Alternative non-linear predictive control under constraints applied to a two-wheeled nonholonomic mobile robot

Gaspar Fontineli Dantas Júnior[*1], João Ricardo Tavares Gadelha[*2], Carlor Eduardo Trabuco Dórea[*3]
Departamento de Engenharia da Computação e Automação
Universidade Federal do Rio Grande do Norte
Natal, Brazil
gsatnad@yahoo.com[*1], joaoricardotg@gmail.com[*2], cetdorea@dca.ufrn.br[*3]

Abstract— This paper presents a novel algorithm for Model Predictive Control (MPC) of a nonlinear class that can be applied to systems subject to linear state and control constraints. A piecewise linear representation of the nonlinear system is obtained, by fixing the sequence of future inputs. A quadratic optimization problem is then repeatedly solved until convergence is reached. In this case, it is proved that no prediction error occurs and, consequently, the state and control constraints are satisfied by the original nonlinear system. The proposed technique is applied to a two-wheeled nonholonomic mobile robot, known to be difficult to control with linear techniques.

Keywords— Model Predictive Control; Nonlinear Model Predictive Control, Control Under Constraints, Nonlinear Systems, Nonholonomic Two Wheeled Robot.

I. INTRODUCTION

Model predictive control (MPC) has a long history in the field of control engineering. It is one of the few areas that has received on-going interest from researchers in both the industrial and academic communities. Four major aspects of model predictive control make the design methodology attractive to both practitioners and academics. The first aspect is the design formulation, which uses a completely multivariable system framework where the performance parameters of the multivariable control system are related to the engineering aspects of the system; hence, they can be understood and 'tuned' by engineers. The second aspect is the ability of the method to handle both 'soft' and hard constraints in a multivariable control framework. This is particularly attractive to industry where tight profit margins and limits on the process operation are inevitably present. The third aspect is the ability to perform on-line process optimization. The fourth aspect is the simplicity of the design framework in handling all these complex issues. [1].

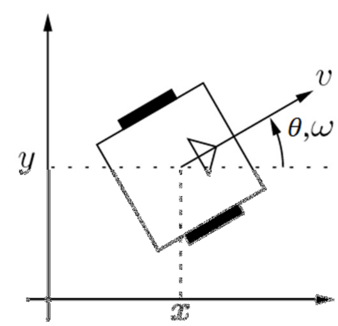
A wheeled mobile robot (WMR) is defined as a wheeled vehicle that can move autonomously without assistance from external human operator. The WMR is equipped with a set of motorized actuators and, sometimes, with an array of sensors, which help it to carry out useful work [2].

Wheeled mobile robots (WMRs) are increasingly present in industrial and service robotics, particularly when flexible motion capabilities are required on reasonably smooth grounds and surfaces [3]. The problem of autonomous control of WMRs has attracted the interest of researchers in view of its theoretical challenges.

In the absence of workspace obstacles, the basic motion tasks assigned to a WMR may be reduced to following a given trajectory or reaching a given destination. From a control viewpoint, the second problem is easier than the first.

A schematic diagram of the robot is presented on figure 1. The configuration is represented by its position on the Cartesian space (x and y, that is the position of the robot-body center with relation to a referential frame fixed on the workspace), and by its orientation θ (angle between the robot orientation vector and the reference axis X, fixed on the workspace).

Figure 1: Schematic diagram of a two-wheeled nonholonomic robot.



In this paper is proposed to make a two-wheeled differentially driven nonholonomic mobile robot reach a given destination. A piecewise linear representation of the nonlinear system is obtained, by initially fixing the sequence of future inputs. A quadratic optimization problem is then repeatedly solved until convergence is reached so that, the first element of the optimized input vector can be used in the system. In this case, it is proved that no prediction error occurs and, consequently, the state and control constraints are satisfied by the original nonlinear system.

II. CONTROL STRATEGY

A. Model Predictive Control applied to a Linear System

In the case where the system is in linear format, we have the following representation in discrete time:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ z(k) = Cx(k) \end{cases} \quad (1)$$

Where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the input vector, $z \in \mathbb{R}^p$ is the output vector and $k \in \mathbb{N}$ is the sampling instant.

To find the prediction equation in the absence of disturbances and noise when the measurement of the full state vector is available [5,8], we have:

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \vdots \\ \hat{x}(k+H_u|k) \\ \hat{x}(k+H_u+1|k) \\ \vdots \\ \hat{x}(k+H_p|k) \end{bmatrix} = \begin{bmatrix} A \\ \vdots \\ A^{H_u} \\ A^{H_u+1} \\ \vdots \\ A^{H_p} \end{bmatrix} x(k) + \begin{bmatrix} B \\ \vdots \\ \sum_{i=0}^{H_u-1} A^i B \\ \sum_{i=0}^{H_u} A^i B \\ \vdots \\ \sum_{i=0}^{H_p-1} A^i B \end{bmatrix} u(k-1)$$

$$+ \begin{bmatrix} B & \cdots & 0 \\ AB+B & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_u-1} A^i B & \cdots & B \\ \sum_{i=0}^{H_u} A^i B & \cdots & AB+B \\ \vdots & & \vdots \\ \sum_{i=0}^{H_p-1} A^i B & \vdots \;\cdots & \sum_{i=0}^{H_p-H_u} A^i B \end{bmatrix} \begin{bmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{bmatrix} \quad (2)$$

Where H_p is the prediction horizon, H_u is the control horizon, $\hat{x}(k)$ is the estimated state vector and $\Delta\hat{u}$ is the estimated difference of the control variable.

Thus, the prediction of the system output ($Z(k)$) is given in matrix form by:

$$Z(k) = \Psi x(k) + \Upsilon u(k-1) + \Theta \Delta U(k) \quad (3)$$

Where:

$$\Psi = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & C & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & C \end{bmatrix} \begin{bmatrix} A \\ \vdots \\ A^{H_u} \\ \vdots \\ A^{H_p} \end{bmatrix} \quad (4)$$

$$\Theta = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & C & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & C \end{bmatrix} \begin{bmatrix} B & \cdots & 0 \\ AB + B & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_p} BA^i & \cdots & \sum_{i=0}^{H_p-H_u} BA^i \end{bmatrix} \quad (5)$$

$$\Upsilon = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & C & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & C \end{bmatrix} \begin{bmatrix} B \\ B + BA \\ \vdots \\ \sum_{i=0}^{Hu-1} BA^i \\ \vdots \\ \sum_{i=0}^{Hp-1} BA^i \end{bmatrix} \quad (6)$$

Where the tracking error is given by:

$$\varepsilon(k) = \tau(k) - \Psi x(k) - \Upsilon u(k-1) \quad (7)$$

In the event of constraints in $\Delta U(k)$, $U(k)$, $Z(k)$ and $X(k)$, respectively, they are treated as follows:

$$E \begin{bmatrix} \Delta U(k) \\ 1 \end{bmatrix} \leq 0 \quad (8)$$

$$F \begin{bmatrix} U(k) \\ 1 \end{bmatrix} \leq 0 \quad (9)$$

$$G \begin{bmatrix} Z(k) \\ 1 \end{bmatrix} \leq 0 \quad (10)$$

$$H \begin{bmatrix} X(k) \\ 1 \end{bmatrix} \leq 0 \quad (11)$$

Suppose:

$F = [F_1, F_2, \ldots, F_{H_u}, f]$, where f is the last column of F

$G = [\Gamma, g]$, g is the last column of G

$H = [\mathrm{P}, p]$, p is the last column of P

$W = [1, w]$, w is the last column of W

These inequations can be rewritten in dependency with ΔU, according to equation (3) in the following matrix form:

$$\begin{bmatrix} F \\ \Gamma\Theta \\ W \\ P\gamma \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} -F_1 u(k+1) - f \\ -\Gamma[\Psi x(k) + \Upsilon u(k-1)] - g \\ w \\ -P(\alpha x(k) + \beta u(k-1)) - p \end{bmatrix} \quad (12)$$

The predictive control algorithms utilize cost functions in order to penalize deviations of the output from the reference. Based on this error, one can obtain:

$$V(k) = \sum_{i=H_w}^{H_p} \| \hat{z}(k+i|k) - r(k+i|k) \|_{Q(i)}^2 + \sum_{i=0}^{H_u-1} \| \Delta\hat{u}(k+i|k) \|_{R(i)}^2 \quad (13)$$

Where $V(k)$ is the performance index (cost function) of finite horizon to be optimized, $H_w > 1$, $\hat{z}(k|k)$ is the predicted controlled output, $r(k|k)$ is the reference trajectory, Q is the weighting matrix of the tracking error, R is the weighting matrix of the control effort (Q and R are positive definite matrices).

Equation (13) can be written in the following matrix form:

$$V(k) = ||Z(k) - \tau(k)||_Q^2 + ||\Delta U(k)||_R^2 \quad (14)$$

Where:

$$Z(k) = \begin{bmatrix} \hat{z}(k+H_w|k) \\ \vdots \\ \hat{z}(k+H_p|k) \end{bmatrix} \quad (15)$$

$$\tau(k) = \begin{bmatrix} \hat{r}(k+H_w|k) \\ \vdots \\ \hat{r}(k+H_p|k) \end{bmatrix} \quad (16)$$

$$\Delta U(k) = \begin{bmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{bmatrix} \quad (17)$$

Subject to the weighting matrices given by:

$$Q = \begin{bmatrix} Q(H_w) & 0 & \dots & 0 \\ 0 & Q(H_w+1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q(H_w) \end{bmatrix} \quad (18)$$

$$R = \begin{bmatrix} R(0) & 0 & \dots & 0 \\ 0 & R(1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R(H_u-1) \end{bmatrix} \quad (19)$$

It is possible to write the cost function as follows:

$$V(k) = Constant - \Delta U(k)^T G + \Delta U(k)^T H \Delta U(k) \quad (20)$$

Where:

$$G = 2\Theta^T Q E(k) \quad (21)$$

$$H = \Theta^T Q \Theta + R \quad (22)$$

E(k) is the tracking error vector.

To find the optimal $\Delta U(k)$, it is needed to find the gradient of $V(k)$ and set it to zero. So the optimal set of future input moves can be found by:

$$\Delta U(k) = \frac{1}{2} H^{-1} G \quad (23)$$

Since the matrix H is usually ill-conditioned, the best way to find the optimal ΔU, respecting the constraints described in the inequation (12), is using Quadratic Programming.

B. Alternative Nonlinear Model Predictive Control

For the application of the proposed control method, the nonlinear system represented in discrete time must be written in the following form [6,7]:

$$\begin{cases} x(k+1) = f\big(x(k),u(k)\big)x(k) + g\big(x(k),u(k)\big)u(k) \\ z(k) = Cx(k) \end{cases} \quad (24)$$

Where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the input vector, $z \in \mathbb{R}^p$ is the output vector and $k \in \mathbb{N}$ is the sampling instant.

Considering the prediction equation of state i steps ahead, one has:

$$\hat{x}(k+1+i) = f\big(\hat{x}(k+i),\hat{u}(k+i)\big)\hat{x}(k+i) + g\big(\hat{x}(k+i),\hat{u}(k+i)\big)\hat{u}(k+i) \quad (25)$$

It is possible to notice that if $\hat{x}(k+i)$ and $\hat{u}(k+i)$ were fixed over the horizon $i = 0,1,\dots H_p - 1$, it would be possible to find the equation of state in a piecewise linear system described by:

$$\hat{x}(k+1+i) = A_i(\hat{x},\hat{u})\hat{x}(k+i) + B_i(\hat{x},\hat{u})\hat{u}(k+i) \quad (26)$$

Where:

$$A(i)(\hat{x},\hat{u}) = f\big(\hat{x}(k+i),\hat{u}(k+i)\big)$$

$$B(i)(\hat{x},\hat{u}) = g\big(\hat{x}(k+i),\hat{u}(k+i)\big) \quad (27)$$

However, $\hat{x}(k+i)$ and $\hat{u}(k+i)$ are not known over the horizon since $\hat{x}(k+i)$ depends on the future inputs $\hat{u}(k+i)$.

Nevertheless, it is possible to describe an algorithm that converges to the correct predicted values, as follows:

Differently from the linear case, where the matrices A, B and C are constant, based on equation (3), one can find:

$$Z(k) = \Psi' x(k) + \Upsilon' u(k-1) + \Theta' \Delta U(k) \quad (28)$$

Where:

$$\Psi' = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & C & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & C \end{bmatrix} \begin{bmatrix} A(1) \\ \vdots \\ A(1)A(2)\dots A(Hu) \\ \vdots \\ A(1)\dots A(Hu)\dots A(Hp) \end{bmatrix} \quad (29)$$

$$\Upsilon' = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & C & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & C \end{bmatrix} \begin{bmatrix} B(1) \\ B(2) + A(2)B(1) \\ \vdots \\ B(H_u) + A(H_u)(line(H_u-1)) \\ \vdots \\ B(H_p) + A(H_p)(line(H_p-1)) \end{bmatrix} \quad (30)$$

$$\Theta' = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & C & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & C \end{bmatrix} \begin{bmatrix} B(1) & \dots & 0 \\ B(2) + A(2)B(1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ B(H_p) + A(H_p)(line(H_p-1)) & \cdots & B(1) \end{bmatrix} \quad (31)$$

The proposed controller is explained in the following steps:

1. Take a guess and fix what could be the future inputs $\hat{u}(k+j),\ j = 1,2,\dots H_u - 1$. Use it to calculate $\hat{x}(k+i),\ i = 1,2,\dots H_p$ with the equation (24).

2. Substitute $\hat{u}(k+i)$ and $\hat{x}(k+i)$ in equation (27) to obtain a piecewise linear representation of the nonlinear system for each i: $A(i)$ and $B(i)$ $i = 1,2,\dots H_p$.

3. Formulate the equation (28).

4. Similarly to the Linear MPC, Use equation (23) to find the optimal ΔU.

5. Repeat steps 2, 3 and 4 until the norm vector between the new $\hat{u}(k+j)$ and the old $\hat{u}(k+j)$ is within a predetermined tolerance.

6. Once the norm vector reaches a desired value, the actual control input becomes the first sample of $\hat{u}(k)$.

7. Repeat steps 1 to 6 for each sampling instant k of the algorithm.

There will be an error between the predicted states and the states that would be generated by the nonlinear model due to the following reason: in step 4, the future values of the states are calculated from the application of optimal control sequence calculated in step 3 and the non-linear model. In step 2, these future values are used to form the piecewise linear model in the optimization of the next iteration.

However, this piecewise linear model only corresponds to the non-linear model if the future values of the states did not change from one iteration to another, i.e., this would occur if the optimal sequence $\hat{u}(k+j)$ did not modify from one iteration to the next. Thus, if this algorithm converges, the final values of $\hat{x}(k+i)$ and $\hat{u}(k+i)$ predicted by the piecewise linear model are equal to those predicted by the original nonlinear model.

Thus, if the optimization problem has a feasible solution, it ensures that constraints (12) are met along the prediction horizon. This is the largest advantage of the proposed method compared to those based on approximations of the model that cannot guarantee the satisfaction of constraints in the states.

Unlike strategies that focus on the direct resolution of the non-convex optimization problem, the strategy proposed here does not guarantee optimality. $V(k)$ Provides however, solutions that improve at each iteration of the algorithm.

III. ROBOTIC APPLICATION

The proposed robot is described by the following kinematic model:

$$\dot{x} = \cos(\theta)\, v$$
$$\dot{y} = \text{sen}(\theta)\, v$$
$$\dot{\theta} = \omega \tag{32}$$

Where v is the forward velocity, ω is the steering velocity, $(x;\ y)$ is the position of the mass center of the robot moving in the plane and θ denotes its heading angle from the horizontal axis.

It can be transformed into the following state space representation:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x\\ y\\ \theta\end{bmatrix} + \begin{bmatrix}\cos(\theta) & 0\\ sen(\theta) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}$$
$$z = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\ y\\ \theta\end{bmatrix} \tag{33}$$

Where z is the system output and it was decided to be used to control, independently, all three state variables.

The dynamic model is derived from the actuators dynamics and the robot dynamics parameters, like mass, inertia momentum and friction coefficients [4].

The suggested model indicates that the problem is truly nonlinear based on the matrix B, thus linear control is ineffective, even locally, and innovative design techniques are needed [3].

Here, the velocities v and ω are taken as the inputs and are subject to the following constraints:

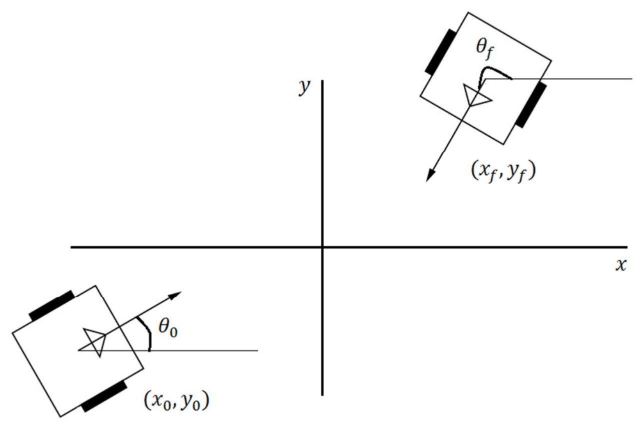
$$|u(t)| \leq u_{max}$$
$$x_{min} \leq x(t) \leq x_{max}$$
$$\forall t \tag{34}$$

Approximating the derivatives for finite differences using a sampling period equal to T, one obtains the following discrete-time model:

$$\begin{bmatrix}x(k+1)\\ y(k+1)\\ \theta(k+1)\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x(k)\\ y(k)\\ \theta(k)\end{bmatrix} + \begin{bmatrix}\text{T}\cos(\theta) & 0\\ Tsen(\theta) & 0\\ 0 & T\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}$$
$$z(k) = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x(k)\\ y(k)\\ \theta(k)\end{bmatrix} \tag{35}$$

Basically, the proposed controller is used to steer the robot from a starting position (x_0, y_0) with heading angle θ_0 to a (x_f, y_f) heading angle θ_f, according to Figure 2:

Figure 2: Control objective



IV. RESULTS

In all cases, the following conditions will be the same:

$$T = 0{,}1s,\ H_p = 10,\ H_u = 5,\ Q = 1,\ R = 0{,}1,$$
$$(x_0, y_0) = (0,0), |u(t)| \leq \begin{bmatrix}1\\ 0.005\end{bmatrix}$$

A. Verifying constraints applied to the states variables

Keeping the heading angle constant $\theta_0 = \theta_f = 0{,}7854$ rad, and $(x_f, y_f) = (4,5)$, figures 3, 4 and 5 shows the results when no constraint is applied to the state variables and figures 6,7 and 8 shows results when it is, as follows:

Figure 3: State variables

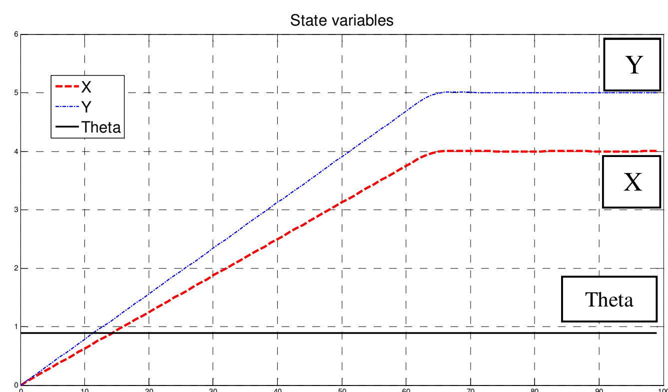


Figure 4: Input

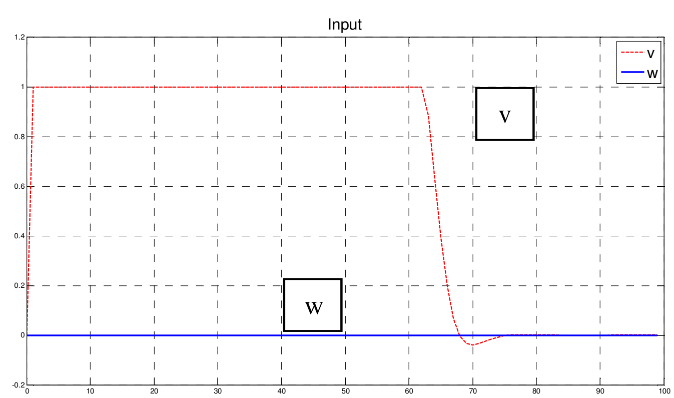


Figure 5: Robot trajectory

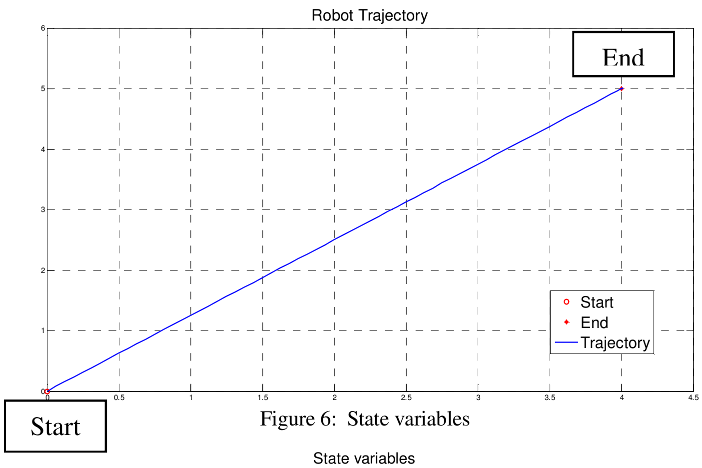


Figure 6: State variables

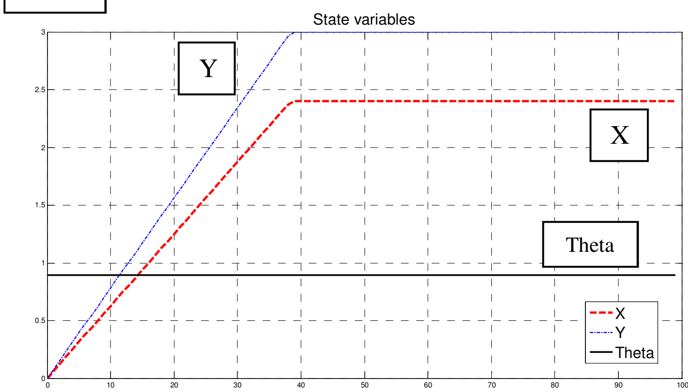


Figure 7: Input

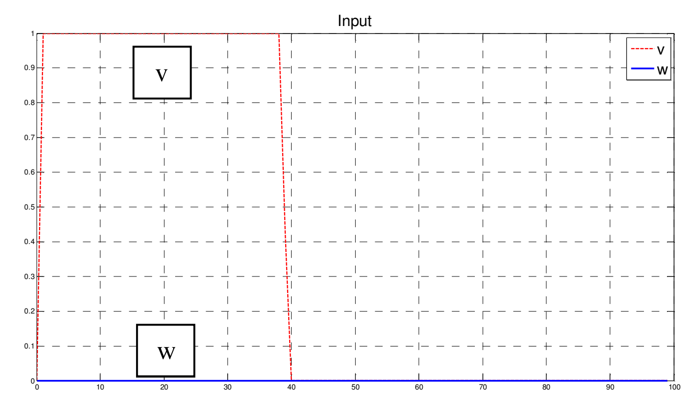
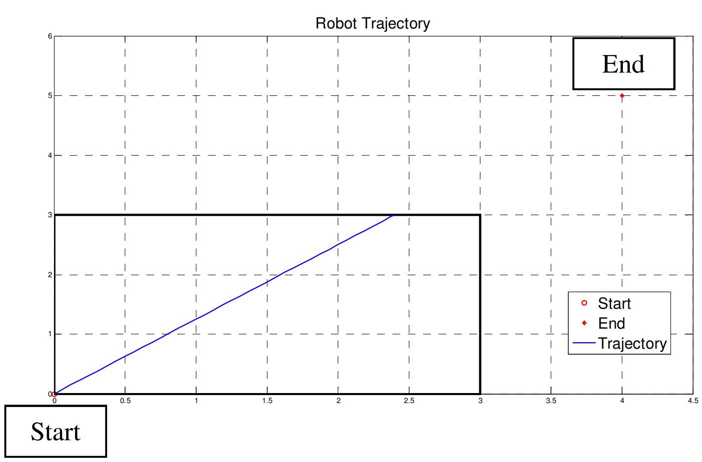


Figure 8: Robot trajectory



The constraint applied to state variables was:

$$x_{min} = \begin{bmatrix} 0 \\ 0 \\ -inf \end{bmatrix} \qquad x_{max} = \begin{bmatrix} 3 \\ 3 \\ +inf \end{bmatrix}$$

It is possible to notice that once one of the state variables reaches the imposed constraint, the robot stops (figure 8) according with the imposed constraint.

B. *Verifying the behavior when the heading angle varies*

Given $(\theta_0 \neq \theta_f)$, $\theta_0 = 1{,}3963$ rad $\theta_f = 0{,}5236$ rad, figures 9, 10 and 11 shows results for $(x_f, y_f) = (20, 20)$, while figures 12,13 and 14, for $(x_f, y_f) = (20, 10)$, as follows:

Figure 9: State variables

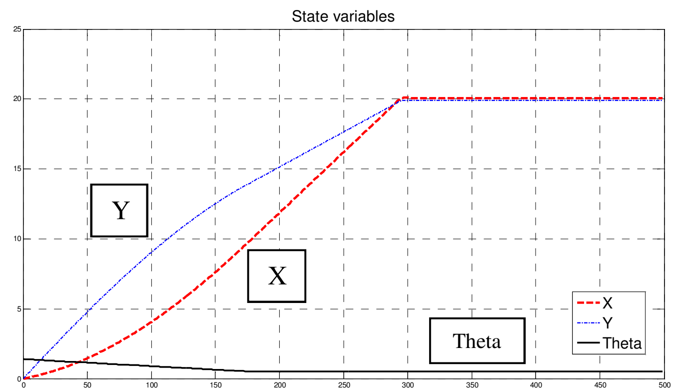


Figure 10: Input

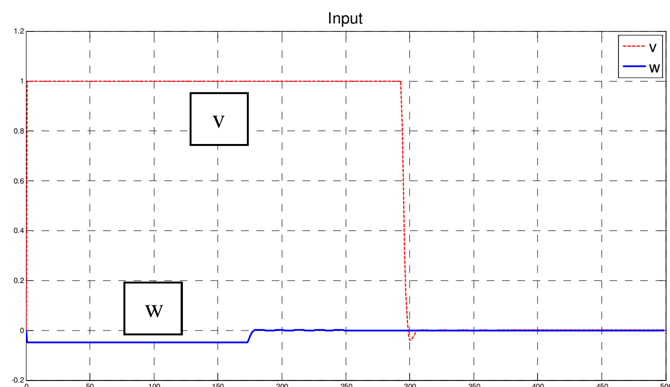


Figure 11: Robot trajectory

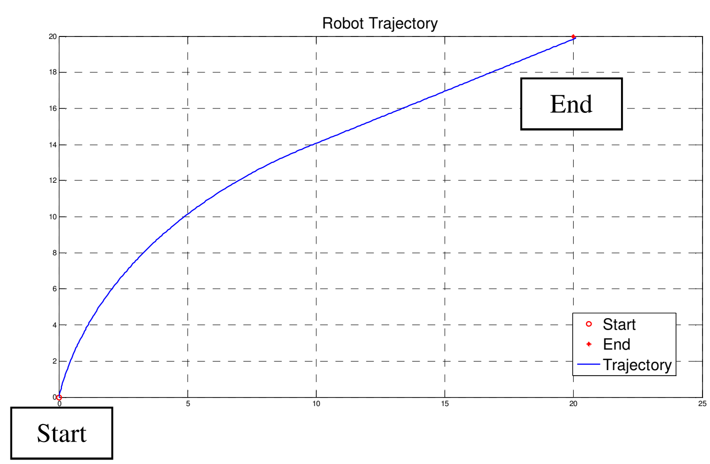


Figure 12: State variables

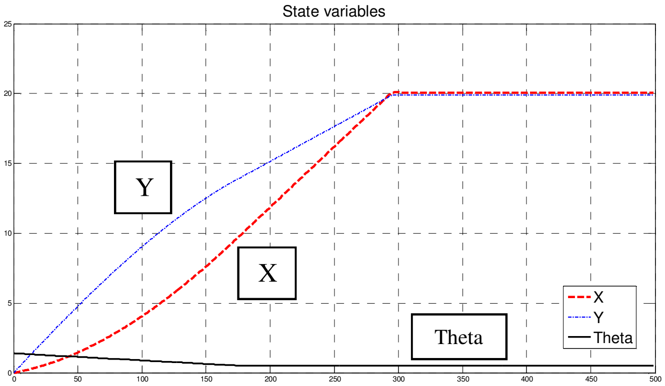


Figure 13: Input

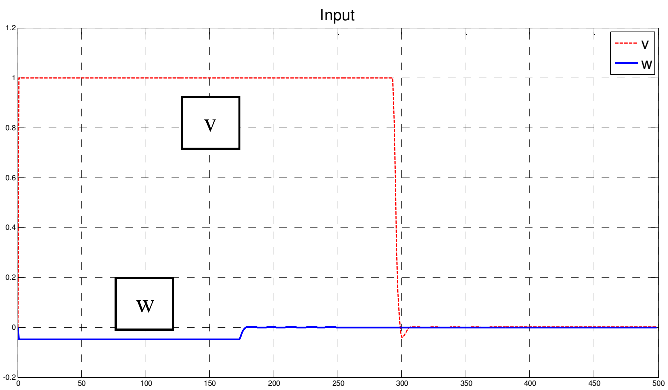
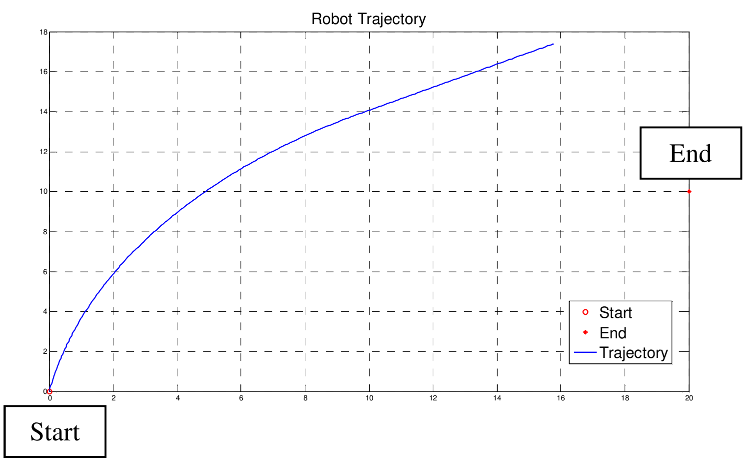


Figure 14: Robot trajectory



It is possible to notice that once the reference was changed to $(x_f, y_f) = (20, 10)$ the robot is unable to reach it (figure 14) as expected.

V. CONCLUSION

After analyzing the results, it was possible to visualize the satisfaction of the imposed constraints which is one of the advantages for the use of the proposed MPC, but because of the nonholonomic restrictions, constant reference and the number of the state variables higher than the number of inputs, the robot is not capable to reach the desired reference when heading angle varies.

A proposal for future work would be to provide a set of coordinates for a predetermined trajectory to be followed as a reference, or even the use of a more elaborated model that will allow the proposed controller show its full capabilities. Once it is done, comparison of results with other nonlinear methods of control would be motivating.

REFERENCES

[1] WANG, L. **Model Predictive Control System Design and Implementation**. Advances in Industrial Control. Springer, 2008.

[2] SIDEC, N. **Dynamic Modeling and Control of Nonholonomic Wheeled Mobile Robot Subjected to Wheel Slip**. PhD Thesis, Vanderbilt University, USA, 2008.

[3] SCHRAFT, R. D., SCHMIERER, G. **Serviceroboter**. Springer. Berlin,1998

[4] LUCA, A.; ORIOLO, G.; VENDITTELLI, M. **Control of Wheeled Mobile Robots: An Experimental Overview**. Universita degli Studi di Roma, Italy.

[5] CAMACHO, E. F.; BORDONS, C. **Model Predictive Control**. Springer-Verlag Ltd, 2004.

[6] FONTES, A.; DÓREA, C.; GARCIA, M. **An iterative algorithm for constrained mpc with stability of bilinear systems**, Proc. 16th Mediterranean Conference on Control andAutomation, Ajaccio, pp. 1526–1531, 2008.

[7] KOCH, S.; DURAISKI, R.; FERNANDES, P.; TRIERWEILER, J. **NMPC with statespace models obtained through linearization on equilibrium manifold**. Proc. Int. Symp. on Advanced Control of Chemical Process, pp. 1015–1020, 2006.

[8] MACIEJOWSKI, J. M. **Predictive Control with Constraints**. Person Education Limited, 2002.

[9] MATLAB. **version 7.10.0** (R2010a), The MathWorks Inc., Natick, Massachusetts, EUA, 2010.