Path Planning for Autonomous Vehicles based on Nonlinear MPC with using a 3DOF Kinematic Bicycle Model

Jonas Wagner

***Abstract*—In this project, NOVA's autonomous vehicle, Hail Bopp, is modeled as a 3DOF Kinematic Bicycle Model and then used in a nonlinear MPC-based path-planning approach. The MPC solution is then used in a closed-loop simulation and the feasibility of the approach is explored.**

I. Introduction

Nova is UT Dallas's autonomous driving group that is run by undergrads as part of the Applied Systems Lab. The objective for nova is to develop an open-source autonomous driving stack (Navigator) for use by researchers. Specifically the Applied Systems Lab will eventually apply and test out the cyber-physical system security techniques and how they can be used on autonomous vehicles. [**?**]

Fig. 1. Hail Bopp with labeled sensors.

The primary vehicle used by Nova is Hail Bopp which is a Polaris Gem e6, seen in **??**, and is modified to allow for autonomous operation. There are many sensors added to the system (i.e. GPS, cameras, radar, lidar) that are then used in an entire perception stack to determine the current state of the vehicle and the surrounding environment.

The environment perception stack is shown operating on the simulated vehicle, done in CARLA [**?**], [**?**], along with the test run around campus in **??**.

The author is with the Mechanical Engineering Department at the University of Texas at Dallas, Richardson, TX, USA `jonas.wagner@utdallas.edu`

Additional acknowledgements for this project for Jusin Ruths (research advisor) and Justin Koeln (course instructor).

The future implementation of this work would not be possible without the work done by the amazing undergraduate students within the Nova Lab.

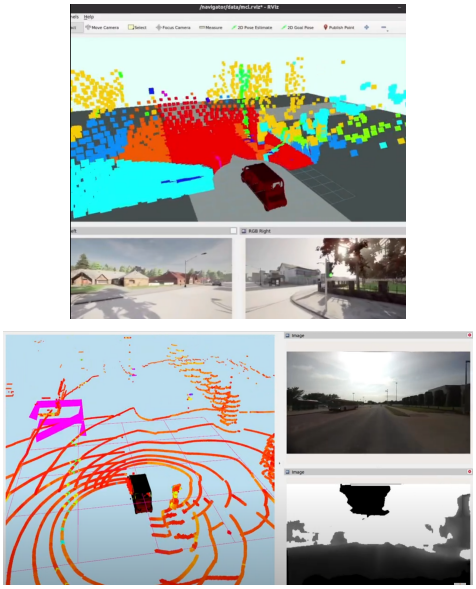

Fig. 2. NOVA perception stack in operation within simulated environment (top) and around campus (bottum).

II. Problem Definition

A. Path Planning

The objective of this project was to create high-fidelity MPC controller produce an optimal trajectory to reach a waypoint given the occupancy map of the environment. This controller will then be used to generate training data for a neural network which will be able to perform an approximation of this controller in real-time.

From the perception stack, the current vehicle states (local and global) will be known to some uncertainty and the surrounding environment will be processed into a predicted occupancy map. This occupancy map will be assumed to already have weights corresponding to where it is safe/ideal for the vehicle to be in the future.

B. Model Definition

1) 3-DOF Bicycle Model: Hail-Bopp is considered to be a standard 4-wheel ackermann-steering vehicle whose approximation to a 3 degree of freedom bicycle model is detailed in [**?**] and [**?**]. A diagram of the bicycle model is shown in **??**.

Due to computation complexity, the kinematic model derivation is used which focuses solely on the relationship between

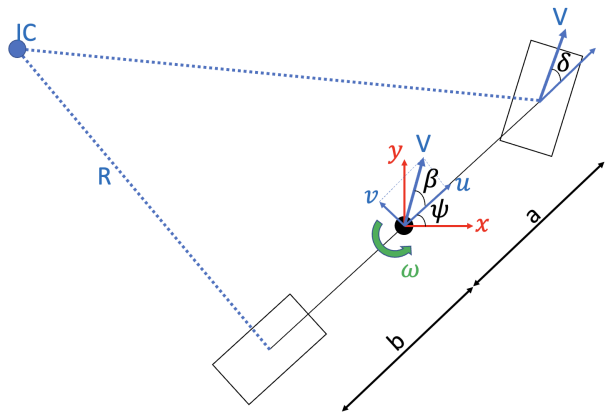

Fig. 3. A diagram of the 3-dof bicycle model.

the system inputs and outputs. This does ignore some dynamics, namely the relationship between the actual forces and torques needed to induce vehicle movement, but still incorporates the important nonlinear relationships between steering angle and the vehicle rotation.

These simple kinematic model equations of motion are given in (**??**):

$$\begin{cases} \dot{x} = V\cos(\psi+\beta) \\ \dot{y} = V\sin(\psi+\beta) \\ \dot{\psi} = \dfrac{V\cos(\beta)}{l_f+l_r}(\tan(\delta_f)-\tan(\delta_r)) \\ \dot{\theta} = \psi \end{cases} \tag{1}$$

where

$$\beta = \tan^1\left(\frac{l_f\tan(\delta_r)+l_r tan(\delta_f)}{l_f+l_r}\right) \tag{2}$$

2) Model Discretization: The nonlinear kinematic model is discretized using an RK4 approximation with a zero-order hold input.

For a CT-nonlinear time-invarient system, $\dot{x} = f_{ct}(x,u)$, the update equation for step-size Δt is calculated as in (**??**):

$$x_{k+1} = f_{RK4}(x_k,u_k) = x_k + \frac{\Delta t}{6}(k_1+2k_2+2k_3+k_4) \tag{3}$$

where

$$\begin{aligned} k_1 &= f_{ct}(x_k,u_k) \\ k_2 &= f_{ct}(x_k+\frac{k_1}{2}\Delta t) \\ k_3 &= f_{ct}(x_k+\frac{k_2}{2}\Delta t) \\ k_4 &= f_{ct}(x_k+k_3\Delta t) \end{aligned}$$

The selected discretization step-size was tested to range between $\Delta t \in [0.01, 5]$ and unlike the Euler update approximation, the RK4 approximation seemed effective at larger time-steps (specifically $\Delta t = 1$ [s] for the selected results).

III. Problem Solution

The MPC Controller for path planning is formulated with the current state and model update equations as hard constraints, an objective to minimize the time and distance to reach the future waypoint, and introducing the occupancy map as soft-constraints within the objective function.

A. Model Constraints

1) Update Equations: Using the discretized model, an update function $x_{k+1} = f_{RK4}(x_k,u_k)$ which is implemented for each timestep.

2) Input Constraints: To represent the input constraints of the actual vehicle, interval constraints upon both the value and change of the inputs:

$$\begin{aligned} V &\in [0,10]\ \text{[m/s]} & \theta &\in \pm\pi/3\ \text{[rad]} \\ \dot{V} &\in [-4,2]\ \text{[m/s}^2\text{]} & \dot{\theta} &\in \pm\pi/10\ \text{[rad/s]} \end{aligned}$$

This is converted into two half-space representation constraints: $h_u(u_k) \le 0$ and $h_{\dot{u}}(u_{k+1},u_k) = h_{\dot{u}\left(\frac{|u_{k+1}-u_k|}{\Delta t}\right)\le 0}$. These half-space representations are visualized as two sets shown in **??**.

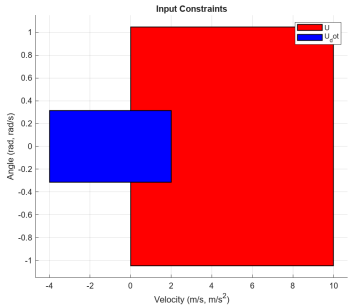

Fig. 4. Halfspace visualization of the input constraints.

B. Cost Function

The primary cost function will be linear from the system state as derived the cost map as the sum of the region that the vehicle would occupy. An example is shown in **??**.

Fig. 5. An example occupancy map provided from the perception stack. The

This is first done by directly converting the occupancy map of the vehicle in positions into a function of the system state, visualized in **??**, as just $f_{map}(x_k)$. Note that this is also normalized and the function itself is currently treated

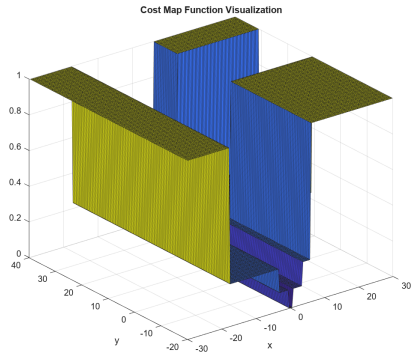

Fig. 6. An example cost map after converting into a direct function of system state and further interpolating.

as a piecewise function (unfortunately causing issues within implementation).

Additionally, an objective will be included that describes the forward progression along a proposed route, $f_{progress}(x_k)$.

As was found out within actual implementation, the occupancy map was additionally simplified to a static function relative to the initial path for each time-step to remove the piecewise-nature or interpolation required by the full cost-map.

The results section will include simulation results for specific demonstrative cost-functions. Note that in the results, the actual path is generated each time-step and the primary visualization as that which closes the loop and actually implements the desired inputs (i.e. actual MPC). In the actual system, the PID would track the generated state-trajectory while the inputs are not explicitly used (i.e. similar to hieratical MPC).

IV. Simulation and Results

The actual system was modeled MATLAB using multiple toolboxes (yalmip [**?**], MPT3 [**?**], gurobi [**?**], and IPOPT [**?**]).

Many simulations were run and the few included here are used to demonstrate the simpler operations. The follwoing parameters were used within these results:

- $N = 15$
- $\Delta t = 1$ [s]
- $h_u(u_k) \leq 0$
- $h_{\dot{u}}(u_{k+1}, u_{k-1}) = \frac{u_{k+1} - u_k}{\Delta t} \leq 0$

The simulations were ran with $T = 10$ [s] and the results for both fmincon and ipopt are provided.

A. 90° *turn*

1) Turn with optimal lane reference: One success includes a 90 degree turn and just maximization of the distance in that direction. The specific cost function specified $f_{map} = (x_k - 0)^2 + (\theta_k - \pi/2)^2$ and $f_{progress} = -y_k$. The closed loop trajectory for both fmincon and ipopt are similar, path compared in **??**, while the computation time is very different with 665 [s] and 85 [s] for fmincon and ipopt respectivly.

The plot of the state and input trajectories for the ipopt simulation is shown in **??**. As can be seen, the input is well planned to turn as much as possible at first and then optimize

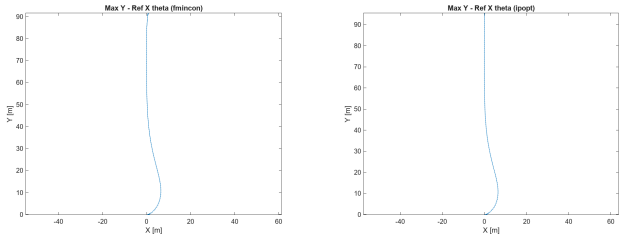

Fig. 7. Quiver plot of the closed-loop simulation for $J_k = -y_k + (x_k - 0)^2 + (\theta_k - \pi/2)^2$ using fmincon (left) and ipopt (right).

for distance in the Y direction by having an overdamped response for returning to the center of the plot. The fmincon

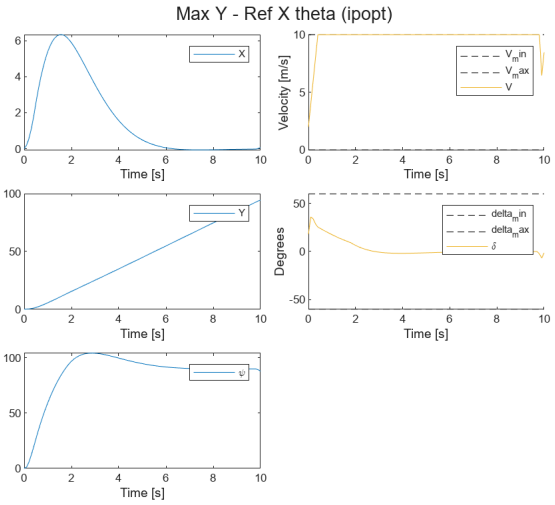

Fig. 8. Closed loop simulation to maximize the Y position with $J_k = -y_k + (x_k - 0)^2 + (\theta_k - \pi/2)^2$ using ipopt.

version is attached in the Appendix as **??**.

2) Removing the cost map term: When testing the ability for the controller to operate with solely an unstructured "go as far that direction", i.e. maximize in the y-direction ($f_{progress} = -y_N$) at the final time-step. The fmincon solution was effective in its goal while the ipopt solution would be classified as a failure. The path comparison can be seen in **??**.

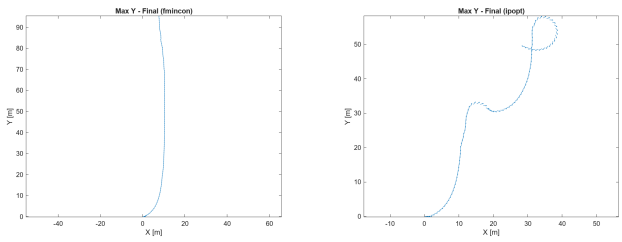

Fig. 9. Quiver plot of the closed-loop simulation for $J = y_N$ using fmincon (left) and ipopt (right).

When looking into the simulation in more detail, it is clear that the fmincon,shown in **??**, followed a similar trajectory as when the cost map component was included, with the biggest

difference being that it does not return to the center within the x direction.

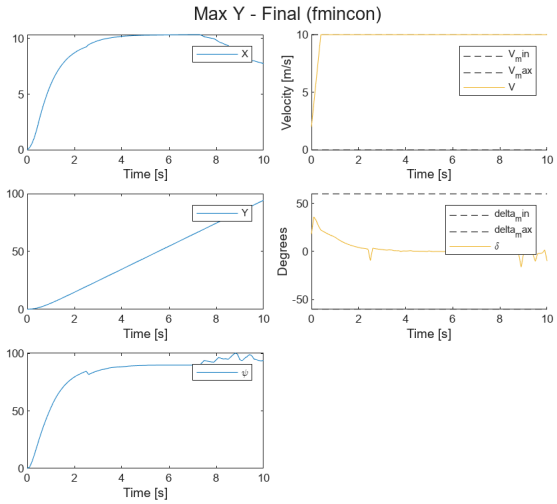

Fig. 10. Closed loop simulation to maximize the Y position with $J_N = -y_N$ using fmincon.

The results for the ipopt solver, shown in **??**, is more interesting and certainly classified as a "failure". Although forward progress made, the vehicle appears to get up to speed as quickly as possible than makes a small mistake in turning until eventually having to correct, over-correct, and then do a donut to hopefully get going in the correct direction again.

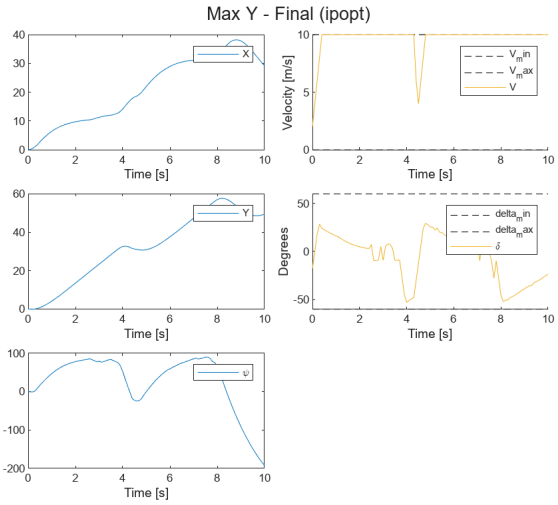

Fig. 11. Closed loop simulation to maximize the Y position with $J_N = -y_N$ using ipopt.

B. U-turn

When testing a more complicated case, such as a U-turn, produced even more interesting results.

1) Unstructured turn around: Implementation with only the $f_{progress}$ term, resulted in different turning behaviors requiring overcorrection for both solvers as seen in **??**. When looking

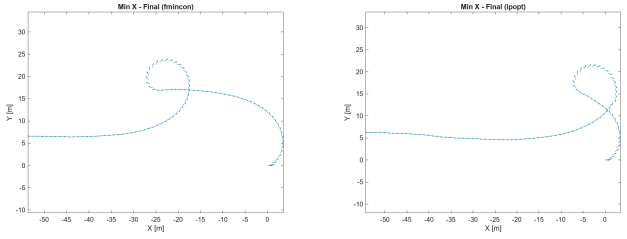

Fig. 12. Quiver plot of the closed-loop simulation for $J = x_N$ using fmincon (left) and ipopt (right).

at the trajectories in more detail, shown in **??** in the appendix, the general shape of the response (turning, under correction, overcorrection, and donut) is similar but with different delays in response.

2) Cost-map term included: Implementation of the U-turn with a cost-map term, $f_{map} = (y_k - 0)^2/100 + (\theta_k - \pi)^2$, was successful for the ipopt case as seen in **??**; however not so for the fmincon case (**??** shown in appendix) as it ended up with way too many overcorrection turns. An interesting

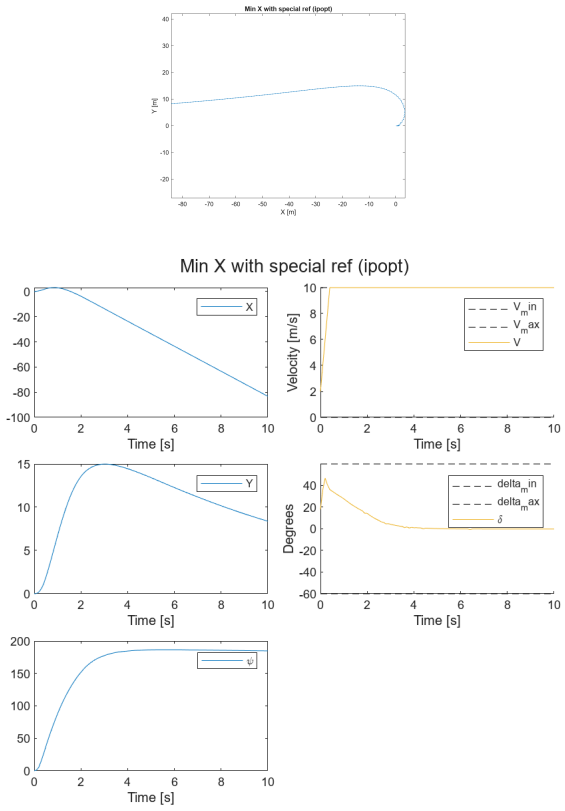

Fig. 13. Closed loop simulation with $f_{map} = (y_k - 0)^2/100 + (\theta_k - \pi)^2$ using ipopt.

note about the f_{map} term is that if the cost map function ends up being a reference to $-180°$ instead of $+180°$, $f_{map} = (y_k - 0)^2/100 + (\theta_k + \pi)^2$, then the vehicle would do the U-turn in the other direction.

V. Conclusion

The biggest conclusion from this project is that that current nonlinear MPC setup is not a feasible solution for NOVA to do path planning, even in the case for just generating training data for the neural network. It was found that it is possible to create specific f_{map} functions that could be effective; however the creation of these from the discrete values provided from the perception stack would be difficult to create with no gaurentee that it may work.

Additionally, this project was a major demonstration of how using different optimization solvers can have a major impact on the computation time, but also the results themselves. A rough testing comparison of the different testing times is included in **??** and clearly IPOPT was far faster in every test, but even so it is not nearly fast enough to generate a lot of training data let alone for online operation.

TABLE I
Computation time comparison between FMINCON and IPOPT

	FMINCON	**IPOPT**
Min Y - Unstructured	243.79	164.75
Min Y - Final	235.91	155.6
Min Y - Ref X and theta	438.67	71.79
Min X - Final	**1077.6**	**164.58**
Min X with +180	395.71	191.69
Min X with special ref	**344.52**	**165.16**
Max Y - Final	**331.71**	**140.49**
Max Y - Ref X theta	**665.82**	**85.492**

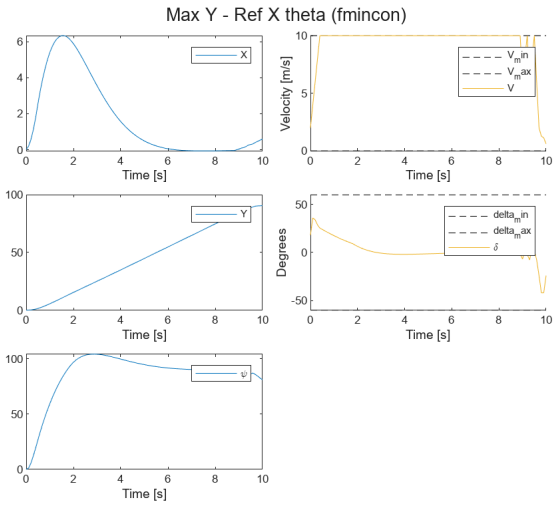

Fig. 14. Closed loop simulation to maximize the Y position with $J_k = -y_k + (x_k - 0)^2 + (\theta_k - \pi/2)^2$ using fmincon.

References

[1] NOVA, "nova-utd.github.io," 2023. [Online]. Available: https://nova-utd.github.io/

[2] carla simulator, "Carla." [Online]. Available: https://github.com/carla-simulator/carla

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[4] D. Casanova, "On minimum time vehicle manoeuvring: The theoretical optimal lap," 2000.

[5] P. P. Ramanata, "Optimal vehicle path generator using optimization methods," Ph.D. dissertation, Virginia Tech, 1998.

[6] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in matlab," in *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[7] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proc. of the European Control Conference*, Zürich, Switzerland, July 17–19 2013, pp. 502–510, http://control.ee.ethz.ch/~mpt.

[8] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com

[9] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:14183894

Appendix

Github

See my github repo for this project for all the source code and additional results: https://github.com/jonaswagner2826/MECH6V29_MPC_FinalProject

Matlab results

MATLAB code is included in submission.

Additional Results

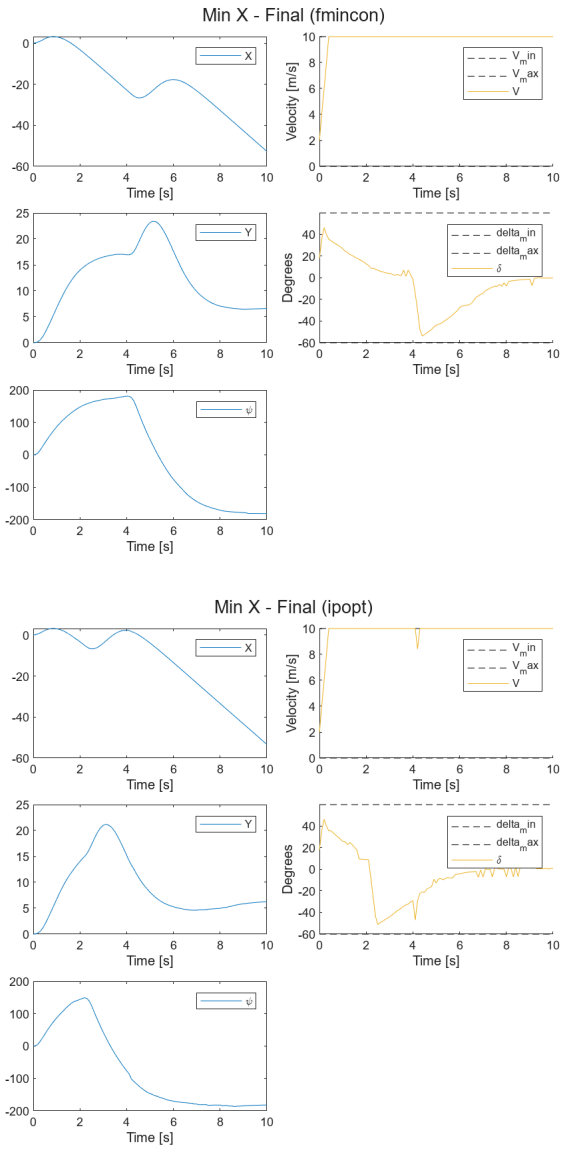

Fig. 15. Closed loop simulation to maximize the Y position with $J_N = x_N$ using fmincon(top) and ipopt (bottom).

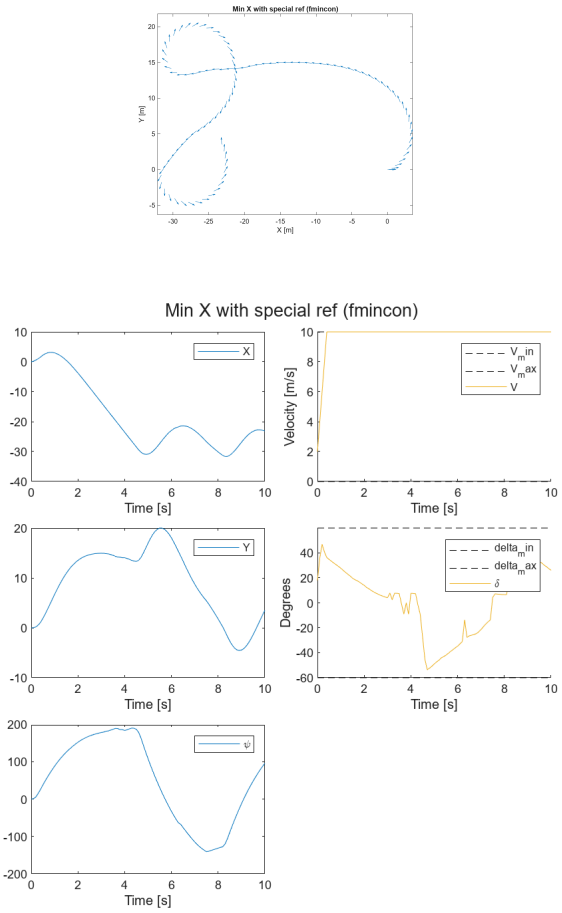

Fig. 16. "Failed" fmincon case for the cost-map term version of a U-turn