ROBÓTICA 2007 - 7th Conference on Mobile Robots and Competitions
Centro Paroquial de Paderne, Albufeira, Portugal
April 27, 2007

Predictive Control Applied to 5DPO - RoboCup Middle-size Omnidirectional Robots

André Scolari Conceição*, A. Paulo Moreira, Paulo J. Costa
Department of Electrical and Computer Engineering
University of Porto
Porto - Portugal.
[scolari,amoreira,paco]@fe.up.pt

***Abstract*—This paper presents a nonlinear model based predictive controller (NMPC) for trajectory tracking of the Middle-size Omnidirectional Robots from the 5DPO Robotic Soccer team. The strategy proposed uses methods of numerical optimization to perform real time nonlinear minimization of the cost function. The cost function penalizes the robot position error, the robot orientation angle error and the control effort. Experimental results of the trajectories following and the performance of the methods of optimization are presented.**

I. INTRODUCTION

Nowadays, the prediction concept in the mobile robotics is very important, because the robots are inserted more and more in dynamic environments(for example: robotic soccer, manufacturing plants, agriculture). Above all, in Robotic soccer's application the robots need to execute trajectories quickly and with a perfect position to the objective, for example, positioning to the ball, or to the goal, or to avoid dynamic obstacles. Therefore, the capacity to predict the control actions to follow a path or to avoid obstacles is very useful.

Several applications of predictive control techniques can be found in the literature. The article [1] talks about the way of implementing a model-based predictive controller for mobile robot navigation using genetic algorithms to perform online nonlinear optimization. The cost function was defined as a quadratic function of sum of the future errors in reference tracking. Reference [2] presents a tracking method for a mobile robot which combines predictive control and fuzzy logic control, where the predictive control is used to predict the position and the orientation of the robot, while the fuzzy control is used to deal with the non-linear characteristics of the system. A path tracking scheme for mobile robot based on neural predictive control is presented in [3], where a multi-layer back-propagation neural network is employed to model non-linear kinematics of the robot. In [4], a neural network multilayer perceptron has been trained to reproduce the MBPC behaviour in a supervised way. In [5], the minimization of the cost function has to be carried out by a numerical optimization method, a neural network is used to solve the problem. Another strategy using neural network is used in [6], where a neural-network-based technique for developing nonlinear dynamic models from empirical data for an model predictive control (MPC) algorithm is presented. A genetic algorithm based predictive control strategy which allows the minimization of a nonlinear cost function in real time is presented in [7].

In this paper a new approach of predictive controller is presented, where methods of numeric optimization are used to obtain the minimization of the controller's cost function. This approach became possible when we obtained good times of minimization of the controller's cost function (< 30 milliseconds). Nowadays with potent processors in personal computers, the use of optimization algorithms became viable.

The paper is organized as follows. Section II presents a brief description of the omni-directional mobile robot. Section III presents the NMPC elements: NMPC scheme, prediction model, cost function and control law. In this section the optimization methods are presented. The experimental results of the NMPC are presented in section IV. Finally, the conclusions are drawn in section V.

II. ROBOT DESCRIPTION

The robot, Fig. 3(a), is equipped with four omni-directional wheels connected to geared motors. Connected to each wheel there is a industrial encoder to measure its speed. Each pair wheel-encoder is connected to a controller board. This board has a microcontroller that measures the wheel speed and implements a local controller. This controller maintains the requested speed and is based on PID. This low level loop has a sampling frequency of 100Hz. The four controllers are connected to the PC by a RS-232 link running at 115200 baud. The robot has a standard PC motherboard with a 2GHz Celeron processor. Nevertheless, there is no hard disk and the Linux OS and the programs are stored in a 256 MBytes Compact Flash Card connected to the IDE. Another very important module is the one that deals with the image captured by the onmidirectional vision system and extracts the most important features. This information is used to construct an estimation of the robot position. The vision camera also provides the sample time control of the robot (t_s=40 milliseconds). The mobile robot was built for the 5DPO Robotic Soccer team from the Department of Electrical and Computer Engineering at the University of Porto at Porto, Portugal.

*The author is supported by the Program Alβan, the European Union Program of High Level Scholarships for Latin America, scholarship n.E04D028256BR

III. Model Predictive Controller

The strategy of the MPC controller is characterized by scheme represented in Fig. 1. A process model is used to predict the future outputs, based on past and current values and on the proposed future control actions. These actions are calculated by optimization of a cost function. Take into account the trajectory tracking problem, the cost function J of the predictive controller is defined as follow:

$$\begin{aligned} J(N_1, N_2, N_u) &= \sum_{j=N_1}^{N_2} \lambda_1 \left([\hat{x}(k+j|k) - x_c(k+j)]^2 + \right. \\ &\qquad \left. [\hat{y}(k+j|k) - y_c(k+j)]^2 \right) + \\ &\quad \sum_{j=N_1}^{N_2} \lambda_2 [\hat{\theta}(k+j|k) - \theta_c(k+j)]^2 + \\ &\quad \sum_{j=1}^{N_u} \lambda_3 [\Delta U(k+j-1)]^2 \end{aligned} \tag{1}$$

where the first term in J penalizes the position error, the second term penalizes the orientation angle error and the third penalizes the control effort(ΔU). N_1 and N_2 ate the minimum and maximum prediction horizons and N_u is the control horizon. The coefficients λ_1, λ_2 and λ_3 are penalty factors, which are usually chosen to be constant along the time. $\hat{Y}(k+j|k) = [\hat{x}(k+j|k) \;\; \hat{y}(k+j|k) \;\; \hat{\theta}(k+j|k)]^T$, is an j step prediction of the robot position and orientation angle made at instant k, and $W(k+j) = [x_c(k+j) \;\; y_c(k+j) \;\; \theta_c(k+j)]^T$ is desired robot position and orientation angle. $\Delta U(k+j-1) = [\Delta v(k+j-1) \;\; \Delta vn(k+j-1) \;\; \Delta w(k+j-1)]^T$ is the control effort, where v and vn are the robot linear velocities and w is robot angular velocity, which are the control variables.

A. *Reference trajectory of the controller, W*

The predictive controller needs the desired positions and orientation angles of the robot for the next N periods of time, see Fig. 2. The trajectory R is defined as points in the world frame(OXY): $R(i) = [x_r(i) \; y_r(i) \; \theta_r(i)]^T$, $i = 0, 1, ..., M$, where M is the total number of the trajectory points. The initial position $(x_c, y_c)_k$ is located at the intersection between the desired trajectory and its perpendicular, traced from the robot position (x, y). The next N points are spaced equally on the trajectory R by ΔS(meters), which is a design parameter. The desired orientation (θ_c) is the linear variation between the reference angular position $\theta_r(i)$ and $\theta_r(i+1)$ of the trajectory R:

$$\theta_c = \theta_r(i)(1-d) + \theta_r(i+1)d, \tag{2}$$

where d is the projection from robot position(x, y) to line segment $(x_r, y_r)_i$ and $(x_r, y_r)_{i+1}$, it is normalized to length of the line segment.

B. *Prediction model*

The use of the process model is determined by the necessity to predict the future positions and orientation angle of the robot at future instants, $\hat{Y}(k+j|k) = [\hat{x}(k+j|k) \;\; \hat{y}(k+j|k) \;\; \hat{\theta}(k+j|k)]^T$. The omni-directional mobile robot model is developed based on the dynamics, kinematics and DC motors of the robot.

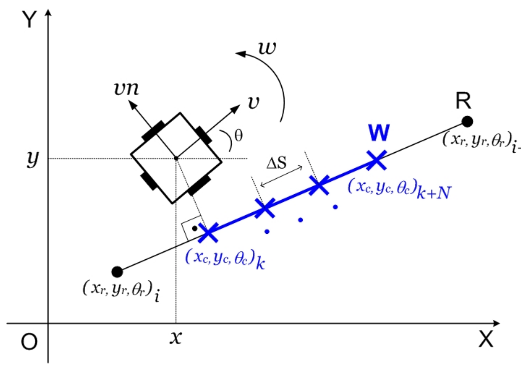
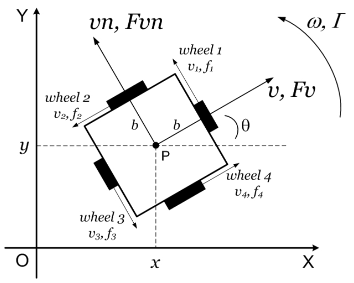


Fig. 2. Controller reference trajectory, W.

(a) Mobile robot.



(b) Geometric parameters and coordinate frames.

Fig. 3. Omni-Directional robot

The World frame(OXY), the robot's body frame and the geometric parameters is shown in Fig. 3(b). The following symbols, in SI unit system, are used to modelling:

- $b \; (m) \rightarrow$ distance between the point P and robot's wheels
- $M \; (kg) \rightarrow$ robot mass
- $r \; (m) \rightarrow$ wheel radius
- $l \; \rightarrow$ motor reduction
- $v, vn \; (m/s) \rightarrow$ linear velocities of the robot
- $w \; (rad/s) \rightarrow$ angular velocity of the robot
- $\theta \; (rad) \rightarrow$ orientation angle of the robot
- $J \; (kg.m^2) \rightarrow$ robot inertia moment
- $B_v, \; B_{vn} \; (N/(m/s)) \rightarrow$ viscous friction related to v and v_n
- $B_w \; (N/(rad/s)) \rightarrow$ viscous friction related to w
- $C_v, \; C_{vn} \; (N) \rightarrow$ coulomb friction related to v and v_n
- $C_w \; (N.m) \rightarrow$ coulomb friction related to w
- $F_v, F_{vn} \; (N) \rightarrow$ traction forces of the robot
- $\Gamma \; (N.m) \rightarrow$ rotation torque of the robot
- $v_1, v_2, v_3, v_4 \; (m/s) \rightarrow$ wheels linear velocities
- $f_1, f_2, f_3, f_4 \; (N) \rightarrow$ wheels traction forces
- $T_1, T_2, T_3, T_4 \; (N.m) \rightarrow$ wheels rotation torque

1) Robot Dynamics: By Newton's law of motion and the robot's body frame, in Fig. 3(b), we have

$$F_v(t) = M\frac{dv(t)}{dt} + B_v v(t) + C_v sgn(v(t)) \tag{3}$$

$$F_{vn}(t) = M\frac{dvn(t)}{dt} + B_{vn} vn(t) + C_{vn} sgn(vn(t)) \tag{4}$$

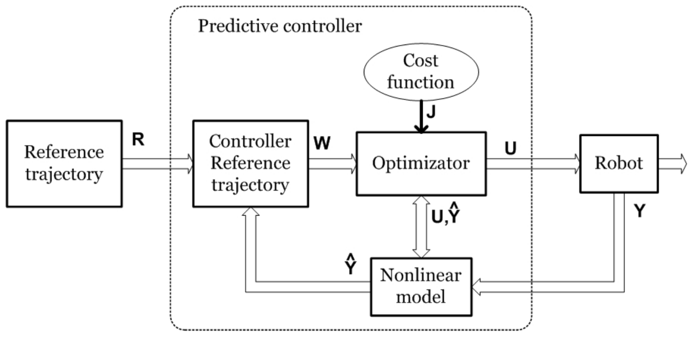


Fig. 1. Scheme of the predictive controller.

$$\Gamma(t) = J\frac{dw(t)}{dt} + B_w w(t) + C_w sgn(w(t)) \quad (5)$$

where,

$$sgn(\alpha) = \begin{cases} 1, & \alpha > 0, \\ 0, & \alpha = 0, \\ -1, & \alpha < 0. \end{cases}$$

The relationships between the robot's traction forces and the wheel's traction forces are,

$$F_v(t) = f_4(t) - f_2(t) \quad (6)$$

$$F_{vn}(t) = f_1(t) - f_3(t) \quad (7)$$

$$\Gamma(t) = (f_1(t) + f_2(t) + f_3(t) + f_4(t))b \quad (8)$$

The wheel's traction force(f) and the wheel's torque(T), for of each DC motor, is as follow:

$$f(t) = \frac{T(t)}{r} \quad (9)$$

$$T(t) = l.K_t.i_a(t) \quad (10)$$

where K_t is motor torque constant and $i_a(t)$ is the armature current. The armature current is limited to save battery,

$$0 \leq i_a(t) \leq i_{max}$$

where i_{max} is a design parameter.

The dynamics of each DC motor can be described using the following equations,

$$u(t) = L_a\frac{di_a(t)}{dt} + R_a i_a(t) + K_v w_m(t) \quad (11)$$

$$T(t) = K_t i_a(t) \quad (12)$$

where L_a is the armature inductance, R_a is the armature resistance, $w_m(t)$ is the rotor angular velocity in rad/sec, k_v is the emf constant. In SI unit system, the values of K_t and K_v are identical, see [8]: $K_t(N.m/A) = K_v(Volts/(rad/sec))$. The armature voltage is $u(t)$, with input signal constraints,

$$-24 \leq u(t) \leq 24 \quad \text{for} \quad t \geq 0.$$

2) Robot Kinematics: By geometric parameters of the robot and the robot's body frame, in Fig. 3(b), is possible to derive the motion equations,

$$\begin{array}{l} \frac{dx(t)}{dt} = v(t)cos(\theta(t)) - vn(t)sen(\theta(t)) \\ \frac{dy(t)}{dt} = v(t)sen(\theta(t)) + vn(t)cos(\theta(t)) \\ \frac{d\theta(t)}{dt} = w(t) \end{array} \quad (13)$$

The relationships between wheel's linear velocities (v_1, v_2, v_3 and v_4) and robot velocities (v,vn and w) are,

$$\begin{array}{l} v_1(t) = vn(t) + bw(t) \\ v_2(t) = -v(t) + bw(t) \\ v_3(t) = -vn(t) + bw(t) \\ v_4(t) = v(t) + bw(t) \end{array} \quad (14)$$

where $x(t)$ and $y(t)$ is the localization of the point P, and $\theta(t)$ the orientation angle of the robot.

C. Control law

The robot model is nonlinear, so an analytical solution cannot be obtained and an iterative method of optimization in real time should be used. In order to obtain control values $U(k + j|k)$ it is necessary to minimize the cost function J, in 1. Between many different methods of numerical optimization, in [9]-[14], methods with low computational time and simple implementation was chosen for test in the controller. Three optimization methods had been tested:

- The method of steepest descent, also known as the gradient method (G);
- Nonlinear Conjugate Gradient method - Fletcher Reeves(GCFR);
- Nonlinear Conjugate Gradient method - Polak Robiere(GCPR).

The method of steepest descent is the simplest example of a gradient based method for minimizing a function of several variables. It has a simple implementation, but with slow convergence. The conjugate gradient method is slightly more complicated than steepest descent, but it converges faster than steepest descent. This methods make good optimization progress because it is based on gradients.

The algorithm 1 presents the method of steepest descent for minimizing $f(x)$.

The algorithm 2 presents the Fletcher Reeves and Polak Robiere conjugate gradient method for minimizing $f(x)$. The methods Flecher Reeves and Polak Robiere differ only in the

Algorithm 1 - Steepest descent

```
1. Given $x_0$, set $k = 0$.
2. Compute $d_k = -\nabla f(x_k)$,
   where $\nabla f$ denotes the gradient of $f$.
3. If $d_k \leq \epsilon$, stop. Else, find the new point $x_{k+1} = x_k + \alpha d_k$,
   $\alpha$ is the size of the step in the direction of the travel;
   $x_k$ and $x_{k+1}$ are the variables values in the $k$ and $k+1$;
   $\epsilon$ is the stop criterion.
4. Increment $k = k + 1$, go to step 2.
```

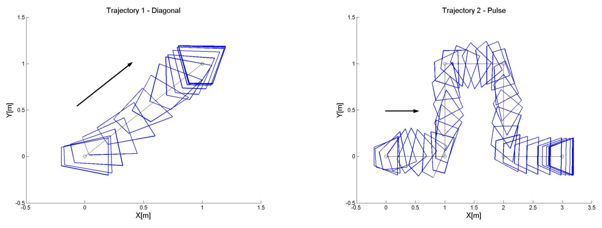
choice of β, where β^{FR} denotes the Flecher Reeves method and β^{PR} denotes the Polak Robiere method.

Algorithm 2 - Conjugate Gradient methods - Fletcher Reeves and Polak Robiere

```
1. Given $x_0$, compute $d_0 = -\nabla f(x_d)$, $k = 0$;
   $\nabla f(x)$ is the vector of gradients of the cost function at point $f(x)$;
2. Using $d_k$, compute $x_{k+1} = x_k + \alpha d_k$,
   where $\alpha$ is the step size that minimizes $f(x_k + \alpha d_k)$.
3. Compute $\nabla f(x_{k+1})$, if $\nabla f(x_{k+1}) \leq \epsilon$, stop. Else
       $$\beta_{k+1}^{FR} = \frac{\nabla f(x_{k+1})^T \nabla f(x_{k+1})}{\nabla f(x_k)^T \nabla f(x_k)}$$
   or
       $$\beta_{k+1}^{PR} = \frac{\nabla f(x_{k+1})^T (\nabla f(x_{k+1}) - \nabla f(x_k))}{\nabla f(x_k)^T \nabla f(x_k)},$$
   $x_k$ and $x_{k+1}$ are the variables values in the $k$ and $k+1$;
   $\epsilon$ is the stop criterion.
4. Compute new direction $d_{k+1} = -\nabla f(x_{k+1}) + \beta_{k+1} d_k$.
5. Increment $k = k + 1$, go to step 2.
```

IV. EXPERIMENTAL RESULTS

The proposed control strategy has been tested with the mobile robot following two trajectories, and it was repeated for the 3 algorithms of the optimization (G,GCFR and GCPR). The first trajectory, Fig. 4(a), is a movement of rotation and translation in the same time. The second trajectory has special features, as sudden change of direction and orientation to the robot, in order to test the controller in hard condition, see Fig. 4(b).



(a) "Diagonal" trajectory. (b) "Pulse" trajectory.

Fig. 4. Trajectories of test.

Firstly, we made simulations with the controller using the robot's model. The first objective of these simulations was to define a first calibration in the controller's parameters, as prediction and control horizons(N_1, N_2 and N_u), penalty factors (λ_1, $\lambda_2$and λ_3) and stop criterion of the optimization algorithms(ϵ). Another objective was to verify the computational time to calculate the minimization of the cost function. Informations about the computational time are extremely important to define the controller's parameters. The robot uses a sample time equal to 40 milliseconds, then it is necessary to project the controller to respect this restriction of time. In the experimental tests with the robot was necessary to insert a restriction in the time used by the optimizator of the predictive controller. It was defined as 20 the maximum number of iterations of the algorithms. That means that the minimization of the cost function will be truncated in the iteration 20, in case it has not found a minimum.

A. Experimental test 1

For the first experimental test with the robot and after the simulations results, the controller parameters have been chosen as follows in table I. In this test, we used a prediction horizon of 10 samples ($N = N_2 - N_1 = 10$), because it was verified in the simulation results that the optimization algorithms get a good minimization of the cost function in these conditions. The table II and the figures 5 and 6, show the results of the trajectory's following, for the three methods of optimization. For the two trajectories, the method of the conjugated gradients GC_PR is the fastest in the minimization of the cost function, as in the simulated tests, see the table II. This method (GC_PR) presents robot position errors slightly larger than the other methods, but this difference in the errors is not significant. It was verified that the time for an iteration of the cost function minimization is approximately 1 millisecond for all methods, due to the algorithms almost have the same complexity for implementation. Usually the algorithms possess a larger contribution for the solution in the first iterations and a small contribution in you finish, as shown in the last iterations in the Fig. 5(d) and 6(d), therefore the truncation in the algorithms did not harm in a preoccupying way the trajectory's following. Table III shows the amount of truncations and the number of samples along the robot navigation.

N_1	N_2	N_u	λ_1	λ_2	λ_3	ϵ	$\Delta S[m]$
0	10	1	2	1	1	$1e-3$	0.04

TABLE I

PARAMETERS OF THE CONTROLLER, TEST 1.

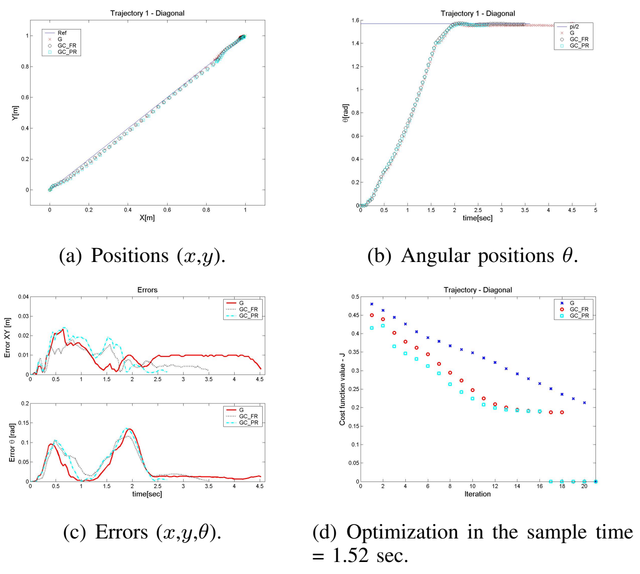
Diagonal trajectory				
Method	Iteration numbers			Optimization time
	Maximum	Minimum	mean	mean[milisec]
G	20	4	9.13	7.42
GC_FR	20	4	7.98	6.65
GC_PR	16	4	7.31	6.21
Pulso trajectory				
Method	Iteration numbers			Optimization time
	Maximum	Minimum	mean	mean[milisec]
G	20	4	13.15	11.19
GC_FR	20	4	10.21	8.57
GC_PR	20	4	8.75	7.25

TABLE II

DIAGONAL AND PULSE TRAJECTORY - TEST 1.

Diagonal trajectory		
Method	Number of truncation	Total of samples
G	4	115
GC_FR	1	89
GC_PR	0	69
Pulse trajectory		
Method	Number of truncation	Total of samples
G	48	213
GC_FR	10	196
GC_PR	1	184

TABLE III
TRUNCATION - TEST 1.



(a) Positions (x,y). (b) Angular positions θ.

(c) Errors (x,y,θ). (d) Optimization in the sample time = 1.52 sec.

Fig. 5. Diagonal trajectory - test 1.

B. Experimental test 2

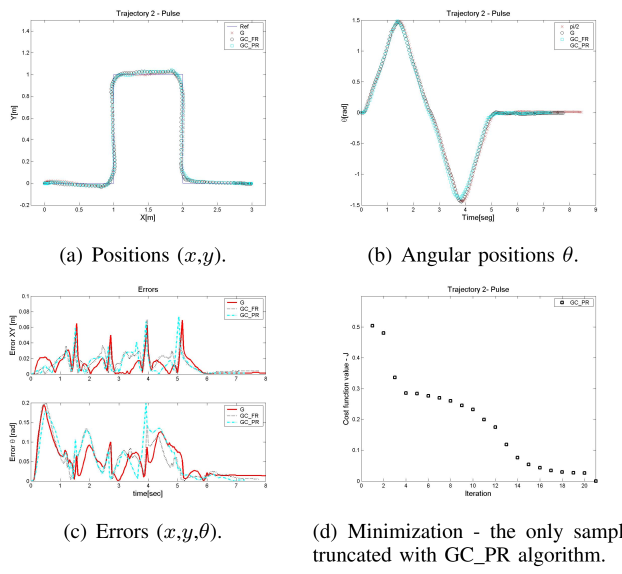
In the experimental test 2 the controller parameters in the table IV were used. In this test a larger horizon of the

N_1	N_2	N_u	λ_1	λ_2	λ_3	ϵ	$\Delta S[m]$
0	12	1	2	1	1	$1e-3$	0.04

TABLE IV
PARAMETERS OF THE CONTROLLER, TEST 2.

prediction was used ($N = 12$), to verify the performance of the optimizator and the behavior of the robot in the following of the trajectories. Fig. 7 and Fig. 8 show the results of the trajectory's following, for the three methods of optimization. Fig. 7(c) and Fig. 8(c) show the errors of the position (x, y) and posture (θ) of the robot. The linear(v,vn) and angular(w) velocities of the robot are shown in the figures 7(d) and 8(d). The table V shows information about the number of iteration of the optimizator. The method GC_PR was the fastest again, and any truncation did not happen, see the table VI.

In the experimental test 2 the number of iterations reduced comparing with the test 1. With the increase of the prediction



(a) Positions (x,y). (b) Angular positions θ.

(c) Errors (x,y,θ). (d) Minimization - the only sample truncated with GC_PR algorithm.

Fig. 6. Pulse trajectory - test 1.

horizon ($N = 12$), it is probable that the errors in the trajectory following have an increase, mainly in trajectories with abrupt changes of direction, as the pulse trajectory. The behavior of the robot tends to predict the control actions, avoiding the points of abrupt changes, for example the position $x = 1$ and $y = 0$ of the pulse trajectory, see Fig. 8(a).

Another important characteristic is that the quantity of iterations of the optimization algorithms increases with the decrease of the prediction horizon, consequently it increases the time of calculation of the signs of control of the robot in each sampling. A good prediction horizon range is larger than 8 and smaller than 12 ($8 \leq N \leq 12$), it is a good commitment between the number of iterations of the optimization algorithms and the behavior of the mobile robot.

In both experimental tests, the conjugated gradient methods are faster than the steepest descent method.

Diagonal trajectory				
Method	Iteration numbers			Optimization time
	Maximum	Minimum	mean	mean[mseg]
G	20	4	9.37	8.23
GC_FR	16	4	7.32	6.53
GC_PR	14	4	6.80	6.13
Pulse trajectory				
Method	Iteration numbers			Optimization time
	Maximum	Minimum	mean	mean[mseg]
G	20	4	12.01	10.76
GC_FR	20	4	8.45	7.63
GC_PR	14	4	6.93	6.27

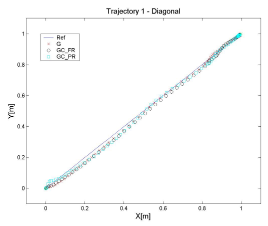
TABLE V
DIAGONAL AND PULSE TRAJECTORY - TEST 2.

Diagonal trajectory		
Method	Number of truncation	Total of samples
G	2	97
GC_FR	0	89
GC_PR	0	76

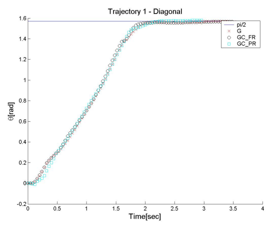
Pulse trajectory		
Method	Number of truncation	Total of samples
G	16	189
GC_FR	1	182
GC_PR	0	180

TABLE VI

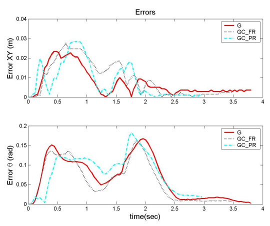
TRUNCATION - TEST 2.



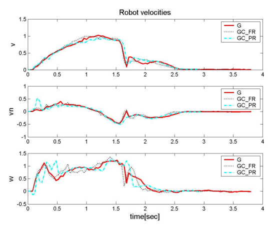
(a) Positions (x,y).



(b) Angular positions θ.



(c) Errors (x,y,θ).

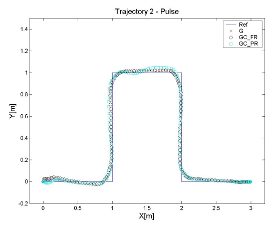


(d) Robot velocities.

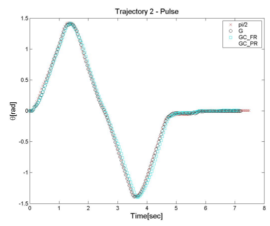
Fig. 7. Diagonal trajectory - test 2.

V. CONCLUSIONS

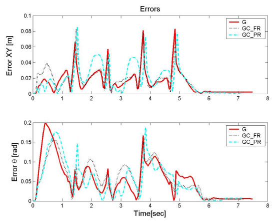
A nonlinear model based predictive controller has been proposed for a mobile robot path tracking problem. A new approach of predictive controller is presented, where methods of numeric optimization are used to obtain the minimization of the cost function of the predictive controller. The structure of the controller allows a vast capacity of calibration. The optimization algorithms, mainly the methods based on conjugate gradients, present good times of minimization of the cost function, allowing its use in the predictive controller. Some experimental results have shown the good performance of the strategy of the control proposed.



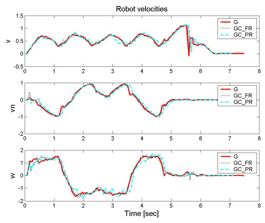
(a) Positions (x,y).



(b) Angular positions θ.



(c) Errors (x,y,θ).



(d) Robot velocities.

Fig. 8. Pulse trajectory - test 2.

REFERENCES

[1] D.R. Ramirez, D. Limon, J. Gomez-Ortega and E.F. Camacho, *Nonlinear MBPC for mobile robot navigation using genetic algorithms, International Conference on Robotics & Automation,* vol. 3, pp. 2452-2457, 1999.

[2] X. Jiang, Y. Motai and Xingquan Zhu, *Predictive fuzzy control for a mobile robot with nonholonomic constraints, 12th International Conference on Advanced Robotics, ICAR '05,* pp.58-63, 2005.

[3] D. Gu and H. Hu, *Neural predictive control for a car-like mobile robot, Robotics and Autonomous Systems Journal,* vol.39(2), pp.73-86, 2002.

[4] J. Gomez-Ortega and E.F. Camacho, *Neural Network MBPC for Mobile Robots Path Tracking, Robotics and Computer Integrated Manufacturing Journal,* vol. 11(4), pp.271-278, 1994.

[5] E.F. Camacho and C. Bordons, *Model Predictive Control,* Springer-Verlag, 2004.

[6] S. Piche, B. Sayyar-Rodsari, D. Johnson and M. Gerules, *Nonlinear model predictive control using neural networks, IEEE Control Systems Magazine,* vol. 20(3), pp.53-62, 2000.

[7] J.E. Normey-Rico, J. Gomez-Ortega, I. Alcalá-Torrego and E.F. Camacho, *Low time-consuming implementation of predictive path-tracking control for a "Synchro-drive" mobile robot, 5th International Workshop on Advanced Motion Control,* pp.350-355, 1998.

[8] Benjamin C. Kuo, *Automatic Control Systems,* John Wiley & Sons, Inc, 1995.

[9] R. Fletcher, *Practical methods of optimization,* John Wiley & Sons, 1987.

[10] R. Fletcher and C.M. Reeves, *Function minimization by conjugate gradients, The Computer Journal,*vol 7(2), pp.149-154, 1964.

[11] L.M. Grañ̃a Drummond and B.F. Svaiter, *A steepest descent method for vector optimization, Journal of Computational and Applied Mathematics,*vol 175(2), pp.395-414, 2005.

[12] W.H. William and Z. Hongchao, *A New Conjugate Gradient Method with Guaranteed Descent and an Efficient Line Search, SIAM J. on Optimization,* vol.16(1), pp. 170-192, 2005.

[13] W.H. William and Z. Hongchao, *Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent, ACM Transactions on Mathematical Software,* vol.32(1), pp. 113-137, 2006.

[14] J. Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain, Technical report, School of Computer Science, Carnegie Mellon University,* 1994.