Rule-based Optimal Control for Autonomous Driving

Agenda

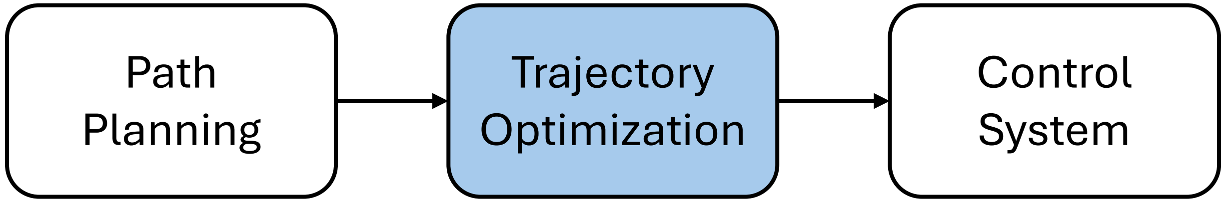
- Introduction to the Project
- Problem formulation
- Approach to solve – Algorithm
- Results
- Tools

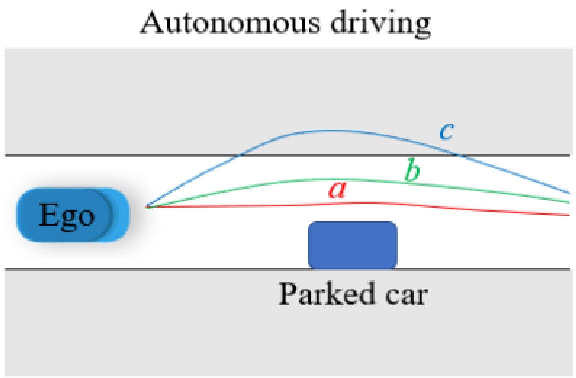
Highlights of the Project

- Modified implementation of a Research Paper
- Optimal Control Technique
- Optimization methodology
- MATLAB Optimization tool chain
- C++ Optimization tool chain

Introduction to the Project

- With higher levels of Autonomy, the decision-making capabilities increase
- The vehicles on road must follow certain driving behaviors and traffic rules
- These rules and behaviors can be assigned certain priority and modeled as constraints
- The control algorithms must consider these rules along with on-road driving constraints to find an optimal path in the given environment
- Aim of the project is to find an optimal trajectory based on an iterative optimization algorithm with prioritized soft constraints





Problem Formulation

- We essentially want to **solve an optimization problem** with soft constraints
- How to model **soft constraints** on a non-linear system?

- **Control Lyapunov Function**

  - CLF is used as a stability function for non-linear control systems
  - CLF is needs be designed – *V(x)*
  - It is an exponentially decaying function
  - It helps to construct stabilizing inputs
  - We can construct a CLF constraint on our system based on its decaying condition.

- **Control Barrier Function**

  - CBF is used for imposing safety critical constraints for a non-linear control system
  - We need to design the CBF – *B(x)*
  - We can design multiple CBFs
  - Used in combination with CLF for stability and safety

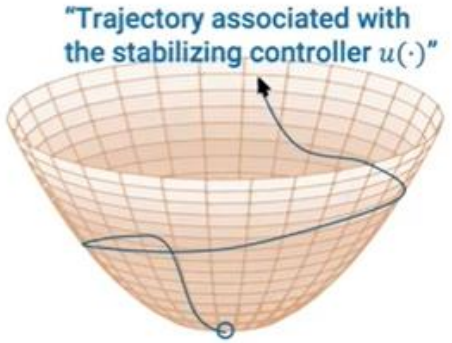
Famous concepts in non-linear controls for MPC optimization formulation

Control Lyapunov Function

- Consider an affine control system: $\boxed{\dot{x} = f(x) + g(x)u}$
- We design our CLF – ***V(x)***, such that the state reaches the target destination (or set) and impose a stability constraint

- CLF general form: $\boxed{L_f V(x) + L_g V(x)u + \epsilon V(x) \leq \delta_e}$

This is adding a stability constraint to the system along with the desired target reaching constraint



L_f	- Lie derivative along f(x)
$V(x)$	- Lyapunov Function
δ_e	- CLF Slack Variable

- ϵ is a tuning constant. As ϵ gets bigger, the CLF constraint imposes stricter condition. It requires $V(x)$ to decay more quickly.

Control Barrier Function

- CBFs are used to add safety critical constraints which help the system to avoid unsafe sets
- There two forms of CBFs, depending on the degree of CBF function ***B(x)*** chooses

1. CBF general form: $\boxed{L_f B(x) + L_g B(x) u + \gamma_i B(x) - \delta_i \geq 0}$

2. HOCBF(2nd degree): $\boxed{L_f^2 B(x) + L_f L_g B(x) u + \gamma_i B(x) - \delta_i \geq 0}$

$B(x)$ - CBF

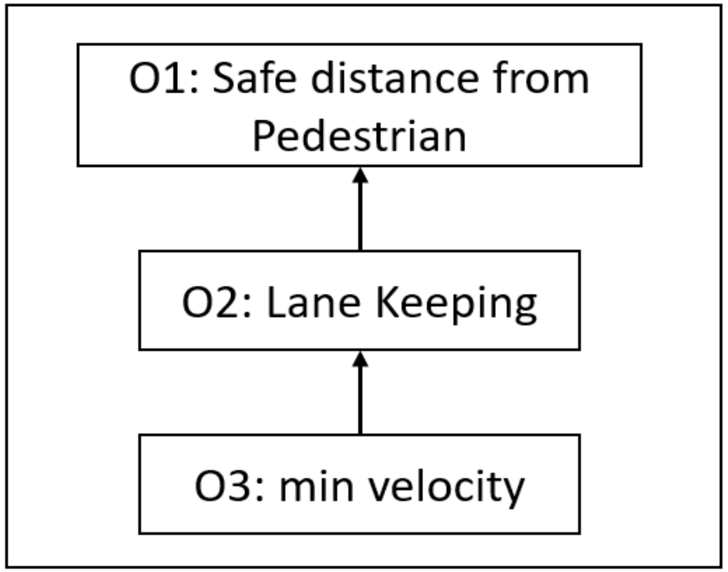
δ_i - CBF Slack Variable

- Gamma is the tuning parameter for CBFs, which is controls the decay rate of the function.

Priority Structure in Constraints

- The constraints in the problem are also assigned certain priority
- The framework recursively relaxes the constraints based on the priority structure
- Here we consider 3 constraints:

1. Safe distance from pedestrian
2. Lane Keeping
3. Minimum velocity



Optimization Problem

Dynamics Model

$$\dot{x} = f(x) + g(x)u$$

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \dot{\theta} \end{bmatrix}$$

Cost Function

$$\min_{u,\delta_e,\delta_i} J = \int_0^{t_f} [c_1 v^2 + c_2 \dot{\theta}^2 + c_3 \delta_e^2 + c_4 \delta_1^2 + c_5 \delta_2^2 + c_6 \delta_3^2]\, dt$$

v	Linear Velocity
ω	Angular Velocity
δ_e	CLF Slack Variable
δ_1	Pedestrian Distance CBF Slack Variable
δ_2	Lane-keeping CBF Slack Variable
δ_3	Minimum Velocity CBF Slack Variable

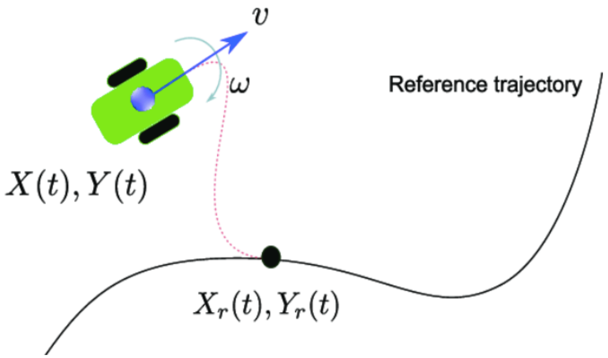
Constraints

1. State and Control

$$v_{min} \leq v \leq v_{max} \quad \forall t \in [0, t_f]$$
$$\dot{\theta}_{min} \leq \dot{\theta} \leq \dot{\theta}_{max} \quad \forall t \in [0, t_f]$$

$$y_{min} \leq y(t) \leq y_{max} \quad \forall t \in [0, t_f]$$
$$X(t_f) = X_{goal}$$

2. Control Lyapunov Function (CLF Constraint)



Lyapunov Function:

$$V(x) = \|y\|^2 = (x - x_{ref})^2 + (y - y_{ref})^2 + (\theta - \theta_{ref})^2$$

Constraint:

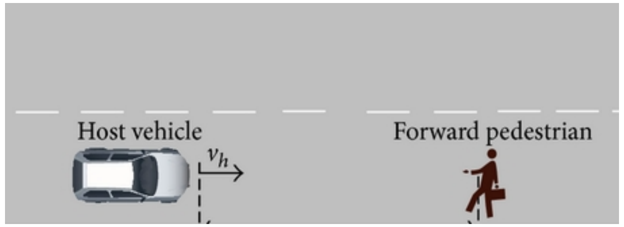
$$2v(x - x_{ref}) \cos\theta + 2v(y - y_{ref}) \sin\theta + 2(\theta - \theta_{ref})\dot{\theta} + \epsilon\|y\|^2 \leq \delta_e$$

Constrained (continued)

3. Control Barrier Function (CBF Constraint)

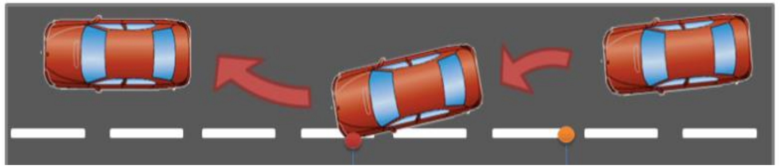
- Pedestrian Avoidance (HOCBF): **Function:** $B(x) = (x - x_p)^2 + (y - y_p)^2 - r^2$

  Constraint: $\gamma_1[(x - x_p)^2 + (y - y_p)^2 - r^2] + 2v(x - x_p)\cos\theta + 2v(y - y_p)\sin\theta - \delta_1 \geq 0$



- Lane Keeping: **Function:** $B(x) = y - l$

  Constraint: $\gamma_2(y - l) + v\cos\theta - \delta_2 \geq 0$

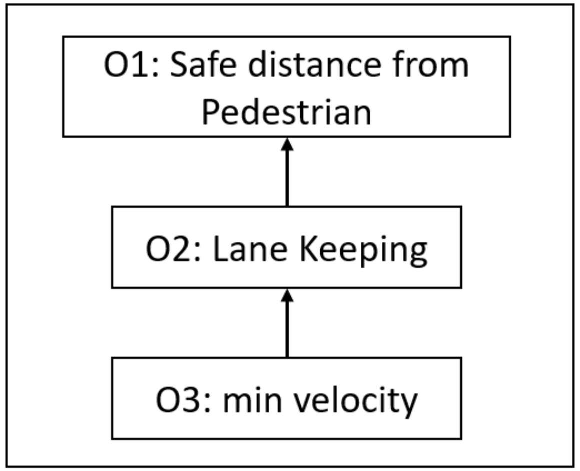


- Min Velocity: **Function:** $B(x) = v - v_{min}$

  Constraint: $\gamma_3 v - \gamma_3 v_{min} - \delta_3 \geq 0$

Priority structure algorithm - Iterative

The algorithm iterates through the power set created which specifies which CBFs are relaxed in a priority order.



Algorithm 1 Recursive relaxation algorithm

Input: System dynamics, initial condition, control bounds, state constraints, rules power set R, reference trajectory $\mathcal{X}$.

Output: Optimal ego trajectory.

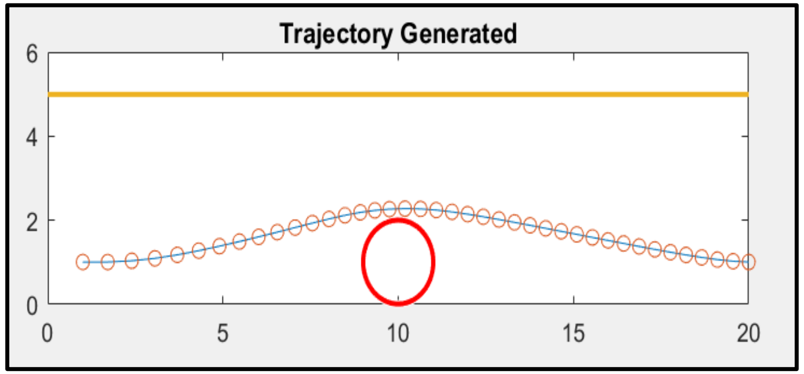
m rules
$k = 0$
while $k \leq 2^m$ **do**
  Get S_k, k^{th} priority scenario of relaxed rules from R.
  Solve optimization problem with S_k
  if the above problem is feasible **then**
    Generate optimal trajectory $\mathcal{X}^*$.
  end if
end while

Power Set

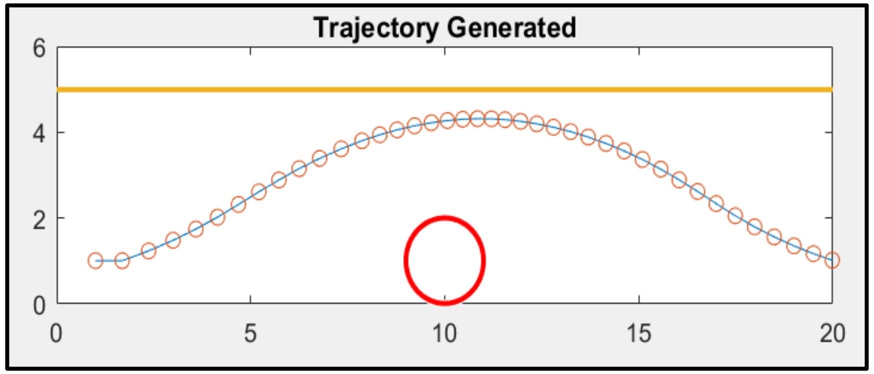
$$\{\{\emptyset\}, \{O_3\}, \{O_2\}, \{O_3, O_2\}, \{O_1\}, \{O_3, O_1\}, \{O_2, O_1\}, \{O_3, O_2, O_1\}\}$$

Tuning Parameters

- Selecting appropriate hyperparameters is very important. They directly affect the performance of the controller.



γ = 0.1



γ = 0.5

- The graph shows the effect of changing hyperparameter γ associated with the Pedestrian safe circle
- Inappropriate values of hyperparameters can also render an infeasible solution

Results

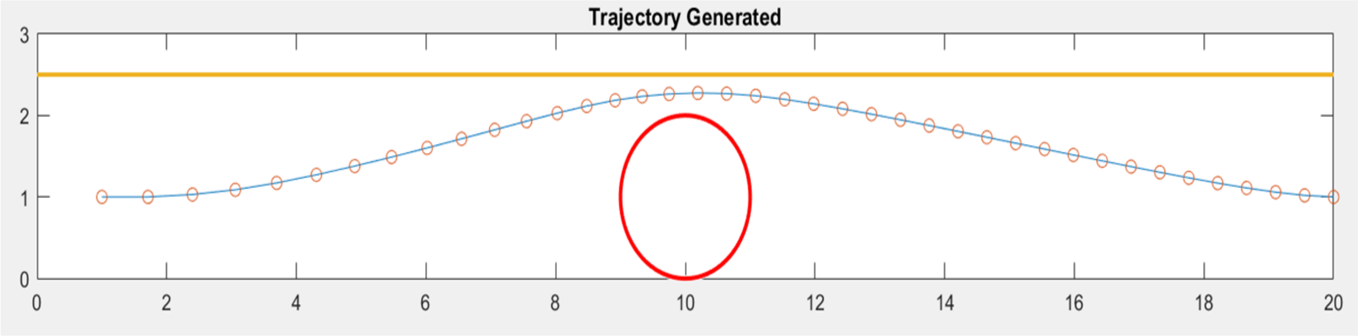
We will look at 4 testing scenarios

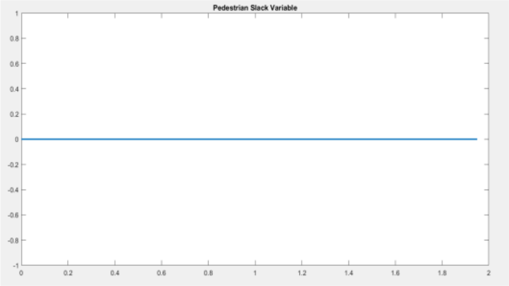
1. No CBF constraints are relaxed (corresponds to relaxing {∅})
2. Iteratively Lane keeping constraint is relaxed (corresponds to relaxing {O2})
3. Iteratively Pedestrian constraint is relaxed (corresponds to relaxing {O1})
4. Iteratively Lane keeping and Pedestrian constraints are relaxed (corresponds to relaxing {O1,O2})

Scenario 1

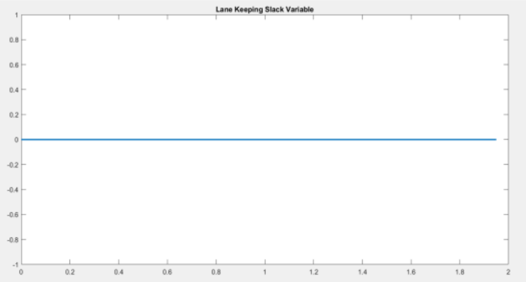
- All constraints act as hard constraints; hence we can see the slack variables for all CBFs to be zero.

- The trajectory follows a path between the gap of lane boundary and pedestrian safe circle.

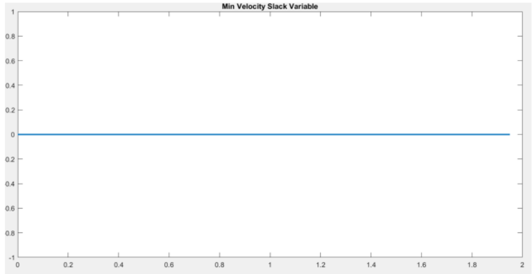




Pedestrian CBF Slack



Lane-keeping CBF Slack



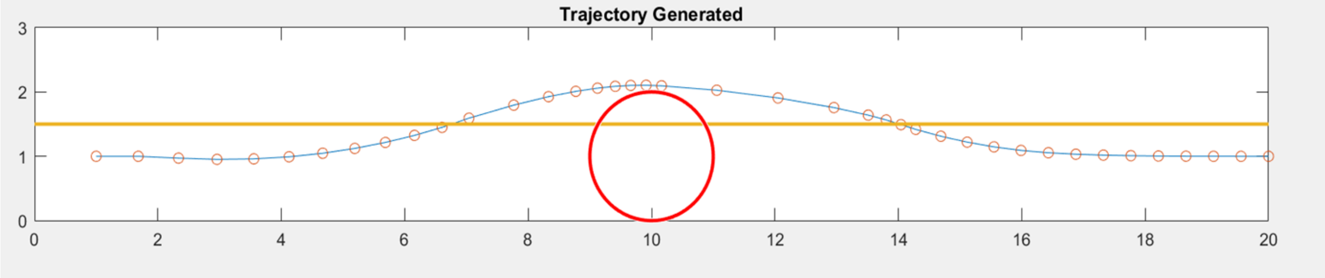
Minimum velocity CBF Slack

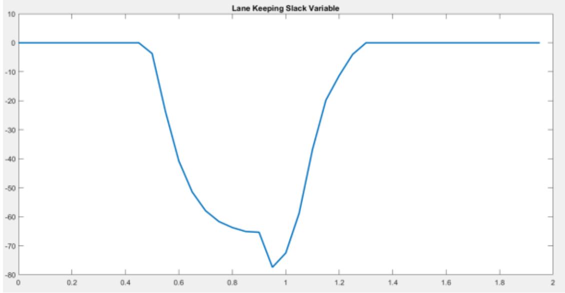
Scenario 2

- Iterative power set finds optimal trajectory when lane keeping constraint is relaxed.

- Lane keeping constraint is violated without violating the pedestrian and min velocity constraint.

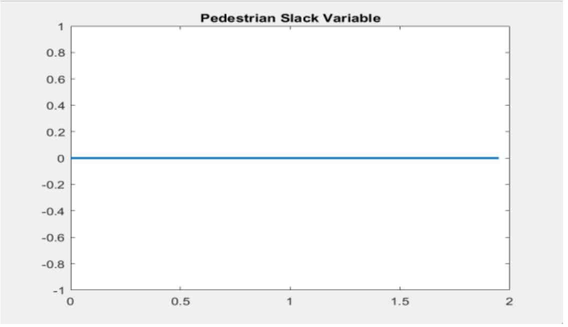
- The lane keeping slack variable has a non-zero value, but only when constraint is violated.



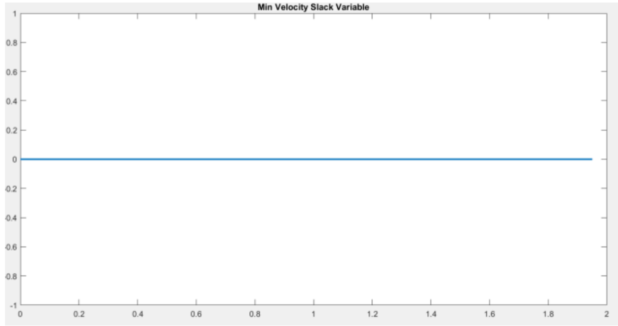


Lane-keeping CBF Slack

Lane keeping slack variable is negative, signifying constraint loosening.



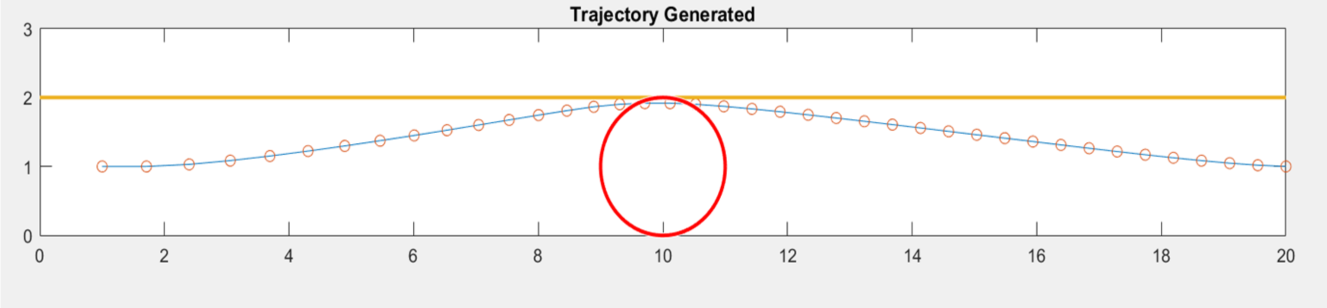
Pedestrian CBF Slack

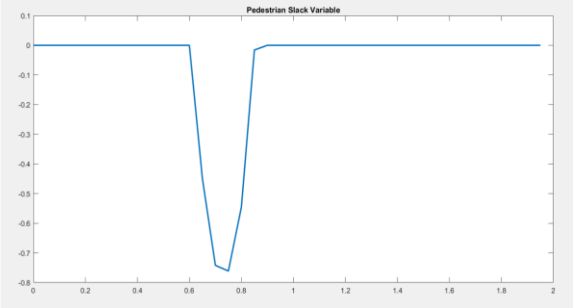


Minimum velocity CBF Slack

Scenario 3

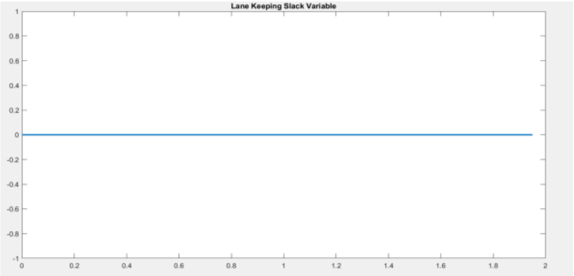
- Here the lane keeping is made a hard constraint for test case purposes.
- We can see that the trajectory violates the pedestrian safe circle, since there is no other feasible trajectory.
- However, it tries to stay as far away as possible from the pedestrian to minimize constraint violation.



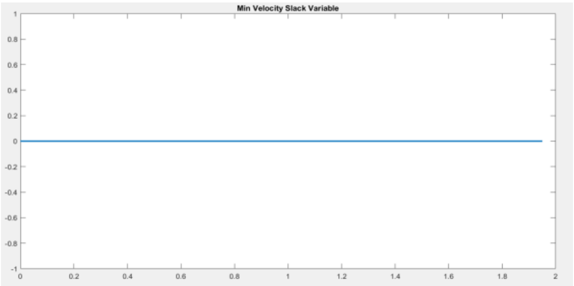


Pedestrian CBF Slack

Pedestrian distance slack variable is negative, signifying constraint loosening.



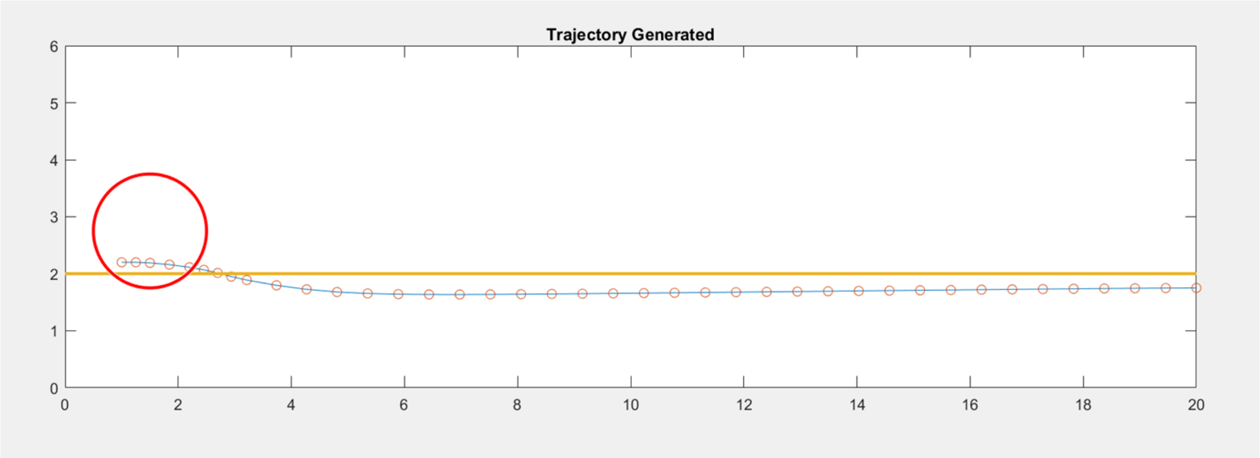
Lane-keeping CBF Slack

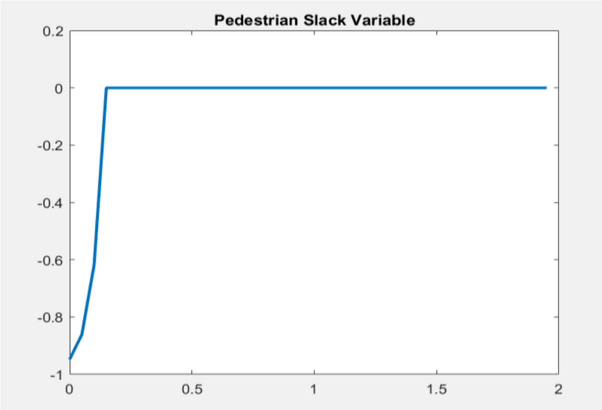


Minimum velocity CBF Slack

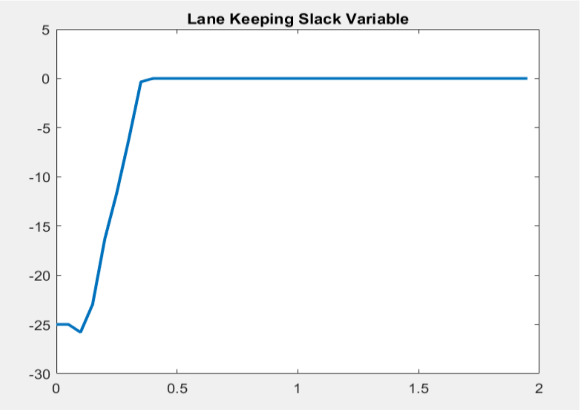
Scenario 4

- Here both lane keeping and pedestrian constraints are relaxed at the starting point.
- We can see that the trajectory tries to move out of the circle as fast as possible.
- However, it tries to exit the pedestrian circle and go inside the lane as soon as possible to minimize constraint violation.





Pedestrian CBF Slack



Lane-keeping CBF Slack

Tools

- MATLAB Optimization Toolbox – non-linear optimization function ‘fmincon’
- C++ - CasADi library and IPOPT optimizer for non-linear optimization

Thank you for your time!

Questions?