Realtime Application of Constrained Predictive Control for Mobile Robot Navigation

Ibtissem Malouche and Faouzi Bouani
Universite de Tunis El Manar, Ecole Nationale d'Ingenieurs de Tunis,
Laboratoire Analyse, Conception et Commande des Systemes, LR11ES20 Tunis, Tunisia

***Abstract*—This work addresses the implementation issue of constrained Model Predictive Control (MPC) for the autonomous trajectory-tracking problem. The chosen process to control is a Wheeled Mobile Robot (WMR) described by a discrete, Multiple Input Multiple Output (MIMO), state-space and linear parameter varying kinematic model. The main motivation of the constrained MPC usage in this case relies on its ability in considering, in a straightforward way, control and states constraints that naturally arise in trajectory tracking practical problems. The efficiency of the presented control scheme is validated through experimental results on a two wheeled mobile robot using both STM32F429II and STM32F407ZG microcontrollers. The controller implementation is facilitated by the usage of the automatic C code generation and interesting optimization before real-time execution. Based on the experimental results obtained, the good performance and robustness of the proposed control scheme are established.**

Keywords*—*Embedded C; STM32; microcontrollers; constrained model predictive control; otpimization

I. Introduction

The gap between theory and practice has been discussed for many years across the spectrum of academia and industry [1]. One of the reasons that explains this gap is that control systems for the first time is often found too abstract and theoretical in nature; that is, too many mathematical equations as well as block diagrams. Furthermore, the use of control algorithms simulation only will not provide sufficient illustration of the real physical application of using control theory in solving engineering problems. The above shortcomings can be avoided throughout control system experimentation which is an important way to go through practical applications of control theory so as to overcome the above-mentioned difficulty. Hence real hands-on experiments or design problems are an alternative way of augmenting the conventional way of dealing with control theory, as it can be related to real engineering applications, such as modeling, controller design, and implementation.

Moreover, although some control algorithms has been found to be quite a robust type of control in most reported applications [2], [19], their implementation on low-cost system on chip solutions (such as microcontrollers) has been historically hindered by many restrictions and constraints [11]–[18]. Among these constraints are mathematical complexities, which are not a problem in general for the research control community but represent a drawback for the use in practice. The time-to-market delays and possible design errors if the algorithm is manually written in embedded C language on one side and the high computing and associated memory demands of the algorithm on the other side are restricting the implementation of these control algorithms.

In fact, modern applications generally involve many computationally demanding iterations and have strong requirements on resource optimization. Therefore, the main drawback of complex control algorithms such as Model Predictive Control (MPC), is the need to solve mathematical program on line to compute the control action [3]. This computation prevents the application of MPC in several contexts, mainly because the computer technology needed to solve the optimization problem within the sampling time which is too expensive or sometime infeasible [5]. In fact, the resource constraints associated with embedded systems, combined with non-optimized software components used for their implementation, introduce non-determinism in the real-time system. For control systems this is of particular concern, since timing variations induced by the implementation degrade the control performance.

To cope with the above mentioned shortcoming and problems, the goal of this work is the implementation of constrained MPC algorithms on a fast system with no manual embedded C coding effort. The trade-offs between data size and computation speed, versus numerical precision and effectiveness of the computed control action is also focused on. By using MPC it follows that the tuning parameters are directly related to a cost function which is minimized in order to obtain an optimal control sequence; constraints on state, control inputs and control inputs deviation can be considered in a straightforward way [3]–[5], [7]. This objective is challenging especially in case of system with fast dynamics [6], system described with Multiple Input Multiple Output (MIMO) and parameter varying model. In this direction, the proposed control system is the nonholonomic two Wheeled Mobile Robot (WMR). The usage of the WMR as a plant is motivated by the following reasons: 1) The selected kinematic model of the WMR belongs to the class of MIMO, non-square, linear parameter varying which is challenging for control; 2) despite the apparent simplicity of the kinematic model of a WMR, the existence of nonholonomic constraints turns the design of stabilizing control laws for those systems in a considerable challenge [21]; 3) in recent years, autonomous mobile robots are finding widespread application in many areas and are at the heart of most modern control systems [22].

As a result, validating the proposed control framework in such a challenging systems will expand the number of addressable real time control applications.

On other hand, a general control system of two-wheeled robot usually has a complex structure due to plenty of sensors, which price is generally expensive. A control system with

DC motor driver is proposed instead, using low cost STM32 Discovery kits. These kits allow the controller implementation on high-performance MCUs with ARM Cortex-M4 core. Experimental results on a real WMR exhibit that the proposed control scheme yields some interesting results on autonomous trajectory tracking problem. In fact, compared to existing works such as [9] and [10], the proposed automatic C code generation saves time, avoids possible design errors with no C code manual coding effort. With regards to [3], the proposed implementation framework yields much more interesting results in term of execution speed.

The outline of this paper is as follows. In Section II, the kinematic robot model as well as the proposed MPC algorithm are presented. Implementation results demonstrating the good performance and robustness of the proposed controller are presented in Section III. Section IV concludes the paper.

II. Model Predictive Control Algorithm

The basic idea of predictive control consists in calculating, at each sampling instant, a control sequence on a prediction horizon aimed at minimizing a quadratic cost function. The control algorithm is based on the following:

(i) The use of a model to predict, on a future horizon, the output of the process,

(ii) Computing the control sequence which minimizes a performance criterion which involves a sequence of the predicted output.

The following section states the robot model presentation followed by the proposed MPC algorithm investigation.

A. Robot model presentation

We consider a WMR made up by a rigid body and non-deforming wheels (Fig. 1). It is assumed that the vehicle moves without slipping on a plane, i.e., there is a pure rolling contact between the wheels and the ground [21], [22]. Referring to [21], we can write the kinematic model of the WMR as in the following system:

$$\begin{cases} \dot{x} = v\cos(\theta), \\ \dot{y} = v\sin(\theta), \\ \dot{\theta} = w. \end{cases} \quad (1)$$

or else, in a more simplified form:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}). \quad (2)$$

where **x** describes the configuration (position and orientation) of the wheels axis's center, *C*, with respect to a global inertial frame $\{O, X, Y\}$. **u** is the control input vector, where *v* and *w* are the linear and the angular velocities, respectively.

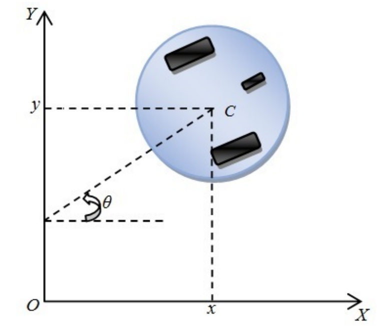


Fig. 1. Robot in (OXY) plan.

Now, considering a sampling period T_s, by applying the Euler's approximation to (1), one can obtain the following discrete-time model for the robot motion:

$$\begin{aligned} x(k+1) &= x(k) + v(k)cos(\theta(k))T_s, \\ y(k+1) &= y(k) + v(k)sin(\theta(k))T_s, \\ \theta(k+1) &= \theta(k) + w(k)T_s. \end{aligned} \quad (3)$$

or, in a compact representation,

$$\mathbf{x}(k+1) = f_d(\mathbf{x}(k), \mathbf{u}(k), T_s). \quad (4)$$

On the other hand, the problem of trajectory tracking can be stated as to find a control law such that:

$$\mathbf{x}(k) - \mathbf{x}_{\mathrm{r}}(k) = 0. \quad (5)$$

Where, $\mathbf{x}_r = [x_r \quad y_r \quad \theta_r]^T$ is a known, pre-specified reference trajectory. It is usual in this case to associate to this reference trajectory a virtual reference robot, which has the same model than the robot to be controlled. A linearized discrete time model of the system, described by (6), is obtained by computing an error model with respect to a reference car. This is ensured by expanding the right side of (3) in Taylor series around the point $(\mathbf{x}_r, \ \mathbf{u}_{r,})$ and using forward differences.

$$\begin{aligned} \tilde{\mathbf{x}}(k+1) &= A(k)\tilde{\mathbf{x}}(k) + B(k)\tilde{\mathbf{u}}(k), \\ \tilde{\mathbf{y}}(k) &= C(k)\tilde{\mathbf{x}}(k). \end{aligned} \quad (6)$$

with $A(k) = \begin{bmatrix} 1 & 0 & -vr\sin(\theta r(k))T_s \\ 0 & 1 & vr\cos(\theta r(k))T_s \\ 0 & 0 & 1 \end{bmatrix}$,

$$B(k) = \begin{bmatrix} \cos(\theta r(k))T_s & 0 \\ \sin(\theta r(k))T_s & 0 \\ 0 & T_s \end{bmatrix},$$

$C(k) = I_3$, I_3 is the (3×3) identity matrix. $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_r$ represents the error with respect to the reference car, $\tilde{\mathbf{u}} = \mathbf{u} - \mathbf{u}_r$ is its associated error control input with $\mathbf{u}_r = [v_r \quad w_r]^T$ and $\tilde{\mathbf{y}} = \mathbf{y} - \mathbf{y}_r$ denotes the system output. It is easy to see that when the robot is not moving (i.e., v_r = 0), the linearization around a stationary operating point is not controllable. However, this linearization becomes controllable as long as the control input **u** is not zero [21].

B. MPC Direct Output Method

In this paper we consider the Model Predictive Control algorithm, where the objective function is rewritten as standard quadratic form. At that point, we target to solve the Quadratic Problem (QP) in using interior-point method, widely used in applications, to solve problems iteratively such that all iterates satisfy the inequality constraints strictly. By extending the output direct method equations provided in [20] to MIMO model process and if we consider (6) for the discrete state space model, the following output at time $(k+j)$ is obtained:

$$\begin{gathered}\mathbf{y}(k+j|k) = \mathbf{y}_r(k+j|k) + CA^j\mathbf{x}(k) - CA^j\mathbf{x}_r(k) \\ + \sum_{i=0}^{j-1} CA^{j-1-i}B\mathbf{u}(k+i) - \sum_{i=0}^{j-1} CA^{j-1-i}B\mathbf{u}_r(k+i).\end{gathered} \tag{7}$$

The MPC based on a state space model aims to minimize the quadratic criterion given by:

$$\begin{aligned}J = &\sum_{s=1}^{m}\sum_{j=1}^{H_c} \lambda_{1_s}\ [\mathbf{u}_s(k+j-1)]^2 \\ &+ \sum_{s=1}^{n}\sum_{j=1}^{H_p} \lambda_{2s}[\mathbf{y}_s(k+j|k) - \omega_s(k+j)]^2.\end{aligned} \tag{8}$$

Here, H_p is the prediction horizon, H_c is the control horizon, $\lambda_1 = [\lambda_{1s}]_{s=1:2} = [\ \lambda_{11} \quad 00 \quad \lambda_{12}\ ]$ is the input weighting matrix with *(m×m)* dimension.

The output error weighting matrix with *(n×n)* dimension is given by:

$$\lambda_2 = [\lambda_{2s}]_{s=1:n} = \begin{bmatrix} \lambda_{21} & 0 & 0 \\ 0 & \lambda_{22} & 0 \\ 0 & 0 & \lambda_{23} \end{bmatrix}, \omega = [\omega_s]_{s=1:n}^T$$

is the set-point and $\omega_s\,(k+j)$ denotes the set-point at time *(k + j)*, $\mathbf{u} = [\mathbf{u}_s]_{s=1:m}^T$ and $\mathbf{y} = [\mathbf{y}_s]_{s=1:n}^T$ denote the inputs and outputs, respectively.

It is assumed that there is no control action after time *(k+H$_c$-1)*, i.e. *u(k+i) = 0* for *i > (H$_c$−1)*.

Since the MPC is a receding horizon approach, only the first computed control input **u** is implemented.

In matrix presentation, the objective function can be expressed as:

$$J \ = \ U^T\Lambda_1 U + (Y-W)^T\Lambda_2(Y-W), \tag{9}$$

in which $W = [\omega\,(k+1)\,,\ \ldots\,,\ \omega\,(k+H_p)]^T$ is the set point matrix, Λ_1 is a weighting matrix with *(mH$_c$× mH$_c$)* dimension and Λ_2 is *(nH$_p$× nH$_p$)* matrix.

As in [20], the output sequence on H_p can be written as follows:

$$\tilde{Y} = \ L\tilde{U} + \ MA\tilde{x}\,(k)\,, \tag{10}$$

where:

$$\tilde{Y} = \begin{pmatrix} y\,(k+1) - y_r\,(k+1) \\ \vdots \\ y\,(k+Hp) - y_r\,(k+Hp) \end{pmatrix},$$

$$\text{and } \tilde{U} = \begin{pmatrix} u(k) - u_r(k) \\ \vdots \\ u(k+H_c-1) - u_r(k+H_c-1) \end{pmatrix}.$$

The L matrix with the *(nH$_p$×mH$_c$)* dimension and M which is an *(nH$_p$×n)* dimensional matrix are given by:

$$L = \begin{pmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \cdots & 0 \\ CA^{Hp-1}B & CA^{Hp-2}B\cdots & \cdots & CA^{Hp-Hc}B \end{pmatrix},$$

$$M = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{Hp-1} \end{pmatrix}. \tag{11}$$

One can rewrite the objective function in a standard quadratic form:

$$J\,(U) \ = \frac{1}{2}U^TGU \ + \ g^TU \ + \ c_0, \tag{12}$$

where:

$$\begin{aligned}&G = 2(L^T\Lambda_2 L + \Lambda_1), \\ &g = L^T\Lambda_2(M\tilde{x} - W), \\ &c_0 = (MA\tilde{x} - W)^T(MA\tilde{x} - W), \\ &\Lambda_1 = diag(\lambda_1 ... \lambda_1), \\ &\Lambda_2 = diag(\lambda_2 ... \lambda_2).\end{aligned}$$

In this way, the constrained MPC problem is formulated as a compact Quadratic Problem (QP) described by (12):

$$\begin{aligned}&\min_{U\in R^{mH_c}} J\,(U) \ = \tfrac{1}{2}U^TGU \ + \ g^TU \ + \ c_0, \\ &s.t. \\ &F^TU \leq b.\end{aligned} \tag{13}$$

G is (*mH$_c$* × *mH$_c$*) matrix, *g* is (*mH$_c$* × 1) vector, *F* is (*q* × *q*) matrix where *q* is the number of inequality constraints, *q* is equal to (6*mH$_c$*) in our case, and *b* is a (6*mH$_c$* × *1*) vector.

The QP problem is subject to linear inequality constraints on the system inputs, system inputs deviation *(Δu = u(k)-u(k-1))* and system outputs as follows:

$$\begin{aligned}&U_{\min} \leq U \leq U_{\max}, \\ &\Delta U_{\min} \leq \ \Delta U \leq \Delta U_{\max}, \\ &Y_{\min} \leq Y \leq Y_{\max},\end{aligned} \tag{14}$$

in which:

$$\Delta u_{\min} = \begin{pmatrix} \Delta v_{\min} \\ \Delta w_{\min} \end{pmatrix}, \Delta U_{\min} = \begin{pmatrix} \Delta u_{\min} \\ \vdots \\ \Delta u_{\min} \end{pmatrix} \in \mathrm{R}^{mH_c},$$

$$\Delta u_{\max} = \begin{pmatrix} \Delta v_{\max} \\ \Delta w_{\max} \end{pmatrix}, \quad \Delta U_{max} = \begin{pmatrix} \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{pmatrix} \in \mathrm{R}^{mH_c},$$

$$u_{\min} = \begin{pmatrix} v_{\min} \\ w_{\min} \end{pmatrix}, \quad U_{\min} = \begin{pmatrix} u_{\min} \\ \vdots \\ u_{\min} \end{pmatrix} \in \mathrm{R}^{mH_c},$$

$$u_{\max} = \begin{pmatrix} v_{\max} \\ w_{\max} \end{pmatrix}, \quad U_{\max} = \begin{pmatrix} u_{\max} \\ \vdots \\ u_{\max} \end{pmatrix} \in \mathrm{R}^{mH_c},$$

$$y_{\min} = \begin{pmatrix} x_{\min} \\ y_{\min} \\ \theta_{\min} \end{pmatrix}, \quad Y_{\min} = \begin{pmatrix} y_{\min} \\ \vdots \\ y_{\min} \end{pmatrix} \in \mathrm{R}^{nHp},$$

$$y_{\max} = \begin{pmatrix} x_{\max} \\ y_{\max} \\ \theta_{\max} \end{pmatrix}, \quad Y_{max} = \begin{pmatrix} y_{\max} \\ \vdots \\ y_{\max} \end{pmatrix} \in \mathrm{R}^{nHp}.$$

Replacing Y in (14) by its value in equation (10), and by considering ΔU value at each iteration *(k)*, we obtain:

$$\begin{array}{l} Y_{\min} \ \leq \ L\tilde{U} + MA\tilde{x}\,(k) \ - Y_{\text{ref}} \leq Y_{\max} \ , \\ \Delta U_{\min} \leq u(k) - u(k-1) \leq \Delta U_{\max}. \end{array} \tag{15}$$

As a consequence, the following inequalities will be easily obtained:

$$U \leq (L)^{+}(LU_r + Y_{\max} + Y_{ref} - \ MA\tilde{x}\,(k)). \tag{16}$$

$$-U \leq -(L)^{+}(LU_r + Y_{\min} \ + Y_{\text{ref}} - \ MA\tilde{x}\,(k)). \tag{17}$$

$$U \ \leq \Delta U_{\max} + U(\mathrm{k}-1). \tag{18}$$

$$-U \ \leq -\Delta U_{\min} - U(\mathrm{k}-1). \tag{19}$$

The operator $(^{+})$ denotes the Moore-Penrose pseudo-inverse operator. Hence, we obtain:

$$F^T = \begin{bmatrix} -I_c \\ I_c \\ -I_c \\ I_c \\ -I_c \\ I_c \end{bmatrix}, \quad b = \begin{bmatrix} -U_{\max} \\ U_{\min} \\ -\Delta U_{\max} - U(k-1) \\ \Delta U_{min} + U(k-1) \\ -(L)^{+}(LU_r + Y_{\max} + Y_{\text{ref}} - MA\tilde{x}) \\ (L)^{+}(LU_r + Y_{\min} + Y_{\text{ref}} - MA\tilde{x}) \end{bmatrix}, \tag{20}$$

with $I_c \in \mathrm{R}^{mH_c \times mH_c}$ is the identity matrix.

C. Quadratic Problem: Interior Point Method

To set up the equations enabling the interior-point method design, we use the general theory on constrained optimization by defining a Lagrangian function and setting up Karush Kuhn-Tucker (KKT) conditions for the QP's that we wish to solve. The KKT-conditions (21) are conditions that must be satisfied for a vector U to be a solution of a given QP.

$$\begin{array}{l} \mathrm{GU} + g - F\lambda_l = 0, \\ \mathrm{s} \ - \ \mathrm{F}^T\mathrm{U} \ + \ \mathrm{b} \ = \ 0, \\ s_i\lambda_{li} \geq 0, \end{array} \tag{21}$$

with λ_{li} is Langrange multiplier of the ith inequality constraints, s_i is the ith element of the slack vector s satisfying:

$$s = F^T U - b, \ s \geq 0. \tag{22}$$

The full algorithm is stated in Fig. 2.

The algorithm requires a starting point (x_0, λ_{l0}, s_0) which does not need to be in the feasible region. In fact, this is accomplished by requiring that (λ_{l0}, s_0) > 0 and having the right hand side of (23) containing the residual vectors r_d, r_p and $r_{s\lambda}$ instead of zeros to prevent infeasibility [23].

Fundamental concepts related to this method such as central path, the stopping criteria, the predictor complementary measure for the computed Newton step μ_{aff}, the search direction $(\Delta x, \Delta\lambda_l, \Delta s)$, the complementary measure μ and the centering parameter σ are detailed in [23]. In the algorithm, the matrix S = $diag(S_1,S_2,..S_m)$ in which $S_i$are the slack vectors, $\Lambda_l = diag(\lambda_{l1}, \lambda_{l2}, ..., \lambda_{lm})$ is a diagonal matrix with the elements of the Lagrange multiplier on the diagonal, μ denotes the complementary measure, $(\Delta x_{aff}, \Delta\lambda_{aff}, \Delta s_{aff})$ is the affine scaling direction, λ_{aff} is the step length and $e = (1, 1, ...1)^T$ is a $(m \times 1)$ vector containing ones.

For implementation, v_r is measured 0.2 m/s. Since we are dealing with nonhomolonic WMR, the saturation of the input control **u** should be considered. The maximum allowed values are v_{max} = 0.4 m/s and w_{max} = 5 rad/s while the minimum ones are v_{min} = 0 and w_{min} = -5 rad/s.

```
Off line treatment
Input (x$_0$, $\lambda_{l0}$, s$_0$) and G, F, g, b
Compute residuals and complementarity measure
$r_d = Gx_0 + g - F\Lambda_{l0}$
$r_p = s_0 - F^T x_0 + b$
$r_{s\Lambda_l} = S_0\Lambda_{l0}e$
$\mu = \frac{s_0^T \lambda_{l0}}{m}$
Online treatment
While loop : terminate if stopping criteria (*) are satisfied
Predictor step:
Solve (23) to obtain an affine scaling direction ($\Delta x_{aff}$, $\Delta\lambda_{aff}$, $\Delta s_{aff}$) for setting up the right hand side for the corrector step and to obtain a good value of the centering parameter $\sigma$:
$$\begin{bmatrix} G & F & 0 \\ F^T & 0 & I \\ 0 & S & \Lambda_l \end{bmatrix} \begin{bmatrix} \Delta x_{aff} \\ \Delta\lambda_{aff} \\ \Delta s_{aff} \end{bmatrix} = - \begin{bmatrix} r_d \\ r_p \\ r_{s\lambda} \end{bmatrix}. \quad (23)$$
Compute $\alpha_{aff}$
    $\lambda_l + \alpha_{aff}\Delta\lambda_{aff} \geq 0$
    $s + \alpha_{aff}\Delta s_{aff} \geq 0$
Compute $\mu_{aff}$
$\mu_{aff} = \frac{(s + \alpha_{aff}\Delta s_{aff})^T(\lambda_l + \alpha_{aff}\Delta\lambda_{aff})}{m}.$
    Compute centering parameter $\sigma$
    $\sigma = \left(\frac{\mu_{aff}}{\mu}\right)^3$
    Corrector and centering step:
    Solve (24) to obtain search direction
$$\begin{bmatrix} G & F & 0 \\ F^T & 0 & I \\ 0 & S & \Lambda_l \end{bmatrix}$$
$$\begin{bmatrix} \Delta x \\ \Delta\lambda_l \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_d \\ r_p \\ r_{s\lambda} + \Delta S_{aff}\Delta\Lambda_{aff}e - \sigma\mu e \end{bmatrix} (24)$$
    Compute $\alpha$
  $\lambda_l + \alpha\Delta\lambda_l \geq 0$
  $s + \alpha\Delta s \geq 0$
Update (x, $\lambda_l$, s)
Update residuals and complementarity measure
$r_d = GU + g - F\lambda_l$
    $r_p = s - F^T x + b$
    $r_s\lambda = S\Lambda_l e$
    $\mu = \frac{s^T\lambda_l}{q}$
    End while loop
```

Fig. 2. Interior point algorithm.

(*) Stopping criteria are:
- Reach Z which is the maximum number of iterations to ensure that the algorithm stops. Z belongs to [50,200],
- $||rd|| \leq \varepsilon$ and $||rp|| \leq \varepsilon$, with ε = 10^{-16}.

III. Implementation Results

In order to demonstrate the effectiveness of the proposed controller scheme, the following section presents the MPC controller implementation steps on two wheeled mobile robot.

A. Implementation Software and Hardware Environment

The considered hardware environment is composed by:

* Two wheeled mobile robot with 80 RPM as no-load rotating speed at 6 V, wheel diameter is equal to 65 mm and with width and weight respectively equal to 10.2 mm and 305 g.

Fig. 3 provides an overview of the used robot.

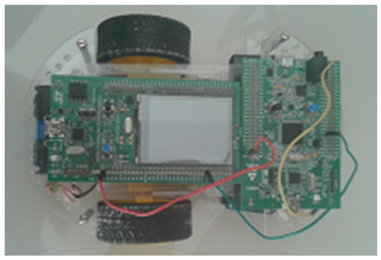


Fig. 3. WMR robot used for the implementation.

* STM32F407 and STM32F429 32-bit are high performance MicroController Units (MCUs). they include ARM Cortex-M4 core, floating point unit and built-in single-cycle multiply-accumulate (MAC) instructions. The Adaptive Real-Time Accelerator combined with STMicroelectronics 90 nm technology provides linear performance up to 172 MHz. These MCUs includes 1 MB of on-chip Flash memory and 192 KB of SRAM. These features expand the number of addressable real time control applications.

* STM32F4DISCOVERY and STM32F429DISCO are low cost kits based on STM32F407 and STM32F429 MCUs respectively and designed to help design engineers developing their applications easily. STM32F429DISCO integrates the gyroscope L3GD20 which is a low-power three-axis angular rate sensor includes a sensing element capable of providing the measured angular velocity w =[w_x,w_y,w_z] to the external world through a digital interface Serial Peripheral Interface (SPI) in our cases. For efficient control, the angular velocity is generally coupled with the linear velocity for the WMR position calculation. That's why, the STM32F4DISCOVERY, embedding an ST MEMS accelerometer, is also considered.

* Arduino Motor Shield based on the L298, is used to drive the two DC motors with control of the speed and direction of each one independently.

The software environment consists of:

* MDK-ARM is a software development and debugging environment for ARM-based microcontroller devices. It is specially designed for microcontroller applications. Its C compiler is the only compilation tool co-developed with the ARM processors, and specially designed to optimally support the ARM architectures.

* Embedded MATLAB Coder works with Real-Time Workshop to convert code from a dynamically typed language (MATLAB) to a statically typed language (C).

B. Implementation Framework

Based on the measured angular velocity, the robot angular position output θ is calculated based on the following formula:

$$\theta(k) = \theta_0 + T_s \sum_{i=0}^{k} w(i), \tag{25}$$

with θ_0 is WMR angular position at *k =0*.

In order to measure *x* abscissa and *y* ordinate, the gyroscope sensor is combined with an accelerometer MEMS sensor in the STM32F407 Discovery kit aiming to provide the robot acceleration *a(k)*. The integration of the acceleration provides the linear velocity *v(k)*.

$$v(k) = v_0 + T_s \sum_{i=0}^{k} a(i). \tag{26}$$

in which v_0 is WMR linear velocity at *k =0*. Robot positions (*x* and *y*) are then calculated based on the following equations with x_0 and y_0 are respectively WMR abscissa and ordinate at *k =0*.

$$x(k) = x_0 + v(k)T_s \sum_{i=0}^{k} \cos(\theta(i)). \tag{27}$$

$$y(k) = y_0 + v(k)T_s \sum_{i=0}^{k} \sin(\theta(i)). \tag{28}$$

These equations allow the robot positions estimation from the real measurements obtained from sensors. However, it's known that accelerometers have an unwanted phenomenon called drift caused by a small DC bias in the acceleration signal. The presence of drift can lead to large integration errors. If the acceleration signal from a real accelerometer was integrated without any filtering performed, the output could become unbounded over time. That's why to solve the problem of drift, a pass-band filter is used to remove the DC component of the acceleration signal. The frequency response of the filter must have a very low and very high cutoff frequencies compared to the bandwidth of the signal. By filtering before integrating, drift errors are eliminated. The block diagram of the closed loop system is illustrated by Fig. 4.

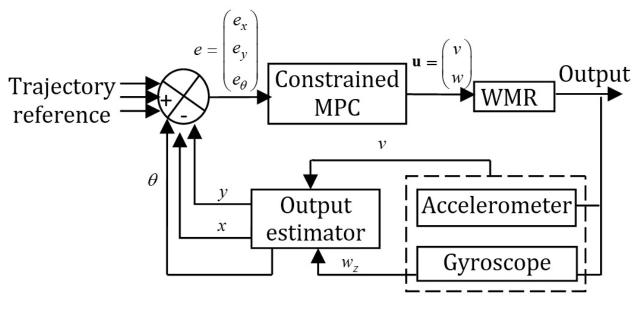

$e = \begin{pmatrix} e_x \\ e_y \\ e_\theta \end{pmatrix}$
$\mathbf{u} = \begin{pmatrix} v \\ w \end{pmatrix}$
Trajectory reference
+
-
Constrained MPC
WMR
Output
v
θ
y
x
Output estimator
Accelerometer
Gyroscope
w_z

Fig. 4. WMR closed loop implementation diagram.

Fig. 5 and 6 summarize needed steps for the proposed controller implementation.

Steps executed during Systick interrupt handler should not exceed the sampling time T$_s$=0.01052 s.

Hint: knowing that w_{max} and v_{max} parameters are subject of change according to battery depletion over time, terrain type,... we propose in this paper a real-time measurement of w_{max} parameter before trajectory tracking. This is done by turning the robot around (*OZ*) axis with maximum allowed speed and measuring the angular velocity which corresponds to w_{max}. v_{max} is then deduced given that a linear velocity is the product of the WMR radius *r* and angular velocity.

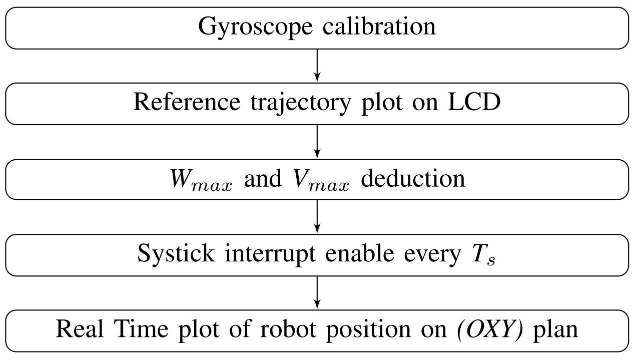

Gyroscope calibration
Reference trajectory plot on LCD
W_{max} and V_{max} deduction
Systick interrupt enable every T_s
Real Time plot of robot position on *(OXY)* plan

Fig. 5. Systick handler treatment.

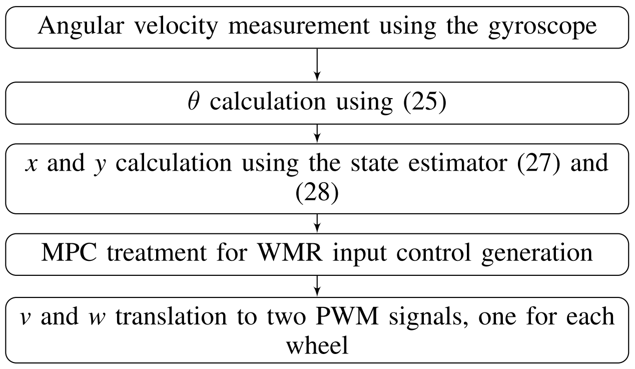

Angular velocity measurement using the gyroscope
θ calculation using (25)
x and *y* calculation using the state estimator (27) and (28)
MPC treatment for WMR input control generation
v and *w* translation to two PWM signals, one for each wheel

Fig. 6. Main program.

C. Automatic Code Generation for the Implementation

For code generation, Embedded Simulink Coder embeds many configuration options and advanced optimization for fine-grain control of the generated code' functions based on the processor architecture. These options allow control function boundaries, preserve expressions and apply optimization on multiple blocks to further reduce code size. The MDK-ARM tool integrates an ARM C compiler including a number of compiler optimization allowing code generation based on chosen microcontroller device and application area. The maximum level of performance optimization is chosen. Steps for automatic C code generation are detailed in [24].

D. Implementation Footprint and Execution Speed Optimization

Control algorithms implementation on system on chip solutions has been historically hindered by the high computing and associated memory demands constraints. To overcome this problem, we propose some optimization hints detailed on [24] which results in a faster and more efficient system-development work-flow. MDK-ARM toolchain is used as development and debugging environment, Among these optimization hints, the usage of the simple precision floating type is particularly relevant. In fact, considering the hardware Floating Point Unit (FPU) of the STM32F429 device rather than the software double precision types (generated by default by MATLAB Embedded Coder) is an optimization hint for performance increase. In fact a simple addition of two variables, declared as double, requires 82 CPU cycles due to double software library call whereas in simple precision case, it requires only 2 CPU cycles to be executed thanks to the hardware FPU usage. To avoid the unnecessary type conversion or confusion, the letter "f" is assigned following the numeric value.

Results provided in Table 1 focus on execution speed results found after applying code optimization techniques on inputs constraints only and in the case of constraints on inputs and outputs.

TABLE I. OPTIMIZATION RESULTS

Constraints	Execution time per sample (ms)
Inputs constraints	2.66
Inputs and inputs deviation constraints	3.89
Inputs, inputs deviation and outputs constraints	4.80

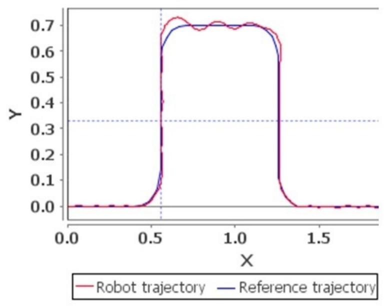
From Table 1, it is deduced that generated code is subject to very interesting execution time optimization. In fact, applying inputs, inputs deviation and outputs constrains does not exceed the half of the sampling period (10 ms) which reflects the effectiveness of the proposed control scheme.

E. Implementation Results: Inputs Constraints

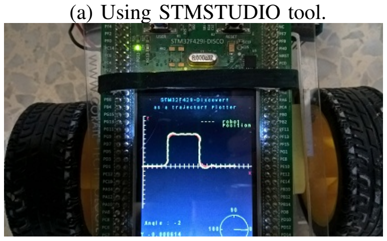
The robot trajectory is displayed using both LCD screen of the STM32 Discovery Kit and the STMStudio tool. This software tool is non-intrusive to the application code and used to monitor STM32 applications while they are running by reading and displaying their variables in real-time.

The maximum allowed linear velocity v_{max} is identified as 0.4 m/s and its minimal value $v_{min} = 0$ m/s. The angular velocities w_{min} and w_{max} are ±5 rad/s. The linear input deviation Δv_{max} and Δv_{min} are measured and fixed to ±0.25 m/s. Similarly, Δw_{max} and Δw_{min} are branded to ±1 rad/s.

Fig. 7 shows that the robot tracks successfully the reference path. This is confirmed by the robot abscissa, ordinate and orientation errors with regards to the reference path presented in Fig. 8, 9 and 10.



(a) Using STMSTUDIO tool.



(b) Using LCD screen.

Fig. 7. WMR trajectory plot.

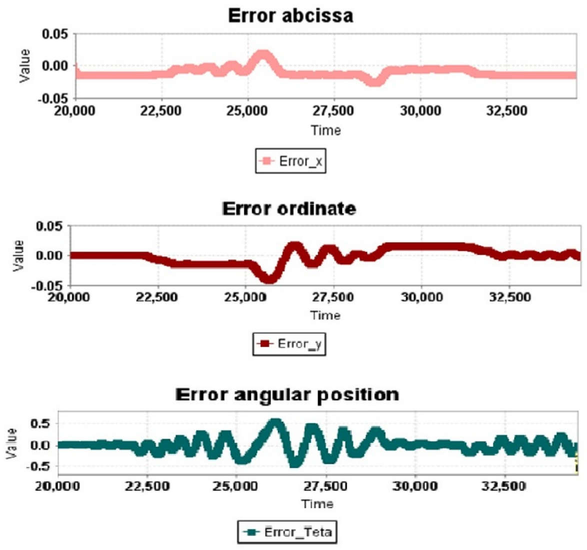


Fig. 8. WMR trajectory errors.

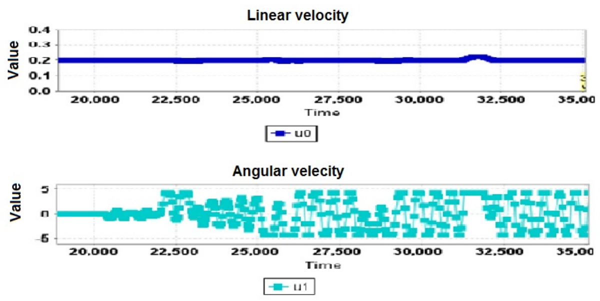


Fig. 9. Control inputs.

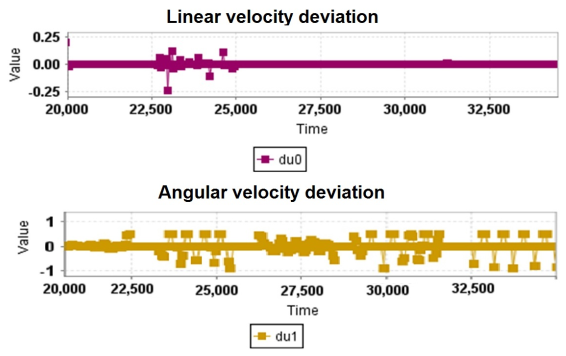


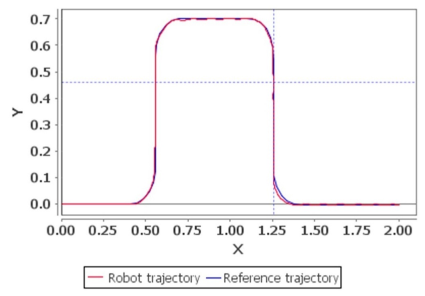
Fig. 10. Control inputs deviation.

F. Implementation Results: Inputs and Output Constraints

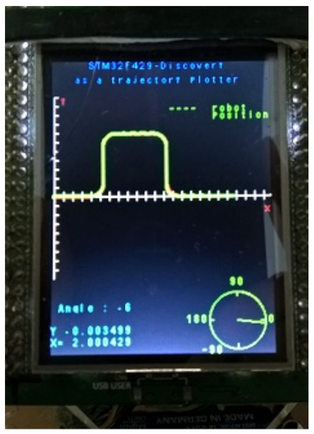
In addition to inputs constraints already defined in the previous paragraph, output constraints are also considered in this section. The output constraints are defined as y_{min} = ω - [0.01 0.01 0.25] and y_{max}= ω + [0.01 0.01 0.25].

Fig. 11 and 12 present the plot of the robot trajectory as well as the reference path. It is concluded from Fig. 11 and 12 that the output constraints are satisfied with errors very close to zero along the path.

From Fig. 11, 12, 13 and 14, it is established that inputs control and inputs control deviations constraints are satisfied and the proposed control method controls successfully the process with a very good set-point tracking.



(a) Using STMSTUDIO tool.



(b) Using LCD screen.

Fig. 11. WMR trajectory plot.

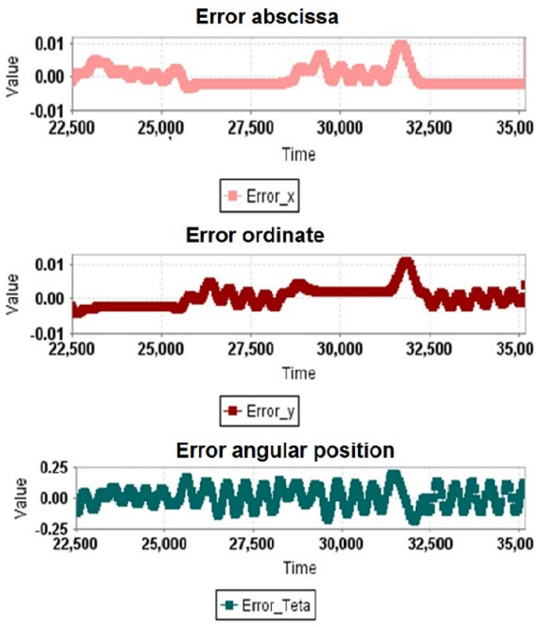


Fig. 12. WMR trajectory errors.

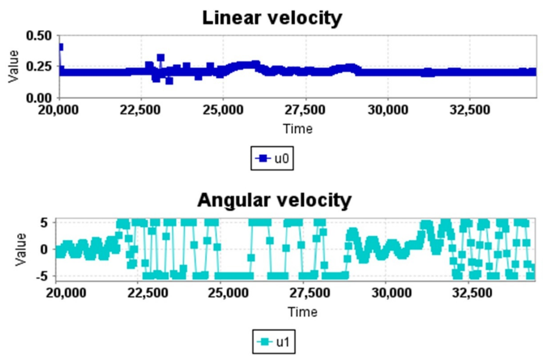


Fig. 13. Control inputs.

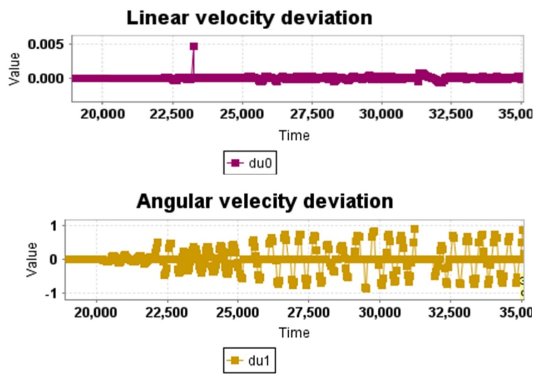


Fig. 14. Control inputs deviation.

IV. CONCLUSION

In this manuscript, Model Predictive Control implementation was focused on taking into account control inputs, inputs deviation and outputs constraints. The proposed control scheme exhibited suitable global performance when applied to wheeled mobile robots (WMR) for trajectory tracking problem.

Implementation results highlight the efficiency of the proposed design method. A framework for embedding MPC controller on a high performed STM32F429 microcontroller has been used. Optimization techniques have been applied to the generated code. Hence, an efficient implementation of the proposed control method yields a low computational burden with a high execution speed. Indeed, based on the experimental results, we noticed that the proposed method control successfully the process with a good set-point tracking.

In forthcoming article, we will focus on performance analysis comparison between the constrained MPC method proposed in this paper and a MIMO adaptive Proportional Integral Derivative (PID) regulator [8] for the trajectory tracking problem.

REFERENCES

[1] Z. Gao and R. R. Rhinehart, *Theory vs. practice: The challenges from industry*, American Control conference, Boston: Massachusetts, June 30 - July 2, 2004.

[2] E. Camacho and C. Bordons, *Model Predictive Control*, Springer, London, 2004.

[3] W. F. Lages and J. A. V Alves, *Real-time control of a mobile robot using linearized model predictive control*, IFAC Proceedings Volumes, vol. 39 no. 16, pp. 968-973, 2006.

[4] R. Sharma, F. Dusek and D. Honc, *Comparitive Study of Predictive Controllers for Trajectory Tracking of Non-holonomic Mobile Robot*, 21st International Conference on Process Control (PC) June 69, Strbske Pleso, Slovakia, 2017.

[5] H. N. Huynh, O. Verlinden and A. V. Wouwer, *Comparative Application of Model Predictive Control Strategies to a Wheeled Mobile Robot*, J Intell Robot Syst, vol. 87 no. 1, pp 8195, July 2017.

[6] D. Q. Mayne, J. B. Rawlings, C. V. Rao and P. O. M. Scokaert, *Constrained model predictive control: Stability and optiMality*, Automatica (Elsevier), vol. 36, pp. 789-814, 2000.

[7] A. Bemporad, F. Borrelli, M. Morari, *Model Predictive Control Based on Linear Programming - The Explicit Solution*, IEEE Transaction On Automatic Control vol. 47 no. 12, pp. 1974-1985, 2002.

[8] I. Malouche and F. Bouani, *A New Adaptive Partially Decentralized PID Controller for Non-square Discrete-time Linear Parameter Varying Systems*, in revision in the International Journal of Control, Automation, and Systems (IJCAS), Springer.

[9] A. A. Kheriji, F. Bouani, M. Ksouri and M. B. Ahmed,*A Microcontroller Implementation of Model Predictive Control*, World Academy of Science, Engineering and Technology International Journal of Electrical and Information Engineering, vol. 5 no. 5, 2011, pp. 600-606.

[10] A. A. Kheriji, F. Bouani and M. Ksouri, *A Microcontroller Implementation of Constrained Model Predictive Control*, International Journal of Electrical and Electronics Engineering, vol.5 no.3, 2011, pp. 199-206.

[11] J. Currie, A. Prince-Pike and D. I. Wilson, *Auto-Code Generation for Fast Embedded Model Predictive Controllers*, 19th International Conference on Mechatronics and Machine Vision in Practice (M2VIP12), Auckland, New-Zealand, 28-30th November 2012, pp. 122-128.

[12] D. W. Clarke , C. Mohtadi and P. S. Tuffs, *Generalized Predictive Control Part I. The Basic Algorithm*, Automatica, 1987, vol. 23 no. 2, pp. 137-148.

[13] K. Ling, S. Yue and J. Maciejowski, *A FPGA implementation of model predictive control*, American Control Conference, Minneapolis: Minnesota, USA, June 2006.

[14] A. Vikstrm, *A study of automatic translation of MATLAB code to C code using software from MathWorks*, Masters Thesis, Lulea University, 2009.

[15] K. Ling, B. Wu, and J. Maciejowski, *Embedded model predictive control (mpc) using a FPGA*, 17th IFAC World Congress, Seoul, Korea, July 2008, pp. 6-11.

[16] A. Kheriji, F. Bouani, and M. Ksouri, *Efficient implementation of constrained robust model predictive control using a state space model*, International Conference on Informatics in Control, Automation and Robotics (ICINCO), Madeira, Portugal, June, 2010, vol. 3, pp. 116-121.

[17] A. Kheriji, F. Bouani, and M. Ksouri, *A Microcontroller Implementation of Constrained Model Predictive Control*, International Journal of Electrical and Electronics Engineering, 2011, vol. 5 no. 4, pp. 272-279.

[18] L. G. Bleris, J. Garcia, M. V. Kothare, and M. Arnold, *Towards embedded model predictive control for System-on-a-Chip applications*, Journal of Process Control, 2006, vol. 16 no. 3, pp. 255-264.

[19] D. I. Wilson and B. R. Young. *The Seduction of Model Predictive Control*, Electrical & Automation Technology, December-January 2006.

[20] K. Watanabet, K. Ikeda, T. Fukuda and S. Tzafestas, *Adaptive generalized predictive control using a state space approach*, International Workshop on Intelligent Robots and Systems IROS, no. 91, Japan: Osaka, 1991, pp. 1609-1614.

[21] F. Khne, W. F Lages, J. M. Gomes. da Silva Jr, *Model Predictive Control of a Mobile Robot Using Linearization*, Mechatronics and Robotics, 2004, Aachen, Germany, pp. 525-530.

[22] L. Pacheco and N. Luo, *Mobile robot local trajectory tracking with dynamic predictive control technics*, International Journal of innovative Computing, Information and control, vol. 7, no 6, June 2011.

[23] R. K. Thomas, *Interior-point algorithms for quadratic programming*, Master Thesis. Technical University of Denmark, 2008

[24] I. Malouche, A. Kheriji and F. Bouani, *Automatic Model Predictive Control Implementation in a High-performance Microcontroller*, International Conference on Systems, Analysis and Automatic Control (SAC), Mahdia, Tunisia, 2015.