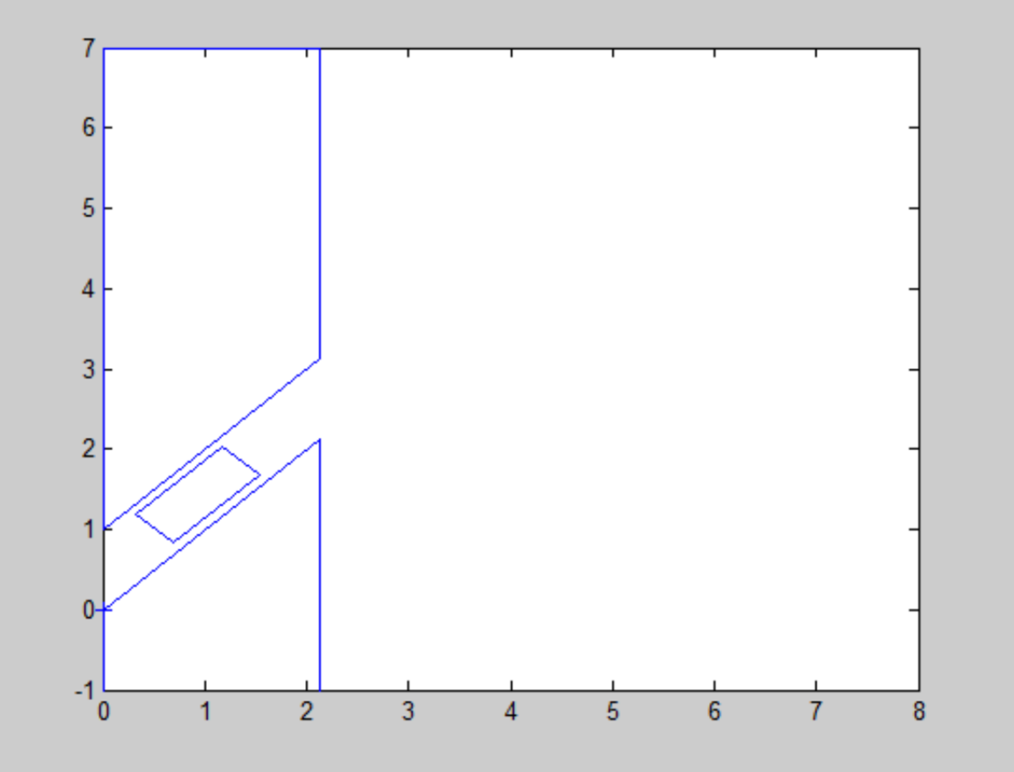
- Corresponding to every step, cost value is calculated. The step corresponding to lowest value of cost function is selected.
- Feasibility of step is checked. If unfeasible, the next step corresponding to lowest value is selected.
- For convergence check, state difference function is created. The result is converged when it reaches the assigned bounds

Specification of Test cases

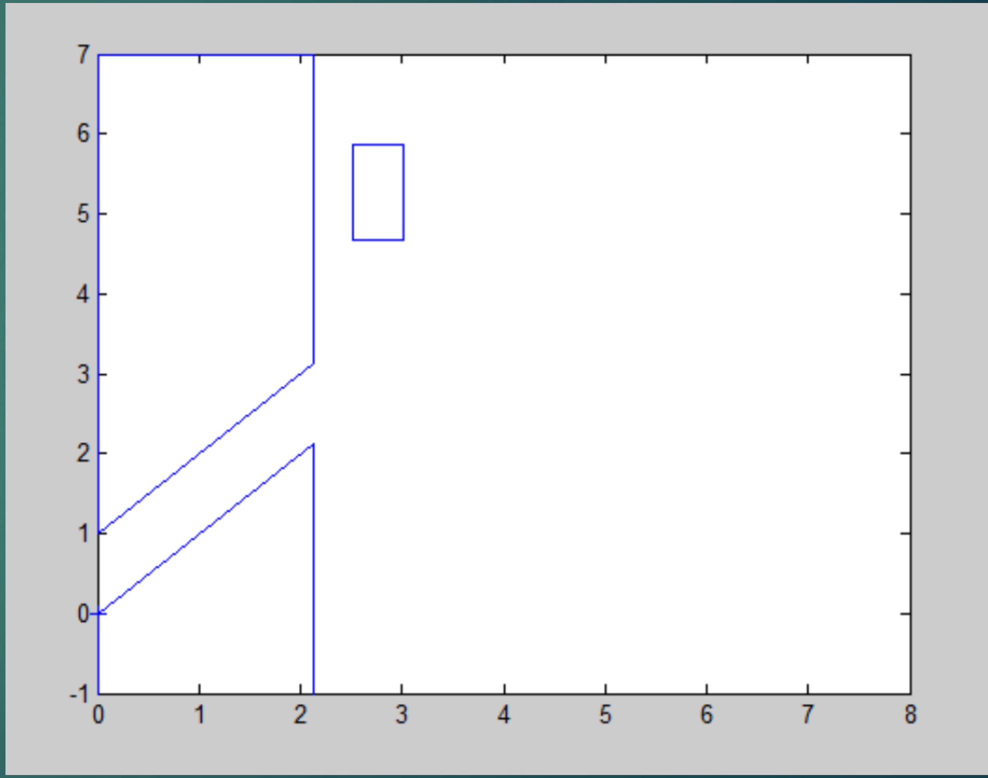
- $\delta_{max} = 30\,°$
- $\delta_{min} = -30\,°$
- n=100
- Step size = 0.01m
- Convergence bound = 0.1

Implementation

Case 1



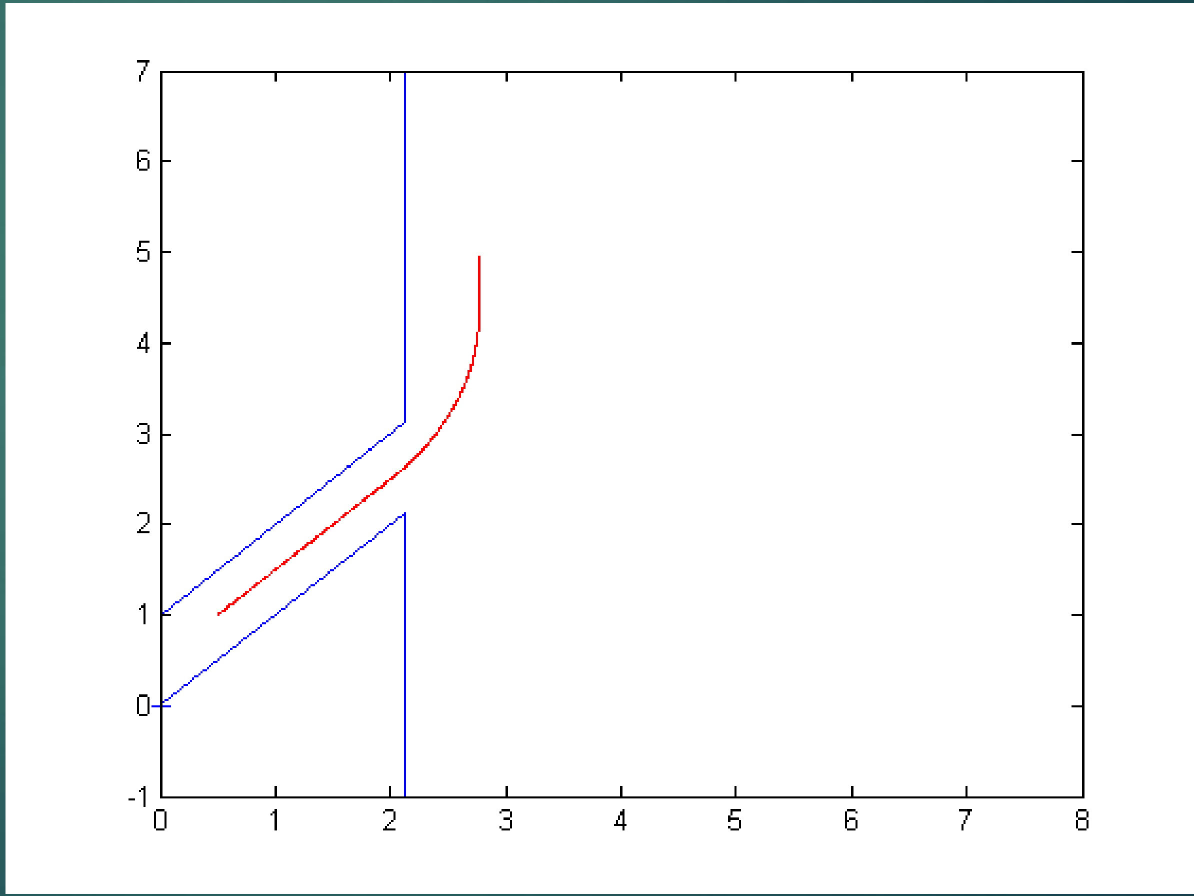
Initial State



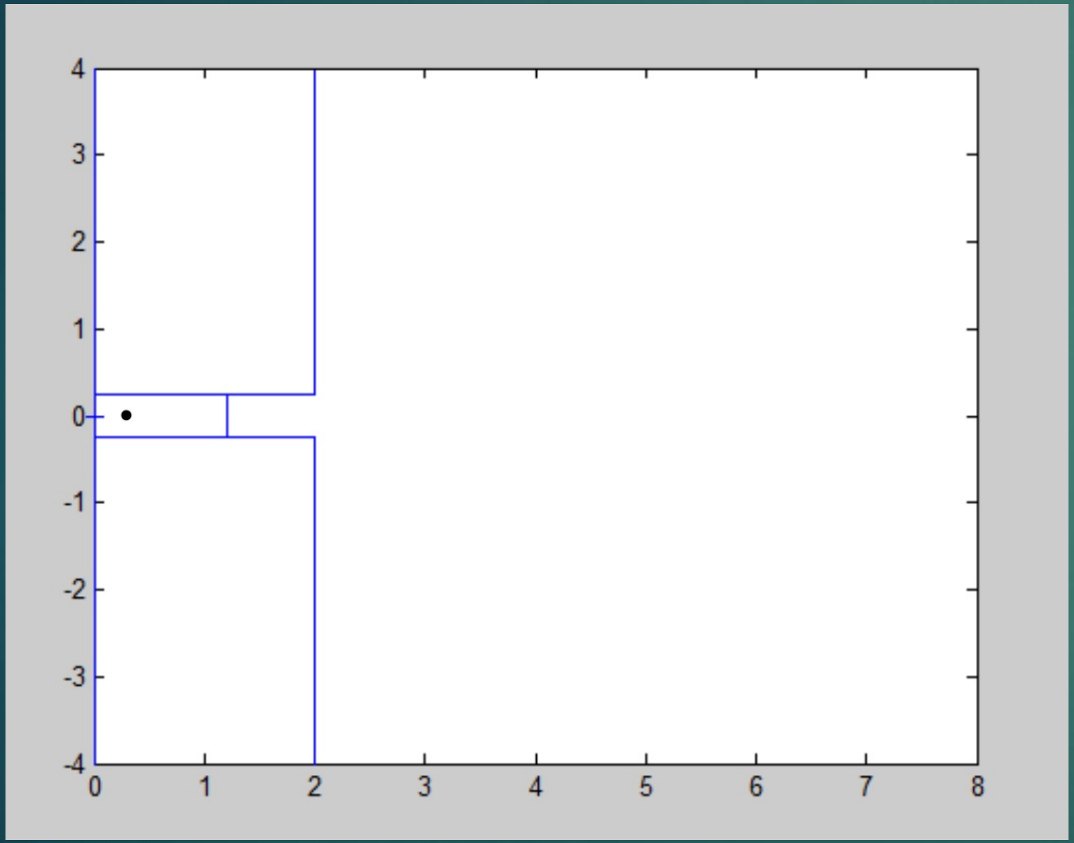
Final State

Implementation
Case 1

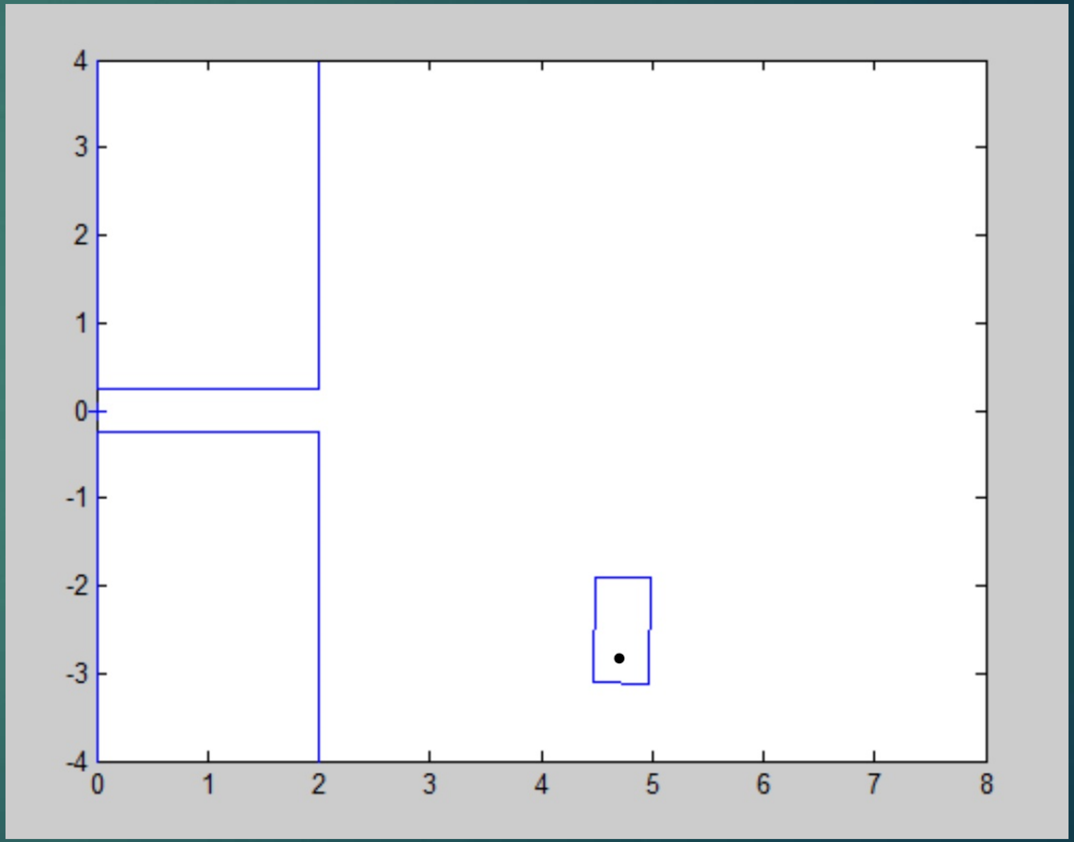
- Adjusting the cost function parameters according to requirement
- The path traced by reference point is shown



Implementation
Case 2



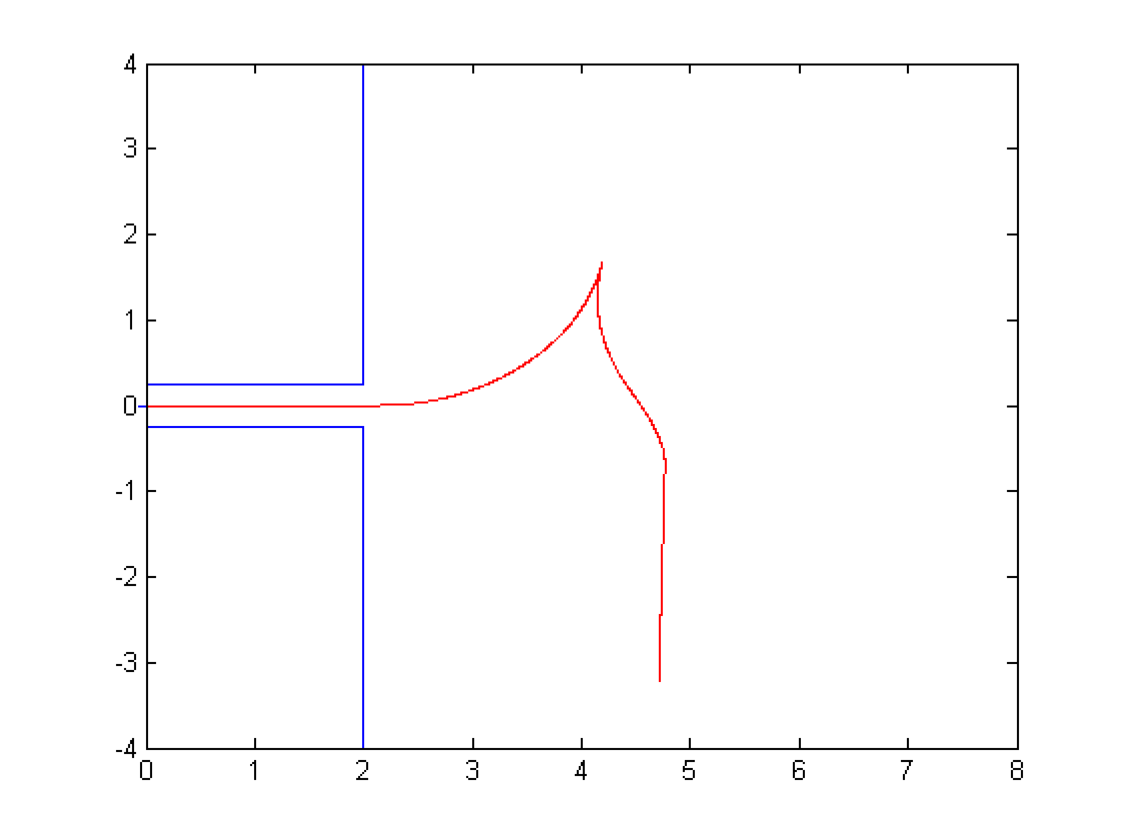
Initial State



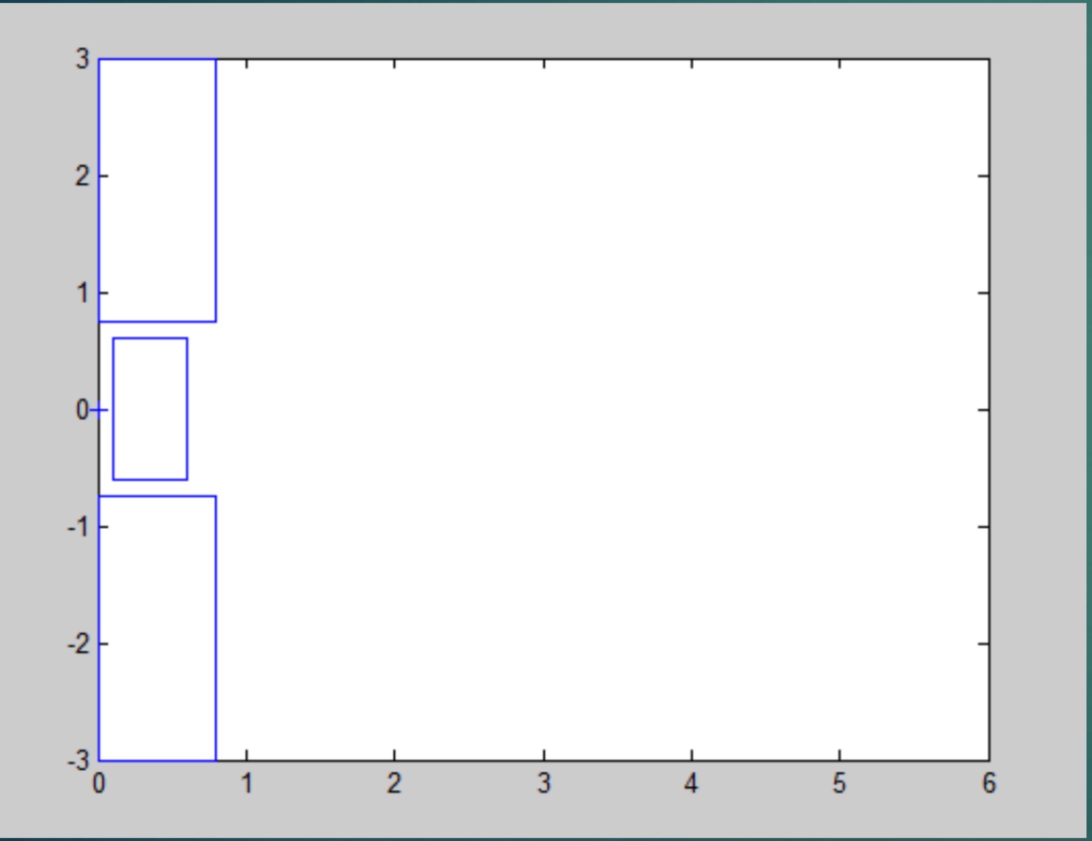
Final State

Implementation
Case 2

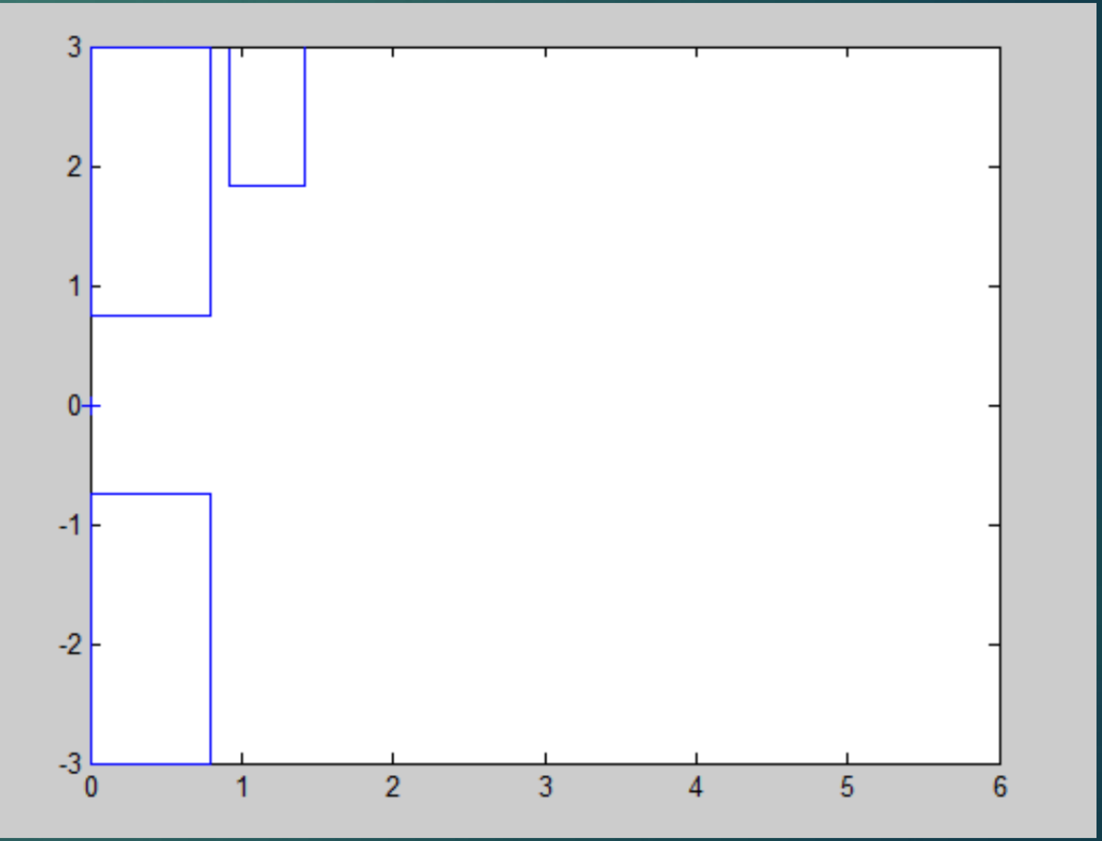
- Cost variables specification:
  - Initially moderate value of both R and r_θ
  - After constrained passage, high value of r_θ is applied
  - During backward motion, value of R adjusted moderately higher than r_θ
- The path traced by reference point is shown



Implementation
Case 2



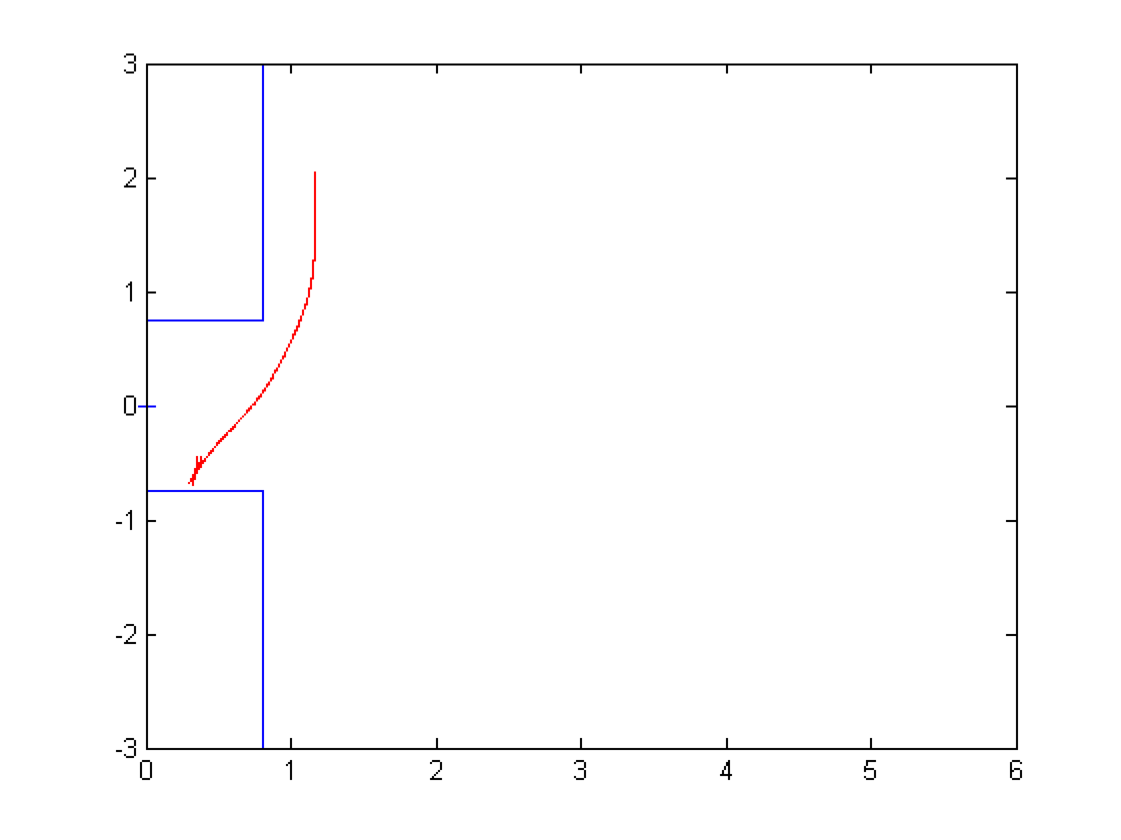
Initial State



Final State

Implementation
Case 2

- To reach the final position, the car go through many to and fro motion
- The path traced by reference point is shown



Thank you