J Intell Robot Syst
DOI 10.1007/s10846-017-0500-2

Comparative Application of Model Predictive Control Strategies to a Wheeled Mobile Robot

Hoai Nam Huynh · Olivier Verlinden · Alain Vande Wouwer

Received: 19 April 2016 / Accepted: 24 January 2017

Abstract Model predictive control strategies refer to a set of methods relying on a process model to determine an optimal control signal by minimising a cost function. This paper reports on the application of predictive control strategies to a wheeled mobile robot. As a first step, friction forces originating from the motor gearboxes and wheels were estimated and a feedforward compensation was applied. Step response tests were then carried out to identify a linear model to design several simple control strategies, such as the Proportional-Integral-Derivative (PID) controller. The PID response constitutes the reference to assess the efficiency of two predictive control strategies: the generalised predictive control (GPC) and the linear quadratic model predictive control (LQMPC) algorithms. These control strategies were tested in simulation with Matlab and EasyDyn (a C++ library for multibody system simulations) and in real life experiments. All three control strategies offer satisfactory reference tracking but MPC allows a reduction of the energy consumption of up to 70 % as a result of set-point anticipation. LQMPC is the best in terms of input activity reduction.

Keywords GPC · LQMPC · Model predictive control · Movement control · PID · Wheeled mobile robot

H. N. Huynh (✉) · O. Verlinden · A. Vande Wouwer
Faculty of Engineering of Mons (UMONS), Boulevard Dolez 31, 7000, Mons, Belgium
e-mail: HoaiNam.HUYNH@umons.ac.be

O. Verlinden
e-mail: Olivier.VERLINDEN@umons.ac.be

A. Vande Wouwer
e-mail: Alain.VANDEWOUWER@umons.ac.be

1 Introduction

The control of mobile robots has been the centre of interest of a lot of research in recent decades. The kinematic model of a wheeled mobile robot (WMR) is rather simple although the existence of non-holonomic constraints and friction make accurate control very challenging. Nevertheless, in practice, constraints on inputs arise and reduce the performance. These issues can be overcome thanks to model predictive control (MPC) through which the control actions that respect actuator limits can be achieved by considering input constraints in the predictive control strategy [1]. This is why model predictive control is often dedicated to vehicle guidance and path-tracking using fixed trajectories [2].

Starting in the seventies, predictive control methods provide a number of possibilities. We find them, especially in the control of machine tools, in the field of robotics and more generally in applications where the trajectories are predetermined [3]. For constrained, complex, multivariable control problems,

Published online: 23 February 2017

model predictive control has become a standard control alternative in the process industries [4]. It is used when the process to be controlled is slow enough to allow its implementation [5]. One major advantage of predictive control is that it takes constraints into account in the strategy development, from the design to the implementation of the control law. In contrast, conventional control methods treat constraints as an after-thought; this is the case of the PID controller using an anti-windup loop. Another thing to consider with model predictive control is that it can use information about future changes concerning the target to anticipate control actions, allowing a considerable reduction in the input required by the actuators.

MPC employs a model of the system to determine the best input sequence that minimises a cost function J. The feedback is introduced by continually updating the predicted behaviour of the system over a prediction horizon. The concept of the receding horizon takes into account the most recent target and measurement data over an input horizon. Model predictive control thus achieves an optimisation problem at each sample period by computing a new control input vector and, at the same time, takes process constraints into account [6]. The optimisation can be turned into a quadratic programming (QP) problem which can be solved by robust solvers leading to global optimal solutions. The philosophy of MPC is similar to human behaviour whereby one selects control actions that will drive the best results over a finite horizon. Then, whenever new observations become available, the decisions are updated to reach the target in the best way [7]. It is an intuitive and a logical concept which is transposed into equations to predict the output over a finite horizon. These equations are summarised in a set of methods called 'MPC methods' which are somewhat similar to the interpretation of the key concepts.

The contribution of this article is focused on the implementation of two model predictive control strategies for the precise and fast positioning of a wheeled mobile robot. The obtained results were then compared to those collected from a PID controller, whose parameters were tuned according to the transfer function of the robot. The first tested predictive method was the generalised predictive control (GPC) algorithm which was introduced by David W. Clarke and his co-workers in the 1980s [8]. Its implementation requires transposing the transfer function of the system into its CARIMA form, thus providing unbiased predictions [9]. The GPC algorithm is an incremental method based on the optimisation of input increments to handle the constraints and the feedforward. The GPC algorithm extends over limited prediction and input horizons whereas the second tested predictive method foresees input over infinite horizons. This second strategy, known as the linear quadratic model predictive control (LQMPC) or the optimal model predictive control (OMPC) algorithm, draws together optimal control given by a linear quadratic regulator (LQR) and some extra degrees of freedom to handle the constraints and the feedforward according to J. A. Rossiter [10]. The LQMPC algorithm was first proposed by Scokaert and Rawlings in 1996 [11]. In order to implement these three controllers, two models of the real system were determined: the first one corresponds to a first order transfer function, identified by trials of step response on the real system, and the second one is a multibody model. A `C++` library, called `EasyDyn`, developed for the simulation of problems represented by second-order (or first-order) differential equations and, more particularly, for multibody systems [12], was used to implement the multibody model. Moreover, a compensation of the friction was taken into account for these models and the real system by adding a constant extra voltage to the two DC motors with a symbol (+/-) chosen with respect to the rotation direction of the wheels.

Although many studies of MPC have been reported in the literature, there is still a need to evaluate and compare these methods in the framework of real-life robotic applications. Future prospects include trajectory tracking of wheeled robots performing dedicated tasks, and medical applications, especially rehabilitation by anticipating patient motion.

2 The Wheeled Mobile Robot

The WMR was designed by the Theoretical Mechanics, Dynamics and Vibration Unit of the Faculty of Engineering of the University of Mons (Belgium). It is made up of two driving wheels, each is coupled to a DC motor (EMG30) powered by a 12 V battery (NP2-12 Yuasa) held under an aluminium frame (Fig. 1). The third support consists of a ball bearing roller at the front of the robot. The control strategies are programmed on a ChipKIT UNO32 board equipped with a PIC32 microcontroller (80

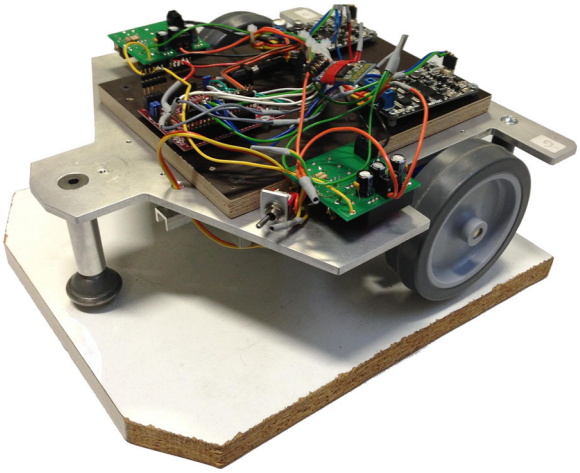

Fig. 1 wheeled mobile robot

MHz 32-bit MIPS, 128K Flash, 16K SRAM). Despite the fact that each motor has embedded rotary encoders, two separate odometers (ISC3004) are used to retrieve the angular positions of each driving wheel. They are mounted on spring brackets to avoid the sliding of the odometer wheels with the ground. Their maximum resolution reaches 1440 pulses per revolution (0.082 mm). The interface between the microcontroller and the motors is provided by two power boards (MD10C Cytron). A current sensor (current transducer LA 25-NP) is connected in series with each motor to evaluate the corresponding friction torque. Finally, a Bluetooth antenna is wired to the ChipKIT in order to transmit data to a computer.

The block diagram of the WMR, which is in fact a double input/double output system is shown in Fig. 2. The two controllers and the two position set points are loaded in the embedded microcontroller. The inputs of each motor are computed by the microcontroller and delivered through the power boards. The control loop is closed by reading the current position of the WMR wheels from the odometers. All position

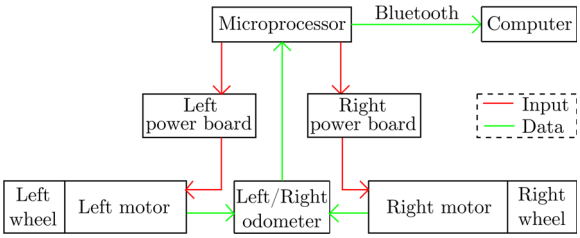

Fig. 2 block diagram of the WMR

data are transferred to a computer at the end of the manoeuvre.

Figure 3 shows the typical steps achieved leading to either control simulations or real life experiments. The position set point is computed for each motor through a C++ routine by using basic mechanical motion formulae and then gathered in files read under Matlab [13]. When cornering, the wheel velocities differentiate by tracking two different set points. Matlab is used to first compute the control parameters or matrices derived from a transfer function model of the WMR, and then performs a simulation of the tested strategy. EasyDyn and the real system are also fed by the control variables computed by Matlab to achieve their own simulation or experiment, taking into account the characteristics of their own model.

3 Modelling

3.1 Multibody Model

The multibody model was built using the C++ library EasyDyn [12, 14] in order to perform numerical simulations before the implementation of the methods on the real system. The model is composed of 8 bodies: the frame, the two motors, the two wheels, the two odometers and the steel ball at the front of the robot. The mass and moments of inertia of the 8 bodies obtained through a Solidworks model are summarised in Table 1.

Since the WMR is a non-holonomic system, it requires the introduction of extra degrees of freedom: the WMR possesses 8 degrees of freedom among which six of them correspond to the free movement of the frame in space and the last two result from the rotation of the wheels (θ_l, θ_r). The contact between

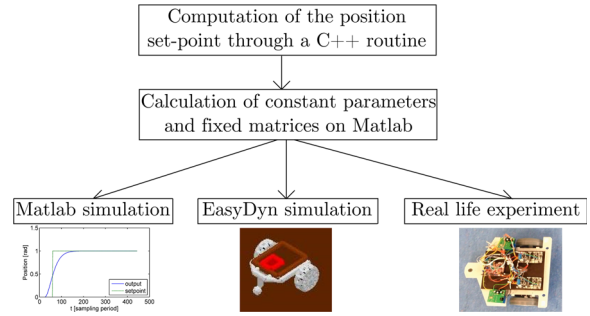

Fig. 3 organization of the control process

Table 1 mass and moments of inertia of the WMR

Body	Mass [kg]	I_{xx} [kg·m^2]	I_{yy} [kg·m^2]	I_{zz} [kg·m^2]
Frame	2.3	4.835e-3	5.45e-3	9.226e-3
Motor	0.37	1.92e-4	1.92e-4	4e-5
Odometer	0.0042	3.66e-7	3.66e-7	7.21e-7
Wheel	0.136	2.1e-4	1.1e-4	1.1e-4
Steel ball	0.00419	1.67e-7	1.67e-7	1.67e-7

the wheels and the ground was modelled with a "tyre type element" to get closer to the slip-free rolling conditions. The tyre model comes from the University of Arizona [15]. The main parameters of the tyre are gathered in Table 2.

Besides the forces of gravity and those generated by the "tyre type element", the model is completed by the definition of the external forces applied to the system, and, more particularly, by the actuators. The equations of a DC motor are expressed as follows:

$$U_{l,r} = L \cdot \frac{di}{dt} + R \cdot i + K_b \cdot \dot{\theta}_{l,r} \tag{1}$$

$$C_m = K_m \cdot i \tag{2}$$

with

- U: nominal voltage;
- L: terminal inductance;
- R: terminal resistance;
- K_m, K_b: torque and speed constants of the motor;
- C_m: motor torque.

The electrical parameters and the constants of the motor were determined experimentally and are related to the motor from the point of view of the wheel

Table 2 tyre parameters

Parameter	Value
Outer radius	0.05 m
Equivalent torus radius	0.0375 m
Vertical stiffness	100000 $\frac{N}{m}$
Vertical damping	30
Nominal vertical force	10 N
Nominal longitudinal stiffness	80 $\frac{N}{m}$
Nominal cornering stiffness	80 $\frac{N}{m}$
Nominal camber stiffness	8 $\frac{N}{m}$
Coulomb friction coef.: static	1
Coulomb friction coef.: dynamic	0.8

Parameter	Value
Terminal resistance	4.5 Ω
Terminal inductance	0.004115 H
Torque/speed constant	0.5335 Nm/A or V/rad/s

Fig. 4 motor parameters

(Table 4). Due to its small value, the inductance term was neglected. Note that the torque of the motor C_m decreases with the wheel speed through a damping coefficient B (Eq. 3). In this case, B=0.005 $\frac{Nms}{rad}$ to match the experimental step response.

$$C_m = K_m \cdot i - B \cdot \dot{\theta}_{l,r} \tag{3}$$

3.2 Black-box Model

The system is represented by a first order transfer function expressed in terms of the Laplace variable s linking the angular velocity of the wheels $\Omega(s)$ to the applied voltage $U(s)$:

$$\frac{\Omega(s)}{U(s)} = \frac{K}{\tau s + 1} \tag{4}$$

The gain K and the mechanical time constant τ can be estimated from step responses of 1 V, between 0 V and 12 V, for each motor (Fig. 5). The experiments were carried out using a compensation of the friction: a feedforward input voltage of 0.66 V was applied to each motor in order to balance the friction of the gearbox and that coming from the contact between the ground and the wheels. The value of this extra voltage was found by measuring the current which is the representation of the friction torque when the wheels angular velocity was constant. The results were averaged for speeds ranging from 0 to 18 rad/s. Although the stiction was measured, it was not taken into account as a continuous drive was assumed. The WMR thus moved immediately as soon as an

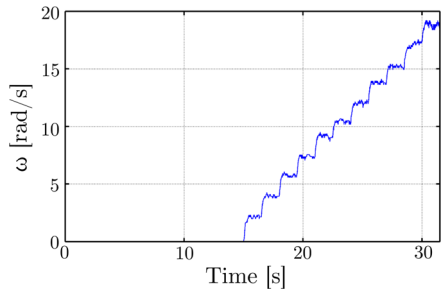

Fig. 5 successive steps (50 Hz)

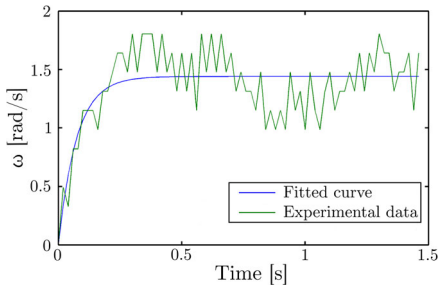

Fig. 6 fitting of the transfer function (step between 10 V-11 V)

additional voltage was applied from its rest position. Figure 6 provides the model identification by fitting a first order transfer function to an experimental step between 10 V and 11 V.

An average model (obtained using the information of 5 step responses for each motor) leads to the following transfer function between the input voltage [V] and the output robot position [rad] (Eq. 5). The same transfer function is used for both motors (left and right).

$$G(s) = \frac{\theta_{l,r}(s)}{U(s)} = \frac{K}{s(\tau s + 1)} = \frac{1.715}{s(0.09s + 1)} \tag{5}$$

4 PID Controller

First, a discrete-time PID controller was designed, whose results will be used to benchmark the MPC strategies. The block diagram of a parallel PID controller is depicted in Fig. 7.

The discrete input u_k was computed by summing 3 contributions involving the discrete error ϵ_k related to the difference between the set point and the discrete output y_k:

$$u_k = p_k + i_k + d_k \tag{6}$$

with

- the proportional term: $p_k = K_p \cdot \epsilon_k$
  - K_p: proportional gain.
- the integral term: $i_k = i_{k-1} + \frac{K_p}{\tau_i} \cdot T_s \cdot \epsilon_k$
  - τ_i: integral time constant;
  - T_s: sampling period.
- the derivative term: $d_k = \frac{\tau_d}{\tau_d + N \cdot T_s} \cdot d_{k-1} + \frac{K_p \cdot N \cdot \tau_d}{\tau_d + N \cdot T_s} \cdot (\epsilon_k - \epsilon_{k-1})$
  - τ_d: derivative time constant;
  - $\frac{1}{N}\tau_d$: pole reducing the high-frequency noise.

The selection of the PID parameters was achieved by loop shaping as shown in Fig. 8. Since the PID and the position transfer function both have an embedded integrator, there is no steady state error. The retained closed-loop system was highly damped by moving the frequency response curve in the Black-Nichols chart to the right. The correct loop shaping was achieved graphically by imposing a phase margin of 80 degrees which prevents any overshoot. On the contrary, pushing the frequency response curve to the left in the Black-Nichols chart would result in some overshoot before returning to the set point. Despite a shorter rise time, this tuning was not suitable for this application as the WMR would move forwards and backwards to reach the set point. The chosen tuning yields a slower response with minimum overshoot for direct positioning. The parameters of the PID were computed through an in-house routine named LIASCA [16] and are presented in Table 3.

An anti-reset wind-up was included to tackle the saturation of the motors. Their input was limited to 12 V. Therefore, the integral term of the discrete-time parallel PID was modified with a tracking time constant

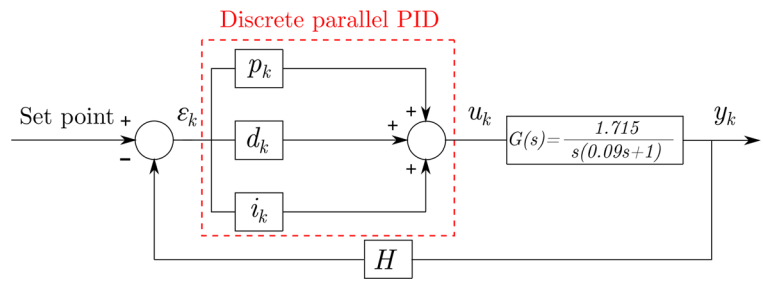

Fig. 7 discrete parallel PID controller

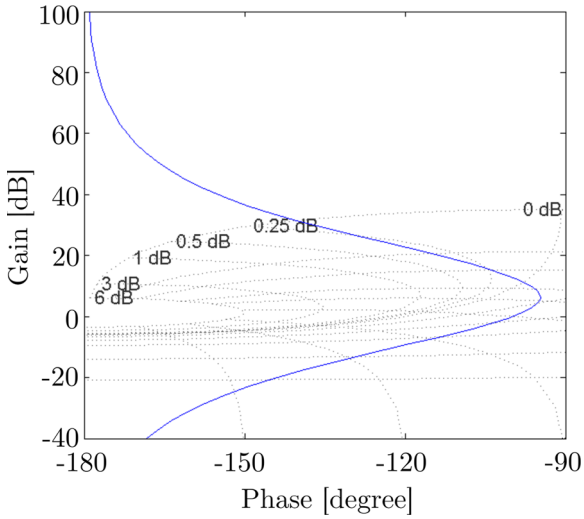

Fig. 8 loop shaping in the Black-Nichols chart

T_t (Eq. 7). The absolute value of the difference between the unconstrained input and the saturation is given by Δu_{sat}.

$$T_t = \sqrt{\tau_i \cdot \tau_d} \qquad i_k = i_{k-1} + \left(\frac{K_p}{\tau_i} \cdot \epsilon_k - \frac{1}{T_t} \cdot \Delta u_{sat}\right) \cdot T_s \tag{7}$$

A step response of 1 radian for the WMR wheels was performed by simulations under Matlab and EasyDyn, and in real life experiments. The responses were slower because of the saturation but there was no overshoot. Moreover, the three responses were almost identical attesting to a good modelling of the WMR. For that simulation, the voltage input reached the motor saturation of 12 V during the transient phase. The computation time for one PID iteration

Table 3 discrete-time parallel PID parameters

PID parameters	
K_p	10.7910 $\frac{V}{rad}$
τ_i	1.2416 s
τ_d	0.0932 s
N	8.1766

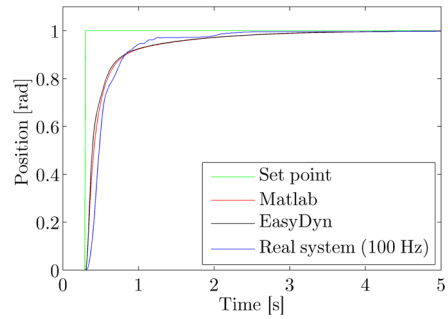

Fig. 9 PID: step-responses of the closed-loop system using a PID anti-reset wind-up (simulations and real life experiments)

was approximately 0.001 s on the embedded microcontroller.

5 GPC Algorithm

5.1 Theoretical Background

The first predictive method tested was the Generalised Predictive Control (GPC) algorithm proposed by Clarke and Tuffs in 1987 [8]. In order to implement the GPC algorithm, the transfer function (Eq. 5) was first transformed into a transfer function model, called a "CARIMA" model (Eq. 8), which includes a disturbance estimate and therefore gives unbiased predictions.

$$a(z^{-1})y_k = b(z^{-1})u_k + T(z^{-1})\frac{\zeta_k}{\Delta} \tag{8}$$

In practice, as the output is measured often, the one-step ahead prediction model used variables of the output and input increments Δu_k and assumed that the best estimate of the future random term ζ_k was zero because it is a zero mean random variable (Eqs. 9 and 10). $T(z^{-1})$ can be treated as a design parameter and could arise from a model identification [8].

$$[a(z^{-1})\Delta]y_k = [b(z^{-1})]\Delta u_k + \cancel{T(z^{-1})\zeta_k} \tag{9}$$

$$A(z^{-1})y_k = b(z^{-1})\Delta u_k \tag{10}$$

with

- $\Delta = 1 - z^{-1}$;
- $A(z^{-1}) = 1 + A_1 z^{-1} + \cdots + A_n z^{-n}$;
- $b(z^{-1}) = b_1 z^{-1} + \cdots + b_m z^{-m}$.

Using the WMR transfer function, the numerical expression of the "CARIMA" model is given by Eq. 11:

$$\begin{aligned}&[1-(\mathrm{e}^{\frac{-T_s}{0.09}}+1)z^{-1}+\mathrm{e}^{\frac{-T_s}{0.09}}z^{-2}]y_k\\&=[(1.715T_s+1.715\cdot 0.09\mathrm{e}^{\frac{-T_s}{0.09}}-1.715\cdot 0.09)z^{-1}\\&\quad+(-1.715T_s\mathrm{e}^{\frac{-T_s}{0.09}}+1.715\cdot 0.09-1.715\\&\quad\cdot 0.09\mathrm{e}^{\frac{-T_s}{0.09}})z^{-2}]\Delta u_k\end{aligned}\tag{11}$$

The output prediction over the prediction horizon n_y is given by Eq. 12 (the control horizons have to be multiplied by the sampling period to convert them into seconds). The prediction matrices are denoted by H, P, Q (according to the notations of J.A. Rossiter [7]).

$$\underset{\rightarrow}{\mathbf{y}}_{k+1}=\underbrace{\mathbf{H}\underset{\rightarrow}{\Delta\mathbf{u}}_k}_{\text{Future input increments}}+\underbrace{\mathbf{P}\underset{\leftarrow}{\Delta\mathbf{u}}_{k-1}+\mathbf{Q}\,\underset{\leftarrow}{\mathbf{y}}_k}_{\text{Past data}}\tag{12}$$

with

- $\mathbf{H}=\mathbf{C}_A^{-1}\mathbf{C}_b$;
- $\mathbf{P}=\mathbf{C}_A^{-1}\mathbf{H}_b$;
- $\mathbf{Q}=-\mathbf{C}_A^{-1}\mathbf{H}_A$.

The arrow vector subscript is used to denote future or past information. For example, $\underset{\rightarrow}{\mathbf{y}}_{k+1}$ represents a set of predictions starting from $k+1$ to $k+n_y$ computed at sample time k (Eq. 13).

$$\underset{\rightarrow}{\mathbf{y}}_{k+1}=\begin{bmatrix}y_{k+1|k}\\y_{k+2|k}\\\vdots\\y_{k+n_y|k}\end{bmatrix}\tag{13}$$

The terms which compose the matrices of prediction are developed in Eqs. 14 to 17 by assuming a prediction in 4 steps.

$$\mathbf{C}_A=\begin{bmatrix}1&0&0&0\\A_1&1&0&0\\A_2&A_1&1&0\\A_3&A_2&A_1&1\end{bmatrix}\tag{14}$$

$$\mathbf{C}_b=\begin{bmatrix}b_1&0&0&0\\b_2&b_1&0&0\\b_3&b_2&b_1&0\\b_4&b_3&b_2&b_1\end{bmatrix}\tag{15}$$

$$\mathbf{H}_A=\begin{bmatrix}A_1&A_2\cdots A_{n-4}&A_{n-3}\cdots A_{n-1}&A_n\\A_2&A_3\cdots A_{n-3}&A_{n-2}\cdots A_n&0\\A_3&A_4\cdots A_{n-2}&A_{n-1}\cdots 0&0\\A_4&A_5\cdots A_{n-1}&A_n\cdots 0&0\end{bmatrix}\tag{16}$$

$$\mathbf{H}_b=\begin{bmatrix}b_2&b_3\cdots b_{m-4}&b_{m-3}\cdots b_{m-1}&b_m\\b_3&b_4\cdots b_{m-3}&b_{m-2}\cdots b_m&0\\b_4&b_5\cdots b_{m-2}&b_{m-1}\cdots 0&0\\b_5&b_6\cdots b_{m-1}&b_m\cdots 0&0\end{bmatrix}\tag{17}$$

The GPC control law was then established in order to determine the optimal future input increment sequence $\underset{\rightarrow}{\Delta\mathbf{u}}_k$ over a control horizon n_u (Eq. 18). Beyond the n_u steps, the input increments are assumed to be equal to zero.

$$\underset{\rightarrow}{\Delta\mathbf{u}}_k=\begin{bmatrix}\Delta u_k\\\vdots\\\Delta u_{k+n_u-1}\\0\\\vdots\\0\end{bmatrix}\tag{18}$$

The performance index is defined by:

$$J=\sum_{i=1}^{n_y}e_{k+i}^2+\lambda\sum_{i=1}^{n_u}(\Delta u_{k+i-1})^2\tag{19}$$

and included the:

- sum of the squared prediction errors between the set point and the output $\underset{\rightarrow}{\mathbf{e}}_{k+1}=\underset{\rightarrow}{\mathbf{r}}_{k+1}-\underset{\rightarrow}{\mathbf{y}}_{k+1}$;
- sum of the squared input increments over the control horizon n_u.

The performance index was minimised with respect to the input increments $\underset{\rightarrow}{\Delta\mathbf{u}}_k$ by substituting the prediction Eq. (12) leading to the GPC control law (Eq. 20).

$$\underset{\rightarrow}{\Delta\mathbf{u}}_k=\mathbf{P}_{rlong}\,\underset{\rightarrow}{\mathbf{r}}_{k+1}-\mathbf{D}_k\underset{\leftarrow}{\Delta\mathbf{u}}_{k-1}-\mathbf{N}_k\,\underset{\leftarrow}{\mathbf{y}}_k\tag{20}$$

with

- $\mathbf{P}_{rlong}=\mathbf{E}_1^T(\mathbf{H}^T\mathbf{H}+\lambda)^{-1}\mathbf{H}^T$;
- $\mathbf{D}_k=\mathbf{P}_{rlong}\cdot\mathbf{P}$;
- $\mathbf{N}_k=\mathbf{P}_{rlong}\cdot\mathbf{Q}$.

The first value of $\underrightarrow{\Delta \mathbf{u}}_k$ was implemented and applied to the motors by extracting the first value of its vector thanks to matrix $\mathbf{E}_1^T$.

$$\underrightarrow{\Delta \mathbf{u}}_k = \begin{bmatrix} \Delta u_k \\ \vdots \\ \Delta u_{k+n_u-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow \underbrace{[I, 0, \cdots, 0]}_{E_1^T} \underrightarrow{\Delta \mathbf{u}}_k = \Delta u_k \tag{21}$$

The feedforward horizon n_a considers advance knowledge of the reference target $\underrightarrow{\mathbf{r}}_{k+1}$.

5.2 Tuning of the GPC Algorithm [7]

The GPC algorithm included four parameters chosen thanks to tests at a sample frequency of 100 Hz:

– The prediction horizon n_y was set to 28 (0.28 s) so that the open-loop prediction matched the closed-loop behaviour. In Fig. 10, the open-loop prediction is the prediction at the instant k (blue curve) whereas the closed-loop behaviour is related to the update of that prediction over the prediction horizon (green curve). It is indeed desirable that the open-loop prediction matches the closed-loop behaviour so that the optimisation is well-posed. In the GPC algorithm, errors are only minimised within the prediction horizon. Consequently, if the prediction horizon is too short, the outcome turns out to be unstable. Figure 11 shows the impact of various prediction horizon n_y values. On the other hand, taking too many errors into account for the prediction horizon tends to slow down the dynamics of the response for this system, as the algorithm becomes more cautious. For this particular wheeled mobile robot, a prediction horizon of n_y=28 was selected as the open-loop prediction gets closer to the closed-loop behaviour.

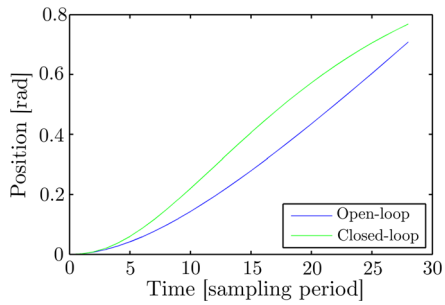

Fig. 10 GPC: prediction horizon n_y=28

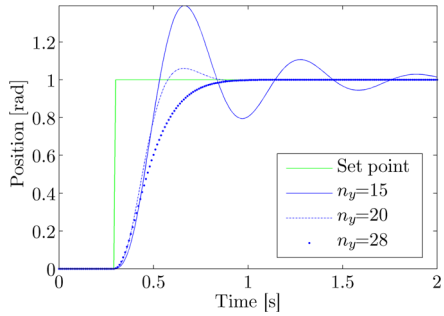

Fig. 11 GPC: selection of the prediction horizon n_y=28

– The feedforward horizon n_a was set to 18 (0.18 s) to anticipate the set point by balancing errors both sides of the set point change for the step response. With this setting, the step response is intended to respond before the imposed set point step. It is one of the main features of predictive control methods which anticipate the set point in order to better distribute the effort from the actuator throughout the whole manoeuvre.

From a mathematical point of view, it is possible to determine the best value of the feedforward horizon n_a by plotting the performance index J as a function of n_a. In practice, a value of n_a is selected before J settles to a minimum in order to avoid an unwanted over-anticipation of the set point (Fig. 12). Over-anticipation may lead to a less accurate prediction and, therefore, a steady state error for the positioning of the mobile robot as well as overshoot.

– The control horizon n_u was set to 1 (0.01 s) to avoid any overshoot of the set point. A larger

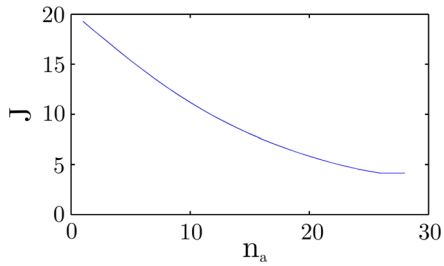

Fig. 12 GPC: selection of the feedforward horizon n_a=18

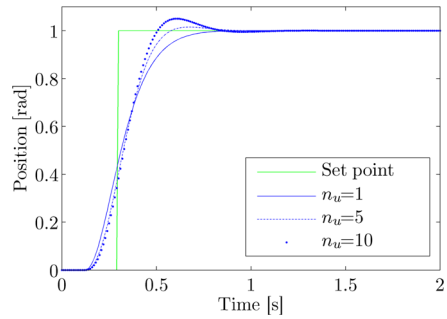

Fig. 13 GPC: selection of the input horizon n_u=1

value of n_u does indeed cause overshoot for a slightly faster output (Fig. 13). For general purpose applications, for which overshoot is not an issue, it is common to increase the value of the control horizon (n_u=4 or more) in order to speed up the dynamics of the system.

– The input weight λ was simply chosen as 1 because an increase of this parameter induces a slower output since the input gets softer for this system (Fig. 14). On the contrary, a reduction of this parameter would hasten the dynamics at the expense of more tracking errors.

5.3 Application of the GPC Algorithm

The unconstrained GPC law was tested by simulations under Matlab and `EasyDyn` as well as in real life experiments. The results of a step response of 1 radian for the WMR wheels with a sampling frequency of 100 Hz are presented in Fig. 15. All responses ensure an offset-free tracking error. Moreover, the motor saturation of 12 V was not reached with the GPC algorithm (a maximum input of 3.1 V was observed)

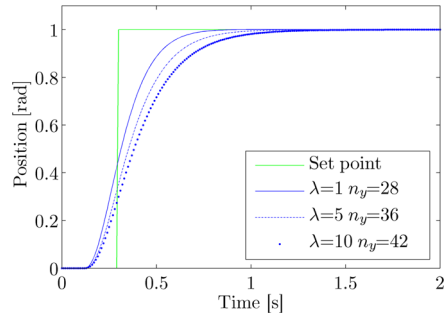

Fig. 14 GPC: selection of the input weight λ=1

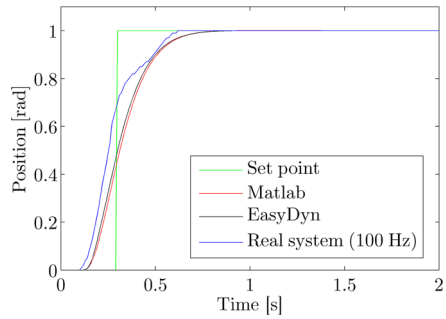

Fig. 15 GPC: unconstrained steps (output)

whereas the PID controller involved a saturation during its transient phase. It can also be seen that the GPC algorithm correctly anticipated the set point.

Since the performance index of the GPC algorithm is quadratic, a quadratic minimisation through quadratic programming (QP) was used to handle the input constraints (Eq. 22). The constraints were then expressed through Matlab under an inequality $\mathbf{M}x \leq \mathbf{d}$ for which matrices $\mathbf{M}$ and $\mathbf{d}$ represent constraints and are developed in [7]. In order to determine the optimal input increment Δu_k at instant k, the performance index J was minimised under the constraints represented by the inequality $\mathbf{M}x \leq \mathbf{d}$.

$$min \quad J = \underbrace{\Delta\underrightarrow{\mathbf{u}}_k^T}_{x^T} \underbrace{(\mathbf{H}^T\mathbf{H}+\lambda\mathbf{I})}_{\mathrm{S}} \underbrace{\Delta\underrightarrow{\mathbf{u}}_k}_{x} - \underbrace{\Delta\underrightarrow{\mathbf{u}}_k^T}_{x^T} \underbrace{2\mathbf{H}^T[\,\underrightarrow{\mathbf{r}}_{k+1} - \mathbf{P}\Delta\underleftarrow{\mathbf{u}}_{k-1} - \mathbf{Q}\,\underleftarrow{\mathbf{y}}_k\,]}_{\mathrm{a}} \quad (22)$$

$$under \quad \mathbf{M}x \leq \mathbf{d}$$

In the present study:

$$\mathbf{M} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (23)$$

$$\mathbf{d} = \begin{bmatrix} 12 \\ 12 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} u_{k-1} \quad (24)$$

Figure 16 shows that the consideration of the constraints considerably improved the behaviour of the system. The obtained overshoot using the constrained GPC law was due to the sliding of the WMR whereas the odometer wheels did not slip. The computation time for one GPC iteration taking constraints into account was approximately 0.0067 s on the embedded microcontroller.

A ramp set point was also tested and is shown in Fig. 17. A closer look shows that the GPC method

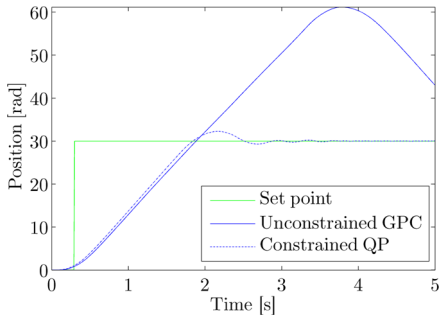

Fig. 16 GPC: Comparison between the constrained and the unconstrained GPC law for a step response of 30 radians

smoothly approached the set point without any overshoot.

6 LQMPC Algorithm

6.1 Theoretical Background

The dual-mode linear quadratic MPC first proposed by Scokaert and Rawlings in 1996 [11] and described by J. A. Rossiter [10] uses infinite control and prediction horizons. A discrete-time state-space model is required (Eqs. 25 and 26). For the sake of simplicity, all quantities are assumed to be scalar, except when the arrow notation arises.

$$x_{k+1} = Ax_k + Bu_k \tag{25}$$

$$y_{k+1} = Cx_{k+1} \tag{26}$$

The LQMPC algorithm employs the optimal control theory through a linear quadratic regulator (LQR) in conjunction with n_c degrees of freedom $\underset{\rightarrow}{\mathbf{c}}_k$ during the first mode called the transient mode (Eq. 27). Beyond the first n_c steps, the terminal mode is activated to get an asymptotic behaviour by assuming that the degrees of freedom are equal to zero (Eq. 28). The degrees of freedom are used to take into account the constraints and the feedforward during the transient phase. On the other hand, the optimal feedback given by the LQR is represented by matrix K. This implies that $[A - BK] = \Phi$ is the transition matrix of the system.

$$\{x_{k+1} = \Phi x_k + Bc_k; \quad u_k = -Kx_k + c_k\}$$
$$k \leq n_c \quad \text{(transient mode)} \tag{27}$$

$$\{x_{k+1} = \Phi x_k; \quad u_k = -Kx_k\} \quad k > n_c \quad \text{(terminal mode)} \tag{28}$$

It is assumed that the global optimum closed-loop trajectory can be determined with a performance index involving infinite horizons where Q and R are the state and input weights respectively (Eq. 29).

The performance index J is minimised with respect to the degrees of freedom $\underset{\rightarrow}{c}_k$ to handle the constraints and the feedforward.

$$\min_{\underset{\rightarrow}{c}_k} J = \sum_{i=1}^{\infty} x_{k+i}^T Q x_{k+i} + u_{k+i-1}^T R u_{k+i-1} \tag{29}$$

Then, an autonomous model is built to combine the two modes into one single prediction by forming an augmented model where the degrees of freedom are treated as extra states. The advance knowledge of the target is also included into the autonomous model with a feedforward horizon n_a. In order to do so, the expected *s*teady *s*tate values for the states and the output are expressed with the subscript: ss. An output disturbance d_k constant over the prediction can be added to the model. It should be estimated and updated at each sample time k by an observer (Eqs. 30 and 31).

$$x_{ss} = Ax_{ss} + Bu_{ss} \tag{30}$$

$$y_{ss} = Cx_{ss} + d_k \tag{31}$$

The steady state model of the prediction can be expressed by Eq. 32 where y_{ss} can be assimilated to the reference r_{k+1}.

$$\begin{bmatrix} x_{ss} \\ u_{ss} \end{bmatrix} = \begin{bmatrix} C & 0 \\ A - I & B \end{bmatrix}^{-1} \begin{bmatrix} r_{k+1} - d_k \\ 0 \end{bmatrix} \tag{32}$$

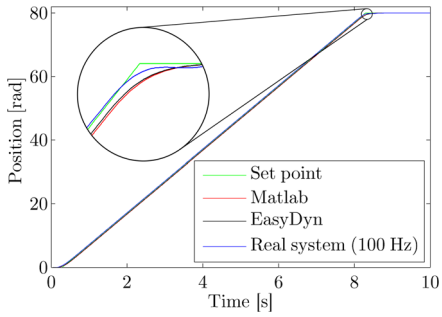

Fig. 17 GPC: ramp set point (output)

The previous equation is put in a compact form through Eq. 33.

$$\begin{bmatrix} x_{ss} \\ u_{ss} \end{bmatrix} = \begin{bmatrix} K_{xr} \\ K_{ur} \end{bmatrix} (r_{k+1} - d_k) \tag{33}$$

It is shown by J. A. Rossiter [10] that the dual-mode predictions can be expressed in terms of deviation variables $\hat{x}_{k+1} = (x_{k+1} - x_{ss})$ and $\hat{u}_k = (u_k - u_{ss})$ (Eqs. 34 and 35).

$$\{\hat{x}_{k+1} = \Phi\hat{x}_k + Bc_k; \quad \hat{u}_k = -K\hat{x}_k + c_k\} \quad k \leq n_c \quad \text{(transient mode)} \tag{34}$$

$$\{\hat{x}_{k+1} = \Phi\hat{x}_k; \quad \hat{u}_k = -K\hat{x}_k\} \quad k > n_c \quad \text{(terminal mode)} \tag{35}$$

Then, in combination with Eq. 33, the one-step ahead prediction and the control input with respect to the target changes for the transient mode (Eqs. 36 and 37) can be expressed as:

$$x_{k+1} = \Phi x_k + [I - \Phi]K_{xr}(r_{k+1} - d_k) + Bc_k \quad k \leq n_c \tag{36}$$

$$u_k = -Kx_k + [KK_{xr} + K_{ur}](r_{k+1} - d_k) + c_k \quad k \leq n_c \tag{37}$$

The feedforward horizon n_a is introduced by assuming that beyond n_a steps the reference remains unchanged (Eq. 38).

$$r_{k+n_a+i} = r_{k+n_a} \quad i \geq 0 \tag{38}$$

The variation of the target is captured with a shift matrix D_R (Eq. 39):

$$\underbrace{\begin{bmatrix} r_{k+2} \\ r_{k+3} \\ \vdots \\ r_{k+n_a+1} \end{bmatrix}}_{\underrightarrow{R}_{k+1}} = \underbrace{\begin{bmatrix} 0 & I & 0 & \cdots \\ 0 & 0 & I & \cdots \\ \vdots & \vdots & \ddots & \cdots \\ 0 & 0 & 0 & I \end{bmatrix}}_{D_R} \underbrace{\begin{bmatrix} r_{k+1} \\ r_{k+2} \\ \vdots \\ r_{k+n_a} \end{bmatrix}}_{\underrightarrow{R}_{k}} \tag{39}$$

The disturbance d_k can be added to each term of $\underrightarrow{R}_{k+1}$ with a column vector L filled with ones to give the right dimensions (Eq. 40).

$$\underrightarrow{R}_{k+1} - Ld_k = D_R(\underrightarrow{R}_{k} - Ld_k) \tag{40}$$

The autonomous model Z can finally be built by grouping the feedforward and the two modes into one single mode (Eqs. 41 and 42).

$$\underbrace{\begin{bmatrix} x_{k+1} \\ \underrightarrow{c}_{k+1} \\ \underrightarrow{R}_{k+1} - Ld \end{bmatrix}}_{Z_{k+1}} = \underbrace{\begin{bmatrix} \Phi & \begin{bmatrix} B & 0 & \cdots & 0 \end{bmatrix} & \begin{bmatrix} (I-\Phi)K_{xr} & 0 & 0 & \cdots & 0 \end{bmatrix} \\ 0 & \begin{bmatrix} 0 & I & 0 & \cdots \\ 0 & 0 & I & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \ddots & \end{bmatrix} & 0 \\ 0 & 0 & D_R \end{bmatrix}}_{\Psi} \times \underbrace{\begin{bmatrix} x_k \\ \underrightarrow{c}_{k} \\ \underrightarrow{R}_{k} - Ld_k \end{bmatrix}}_{Z_k} \tag{41}$$

$$u_k = \underbrace{\begin{bmatrix} -K & \begin{bmatrix} 1 & 0 & 0 & \cdots \end{bmatrix} & \begin{bmatrix} KK_{xr} + K_{ur} & 0 & \cdots & 0 \end{bmatrix} \end{bmatrix}}_{K_Z} \times \underbrace{\begin{bmatrix} x_k \\ \underrightarrow{c}_{k} \\ \underrightarrow{R}_{k} - Ld_k \end{bmatrix}}_{Z_k} \tag{42}$$

The expression of the cost function J in terms of the autonomous model Z can be found thanks to the Lyapunov identity (Eqs. 43 and 44). In the calculation,

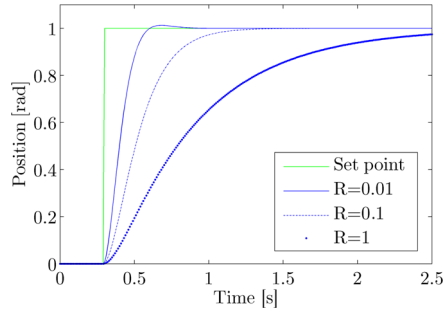

Fig. 18 LQMPC: impact of the input weight R (output)

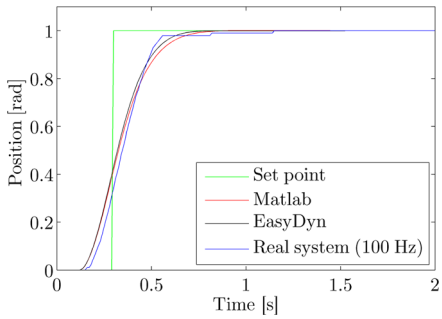

Fig. 19 LQMPC: unconstrained steps (output)

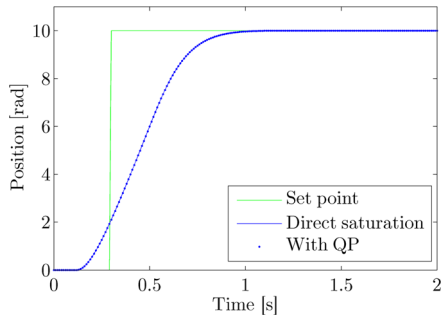

Fig. 21 LQMPC: constraints handling (output)

developed by J. A. Rossiter [10], S is the cost function matrix.

$$J = Z_k^T S Z_k = \begin{bmatrix} x_k \\ \underset{\rightarrow k}{c} \\ \underset{\rightarrow k}{R} - Ld_k \end{bmatrix}^T \underbrace{\begin{bmatrix} S_x & S_{xc} & S_{xr} \\ S_{xc}^T & S_c & S_{cr} \\ S_{xr}^T & S_{cr}^T & S_r \end{bmatrix}}_{\mathrm{S}} \begin{bmatrix} x_k \\ \underset{\rightarrow k}{c} \\ \underset{\rightarrow k}{R} - Ld_k \end{bmatrix} \tag{43}$$

$$\begin{aligned} J = {} & x_k^T S_x x_k + 2x_k^T S_{xc} \underset{\rightarrow k}{c} + \underset{\rightarrow k}{c}^T S_c \underset{\rightarrow k}{c} \\ & + (\underset{\rightarrow k}{R} - Ld_k)^T S_r (\underset{\rightarrow k}{R} - Ld_k) \\ & + 2x_k^T S_{xr} (\underset{\rightarrow k}{R} - Ld_k) + 2 \underset{\rightarrow k}{c}^T S_{cr} (\underset{\rightarrow k}{R} - Ld_k) \end{aligned} \tag{44}$$

The performance index can finally be minimised with respect to the degrees of freedom $\underset{\rightarrow k}{c}$ to generate the feedforward action during the transient mode. Equation (45) gives the optimal value of the degrees of freedom.

$$\frac{dJ}{d\,\underset{\rightarrow k}{c}} = 0 \Rightarrow \underset{\rightarrow k}{c} = -S_c^{-1} \left[S_{cr} (\underset{\rightarrow k}{R} - Ld_k) + S_{xc} x_k \right] \tag{45}$$

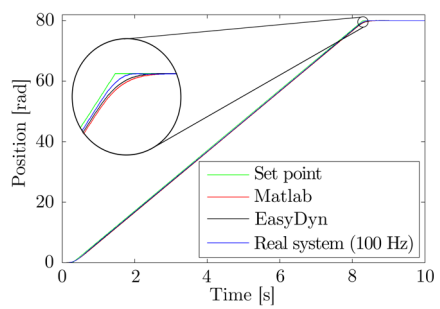

Fig. 20 LQMPC: unconstrained ramps (output)

They are used to define the optimal input at each sample time k taking the feedforward action into account (Eq. 46).

$$u_k = K_Z Z_k - \left[0 \; 0 \; \left[K_{ur} \; 0 \cdots 0 \right] \right] Z_k + u_{ss|k} \tag{46}$$

6.2 Tuning of the LQMPC Algorithm [7]

The chosen number of degrees of freedom was n_c=1 because the S_{xc} matrix tends to be zero. Therefore, for the unconstrained case, the optimal values of the degrees $\underset{\rightarrow k}{c}$ were constant in the receding horizon and only the first value was really implemented into the system.

In order to compare the GPC and the LQMPC algorithms, and in particular the influence of the actuator activity, the input weight R of the LQR can be adjusted to match the output step response of the GPC by choosing R=0.045 and therefore avoiding overshoot (Fig. 18); Q is typically chosen $Q = C^T C$. The feedforward horizon was chosen as n_a=18 (0.18 s) as before.

6.3 Application of the LQMPC Algorithm

The results of a step response of 1 radian are shown in Fig. 19. The responses were quite similar and reached

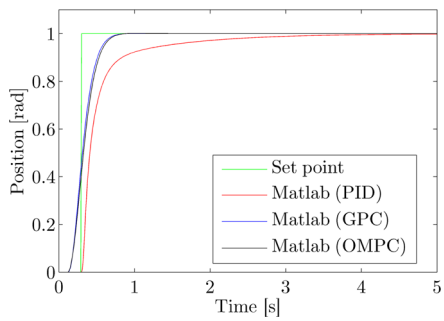

Fig. 22 comparison of step responses (Matlab output)

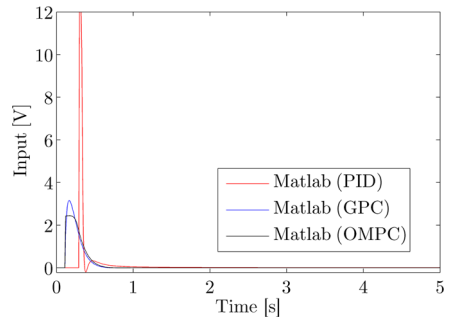

Fig. 23 comparison of step responses (Matlab input)

the set point with an offset-free tracking error. The anticipation of the set point is also visible.

The graph of a set point ramp is shown in Fig. 20. The tracking was also performed well by the LQMPC algorithm. The enlarged image at the end of the simulation shows that the set point was reached smoothly.

The constraints in combination with the feedforward were taken into account using the Gilbert algorithm [17] by defining the MCAS: *Maximal Controlled Admissible Set*. It produces the matrices of constraints F and t in terms of the autonomous model (Eq. 47). These matrices were defined offline and were valid throughout the control. Quadratic programming was used under Matlab to find the values of the degrees of freedom $\underrightarrow{c}_k$ taking the input constraints into account.

$$\underbrace{\begin{bmatrix} M & N & V \end{bmatrix}}_{\mathrm{F}} \begin{bmatrix} x_k \\ \underrightarrow{c}_k \\ \underrightarrow{R}_k - Ld_k \end{bmatrix} \leq t \tag{47}$$

Some Matlab simulations have proven that even if the saturation is active for a short period of time, the number of degrees of freedom n_c must increase considerably to allow the MCAS algorithm to converge properly. Moreover, the dimensions of the constraint matrices rise heavily. For example, with n_c=40, the dimension of the matrix t is [274x1].

Thankfully, in this case, the constraints were rather simple since they were simple saturations. Other Matlab simulations have indeed shown that taking constraints into account with the QP or with simple voltage saturations, such as U_{max}=+12V and U_{min}=-12V, leads to the same results (Fig. 21). The constraint handling was thus implemented with direct saturations rather than a QP on the real system. The computation time for one LQMPC iteration taking saturations into account was approximately 0.004 s on the embedded microcontroller.

7 Comparison of the Control Strategies

The PID controller results were taken as a reference to compare the two predictive control strategies. As previously mentioned, the parameters of the LQMPC algorithm were chosen to match the output of the GPC algorithm step response in order to compare the control inputs of each method. For the three tested controllers, the comparison of the control inputs was performed under Matlab. Figure 22 shows the outputs for step responses of 1 radian under Matlab.

Their voltage inputs are then compared in Fig. 23. It can be seen that the input of the PID saturates whereas the inputs computed by the predictive methods are far smaller. By anticipating the set point, the predictive methods require less energy than classical methods. It can also be distinguished that the input of the LQMPC algorithm is slightly lower than the input required by the GPC algorithm.

A numerical comparison of the three methods for the step response is summarised in Table 4. It is based on the Matlab simulations. The PID controller input is taken as a reference in order to compare the efficiency of the predictive methods.

Figure 24 compares the step responses obtained with the 3 tuned controllers through real life experiments. The results are quite similar to their corresponding Matlab simulations. The predictive methods

Table 4 numerical comparisons for a step response (Matlab)

	Rise time [s]	Settling time [s]	Overshoot [%]	Maximum input [V]	Input benefits [%]
PID	0.51	1.08	0	12	ref.
GPC	0.33	0.38	0	3.1	74
LQMPC	0.34	0.41	0	2.44	80

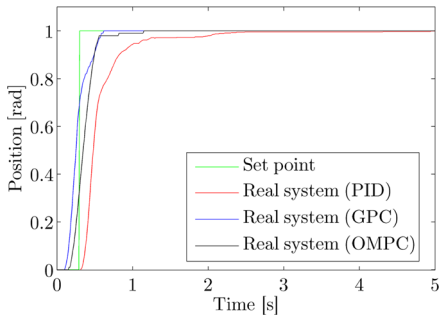

Fig. 24 comparison of step responses (real system at 100 Hz)

greatly improved the behaviour of the WMR reaching the set point faster and without any overshoot.

8 Conclusions

Three control strategies were tested on a wheeled mobile robot: a PID controller was first tuned to compare its performances with two predictive methods. Both predictive strategies were used by taking constraints and a feedforward action into account. It has been shown that predictive methods considerably improve the behaviour of the wheeled mobile robot. As compared to PID control, MPC allows the reduction of the actuator energy consumption up to 70 %. However, along with a heavy computational load, it involves the tuning of several parameters whose impact on the performance is sometimes difficult to assess. Ultimately, by comparing the inputs of the GPC and the LQMPC algorithms for similar outputs, it was observed that the LQMPC algorithm required a smaller input thanks to its infinite input and prediction horizons, leading to less energy consumption. The major weakness of this approach is the lack of consideration for the coupling of wheels. So far, the same control law has been implemented for both motors but it would be more appropriate to consider one unique single control law comprising both motor commands. In this way, the wheeled mobile robot should be responding more to disturbances coming from one of the two wheels by adapting the input of both motors. The GPC algorithm can indeed be converted into a "double input/double output" design by relying on a state space model. On the other hand, the LQMPC algorithm already works with matrices.

Low-pass filters are commonly adopted to reduce the sensitivity to measurement noise and to improve control robustness. A usual solution consists in implementing a T-filter to reduce measurement uncertainties. The T-filter is a low-pass filter reducing the impact of high frequencies and therefore helps to reduce the sensitivity of control laws with respect to measurement noise. Besides, model uncertainty could be processed by algorithms using invariant sets or linear matrix inequality [18]. In this study, using a T-filter would not be useful as the wheeled mobile robot moved on a smooth surface and an integrator was already embedded into the control law to reject disturbance.

The predictive algorithms could eventually be retuned in order to speed up the dynamics of the robot by tuning the values of the λ coefficient (GPC) or the R matrix (LQMPC) while adjusting the corresponding control horizons.

References

1. Lages, W.F., Kühne, F., Gomes da Silva Jr., J.M.: Model Predictive Control of a Mobile Robot Using Linearization. Proc. of the IEEE Mechatronics and Robotics **4**, 525–530 (2004). Germany
2. Pacheco, L., Cufi, X., Luo, N.: Using Model Predictive Control for Local Navigation of Mobile Robots. Advanced Model Predictive Control, Dr. Tao ZHENG **1**, 292–308 (2011)
3. Boucher, P., Dumur, D.: La Commande Prédictive, Méthodes et pratiques de l'ingénieur, Edition Technip, vol. 8, Paris (1996)
4. Bemporad, A., Borrelli, F., Morari, M.: Model Predictive Control Based on Linear Programming - The Explicit Solution. IEEE Transaction On Automatic Control **47**(12), 1974–1985 (2002). doi:10.1109/TAC.2002.805688
5. Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P.O.M.: Constrained model predictive control: Stability and optiMality. Automatica (Elsevier) **36**, 789–814 (2000)
6. Di Ruscio, D.: Model Predictive Control and optimization, Lecture notes, System and Control Engineering, Department of Technology, Telemark University College (2010)
7. Rossiter, J.A.: Model-Based Predictive Control: A Practical Approach, Taylor and Francis Group CRC Press (2003)
8. Clarke, D.W., Mohtadi, C., Tuffs, P.S.: Generalized predictive control - Part I. The basic algorithm. Automatica, Pergamon **23**(2), 137–148 (1987)
9. Mosca, E.: Optimal, Predictive and Adaptive Control. Prentice-Hall, Inc (1995)
10. Rossiter, J.A.: Videos on modelling, control and analysis: Model Predictive Control, University of Sheffield: Department of Automatic Control and System Engineering. https://sites.google.com/a/sheffield.ac.uk/video-lectures-on-modelling-analysis-and-control/ (2014)

11. Scokaert, P.O.M., Rawlings, J.B.: Infinite horizon linear quadratic control with constraints. In: Proceedings, IFAC World Congress, pp. 109–114, San Francisco (1996)
12. Verlinden, O., Kouroussis, G., Conti, C.: EasyDyn: a framework based on free symbolic and numerical tools for teaching multibody systems. In: ECCOMAS Thematic Conference Multibody Dynamics 2005, Madrid, Spain, pp. 21–24 (2005)
13. MATLAB Release: The Mathworks, Inc., Natick, Massachusetts, United States (2011a)
14. Verlinden, O., Ben Fékih, L., Kouroussis, G.: Symbolic generation of the kinematics of multibody systems in EasyDyn: From MuPAD to Xcas/Giac. Theor. Appl. Mech. Lett. **3**, 013012 (2013). doi:10.1063/2.1301312
15. Gwanghun, G.: Vehicle dynamic simulation with a comprehensive model for pneumatic tires, PhD Thesis, University Libraries, University of Arizona (1988)
16. Renotte, C., Vande Wouwer, A., Remy, M.: A Simple Frequency Domain Approach to the Tuning of PID Controllers - Design of an Interactive Software Tool. Journal A. Benelux Quaterly Journal on Automatic Control **42**(3), 23–27 (2001)
17. Gilbert, E.G., Tan, K.T.: Linear systems with state and control constraints: the theory and application of maximal output admissible sets. IEEE Trans. Autom. Control **36**(9), 1008–1020 (1991). doi:10.1109/9.83532
18. Kerrigan, E.C., Maciejowski, J.M.: Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control, Conference on Decision and Control (2000)

Hoai Nam Huynh was born in Hô-Chi-Minh-Ville (Viêt Nam) in 1992. He is a fulltime researcher at the 'Faculté Polytechnique' (Faculty of Engineering) of Mons in the Department of Theoretical Mechanics, Dynamics and Vibration (Belgium). He is funded by the Belgian National Fund for Scientific Research to achieve a PhD thesis about 'Robotic Machining'. He received an engineer degree in mechanics-mechatronics in 2015. His research interests are in the area of robotic machining simulations for cutting parameters optimisation.

Olivier Verlinden was born in 1964. He is a full professor at the 'Faculté Polytechnique' (Faculty of Engineering) of Mons in the Department of Theoretical Mechanics, Dynamics and Vibration (Belgium). His teaching activities include the fields of the motion of mechanical systems (kinematics, statics and dynamics of mechanical systems), computer simulation of multibody system dynamics, sound and vibration, and mechatronics (active mechanical systems, actuators, sensors, microcontrollers). He received the engineer degree in mechanics in 1988 and his PhD concerning the dynamic simulation of flexible multibody systems in 1994. He is involved in research activities in the fields of vibrations (vibrations induced by railway traffic, vibration design of electronic boards), computer simulation of multibody systems and mechatronics.

Alain Vande Wouwer was born in Brussels in 1966. He graduated in electrical engineering from the 'Faculté Polytechnique' (Faculty of Engineering) of Mons (Belgium) in 1988, and received the doctorate degree in applied sciences from the same institution in 1994, together with the European doctorate degree (in collaboration with Stuttgart University). In 1994, he achieved a post-doctoral stay in the Mechanical Engineering Department of Laval University, Québec, Canada. Nowadays, he is full professor in the Control Department of Mons University, Belgium, and coordinator of the research pole BIOSYS, which is dedicated to various aspects of life sciences, ranging from biomedical engineering to industrial biotechnology. His research interests are in mathematical modelling, soft sensing, and control of biological systems and processes, with applications in the pharmaceutical sector, as well as in water treatment and the sustainable production of renewable energies.