DYNAMIC NONLINEAR MODEL BASED PREDICTIVE CONTROL FOR MOBILE ROBOTS

Andrés Rosales Acosta[†]

[†] *Área de Investigación y Desarrollo*
Centro Nacional de Control de Energía – CENACE
Panamericana Sur Km 17½, Santa Rosa. Quito, Ecuador
e-mail: arosales@cenace.org.ec

Abstract– **In this paper, a nonlinear predictive controller (NMPC) to control a unicycle-like mobile robot for trajectory tracking has been developed. A dynamic model of a PIONNER 3-DX mobile robot is used, where external forces and wheels sliding have been considered. Restrictions on control actions and system states are also considered. Simulations results at both tracking and regulation (positioning) are shown, these results show the good performance of the developed controller. Finally, this paper shows that the controller can be implemented in real-time by using an analysis of the calculating times of the NMPC algorithm.**

Key words– **mobile robot, predictive nonlinear control (NMPC), trajectory tracking, dynamic model**

I. INTRODUCTION

The trajectory tracking control for mobile robots is a fundamental problem, which has been exhaustively investigated by the scientific community. Several papers deal with the design of control laws for mobile robots considering their dynamic model, for instance in trajectory tracking (De la Cruz *et al.*, 2006; Dong *et al.*, 2005; Albagul *et al.*, 2004; Yang *et al.*, 1999; Zhang *et al.*, 1998). One of the first investigation results for this problem was dealt in Kanayama *et al.* (1990), where the author uses the Lyapunov theory to design the tracking controller. Nevertheless, this and other controllers do not take into account the restrictions in the control signals because it is a hard task to implement.

Nonlinear predictive control (NMPC) is a most frequently used control-optimization technique in the industry. This methodology has been designed to deal with optimization problems considering restrictions. NMPC is an online optimization algorithm that predicts the system output based on the current states and the system model, it also finds a control feature in open loop by using numeric optimization and applies the first control signal of this optimized feature to the system.

In model based predictive control, due to the use of the receding horizon, the stability analysis is one of the main problems (Peña, 2002). Previous papers have demonstrated that a finite receding horizon can guarantee NMPC stability even for nonlinear systems (Camacho *et al.*, 1998; Peña, 2002), although this strategy is considered heavy due to its computational effort in practice. The NMPC stability analysis with finite receding horizon has been studied in Mayne *et al.* (2000) and Fontes (2001). Recently, the qualities of predictive control have been explored and applied to robotics, papers like Dongbing *et al.* (2006), Hedjar *et al.* (2005), Künhe *et al.* (2005) and Ramírez *et al.* (1999) show different approaches and strategies in MPC with good results for tracking control and regulation. This paper applies NMPC to a dynamic model of a mobile robot and makes an analysis of the calculating times used by the optimization algorithm so that it can implement the controller in real-time.

The paper is organized as follows: Section II describes the NMPC algorithm and the programming schemes used. Section III shows the dynamic model of a mobile robot. Section IV shows the simulation results. Conclusions and future work are detailed in the last Section.

II. PREDICTIVE NONLINEAR CONTROL

A general deterministic nonlinear model at discrete time for a system can be expressed as,

$$\mathbf{x}(k+1)=\mathbf{f}(\mathbf{x}(k),\mathbf{v}(k)) \tag{1}$$

$$\mathbf{y}(k)=\mathbf{h}(\mathbf{x}(k)) \tag{2}$$

where $\mathbf{x}(k)$, $\mathbf{v}(k)$ and $\mathbf{y}(k)$ are the state vector, the control signal and the output system, respectively.

Most of the MPC methods are based on a common scheme (Ramírez *et al.*, 1999). A cost functional $\mathbf{J}$ is defined, which is often a quadratic function with the sum of the norms of future trajectory tracking errors,

$$\mathbf{e}(k+i\mid k)=\mathbf{y}(k+i\mid k)-\mathbf{y}_d(k+i) \tag{3}$$

predicted over a prediction horizon N plus the sum of the norms of predicted increments in the control action, over a control horizon N_u,

$$\mathbf{J}=\sum_{i=1}^{N}\delta_i\left\|\mathbf{e}(k+i|k)\right\|_{\mathbf{Q}}^2+\sum_{i=1}^{N_u}\lambda_i\left\|\Delta\mathbf{v}(k+i-1|k)\right\|_{\mathbf{R}}^2 \tag{4}$$

$$\Delta\mathbf{v}(k+i|k)=\mathbf{v}(k+i|k)-\mathbf{v}(k+i-1|k)$$

where, δ_i and λ_i are penalty sequences usually chosen as constants, $\mathbf{y}_d(k+i)$ is the desired output and the notation $\mathbf{y}(k+i|k)$ means that $\mathbf{y}(k+i)$ is computed with known information at instant k. The future system outputs $\mathbf{y}(k+i|k)$ for $i = 1, \ldots, N$, are predicted by using a process model from the inputs and outputs previous to the time instant k, and from the predicted future control actions $\mathbf{v}(k+i|k)$ for $i = 0, \ldots, N_u$-1, which will be calculated. In addition, $\|\mathbf{x}\|_{\mathbf{Q}}^2$ is defined as $\|\mathbf{x}\|_{\mathbf{Q}}^2 = \mathbf{x}^T\mathbf{Q}\mathbf{x}$ with $\mathbf{Q} > 0$.

In this way, **J** can be expressed as a function that only depends on the future control actions. The objective of predictive control is to obtain a sequence of future control actions [$\mathbf{v}(k)$, $\mathbf{v}(k+1|k)$, …, $\mathbf{v}(k+N_u-1|k)$], so that the predicted outputs $\mathbf{y}(k+i|k)$, using the system model, are near to the reference $\mathbf{y}_\mathrm{d}(k+i|k)$ as possible, along the prediction horizon. This is obtained by means of minimization of **J** with respect to the control variables. After obtaining this sequence, a *receding horizon* strategy is used, which applies only the first control action $\mathbf{v}(k)$ computed. This process is repeated every sampling time.

When nonlinear models are used, MPC depends on finding a solution for a nonlinear programming problem in every sampling step. To solve this problem it is necessary to do the optimization and to solve the system model. Both problems can be implemented by two different ways: sequential or simultaneously (Peña, 2002).

A. Sequential optimization algorithm

In the sequential implementation, a solution at each iteration for the optimization routine is found. The controls are the decision variables, which are processed by the algorithm computing the model solution. Then, this solution is used to evaluate the objective function and the computed value is given to the optimization program. The optimization variable is,

$$\mathbf{z} = \begin{bmatrix} \mathbf{v}(k) & \mathbf{v}(k+1|k) & \cdots & \mathbf{v}(k+N_u-1|k) \end{bmatrix}^T \qquad (5)$$

First, the functional must solve the system model using the vector values **z** and the current state $\mathbf{x}(k)$ applied N times to (1). In this way, the vector sequence $\begin{bmatrix} \mathbf{x}(k+1|k) & \mathbf{x}(k+2|k) & \cdots & \mathbf{x}(k+N|k) \end{bmatrix}$ is obtained, whereas with (2) the sequence output values $\begin{bmatrix} \mathbf{y}(k+1|k) & \mathbf{y}(k+2|k) & \cdots & \mathbf{y}(k+N|k) \end{bmatrix}$ are obtained. Then, the cost functional (4) is evaluated with these values.

The cost functional **J** depends on the predicted outputs, which are based on the state vector and this vector is function of the control actions (optimization variables). Therefore, the outputs must be derived with respect to the control actions from k to (k+N_u-1) to obtain the functional gradient. This is complicated and not always has solution. Therefore, the sequential resolution has no gradient information and it must be obtained by using numeric differentiation, which is computationally negative, because this generates greater calculus cost and convergence problems.

B. Simultaneous optimization algorithm

Unlike the sequential solution, the simultaneous solution and optimization include the states and the controls of the model like decision variables. The model equations are added to the optimization problem like restriction equations. Then, the optimization variable results,

$$\begin{aligned} \mathbf{z} = \big[& \mathbf{x}(k+1) \quad \mathbf{x}(k+1|k) \quad \cdots \quad \mathbf{x}(k+N|k) \\ & \mathbf{v}(k) \quad \mathbf{v}(k+1|k) \quad \cdots \quad \mathbf{v}(k+N_u-1|k) \big]^T \end{aligned} \qquad (6)$$

In other words, the states and control actions are considered like optimization variables. The dimension of this vector is (eN + pN_u), which is bigger than the sequential approach dimension (pN_u), where e and p are sizes of the state vector and control input, respectively. This requires a considerable increase in the optimization variable size in relation to the sequential approach.

The model equations appear like equality restrictions as follows in (6),

$$\mathbf{R} = \begin{cases} \mathbf{x}(k+1|k) = \mathbf{f}(\mathbf{x}(k), \mathbf{v}(k)) \\ \mathbf{x}(k+2|k) = \mathbf{f}(\mathbf{x}(k+1), \mathbf{v}(k+1)) \\ \vdots \\ \mathbf{x}(k+N|k) = \mathbf{f}(\mathbf{x}(k+N-1), \mathbf{v}(k+N_u)) \end{cases} \qquad (6)$$

In this approach, the gradient obtained by the analytic way is simpler, so that, the optimization algorithm can be incorporated in an explicit way. By using the functional in (4) and (7). The equality restrictions gradient is a disperse matrix (8).

$$\nabla\mathbf{J} = \begin{bmatrix} 2\delta_1 \mathbf{G}_{\mathbf{x}_{k+1}}^T \mathbf{Q}\left[\mathbf{h}(\mathbf{x}(k+1|k)) - \mathbf{y}_d(k+1)\right] \\ 2\delta_2 \mathbf{G}_{\mathbf{x}_{k+2}}^T \mathbf{Q}\left[\mathbf{h}(\mathbf{x}(k+2|k)) - \mathbf{y}_d(k+2)\right] \\ \vdots \\ 2\delta_N \mathbf{G}_{\mathbf{x}_{k+N}}^T \mathbf{Q}\left[\mathbf{h}(\mathbf{x}(k+N|k)) - \mathbf{y}_d(k+N)\right] \\ 2\lambda_1 \mathbf{R}\Delta\mathbf{v}(k) - 2\lambda_2 \mathbf{R}\Delta\mathbf{v}(k+1) \\ 2\lambda_2 \mathbf{R}\Delta\mathbf{v}(k+1) - 2\lambda_3 \mathbf{R}\Delta\mathbf{v}(k+2) \\ \vdots \\ 2\lambda_{N_u} \mathbf{R}\Delta\mathbf{v}(k+N_u-1) \end{bmatrix} \qquad (7)$$

$$\nabla\mathbf{R} = \begin{bmatrix} \mathbf{I}_{exe} & -\mathbf{Fx}(k+1) & \cdots & 0_{exe} & 0_{exe} \\ 0_{exe} & \mathbf{I}_{exe} & \cdots & 0_{exe} & 0_{exe} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_{exe} & 0_{exe} & \cdots & \mathbf{I}_{exe} & -\mathbf{Fx}(k+N-1) \\ 0_{exe} & 0_{exe} & \cdots & 0_{exe} & \mathbf{I}_{exe} \\ -\mathbf{Fv}(k) & 0_{exe} & \cdots & 0_{exe} & 0_{exe} \\ 0_{exe} & -\mathbf{Fv}(k+1) & \cdots & 0_{exe} & 0_{exe} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_{exe} & 0_{exe} & \cdots & 0_{exe} & -\mathbf{Fv}(k+N_u-1) \end{bmatrix} \qquad (8)$$

where $\mathbf{G}_{\mathbf{x}_i}^T = \left.\dfrac{\partial \mathbf{h}(\mathbf{x}(k),\mathbf{v}(k))}{\partial \mathbf{x}(k)}\right|_{k=i}$, $\mathbf{F}_{\mathbf{x}_{k+i}} = \left.\dfrac{\partial \mathbf{f}(\mathbf{x}(k),\mathbf{v}(k))}{\partial \mathbf{x}(k)}\right|_{k=i}$, $\mathbf{F}_{\mathbf{v}_{k+i}} = \left.\dfrac{\partial \mathbf{f}(\mathbf{x}(k),\mathbf{v}(k))}{\partial \mathbf{v}(k)}\right|_{k=i}$, $\mathbf{I}_{exe}$ is the identity matrix with e dimension and $\mathbf{0}_{pxe}$ is a null matrix with pxe dimension.

In small problems with few states and a short prediction horizon, a sequential approach is probably more effective (Peña, 2002). Generally, in big problems, a simultaneous approach is more robust, because it is less probable that it fails. In a sequential approach, the addition of restrictions to the states or outputs is more complicated. Besides such model restrictions, control actions restrictions, steady state restrictions, etc, can be added.

III. DYNAMIC MODEL OF MOBILE ROBOT

A system model is necessary to use an MPC strategy. This model will be used to predict the future position and orientation of the controlled system.

Figure 1: Mobile robot PIONNER 3-DX

A unicycle-like mobile robot experimentally validated by using PIONNER 3-DX robots in De la Cruz et al. (2006) will be used. A brief model is shown in Fig. 2 and it is presented in (9). The model with more details can be seen in De la Cruz *et al.* (2006).

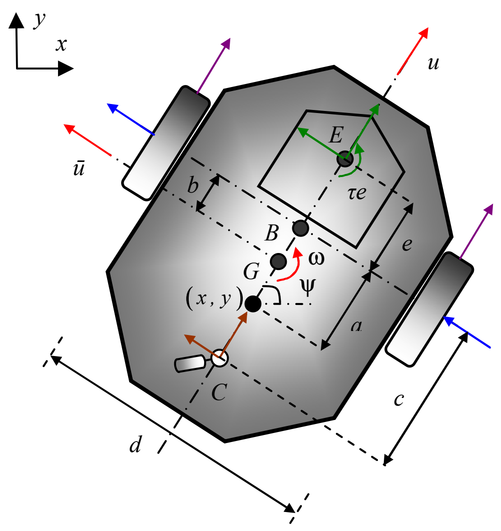

Figure 2: Mobile robot unicycle-like model and its parameters

The robot position is defined by $h=[x,y]^T$, this point is located at a distance a from the rear axis centre of the robot, u and $\bar{u}$ are the longitudinal and side speeds of the mass centre, ω is the angular speed and ψ is the orientation angle of the robot, G is the gravity centre, B is the base line centre of the wheels, E is the work tool placing point and C is the castor wheel placing point.

From the model described in Fig. 2, the dynamic model of mobile robot is obtained as (De la Cruz *et al.*, 2006),

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{u} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} u\cos\psi - a\omega\sin\psi \\ u\sin\psi + a\omega\cos\psi \\ \omega \\ \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}u \\ -\frac{\theta_5}{\theta_2}u\omega - \frac{\theta_6}{\theta_2}\omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{\theta_1} & 0 \\ 0 & \frac{1}{\theta_2} \end{bmatrix} \begin{bmatrix} u_c \\ \omega_c \end{bmatrix} \quad (9)$$

where the mobile robot parameters were validated in De la Cruz *et al.* (2006), these parameters are,

$$\theta_1 = 0.24089 \quad \theta_2 = 0.2424 \quad \theta_3 = -0.00093603$$
$$\theta_4 = 0.99629 \quad \theta_5 = -0.0037256 \quad \theta_6 = 1.0915$$

Equation (9) can be wrote in a compact way as follows,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{g}(\mathbf{v}(t)) \quad (10)$$

where $\mathbf{x}(t) = [x \;\; y \;\; \psi \;\; u \;\; \omega]^T$ is the state vector of the system and $\mathbf{v}(t) = [u_c \;\; \omega_c]^T$ is the control vector.

The dynamic model of the mobile robot can be discretized by using any numeric approximation approach, for example Euler, in other words,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T_0\left[\mathbf{f}(\mathbf{x}_k) + \mathbf{g}(\mathbf{v}_k)\right] \quad (11)$$

where T_0 is the sampling time and the vector of initial conditions is $\mathbf{x}_0(t) = [x_0 \;\; y_0 \;\; \psi_0 \;\; u_0 \;\; \omega_0]^T$.

Equation (12) is the system output,

$$\mathbf{y}(k) = \mathbf{h}(\mathbf{x}(k)) = \mathbf{C}\mathbf{x}(k) \quad (12)$$

where $\mathbf{C}$ is a matrix of oe dimension (o is the state vector size). In this paper, the output equation is given by,

$$\mathbf{y}(k) = \left[x(k) \;\; y(k) \;\; \psi(k)\right]^T \quad (13)$$

IV. SIMULATION AND RESULTS

The maximal absolute values of linear and angular speeds of the mobile robot used in the simulations are 0.5[m/s] and 0.745[rad/s], respectively.

A sampling time T_0 = 0.1[s] was used for all trajectories and the prediction horizon used was $N = N_u$ = 7. Furthermore, we set $\mathbf{Q}$ = diag[1, 1, 0.05], $\mathbf{R} = \mathbf{I}$ (identity matrix) and δ = 28 and λ = 0.8. Linear reference speed was 0.25[m/s] for both reference trajectories.

The reference values used for the state ψ, were computed by using (14),

$$\psi_d = \text{atan}\left(\frac{\dot{y}_d}{\dot{x}_d}\right) \qquad (14)$$

The first reference trajectory applied to the system was an eight-shaped curve defined by,

$$x_d = \sin(t/k), \qquad y_d = \cos(t/k)$$

where $k \in \Re^+$ and in this case k = 10. The initial condition vector was $\mathbf{x}_0(t) = [0.5 \quad 0 \quad 0 \quad 0 \quad 0]^T$.

The trajectory made by the mobile robot to follow the first reference is shown in Fig. 3, where the maximum error in this case was 3[mm].

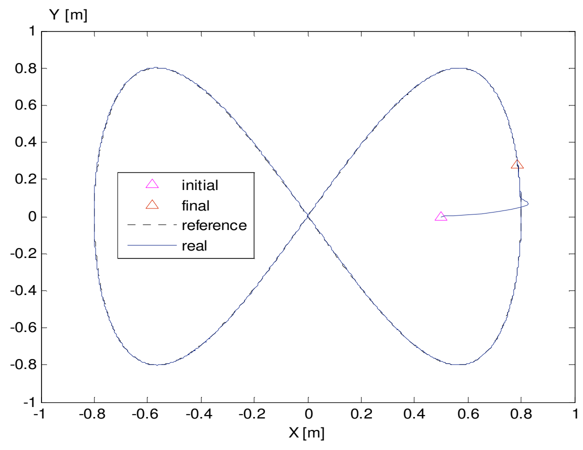

Figure 3: Eight-shaped trajectory for mobile robot

Figure 4 shows the *x-y* mobile robot position and the quadratic error, which tends to zero.

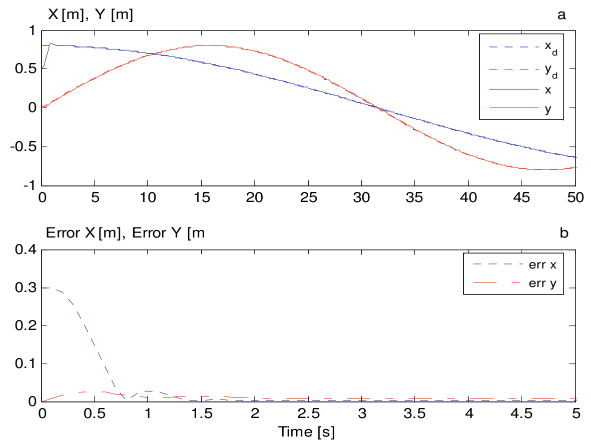

Figure 4: *x-y* mobile robot states (a) and quadratic error (b)

Figure 5 shows the control actions of the system compared with the real linear and angular speeds of the mobile robot.

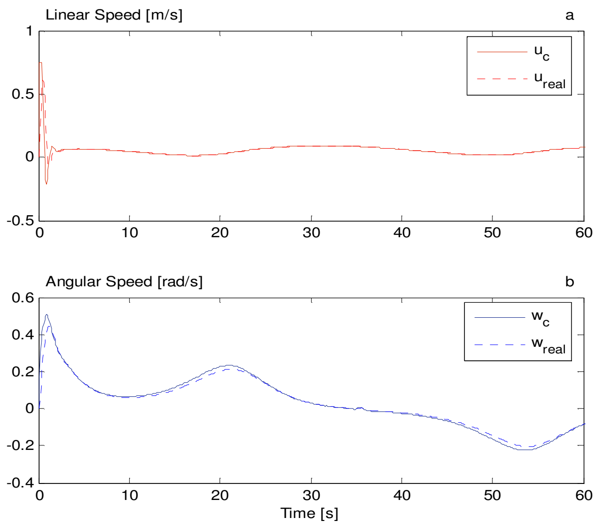

Figure 5: Control actions (a) u_c and (b) ω_c

Figure 6 shows the execution time of the optimization algorithm for each sampling time in the simulations.

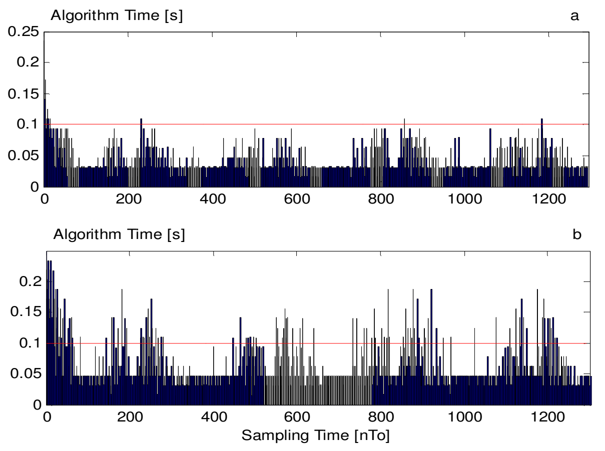

Figure 6: Algorithm execution time during simulation (a) sequential (97% < T_0), (b) simultaneous (72% < T_0)

The sequential algorithm is shown in Fig. 6(a), where 97% of the execution times are smaller than T_0, whereas the simultaneous algorithm is shown in Fig. 6(b), where 72% of the execution times are smaller than T_0.

The second trajectory was applied to check the positioning control of the system,

$$x_d = \begin{cases} 2r\cos(\omega_r t + \pi/4), \\ -r\sqrt{2}, \end{cases} \qquad y_d = \begin{cases} r\sin(2\omega_r t + \pi/4), & 0 \le t < 5\pi \\ -r, & t \ge 5\pi \end{cases}$$

where $r \in \Re^+$. In this case r = 0.4[m] and ω_r = 0.1[rad/s], with initial conditions: $\mathbf{x}_0(t) = [1.4 \quad 0.9 \quad 7\pi/6 \quad 0 \quad 0]^T$.

The trajectory described by the mobile robot for the first tracking of a path and the next positioning in a specific place is shown in Fig. 7.

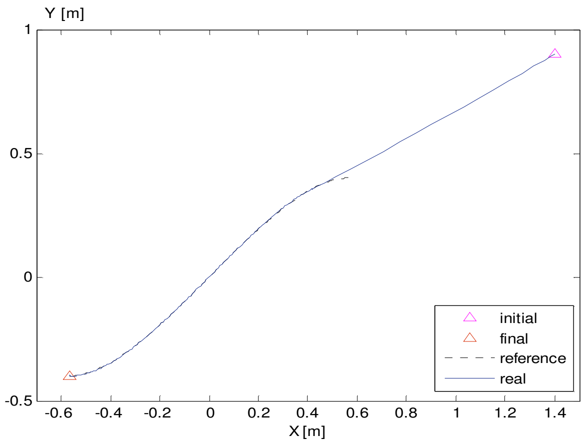

Figure 7: Positioning trajectory for mobile robot

Figure 8 shows the *x-y* position of the mobile robot and the quadratic error, which tends to zero.

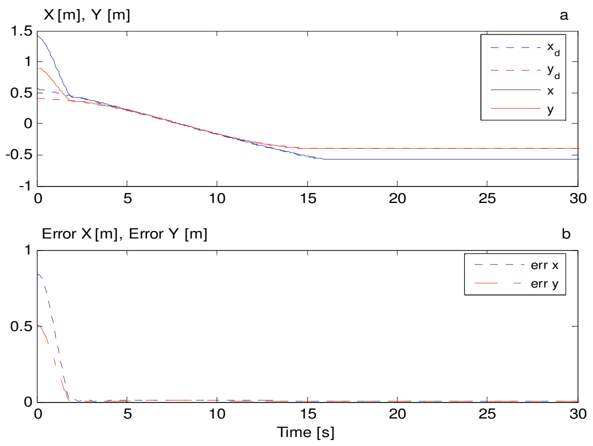

Figure 8: *x-y* mobile robot states (a) and quadratic error (b)

Figure 9 shows the control actions of the system compared with the real linear and angular speeds of the mobile robot. We remark that when the mobile robot arrives to the desired position, it stays with null speed.

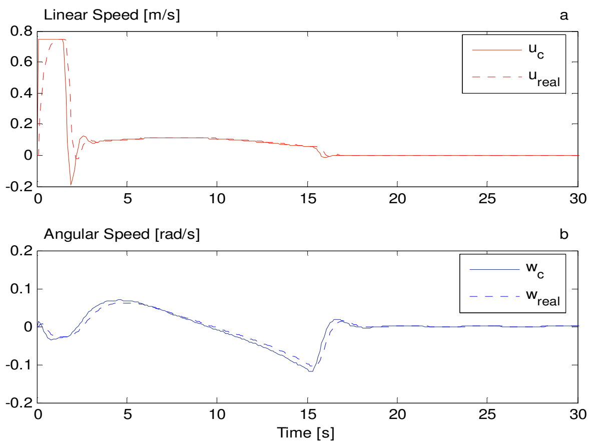

Figure 9: Control actions (a) u_c and (b) ω_c

Figure 10 shows the execution time of the optimization algorithm for each sampling time in the simulations.

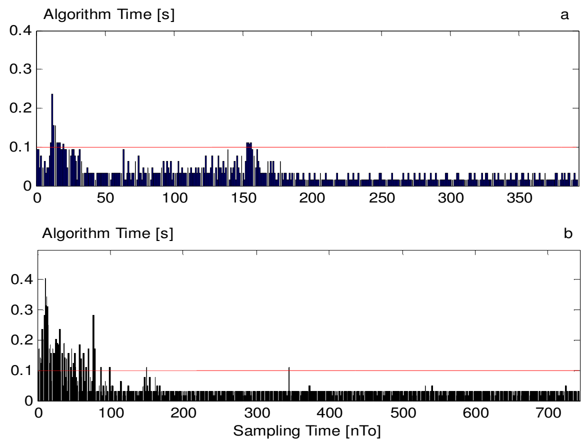

Figure 10: Algorithm execution time during the simulation (a) sequential (93% < T_0), (b) simultaneous (81% < T_0)

The sequential algorithm is shown in Fig. 10(a), where 93% of the execution times are smaller than T_0, whereas the simultaneous algorithm is shown in Fig. 10(b), where 81% of the execution times are smaller than T_0.

V. CONCLUSIONS

The problem of leading a mobile robot through previous computed trajectories has been solved by using NMPC strategy like navigation algorithm. A dynamic holonomic model of a unicycle-like mobile robot has been used.

A good performance of the control system, by means of NMPC, has been achieved for both positioning and trajectory tracking. The results were obtained by using the NMPC sequential approach, by using it, the calculating times were better than the NMPC simultaneous one, as it is observed in the simulations.

The proposed controller can be implemented in real-time as it was shown in the obtained results analysis. By using a suboptimal scheme, good results have been achieved; this proves that by using specific software for this problem, it is possible to reach better results.

Simulations show that most of NMPC calculating times are below the ranges accepted by the mobile robot ($T_0 = 0.1$[s]). Furthermore, preliminary calculating times are higher due to the fact that the algorithm starts from distant initial conditions to the problem optimum.

ACKNOWLEDGEMENTS

This work was partially funded by the German Academic Exchange Service (DAAD – Deutscher Akademischer Austausch Dienst).

REFERENCES

Albagul A. y Wahyudi, “Dynamic Modeling and Adaptive Traction Control for Mobile Robots”, *Annual Conference IEEE IES*, pp. 614-620, (2004).

Camacho E. y Bordons C., *Model Predictive Control in the Process Industry*, Springer-Verlag, (1998).

De la Cruz C. y Carelli R., “Linealización con Realimentación del Modelo Dinámico de un Robot Móvil y Control de Seguimiento de Trayectoria”, *AADECA*, (2006).

Dong W. y Kuhnert K., “Robust Adaptive Control of Nonholonomic Mobile Robot with Parameter and Nonparameter Uncertainties“, *IEEE Transactions on Robotics*, pp. 261-266, (2005).

Dongbing G. y Huosheng H., “Receding Horizon Tracking Control of Wheeled mobile Robots”, *Control Systems Technology*, vol. 14, pp. 743-749, (2006).

F.A.C.C. Fontes, “A General Framework to Design Stabilizing Nonlinear Model Predictive Controllers”, *Sist. Control Lett.*, pp. 127-143, (2001).

Hedjar R., Toumi R., Boucher P. y Dumur D., “Finite Horizon Nonlinear Predictive Control by the Taylor Approximation: Application to Robot Tracking Trajectory”, *Int. J. Appl. Math. Comp. Sci.*, **vol. 15**, pp. 527-540, (2005).

Kanayama Y., Kimura Y., Miyazaki F. y Noguchi T., “A Stable Tracking Control Method for an Autonomous Mobile Robot”, *Proc. IEEE ICRA*, pp. 384-389, (1990).

Künhe F., Gomes J. y Fetter W., “Mobile Robot Trajectory Tracking using Model Predictive Control”, *II IEEE LARS*, (2005).

Mayne D., Rawlings J., Rao C. y Scokaert P., “Constrained Model Predictive Control: Stability and Optimality”, Automatica, pp. 789-814, (2000).

Peña M., *Control basado en Modelos Borrosos*, Tesis de Doctorado – INAUT – UNSJ, (2002).

Ramírez D., Limón-Marruedo D., Gómez-Ortega J. y E. Camacho, “Aplicación del Control Predictivo basado en Modelo No Lineal a la Navegación de un Robot Móvil utilizando Algoritmos Genéticos”, *Métodos Numéricos en Ingeniería*, (1999).

Secchi H., *Control de Vehículos Autoguiados con Realimentación Sensorial*, Tesis de Maestría – INAUT – UNSJ, (1998).

Yang J. y Kim H., “Sliding Mode Control for Trayectory Tracking of Nonholonomic Wheeled Mobile Robots”, *IEEE Transactions on Robotics and Automation*, pp. 578-587, (1999).

Zhang Y., Hong D., Chung J. y Velinsky S., “Dynamic Model Based Robust Tracking Control of a Differentially Steered Wheeled Mobile Robot”, *Proceedings of the American Control Conference*, pp. 850-855, (1998).