M-Analysis for non-convex motion planning using MPC framework*

Jessica Leu[1]

Abstract— **Real-time, safe, and stable motion planning in cluttered environments remains challenging due to the non-convexity of the problem. This paper investigates motion planning in the framework of model predictive control (MPC). As there are many local optima due to non-convexity, it is important to guarantee stability of the non-convex MPC such that the trajectories will not jump around multiple local optima. In order to analyze such a stability problem, a notion of M-analysis is introduced in this paper, which guarantees finite convergence (at least M steps ahead) of the planned trajectories. With such notion, we verify the stability properties of a non-convex MPC which implements the convex feasible set algorithm (CFS) at every MPC step through extensive simulations and experiments. The M-analysis for stability properties provides a tractable tool to understand dynamics of non-convex MPC.**

I. INTRODUCTION

The development of intelligent robots and autonomous vehicles is craving for real-time, safe, and stable motion planning in cluttered dynamic environments. For example, an automated guided vehicle (AGV) on factory floors [1] needs to decide in real time how to bypass multiple human workers and a set of obstacles in the environment in order to approach its target efficiently [2], [3].

In literature, motion planning algorithms usually fall into three categories: graph-search based algorithm, sampling-based algorithm, and optimization-based algorithm. Among these algorithms, this paper focus on model predictive control (MPC) [4], an optimization-based algorithm. MPC has been widely adopted both in academia research and in industrial applications. The fact that MPC controller observes the environment every time before solving the optimization problem allows the system to adjust and re-plan according to the changes of the environment. Its ability to handle input and state constraints also makes it popular in addressing motion planning problems. If the problem is a convex MPC problem, the stability of the closed loop system can be guaranteed by modifying prediction horizon, adding terminal cost, adding terminal equality constraint, or using terminal set constraint instead [5], [6]. These methods can guarantee that the calculated commands and the resulting states are bounded in the presence of bounded disturbance. One common application of motion planning MPC problem is autonomous vehicles [7]. In literatures, stability is guaranteed by setting stability boundary for the control system, i.e., stability is quantified at several vehicle speeds.

[1]Jessica Leu is with the Department of Electrical Engineering, University of California, Berkeley, CA 94720 USA jess.leu24@berkeley.edu

However, the existence of obstacles in the environment introduces non-convex state constraints, which results in non-convex MPC problems. It is still challenging to obtain real-time, safe and stable solutions for non-convex MPCs. An efficient optimization algorithm, i.e., the convex feasible set (CFS) algorithm [8], has been proposed to obtain a safe open-loop trajectory in real time. However, there are many local optima due to non-convexity. Therefore, even the trajectory calculated by CFS at each MPC time step is feasible and smooth, it is possible that the trajectories planned at different time steps will jump around multiple local optima causing the implemented trajectory to be dynamically unreasonable for the robot. Dynamically unreasonable trajectory will cause the robot to execute violent or zigzagging movement that is harmful to the robot's motors and also scares human in the environment. Some work focused on MPC problems with non-convex cost function using the sequential convex optimization method [9]. Although the result is promising, the difficulty of ensuring closed-loop stability and analyzing controller performance for non-convex MPC is also mentioned in the literature.

A standard analytical approach to verify properties of non-convex MPC is needed so that we can estimate whether the trajectory implemented by the MPC controller is smooth and desirable, i.e., the trajectories planned at different time steps do not jump around multiple local optima, and the result of this analysis can serve as an indication of the MPC controller's stability properties.

The contribution of this paper is to provide a new method to analyze the stability of this kind of non-convex MPC problem. In this paper, we first introduce a new notion of stability property called M-analysis and then analyze the properties of the proposed algorithm, non-convex MPC that implements CFS (MPC-CFS). The relationships among the optimal trajectories generated at different steps will be examined, so as to guarantee that the robot trajectory will not experience sudden jumps during trajectory execution. The proposed method addresses the limitation of traditional stability analysis, and can further assure that the planning result is executable for the robot. Simulation studies are performed to test the performance of the implemented CFS algorithm in the MPC framework. The stability features are analyzed using the proposed method. Finally, experiment is carried to verify the algorithm.

The remainder of the paper is organized as follows. Section II provides the problem formulation. Section III shows simulation results. Section IV shows the experiment result (code is publicly available at `github.com/msc-berkeley/MPC-CFS-Matlab`). Section

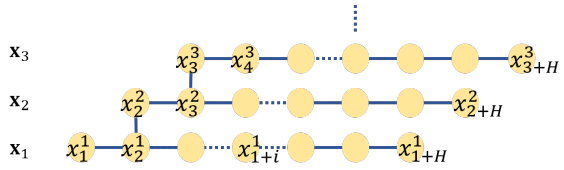


Fig. 1: The execution structure of MPC

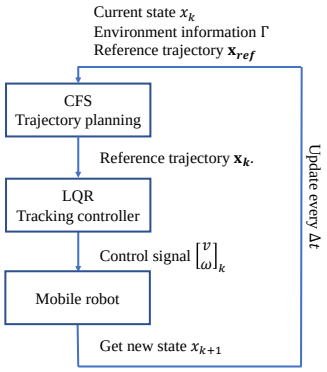


Fig. 2: The overall system control design.

V concludes the paper.

II. PROBLEM FORMULATION

A. Problem and Notations

In MPC (Fig. 2), at each time step t, a future trajectory will be planned by solving an optimization problem. This trajectory is denoted as $\mathbf{x}_k := [x_k, x_{k+1}, x_{k+2}, \cdots, x_{k+H}]$ where the state, x_k, is called the k^{th} action location. H is the prediction horizon. In this paper, $x_k = [x(k) \quad y(k)]^\intercal$ that contains the x and y coordinate of the robot's location in 2-dimensional Cartesian space at time step $t = k$. Note that x_k corresponds to the current position at time step $t = k$. The trajectory will then be executed and the robot will go to x_{k+1}, the $k+1^{th}$ action location, which is the planned action location for the robot at time step $t = k + 1$. The sampling time between two time steps is a constant, denoted as Δt. After reaching the next action location, the robot will again solve the optimization problem and plan a new trajectory and repeat the process. To avoid confusion, the current time step can also be marked as superscript in some cases , e.g., x_{k+1}^k means the planned action location x_{k+1} at time step $t = k$.

At time step $k = t$, given the current state, denoted as $x_0(t = k) = (x, y)|_{t=k}$, the following optimization needs to be solved to obtain $\mathbf{x}_k$,

$$\begin{aligned} \min_{\mathbf{x}_k} \quad & J(\mathbf{x}_k), && (1)\\ s.t. \quad & x_{k+i} \in \Gamma, \forall i = 1, \ldots, H, && (2)\\ & x_k = x_0(k). && (3) \end{aligned}$$

The current state is measured and assigned to the first entry of $\mathbf{x}_k$. Note that the optimization problem for each time step is time-invariant. There are two assumptions in this formulation:

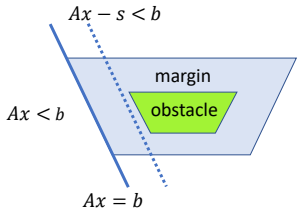


Fig. 3: Illustration of the slack variable.

Assumption 1 (Cost): The cost function is convex and regular, and has the following form

$$J(\mathbf{x}_k) = C_1\|\mathbf{D}\mathbf{x}_k - \mathbf{d}\|_2^2 + C_2\|\mathbf{V}\mathbf{x}_k - \mathbf{v}_{ref}\|_2^2 + C_3\|\mathbf{A}\mathbf{x}_k\|_2^2, \quad (4)$$

where C_1 is the coefficient of the first term, which penalizes the robot's deviation from a reference line so that the robot output trajectory is not too irregular. Matrix $\mathbf{D}$ is a projection matrix that extract all y's from $\mathbf{x}_k$. C_2 is the coefficient of the second term, which penalizes the speed profile of the planned trajectory with regard to a constant speed so that the robot will be time efficient. $\mathbf{V}\mathbf{x}_k$ is the velocity vector. Here, the speed reference is set to be a constant speed going along the positive x-axis. C_3 is the coefficient of the third term, which penalizes the acceleration of the output trajectory so that the motion will be smooth. $\mathbf{A}\mathbf{x}_k$ is the acceleration vector.

Assumption 2 (Constraint): The state constraint Γ is non-convex and its complement is a collection of disjoint convex sets, i.e., each of the obstacle-region is itself convex.

Since the robot dynamics is not included in the MPC problem, a tracking controller is needed. The overall system is shown in Fig. 2. Although the planned trajectory is feasible, tracking error will occur in real world experiment, causing the state ends up to be infeasible. This results in a violent motion because the control command according to the new planning result will immediately pull the robot out of the infeasible area. In order to avoid such problem, we introduce $\mathbf{S}$, the slack variable vector. Introducing slack variables allows the states to violate the original constraint, which is the margin boundary of the obstacles (Fig. 3). However these slack variables are also added to the cost function so that the violation is penalized [10]. The new problem is shown in the following:

$$\begin{aligned} \min_{\mathbf{x}_k} \quad & J(\mathbf{x}_k) + \|\mathbf{S}\|_2^2, && (5)\\ s.t. \quad & x_{k+i} \in \Gamma(s_{k+i}), \forall i = 1, \ldots, H, && (6)\\ & x_k = x_0(k). && (7) \end{aligned}$$

The final trajectory is $[x_k^k, x_{k+1}^{k+1}, \ldots]$, which is equivalent to $[x_k^{k-1}, x_{k+1}^k, \ldots]$ (Fig. 1). As the problem is non-convex, it is possible that the planned trajectories calculated at different time steps enter into different local optima, hence the stability is hard to characterize.

B. M-Analysis for analyzing stability properties

In this paper, we propose a new way, M-analysis, to analyzing stability properties. We say that an action location,

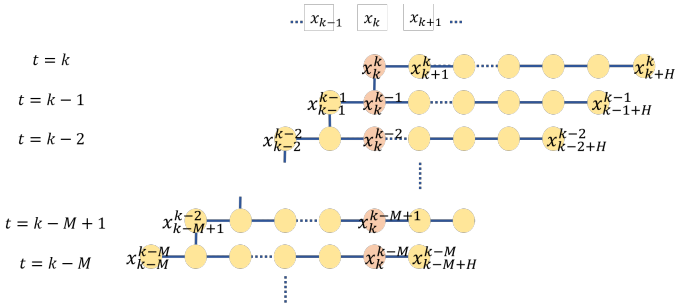


Fig. 4: Illustration of M-analysis.

x_k, is M-settling if for $k - M < t \leq k$, the last M time steps at x_k satisfy: $\|x_k^t - x_k^{t-1}\| \leq \|x_k^{t-1} - x_k^{t-2}\|$ or $\left|\|x_k^t - x_k^{t-1}\| - \|x_k^{t-1} - x_k^{t-2}\|\right| \leq \delta$, where δ is a small threshold. In Fig. 4, the pink-orange-color circles are the action location x_k planned at different time steps that are tested for M-settling property of x_k. The notation implies that the planned state x_k would have smaller and smaller change on each predicted action location between consecutive time steps after time step $k - M$. Under the condition that the environment in the future time step is not going to change much, we say that the non-convex MPC is M-settling if the above inequalities holds for all k. This result can also be expressed as saying the output solutions are all close to one local optimum after time step $k - M$. Note that the problem is always 1-settling, since $\|x_k^t - x_k^{t-1}\| = 0$.

Having such property is good for the robot. M-analysis guarantees that the action location will not change much, and therefore, guarantees smoothness of the output trajectory. If M is sufficiently large, the robot system will not experience sudden change. Moreover, since the planned trajectory is usually tracked by a low-level tracking controller, the larger the M is, the smoother the control commend would be.

C. The Convex Feasible Set Algorithm

Because the problem we are solving has non-convex state constraints, the problem is a non-convex MPC problem. Here we solve the optimization problem at each MPC time step using the convex feasible set algorithm (CFS), which iteratively solve a sequence of sub-problems of the original non-convex problem using convex constraints, e.g., convex feasible set. The CFS algorithm in the MPC structure uses the previous solution, the planned action locations, as a reference. The convex feasible set for a reference point x_r is computed as $\mathcal{F}(x_r) = \{x : A(x_r)x \leq b(x_r)\}$ where $A(x_r)$ is a matrix and $b(x_r)$ is a column vector. At time step $k+1$, the reference is set as $\mathbf{x}_{k+1}^r = [x_{k+1}^k, x_{k+2}^k, \ldots, x_{k+H}^k, x_{k+H+1}^*]$ where

$$x_{k+H+1}^* = \arg\min_{x_{k+H+1}} \|x_{k+H+1}\|_Q^2 + \|x_{k+H}^k - x_{k+H+1}\|_R^2 + \|x_{k+H-1}^k - 2x_{k+H}^k + x_{k+H+1}\|_P^2. \tag{8}$$

If $x_{k+H+1}^* \in \Gamma$, then the optimal solution is $\mathbf{x}_{k+1}^o = \mathbf{x}_{k+1}^r$. If $x_{k+H+1}^* \notin \Gamma$, denote the feasible solution as

$$\bar{x}_{k+H+1} = \arg\min_{x_{k+H+1} \in \Gamma} \|x_{k+H+1}\|_Q^2 + \|x_{k+H}^k - x_{k+H+1}\|_R^2 + \|x_{k+H-1}^k - 2x_{k+H}^k + x_{k+H+1}\|_P^2. \tag{9}$$

Moreover, denote
$\mathbf{x}_{k+1}^u := [x_{k+1}^k, x_{k+2}^k, \ldots, x_{k+H}^k, \bar{x}_{k+H+1}]$.

Because of the construction above, the optimal solution is $\mathbf{x}_{k+1}^o$ at step $k+1$ satisfies that

$$\mathbf{x}_{k+1}^o = \arg\min_{x_{k+i} \in \mathcal{F}(x_{k+i}^o)} J(\mathbf{x}_{k+1}). \tag{10}$$

Note that $J(\mathbf{x}_{k+1}^r) \leq J(\mathbf{x}_{k+1}^o) \leq J(\mathbf{x}_{k+1}^u)$, therefore, the cost of the trajectory is always bounded. The executed trajectory is from those $\mathbf{x}_{k+1}^o$ for different k.

D. Converging property with CFS

In this paper, we will show that the system is stable through simulation. Theoretical proof is left as future work. But here we sketch the procedures. First, we can show that at two consecutive time steps, the difference between the early state should be strictly smaller than the difference between any future state, i.e., $\|x_{k+i}^{k+1} - x_{k+i}^k\| < \lambda \|x_{k+i+1}^{k+1} - x_{k+i+1}^k\|$ for some $\lambda < 1$ and for i sufficiently small. Then we can show that the state is bounded and the difference between the same state at different time steps keeps decreasing.

III. SIMULATION RESULT AND DISCUSSION

The simulation scenario in this work is similar to that of mobile robots operating in factories. To test our algorithm, three kinds of scenarios will be considered. The first one is a static scenario where the environment is completely known, while the other two are dynamic scenarios where the robot will observe changes in the environment. The goal for the robot is to move along a line, $y = 0$, in the positive x-axis direction, while maintaining a constant speed which also points along the positive x-axis.

A. Result of scenario with single static obstacle

In this scenario, there is only one static obstacle. This is the scenario as discussed in section 2, where the MPC problem is time invariant.

The simulation result is shown in Fig. 5a. In the figure, the planned trajectory is marked by gray-star-line of which the gray-color gets darker as time step increases. We can see that the robot successfully track the line while avoiding collision. It is clear from the figure that the last several action locations planned at each time step are one on top of another. This can be taken as an indication of stability.

Figure 5b shows the number M for M-analysis at each action location. The result in Fig. 5b shows that the system is 20-settling once it has run for a sufficient period of time. Note that 20-settling is the highest possible status that a system with prediction horizon of 20 can reach, which is the case in this simulation. If we decrease the sampling time to one fifth of the original, i.e, $\Delta t_{new} = 0.2\Delta t_{original}$, it

takes less time for the system to reach 20-settling (Fig. 5b). In the figure, the system reaches 20-settling at the 49^{th} action location. The run-time at this point is almost equivalent to the run-time at the 10^{th} action location in Fig. 5b. This result indicates that decreasing sampling time will increase M.

Notice that there is a high peak in M around the 35^{th} action location in Fig. 5c (and a jump at 6^{th} in Fig. 5b). This is because these action locations around the peak are close to the obstacle and do not have much space to be adjust. Therefore, these action locations do not change much throughout time and M for M-analysis is higher. We call these action locations which are fixed due to the existence of the obstacle critical cation locations. After passing the critical action locations, the robot again has freedom to adjust its planning to best fit the cost function. The adjustment causes M to drop, but soon after, the system reaches to 20-settling and perfectly track the line with constant speed.

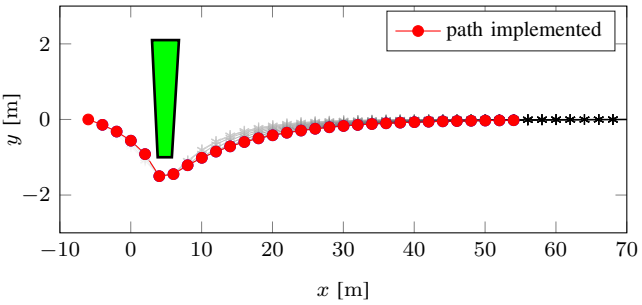
Another way to analyze stability is to look at the cost change. Define path_k as $\mathbf{x}_k^p := [x_k^{k-1}, x_{k+1}^k, \ldots]$ (Fig. 6a). Increase of the index k means the path is starting further and further away from disturbance, i.e., the obstacle, which is located close to the robot's initial location. Therefore, it is expected that the cost will decrease as k increases, which indicates that the robot is stably doing better and better according to what the cost function wants it to do. This expectation meets perfectly with Fig. 6b.

B. Result of scenario with initially unknown static obstacle

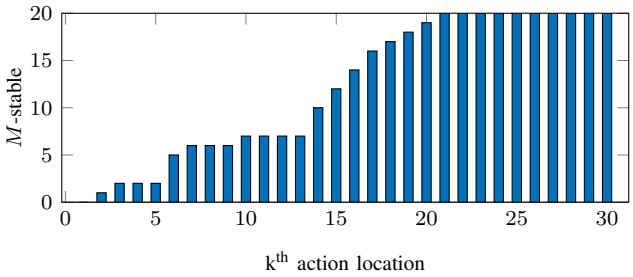
In this scenario, the robot does not know all the obstacle location at the beginning, and has only limited "eye sight," i.e., only the information of the environment within the range starting from 20 meters ahead to its current position. The robot is expected to execute this MPC motion planning and adjust it's planned trajectory to avoid collision once it "sees" the obstacle.

The result of this scenario is shown in Fig. 7a. At the beginning, the third obstacle on the right is unknown to the robot. After detecting the third obstacle, the robot corrects its planned trajectory to avoid the obstacle and completes its intention successfully.

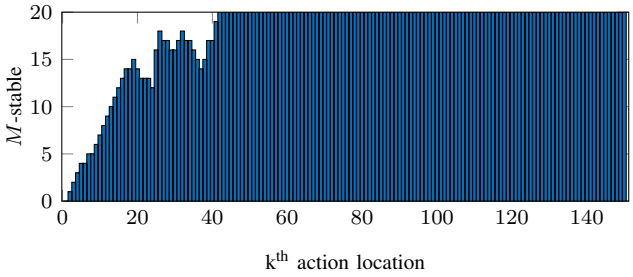
The M-analysis discussed previously can also be applied to this scenario even though the environment is changing. Since the analysis strategy of this work focus on local conversions at each action locations, i.e., every point in the trajectories, the analysis can still be done although the planned locations change a lot due to environment changes. This directly demonstrate the strength of the proposed M-analysis. The result of M-analysis is shown in Fig. 7b. From the plot we can see that M begins to increase slower after the 16^{th} action location, this is exactly where the robot detects the obstacle and plans the new trajectory to avoid collision. The 26^{th} to the 30^{th} are the critical action locations, therefore, have higher M. After passing by the obstacle, the robot adjusts its planning to best lower the cost and then track the line with constant speed steadily. Throughout the process from detection to reaching 20-settling, the system is at least 14-settling. Once the environment stops changing, the robot



(a) Simulation result of single static obstacle.



(b) M-analysis for scenario that has known static obstacle.



(c) M-analysis for scenario that has known static obstacle with increased sampling rate.

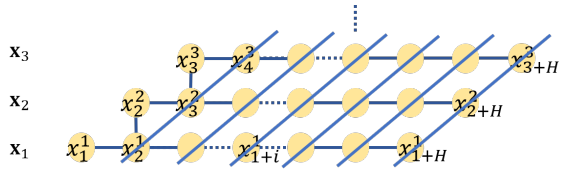
Fig. 5: Scenario that has known static obstacle.

system reaches to 20-settling, which agrees with the result from the static-obstacle-scenario.

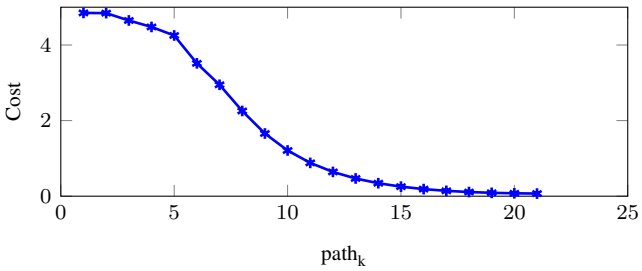
C. Result of scenario with initially unknown moving obstacle

In this scenario, there exists a moving obstacle, which is not observed by the robot at the beginning. Once the robot sees this obstacle, it also gets the information of the obstacle's dynamics, and therefore, is able to predict the future position of the obstacle and conduct planning accordingly.

In Fig. 8a, the existence of the third obstacle on the right is unknown to the robot at the beginning. The obstacle is assumed to be moving in constant speed once detected. However, the obstacle will change its speed at some point after detection. In the simulation, after time step $t = 20$, the obstacle changes its speed and from moving along negative x-axis to positive y-axis. From Fig. 8a (b), we can see that the robot adjusts the planned trajectory to best utilize the space once it detects the speed change and avoids the obstacles successfully. From the M-analysis plot of this scenario (Fig. 8b), we can also see that the system maintains

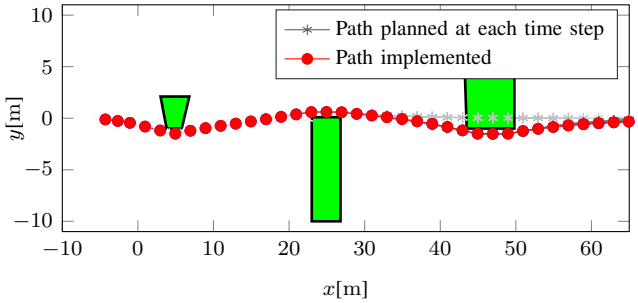


(a) Illustration of path$_k$ (the diagonal lines).

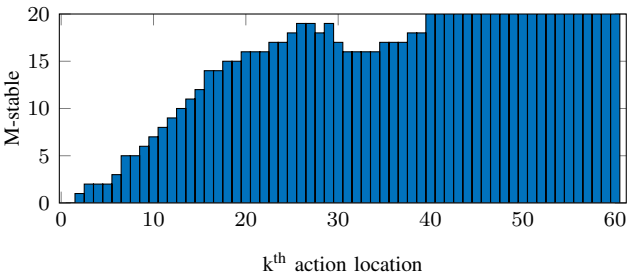


(b) Cost VS path.

Fig. 6: Convergence of path cost.



(a) Simulation result of scenario that has initially unknown static obstacle.



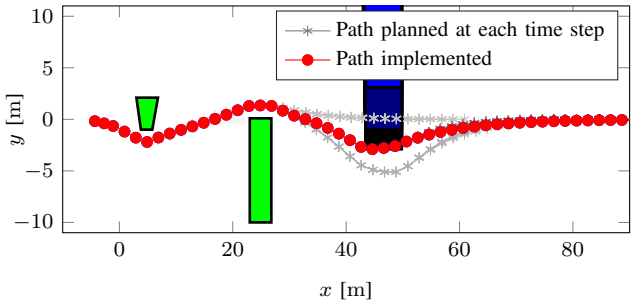
(b) M-analysis for scenario that has initially unknown static obstacle.

Fig. 7: Scenario that has initially unknown static obstacle.

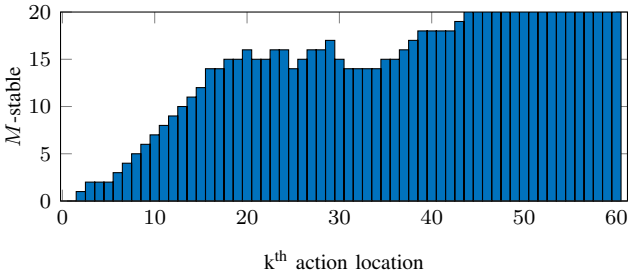
15-settling even though the environment is changing though out time. The result indicates that as long as there is no sudden change accruing close before an action location is executed, the robot can best utilize the new information and successfully plan a smooth trajectory.

D. M-analysis with less stable trajectory

This section shows that M-analysis can also point out the action locations in the implemented trajectory which is dynamically unreasonable. Such trajectory occurs when the environment changes rapidly. In this simulation scenario, we use random initial trajectory and also make the blue obstacle oscillate its moving speed between -1 and 1 on the



(a) Simulation result of scenario that has initially unknown moving obstacles (with speed change).



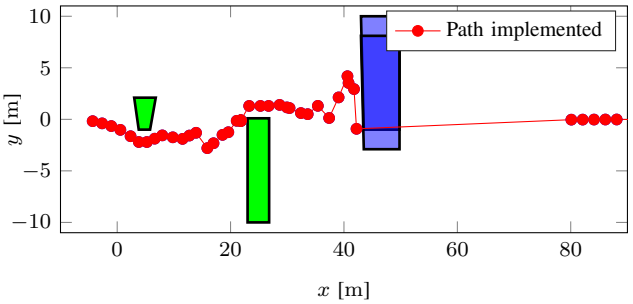
(b) M-analysis for scenario that has initially unknown moving obstacles (with speed change).

Fig. 8: Scenario that has initially unknown moving obstacles (with speed change).

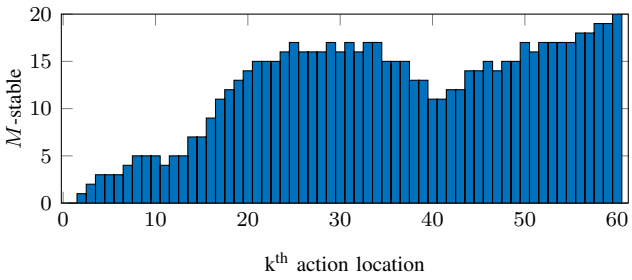
y direction at every time step. From Fig. 9a we can see that the action locations near the obstacle (around $k = 40$) have a zigzagging shape and in Fig. 9b, these action locations also show a decrease in M. This indicates that the trajectory is jumping among local optima and we can say that the MPC controller performance is not as stable compared with the previous scenarios.

E. Relationship between M and H

In industries, choosing a prediction horizon to be large enough is a common way to guarantee stability [5]. Therefore, we examined the relationship between the prediction horizon H and the resulting M for M-analysis. The scenario that has unknown moving obstacle is used here. Define the largest possible M at each action location as $M_{max,k}$, where $M_{max,k} = k - 1$ for $k \leq H + 1$, and $M_{max,k} = H$ for $k > H + 1$. In Fig. 10, systems with larger ($H > 20$) prediction horizon reach 20-settling (which is a guarantee that the location will not change much within 20 sampling steps before it is executed) faster than the system with $H = 20$. Interestingly, all these systems reach $M = 20$ in the range after $k = 27$, which is the critical action location, but before $k = 32$, whereas system with $H = 20$ reaches 20-settling not until after $k = 45$. Figure 11 shows the smallest M after k reaches $H+1$ for each prediction horizon. It is clear that the smallest M among all action location increase with H. Note that large H makes the problem computationally heavy, which should be avoided if possible. We conclude that the horizon can be considered large enough



(a) Simulation scenario resulting in dynamically unreasonable trajectory.



(b) M-analysis for scenario that resulting in dynamically unreasonable trajectory.

Fig. 9: Scenario resulting in dynamically unreasonable trajectory.

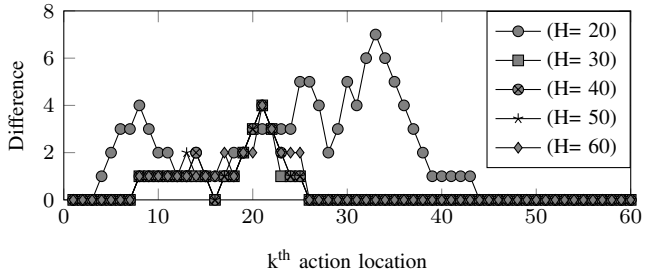


Fig. 10: Difference between simulated M and $M_{max,k}$ for $H = 20$ (with speed change).

if it can cover the last critical action location, in this case, when $H > 27$.

IV. EXPERIMENT RESULT

A. Experiment set up

To verify whether the planned trajectory is dynamically reasonable for a mobile robot to execute, the MPC-CFS (with $H = 20$) controller is tested on TurtleBot3, which is developed by ROBOTIS. TurtleBot3 is a ROS standard platform robot. The MPC-CFS controller is running in MATLAB on a separate laptop with an 2.8GHz Intel Core i7-7700HQ.

Since the dynamics is not included in the MPC problem, we need an additional tracking controller to calculate the suitable command for the robot to execute the planned trajectory. Here we used an iterative LQR (ILQR) controller [11]. The cost function is designed to minimize the tracking error and also the robot's acceleration. The final control input

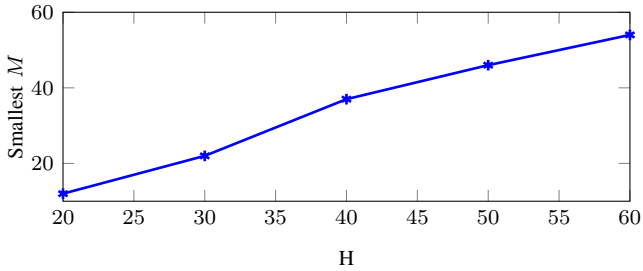


Fig. 11: Smallest M for each prediction horizon.

at the first time step is chosen to be the the control command for the robot.

B. Experiment result

Similar to the simulation, the goal for the robot is to move along a line, $y = 0$, in the positive x-axis direction, while avoiding obstacles and maintaining constant speed which also points along the positive x-axis.

The experiment result is shown in Fig. 12 and Fig. 13, where we can see that the robot can successfully avoid the obstacle and merge back to the original route. Control command is shown in Fig. 14.

The proposed M-analysis can also be applied in this scenario. If we use the original planning problem ((1)∼(3)), the infeasible starting state at each time step will cause the robot to move in a violent manner. However, after introducing slack variables, the planning results in smoother trajectory. From Fig. 15 we can see that after the first several steps, the whole system maintains at least 13-settling. After passing by the obstacle ($k > 80$), M goes to and maintains at 20, which indicates that the robot has settled back to track $y = 0$.

V. CONCLUSION

This paper proposed a new method of analyzing stability properties for non-convex MPC. The proposed method, M-analysis, considered the difference of each action location planned at consecutive time steps. We said the action location k is M-settling if there is smaller and smaller change between consecutive time steps after time step $k - M$ which indicates local conversions of the planned trajectory. Simulation results showed that a robot that implemented the MPC-CFS algorithm was capable of dealing with dynamic environment as long as the changes in the environment were predictable. The stability analysis using M-analysis was performed to show that MPC-CFS had decent stability properties. Finally, experiment on Turtlebot3 showed that the proposed MPC-CFS controller performed well. In the future, we will include the robot dynamics in to the optimization problem formulation and complete the theoretical proof for M-analysis.

REFERENCES

[1] N. Wu and M. Zhou, "Modeling and deadlock control of automated guided vehicle systems," *IEEE/ASME transactions on mechatronics*, vol. 9, no. 1, pp. 50–57, 2004.

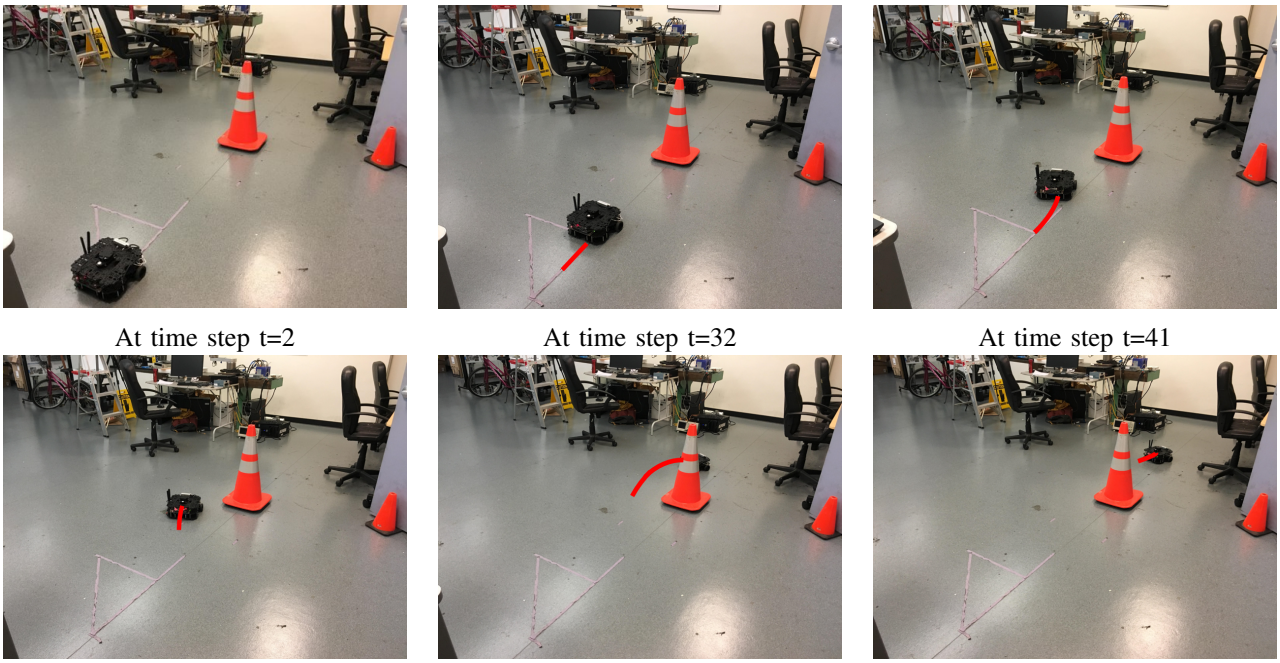


Fig. 12: Experiment result of robot avoiding an obstacle.

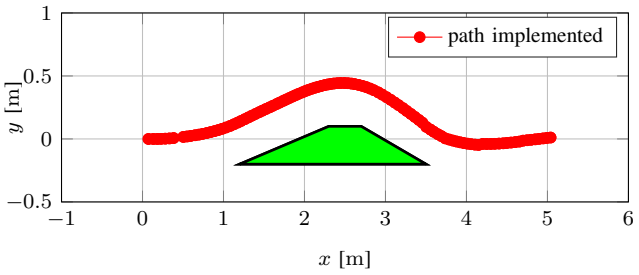


Fig. 13: Implemented trajectory in the experiment.

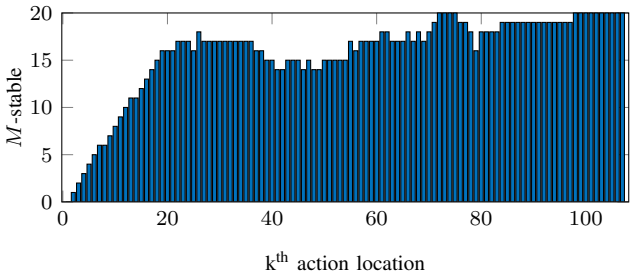


Fig. 15: M-analysis for the experiment.

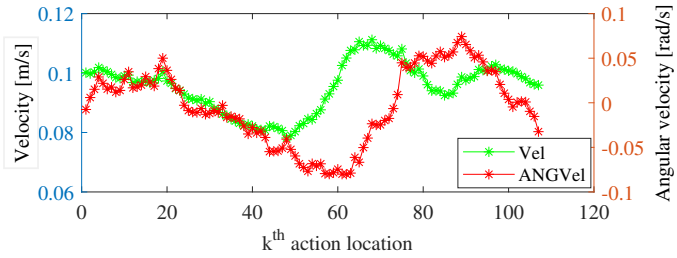


Fig. 14: Control command in the experiment.

[2] J. Wang, J. Steiber, and B. Surampudi, "Autonomous ground vehicle control system for high-speed and safe operation," *International Journal of Vehicle Autonomous Systems*, vol. 7, no. 1-2, pp. 18–35, 2009.

[3] F. Oleari, M. Magnani, D. Ronzoni, and L. Sabattini, "Industrial agvs: Toward a pervasive diffusion in modern factory warehouses," in *Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 233–238.

[4] J. B. Rawlings, "Tutorial: Model predictive control technology," in *American Control Conference, 1999. Proceedings of the 1999*, vol. 1. IEEE, 1999, pp. 662–676.

[5] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[6] D. Limón, T. Alamo, F. Salas, and E. F. Camacho, "On the stability of constrained mpc without terminal constraint," *IEEE transactions on automatic control*, vol. 51, no. 5, pp. 832–836, 2006.

[7] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "Mpc-based approach to active steering for autonomous vehicle systems," *International Journal of Vehicle Autonomous Systems*, vol. 3, no. 2-4, pp. 265–291, 2005.

[8] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[9] T. G. Hovgaard, S. Boyd, L. F. Larsen, and J. B. Jørgensen, "Non-convex model predictive control for commercial refrigeration," *International Journal of Control*, vol. 86, no. 8, pp. 1349–1366, 2013.

[10] J. Chen, C. Liu, and M. Tomizuka, "Foad: Fast optimization-based autonomous driving motion planner," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 4725–4732.

[11] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4906–4913.