Comparison of Stabilizing NMPC Designs for Wheeled Mobile Robots: an Experimental Study

Mohamed W. Mehrez, George K. I. Mann, and Raymond G. Gosine
Intelligent Systems Lab, Faculty of Engineering and Applied Science
Memorial University of Newfoundland
St. John's, NL, Canada
e-mail: {m.mehrez.said, gmann, rgosine}@mun.ca

***Abstract*— In this paper, two stabilizing nonlinear model predictive control (NMPC) designs, namely, final-state equality constraint stabilizing design and final-state inequality constraint stabilizing design have been applied to achieve two wheeled mobile robot's control objectives, i.e. point stabilization and trajectory tracking. In both controllers, final-state constraints are imposed, on the online optimization step, to guarantee the closed loop stability. As shown in the literature, both stabilizing designs were addressed to be computationally intense; thus, their real-time implementation is not tractable. Nonetheless, in this work, a recently developed toolkit implementing fast NMPC routines has been used to apply the two stabilizing designs on a mobile robot research platform after developing a C++ code, coupling the toolkit and the research platform's software. Full scale experiments implementing the two stabilizing designs are conducted and contrasted in terms of performance measures and real-time requirements.**

***Keywords*— mobile robots; nonlinear model predictive control; stability; real-time implementation; coupling code**

I. INTRODUCTION

Wheeled mobile robots (WMRs) have a wide range of applications in industry, discovery, search and rescue, and mapping of unknown environments [1]. The nonholonomic differential drive model (e.g. unicycle) is commonly employed to describe robot's motion kinematics [2]. Point stabilization and trajectory tracking are considered to be the fundamental control problems related to nonholonomic robots [3].

Trajectory tracking control problem intends to control the robot to track a given time-varying trajectory, while the aim of point stabilization or regulation is to drive the robot between two static poses. Because of its relative simplicity, trajectory tracking problem has been extensively studied amongst the robotics society. Numerous tracking techniques are presented in the literature, which include dynamic feedback linearization [4], backstepping [5], and sliding mode [6]. Several studies used a linearized form of the motion equation, around the reference trajectory, to design the control law, e.g. Lyapunov stable time-varying tracking control [7]. According to Brockett's theorem [8], a smooth time-invariant feedback control law doesn't exist for point stabilization. Many studies, solving this problem, are addressed in [9], which include Lyapunov control, piecewise-continuous feedback, nonlinear geometric control, dynamic feedback linearization, and smooth time-varying control. Control designs achieving both control objectives include differential kinematic controller [10], backstepping technique [11], and feedback linearization [4].

Because of its ability to explicitly handle constrained control problems, model predictive control (MPC) has gained more attention in the control society [2]. MPC computes a future control sequence minimizing an objective function, over a predefined prediction horizon, where a set of control actions and system states constraints are satisfied. MPC variants including linear MPC and nonlinear MPC (NMPC) have been used to solve the two discussed control objectives of mobile robots. Studies utilizing linear MPC are presented in [12], and [13]; in these studies, MPC has been used to achieve only the trajectory tracking objective. NMPC, which uses the nonlinear motion model, has been used for tracking problems [15], [16]; regulation problems [9]; and both [2].

Stability of finite horizon MPC cannot be guaranteed trivially [9]; nonetheless, it was proven that it can be assured by using a terminal state equality constraint [17], [18]. Further work showed that the terminal-state equality constraint can be relaxed to a terminal-state inequality constraint, by adding a terminal-state cost [9], [15]. It was claimed that both stabilizing designs are computationally expensive and this limits their real-time applicability, see, e.g. [9], [15], and [17]. Nonetheless, due to the advancements in computing machines and development of efficient numerical algorithms, a considerable number of dynamic optimization and NMPC implementation packages have been developed [19]. A recently developed package (ACADO-Toolkit) [20], which implements fast NMPC routines, has been proven to be open source, user friendly, extensible, self-contained, and computationally versatile, when compared with the previously developed optimization packages [20].

In this work, ACADO-toolkit has been utilized to apply, in real-time, the computationally demanding terminal equality and inequality constraints NMPC designs, in order to achieve the two control objectives of nonholonomic mobile robots. A C++ code, coupling the used toolkit and Pioneer-3AT mobile

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NESRC), the Research and Development Corporation (RDC), C-CORE J.I. Clark Chair, and Memorial University of Newfoundland.

978-1-4799-1740-2/15/$31.00 ©2015 IEEE

robot platform's software, has been developed. Through full-scale laboratory experiments, the performances of the two controllers' designs have been compared and contrasted.

II. PRELIMINARIES

In this section, a brief description of the differential drive robots kinematics is presented together with an explanation of the considered control objectives, i.e. point stabilization and trajectory tracking.

A. Robot Kinematics

Under the nonholonomic constraint of rolling without slipping, a differential drive robot kinematics is given by [15]:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u} \tag{1}$$

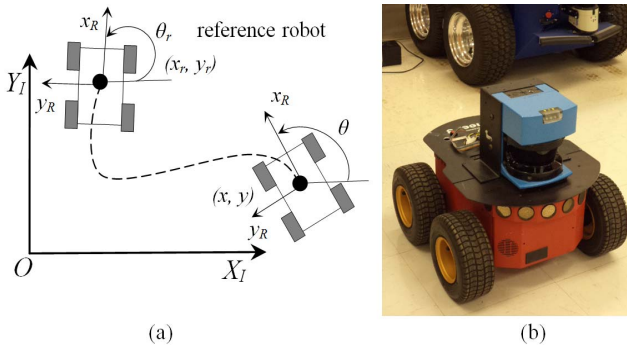
the state and control signal vectors are denoted as $\mathbf{x} = (x, y, \theta)^T$ and $\mathbf{u} = (v, \omega)^T$, respectively. As shown in Fig. 1(a), $(x, y)^T$ represents the Cartesian position of the robot while θ represents its orientation. v and ω are the linear and the angular speeds of the robot, respectively. Despite the simplicity of model (1), it is sufficient to describe the nonholonomic constraints of the differential drive robots; furthermore, as outlined in [21], system (1) has a driftless form with a globally satisfied accessibility rank condition, and hence is controllable.

B. Point Stabilization and Trajectory Tracking

In order to illustrate the two control problems of the mobile robot, a reference robot, shown in Fig. 1(a), is defined with a reference state vector $\mathbf{x}_r = (x_r, y_r, \theta_r)^T$ and a reference control vector $\mathbf{u}_r = (v_r, \omega_r)^T$, and subject to the same constraint as system (1). Thus, its kinematics model can be written as:

$$\dot{\mathbf{x}}_r = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} v_r\cos\theta_r \\ v_r\sin\theta_r \\ \omega_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_r \tag{2}$$

The point stabilization problem is the case when the vector $\mathbf{x}_r$ is constant, and the control vector $\mathbf{u}_r = [0, 0]^T$. On the other hand, for the trajectory tracking problem, vectors $\mathbf{x}_r$ and $\mathbf{u}_r$ have time varying values depending on the chosen reference trajectory. In both cases, the control objective is to control system (1) to track the reference model (2); thus, an error state vector $\mathbf{x}_e$ can be defined as:



(a) (b)

Fig. 1. (a) Differential drive robot kinematics. (b) Pioneer 3-At

$$\mathbf{x}_e = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \tag{3}$$

Then, the error dynamics is obtained by differentiating (3):

$$\begin{aligned} \dot{x}_e &= \omega y_e - v + v_r\cos\theta_e \\ \dot{y}_e &= -\omega x_e + v_r\sin\theta_e \\ \dot{\theta}_e &= \omega_r - \omega \end{aligned} \tag{4}$$

When linearized, the error model (4) becomes:

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix} \mathbf{x}_e + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_e \tag{5}$$

where $\mathbf{u}_e = (u_1, u_2)^T = (-v + v_r\cos\theta_e, \omega_r - \omega)^T$.

The controllability of system (5) is lost when the reference linear velocity and angular velocity values approach the origin; thus, a linear controller cannot be used for point stabilization.

III. STABILIZING MODEL PREDICTIVE CONTROL DESIGNS

In this section, a brief description of the two stabilizing NMPC designs is presented. The general form of the nonlinear control system such as (1) can be expressed as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{6}$$

where $\mathbf{x}(t) \in R^n$ and $\mathbf{u}(t) \in R^m$ are the n dimensional state and m dimensional control vectors, respectively. The control task is to compute an admissible control input $\mathbf{u}(t)$ to drive the system (6) to move toward the equilibrium point defined by ($\mathbf{x}_e(t) = 0$ and $\mathbf{u}_e(t) = 0$). The objective of the controller now is to minimize a cost function given by [15]:

$$J(t, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) = \int_t^{t+T} l(\tau, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) d\tau \tag{7}$$

where $l(\tau, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) = \mathbf{x}_e(\tau)^T Q \mathbf{x}_e(\tau) + \mathbf{u}_e(\tau)^T R \mathbf{u}_e(\tau)$ is the running cost, Q and R are positive definite symmetric weight matrices, and T is the prediction horizon length. As shown in [17] and [18], the stability of the predictive controller can be guaranteed by imposing terminal state equality constraint. In this case, at time t, the online-open-loop optimization problem of the NMPC controller can be formulated as the following:

$$\min_{\mathbf{u}} J(t, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) \tag{8}$$

$$\begin{aligned} \text{subject to}: \ & \dot{\mathbf{x}}(\tau) = \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau)) \\ & \mathbf{u}(\tau) \in U, (\tau \in [t, t+T]) \\ & \mathbf{x}_e(t+T) = 0 \end{aligned} \tag{9}$$

where $U \in R^m$ is a compact set specifying the control input saturation limits, and $\mathbf{x}_e(t+T) = 0$ defines the terminal state equality constraint. As outlined in [18], the controller stability can be proven if the following two assumptions are satisfied.

- The state vector $\mathbf{x}_r \in R^n$ is an equilibrium point for an admissible control value $\mathbf{u}_r \in R^m$; this means that there exists a control value $\mathbf{u}_r \in R^m$ such that $\mathbf{f}(\mathbf{x}_r, \mathbf{u}_r) = \mathbf{0}$.
- The running cost function $l : R^n \times R^m \to R_0^+$ satisfies $l(\mathbf{x}_r, \mathbf{u}_r) = 0$.

By observing system (1) and the running cost $l(\cdot)$, the above assumptions are satisfied. The complete stability proof is presented in [17] and [18].

Authors of [15]proposed another stabilizing NMPC design, with regulation and tracking capabilities, by relaxing final-state equality constraint to final inequality constraint, and adding a Lyapunov function to the cost function as a terminal-state penalty as follows:

$$J(t, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) = \mathbf{g}(\mathbf{x}_e(t+T)) + \int_t^{t+T} l(\tau, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) d\tau \quad (10)$$

where $\mathbf{g}(\mathbf{x}_e(t+T))$ is the terminal state penalty and $l(\cdot)$ is as defined for (7). $\mathbf{g}(\cdot)$ is assumed to be a continuous, differentiable function, and satisfying $\mathbf{g}(\mathbf{0}) = 0$, and $\mathbf{g}(\mathbf{x}_e(t)) > 0$ for all $\mathbf{x}_e(t) \neq 0$. A stability condition is then found and used to find the terminal-state region and the corresponding terminal controller; the terminal-state region replaces the final equality constraints defined in (9), while the terminal state controller is never applied to control the robot. As shown in [15], system (6) is asymptotically stable if a terminal-state controller $\mathbf{u}^L = \left(u_1^L, u_2^L\right)^T$ exists such that the following condition is satisfied:

$$\dot{\mathbf{g}}(\mathbf{x}_e(t)) + l(\tau, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) \leq 0 \quad (11)$$

In order to find the terminal-state region, a Lyapunov function for the terminal-state penalty is defined as follows:

$$\mathbf{g}(\mathbf{x}_e(t+T)) = (1/2)\mathbf{x}_e(t+T)^T \mathbf{x}_e(t+T) \quad (12)$$

By defining the positive definite weight matrices of $l(\cdot)$ as:

$$Q = \mathrm{diag}(q_{11}, q_{22}, q_{33}), \; R = \mathrm{diag}(r_{11}, r_{22}) \quad (13)$$

the terminal state region, specified by the stability condition (11), is obtained as the following:

$$\begin{aligned} &\dot{\mathbf{g}}(\mathbf{x}_e(t)) + l(\tau, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) \\ &= x_{eT}\dot{x}_{eT} + y_{eT}\dot{y}_{eT} + \theta_{eT}\dot{\theta}_{eT} + l(\tau, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) \\ &= x_{eT}u_1^L + y_{eT}v_r \sin\theta_{eT} + \theta_{eT}u_2^L + l(\tau, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) \\ &= x_{eT}u_1^L + y_{eT}v_r \sin\theta_{eT} + \theta_{eT}u_2^L \\ &\quad + q_{11}x_{eT}^2 + q_{22}y_{eT}^2 + q_{33}\theta_{eT}^2 + r_{11}\left(u_1^L\right)^2 + r_{22}\left(u_2^L\right) \end{aligned} \quad (14)$$

the subscript T denotes the terminal state; the terminal state $\mathbf{u}^L$ is now selected to be:

$$u_1^L = -\alpha x_{eT}, \text{ and } u_1^L = -\beta\theta_{eT} \quad (15)$$

with tuning parameters $\alpha \geq 0$, and $\beta \geq 0$; now, the stability condition (11) becomes:

$$\begin{aligned} &\dot{\mathbf{g}}(\mathbf{x}_e(t)) + l(\tau, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) \\ &= -(\alpha - q_{11} - r_{11}\alpha^2)x_{eT}^2 - (\beta - q_{33} - r_{22}\beta^2)\theta_{eT}^2 \\ &\quad + y_{eT}v_r \sin\theta_{eT} + q_{22}y_{eT}^2 \end{aligned} \quad (16)$$

To have a negative derivative of (16) and thus satisfying (11) for stability, the following inequalities are imposed on the weighting parameters (13):

$$\begin{aligned} \alpha - q_{11} - r_{11}\alpha^2 &\geq q_{22} \\ \beta - q_{33} - r_{22}\beta^2 &\geq 0 \end{aligned} \quad (17)$$

in addition, the terminal state region is specified by:

$$\begin{aligned} |x_{eT}| &\geq |y_{eT}| \\ y_{eT}\theta_{eT} &< 0 \end{aligned} \quad (18)$$

Readers can refer to [15] for the complete stability proof.

IV. Implementation

In the work presented here, a software environment for automatic control and dynamic optimization (ACADO-Toolkit) has been utilized to achieve the control tasks. It is an open-source software package written in C++ that is completely self-contained; thus, coupling this package with external packages is optional [20]. In order to be able to satisfy the necessary stability constraints presented in the previous section, it is essential to conservatively maintain a number of prediction horizon steps required for constraints satisfaction; thus, rendering the online optimization problem feasible. Maintaining a reasonably short controller's update rate is also necessary to achieve a good controller performance. As will be presented in the next section, the used toolkit allows for these considerations while real-time requirements are satisfied. A C++ code which couples the used package, and a research platform's input files, has been developed; the developed code instantiates an NMPC execution routine every controller's update step.

Pioneer 3-AT, a research platform mobile robot shown in Fig. 1 (b), is used in the conducted experiments. The robot is a 4-wheel skid steering mobile robot with an embedded computer, Ethernet-based communications, laser scanner, and other autonomous functions. For localization purpose, the robot uses the high accuracy 100 tick encoders with inertial correction for dead reckoning to compensate for skid steering. The computation of the control command vector $\mathbf{u}$ is performed on a 2.5 GHz quad-core machine with a Unix-based operating system and a memory of 6 GB. The calculated control signal is then transmitted over a wireless-based communication connection to a server running on the robot's onboard computer, whereas the robot state vector $\mathbf{x}(t)$ is fedback to the controller through the same connection.

Two sets of experiments are implemented using the two stabilizing designs. In the first set, forward and parallel parking point stabilizations have been conducted. The robot starts from the initial pose $\mathbf{x}_0 = [0,0,0]^T$ (m, m, rad). Under forward parking, the robot is commanded to stabilize at the pose $\mathbf{x}_r = [2,2,\theta_r]^T$, where $\theta_r \in [0, \pi/2, \pi, 3\pi/2]$(rad). Under parallel parking, the robot is commanded to stabilize at the pose $\mathbf{x}_r =$

$[0,2,\theta_r]^T$, where θ_r is as defined before. In point stabilization, the controller update time step is chosen as 0.3 seconds with number of prediction steps N = 30, leading to a prediction horizon time T = 9 seconds. For the two controllers involved, the weights defined in (13) and the parameters defined in (17) are chosen as: $q_{11} = q_{22} = 1$, $q_{33} = 0.2$, $r_{11} = 0.125$, $r_{22} = 0.8$, α = 4, and $\beta = 1$.

In the second set, trajectory tracking task is conducted, where two reference trajectories, i.e. circular-shape (19), and eight-shape (20) trajectories, are considered. Similar to the point stabilization case, the robot starts from the initial pose $\mathbf{x}_0 = [0,0,0]^T$ (m, m, rad). Under tracking, the controller update time step is chosen to be 0.2 seconds with number of prediction steps N = 30. The weights defined in (13) and parameters defined in (17) are chosen as: $q_{11} = q_{22} = 0.5$, $q_{33} = 0.01$, $r_{11} = 0.25$, $r_{22} = 0.125$, $\alpha = 2$, and $\beta = 8$.

$$\begin{bmatrix} x_r(t) & y_r(t) \end{bmatrix}^T = \begin{bmatrix} 0.3+2\sin(0.25t) & -2.3+2\cos(0.25t) \end{bmatrix}^T \quad (19)$$

$$\begin{bmatrix} x_r(t) & y_r(t) \end{bmatrix}^T = \begin{bmatrix} 0.3+1.5\sin(0.3t) & -2.3+2.5\cos(0.15t) \end{bmatrix}^T \quad (20)$$

In the two sets of experiments, to achieve an accurate localization of the robot and to satisfy its actuators saturation bounds, the controller saturation limits are set as follows:

$$\begin{bmatrix} -0.25 & -0.7 \end{bmatrix}^T \le \begin{bmatrix} v & \omega \end{bmatrix}^T \le \begin{bmatrix} 0.6 & 0.7 \end{bmatrix}^T \quad (21)$$

Indeed, the trajectories (19) and (20) are chosen such that the resulting reference velocities are not violating the bounds (21).

V. Experimental Results

Point stabilization results are presented first. Fig. 2, (best viewed in colors) illustrates the executed trajectories of the all cases considered. As can be seen in the subplots of Fig. 2, the two controllers' designs can stabilize the robot to the desired pose in all cases; the robot exhibits similar trajectories in each stabilization case, under the two controllers, except for the parallel parking case when θ_r = 0, see subplots (a) and (b).

Table I summarizes the performance parameters **P** of the point stabilization experiments in terms of the error pose vector $\mathbf{x}_e=[x_e$ (mm),y_e (mm),θ_e (rad)], the average computational cost per time step (C) (milliseconds), and the integral of norm squared actual control inputs $\gamma = \left(\sum_1^k \|\mathbf{u}\|^2 \delta_T \right)$, where k is the number of the controller update steps and δ_T is the used time step (seconds). γ is used as a metric to evaluate the control energy [2]. Fig. 3, (best viewed in colors) shows the actual linear and angular control actions of the robot. As can be seen from Table I, the performance of the two controllers is satisfactory in terms of state vector error $\mathbf{x}_e$ (3). Nonetheless, in the most of the cases, the performance of the final equality constraint controller is relatively better than the second controller. However, as can be noticed from Fig. 3, the robot settles faster to the reference speeds (v=0, ω=0), under the final inequality constraint controller. In fact, as can be seen in Fig.3, under the final equality constraint controller, the robot speeds keep oscillating around the reference values until the end of experiment's running time, i.e. (30 seconds). The final equality constraint imposed on the optimization problem (8) is the reason behind this oscillatory behavior of the robot speeds, which can be resolved by modifying the weight parameters (13) and/or improve model (1) to account for the robot accelerations. The computational costs under both controllers are relatively similar and meet the real-time requirements. While the robot speeds under the two controllers are satisfying the limits (21), final inequality constraint controller showed more conservative energy expenditure in most of the cases. It has to be mentioned here that, when the desired robot position is chosen further than (x_r=2, y_r=2), e.g. (x_r=3, y_r=3), the number of prediction horizon steps N should be largely increased in the case of the final equality constraints controller than the final inequality constraints controller. As per the previous analysis, the overall performance of the final inequality constraint controller has been seen to be more satisfactory and practical in the point stabilization case for the given parameters (13).

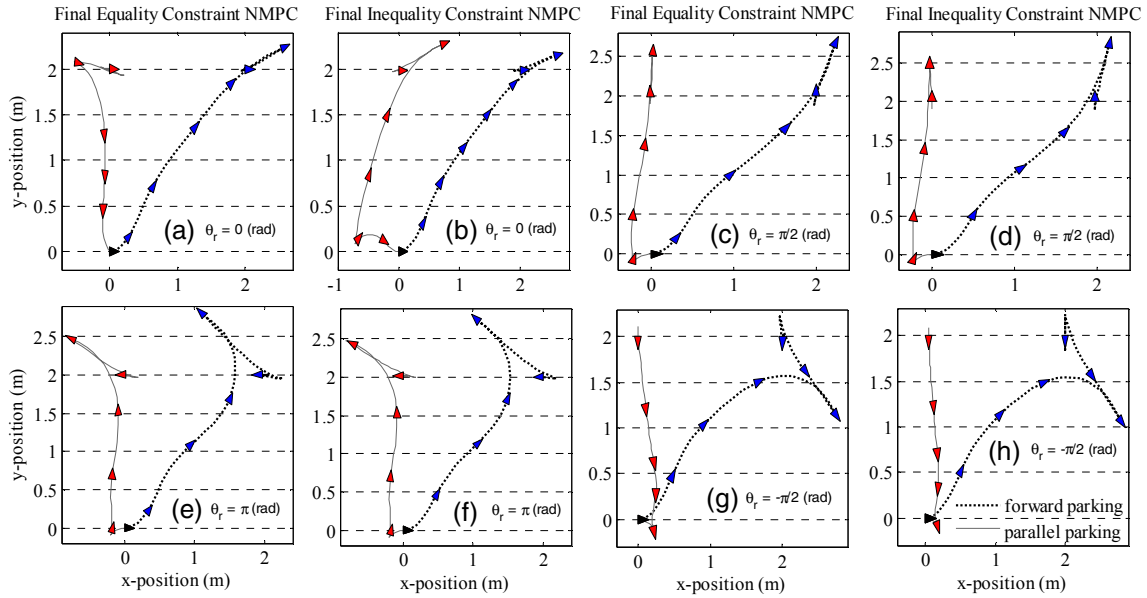


Fig. 2. Point-stabilization closed-loop trajectories under the two controllers: robot initial pose (black-color filled triangle); robot performing forward parking (blue-color filled triangle); and robot performing parallel parking (red-color filled triangle).

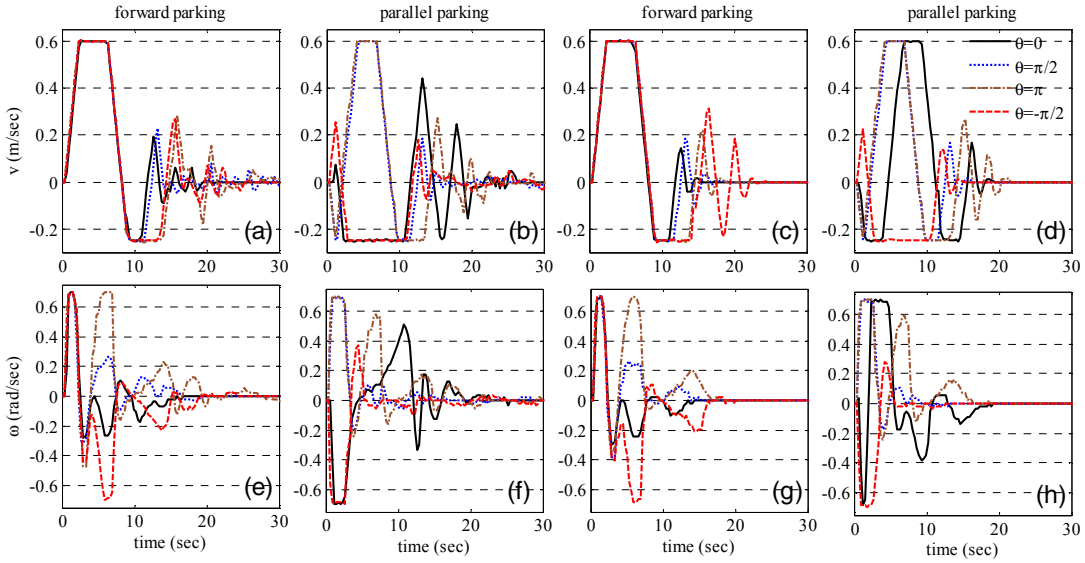


Fig. 3. Point-stabilization control signals. Final equality constraint NMPC: subplots (a, b, e, and f). Final inequality constraint NMPC : subplots (c, d, g, and h).

Trajectory tracking results are presented in the following: Fig. 4, and Fig. 5, (best viewed in colors) illustrate the two controllers' performances. The starting points of trajectories (19) and (20) have been chosen such that an initial error is imposed on the tracking problem. Fig. 4, shows the evolution of the error state vector $\mathbf{x}_e$ over the experiments running time for all cases. For the circular reference (19), the experiments time is set to be (25 seconds), and to (45 seconds) for the eight-shape reference (20); as can be seen the error state vector $\mathbf{x}_e$ converges asymptotically to the origin, under the two controllers, with a settling time of (10 seconds). Fig. 5, shows the actual trajectories executed by the robot, the linear and angular velocities of the robot, and their references; as can be noticed from the subplots of Fig. 5, the two controllers performed adequately well. However, the performance of the final equality constraint is relatively better than the final inequality constraint controller. For trajectory (19), the final equality constraint controller achieved a steady state positional error of (± 2.5 cm), and orientation error of (± 0.02 rad); for the same trajectory, under the final inequality constraint controller, the positional error is (± 3 cm), and the orientation error is (± 0.03 rad). For trajectory (20), the final equality constraint controller achieved a steady state positional error of (± 3.5 cm), and orientation error of (± 0.075 rad); for the same trajectory, under the final inequality constraint controller, the positional error is (± 5 cm), and the orientation error is (± 0.085 rad). In all cases, the average computational time per controller's time step was observed to be within (47 milliseconds). Because the eight-shape trajectory has sharp changes in its reference speeds, errors in the eight-shape trajectory tracking are relatively large when compared to circular-shape tracking. As shown in Fig. 5, in both tracking cases, the control signals did not go beyond their bounds (21). However, a larger overshoot in the angular speed has been observed under the final equality constraint NMPC for the circular trajectory tracking.

TABLE I. POINT STABILIZATION PERFORMANCE PARAMETERS

P	**Forward Parking**							
	Final equality constraint NMPC				***Final inequality constraint NMPC***			
	θ_r				θ_r			
	0	***π/2***	***π***	***- π/2***	***0***	***π/2***	***π***	***- π/2***
x_e	0	-1	36	0	1	17	-2	1
y_e	1	-3	-2	0	13	0	-1	-4
θ_e	0	0	-0.02	0	-0.02	0.02	0.02	-0.02
C	51	41	44	43	41	44	45	45
γ	5.5	5.9	8.1	7.4	5.25	5.67	7	7.7
P	**Parallel Parking**							
	Final equality constraint NMPC				***Final inequality constraint NMPC***			
	θ_r				θ_r			
	0	***π/2***	***π***	***- π/2***	***0***	***π/2***	***π***	***- π/2***
x_e	0	1	-16	0	1	-4	-1	-46
y_e	0	19	0	-3	17	0	-15	-1
θ_e	0	-0.04	0.02	0.02	-0.02	0.02	0.02	-0.02
C	47	44	45	45	41	42	44	44
γ	6.4	5.5	7.1	4.7	6.9	5.3	7	4

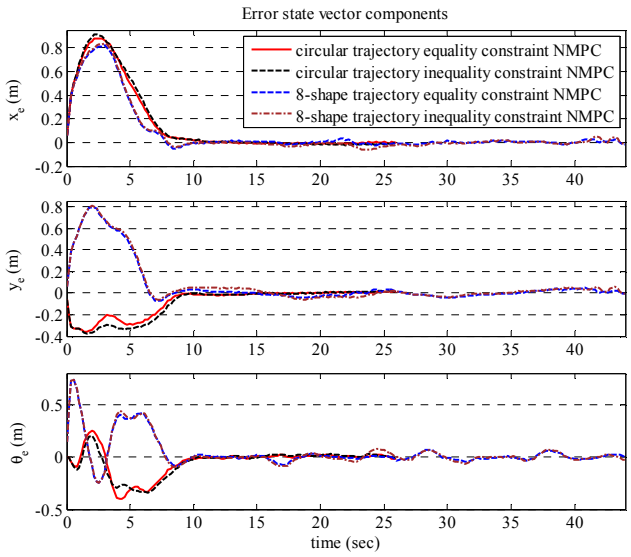


Fig. 4. Trajectory tracking error state vector components

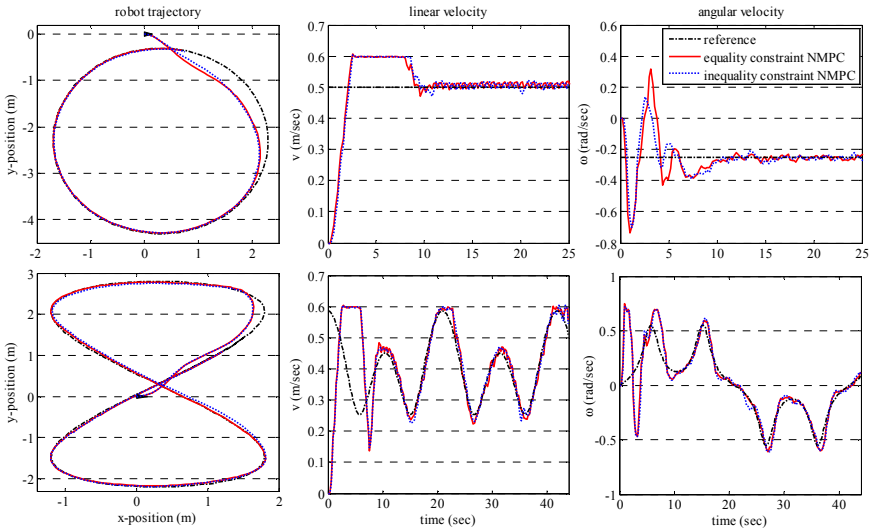


Fig. 5. Trajectory tracking results

VI. Conclusion

The main contribution of this work is the real-time implementation and comparison of two stabilizing NMPC schemes, which have been remarked in the literature to be computationally intense. The use of a toolkit implementing fast NMPC routines rendered this computational problem tractable. In an experimental context, the two controllers' designs have been applied to achieve two common control objectives for mobile robots, i.e. point stabilization and trajectory tracking. In order to perform the experiments, a C++ code, which couples the used package and Pioneer 3-AT's software, has been developed. The performances of the two controllers have been then compared and contrasted in terms of performance measures, and real-time requirements.

References

[1] H. Chen, M.-M. Ma, H. Wang, Z.-Y. Liu, and Z. xing Cai, "Moving horizon H_∞ tracking control of wheeled mobile robots with actuator saturation," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 2, pp. 449-457, 2009.

[2] F. Xie and R. Fierro, "First-state contractive model predictive control of nonholonomic mobile robots," in *Proceedings of the American Control Conference*, pp. 3494-3499, 2008.

[3] G. Klanar and I. . krjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robotics and Autonomous Systems*, vol. 55, no. 6, pp. 460-469, 2007.

[4] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: Design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, 2002.

[5] U. Kumar and N. Sukavanam, 'Backstepping based trajectory tracking control of a four wheeled mobile robot," *International Journal of Advanced Robotic Systems*, vol. 5, no. 4, pp. 403 - 410, 2008.

[6] D. Chwa, "Sliding-mode tracking control of nonholonomic wheeled mobile robots in polar coordinates," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 4, pp. 637-644, 2004.

[7] C. Samson, "Time-varying feedback stabilization of car-like wheeled mobile robots," *The International Journal of Robotics Research*, vol. 12, no. 1, pp. 55-64, 1993.

[8] R. W. Brockett, "Asymptotic stability and feedback stabilization," *Differential Geometric Control Theory*, pp. 181-191, Birkhauser, 1983.

[9] D. Gu and H. Hu, "A stabilizing receding horizon regulator for nonholonomic mobile robots," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 1022-1028, 2005.

[10] W. E. Dixon, D. M. Dawson, F. Zhang, and E. Zergeroglu, "Global exponential tracking control of a mobile robot system via a PE condition," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 30, no. 1, pp. 129–142, Feb. 2000.

[11] T. C. Lee, K. T. Sun, C. H. Lee, and C. C. Teng, "Tracking control of unicycle-modeled mobile robots using s saturation feedback controller," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 2, pp. 305–318, Mar. 2001.

[12] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. Tsengz, "A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems," in *Proceedings of the 46th IEEE Conference on Decision and Control*, pp. 2980-2985, 2007.

[13] G. Klanar and I. . krjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robotics and Autonomous Systems*, vol. 55, no. 6, pp. 460-469, 2007.

[14] J. Backman, T. Oksanen, and A. Visala, "Navigation system for agricultural machines: Nonlinear model predictive path tracking," *Computers and Electronics in Agriculture*, vol. 82, no. 0, pp. 32 - 43, 2012.

[15] D. Gu and H. Hu, "Receding horizon tracking control of wheeled mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 4, pp. 743-749, 2006.

[16] H. Lim, Y. Kang, C. Kim, J. Kim, and B.-J. You, "Nonlinear model predictive controller design with obstacle avoidance for a mobile robot," in *Proceedings of the IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications*, pp. 494-499, 2008.

[17] J. B. Rawlings and K. R. Muske, "Stability of constrained receding horizon control," *IEEE Transactions on Automatic Control*, vol. 38, no. 10, pp. 1512–1516, Oct. 1993.

[18] L. Grüne, and J. Pannek, "*Nonlinear Model Predictive Control: Theory and Algorithms*", 1st ed, in *Communications and Control Engineering*, Springer-Verlag, 2011.

[19] Leineweber DB, Bauer I, Bock HG, and Schloder JP, "An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: theoretical aspects," *Computers and Chemical Engineering*,vol. 27, pp.157–166, 2003.

[20] B. Houska, H. Ferreau, and M. Diehl, "ACADO toolkit - An open source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298-312, 2011.

[21] A. M. Bloch, *Nonholonomic Mechanics and Control*. New York, NY: Springer, 2003.