World Academy of Science, Engineering and Technology
International Journal of Electrical and Information Engineering
Vol:5, No:8, 2011

A Microcontroller Implementation of Constrained Model Predictive Control

Amira Kheriji Abbes, Faouzi Bouani, Mekki Ksouri

Abstract—Model Predictive Control (MPC) is an established control technique in a wide range of process industries. The reason for this success is its ability to handle multivariable systems and systems having input, output or state constraints. Neverthless comparing to PID controller, the implementation of the MPC in miniaturized devices like Field Programmable Gate Arrays (FPGA) and microcontrollers has historically been very small scale due to its complexity in implementation and its computation time requirement. At the same time, such embedded technologies have become an enabler for future manufacturing enterprisers as well as a transformer of organizations and markets. In this work, we take advantage of these recent advances in this area in the deployment of one of the most studied and applied control technique in the industrial engineering. In this paper, we propose an efficient firmware for the implementation of constrained MPC in the performed STM32 microcontroller using interior point method. Indeed, performances study shows good execution speed and low computational burden. These results encourage to develop predictive control algorithms to be programmed in industrial standard processes. The PID anti windup controller was also implemented in the STM32 in order to make a performance comparison with the MPC. The main features of the proposed constrained MPC framework are illustrated through two examples.

Keywords—Embedded software, microcontroller, constrained Model Predictive Control, interior point method, PID antiwindup, Keil tool, C/C++ language.

I. Introduction

Owing to its ability in controlling Muti Input Multi Output systems and in handling constraints arising in industrial applications, the Model Predictive Control (MPC) has become a mature control strategy in the last few years. All these advantages make this method attractive to the academic community and to be emerging in the process industries. Fundamentally, at each step of MPC, an optimal control sequence is computed such that plant outputs follow a given reference trajectories by minimizing the difference between set-points and predicted outputs. This optimal control sequence is found by solving a quadratic program problem. Then, only the optimal current input is applied to the plant and this procedure is repeated at the next sampling instance [1].

In order to predict the future behavior of the process output, we may have a model. In this work, the state space model is considered to compute the control signal. The advantage of this type of model is that the multivariable systems can easily be dealt with. Moreover, extensive amount of literature consider this representation to solve robust model predictive control [2], [3], [4], [5], [6], [7], [8], [9], [10], [11].

A. K.Abbes, F. Bouani, M. Ksouri are in members of the Laboratory of Analysis and Control Systems, National School of Engineering of Tunis Belvedere BP 37, 1002 Tunis e-mails: amira.kheriji@enit.rnu.tn, faouzi.bouani@enit.rnu.tn, mekki.ksouri@enit.rnu.tn.

MPC has been applied in high bandwidth applications such as the slow dynamical systems encountered in chemical process control as well as servomechanisms[12]. In addition, process units as fluid catalic cracking and crude atmospheric distillation have been controlled by MPC controllers for more than two decades [13]. However, in control systems with fast sampling times the MPC has not been introduced yet. The reason for this is that its implementation is explained by its time demanding and its mathematical complexity requirement. In the same time, this control strategy is very desirable for applications with fast dynamical systems. There are few recently works which investigated the MPC in fast devices such as FPGA and Programmable Logic Controller (PLC) [14], [15], [16], [17] and only few works [18] developed the implementation of MPC using a microcontroller but without constraints. The FPGA and the PLC devices have the advantage of a simple implementation of control methods since either their program tools present a straightforward communication with Matlab/Simulink software or the matrix operation blocks are already implemented. Although the FPGA are characterized by a high speed since the maximum frequency can reach even 300 MHz, its drawback is the high power consumption. However, since until recently the priority requirement for embedded systems application is the low power consumption, in this work, we propose an efficient implementation of the proposed constrained MPC firmware using a performed microcontroller characterized by a very low cost and an extensive range of peripherals and featuring three low-power mode.

A microcontroller is described as a computer on a chip because it contains all the features of a full computer including central processor, volatile and non-volatile memories, input and output ports with special features such as serial communication, analog-to-digital conversion and, more recently, signal processing. The presence of microcontroller in semi conductor products is becoming undoubtedly noticeable. This device is used for a variety of industrial applications such as for medicine and bioengineering, aerospace, automotive systems and transportation, microwave ovens, washing machines, integrated secure network systems, etc. Moreover, the advancement of microcontrollers and what they offer combined with their speed, made them more suitable for a large variety of control applications.

In this paper, we propose an efficient constrained Model Predictive Controller firmware for a performed STMicroelectronics microcontroller (STM32). The inverse matrix is implemented using the Gauss-Jordan method presented in [19]. Moreover, the optimization problem is solved using the interior point method explained in [20]. The control application could

benefit from the power features and the flexibility of the STM32F103xB devices. The STM32 keil starter kit based on a JTAG interface and the STM32 board was used to implement the proposed constrained MPC firmware. Since the conversion from m files (Matlab) to C file decreases the performance of the execution time, in this work, we choose to develop all the proposed framework with Keil development tools designed for ARM processor-based microcontroller devices working with C/C++ language. In order to test the effectiveness of our proposed constrained MPC software, the PID antiwindup was also implemented in the STM32 and compared with the proposed constrained MPC software.

The rest of this paper is organized as follows: in section II, a theoretical background which consists of a review of the MPC method as well as the interior point method are presented. Section III states the hardware and the software development tools used in this application. In section IV, a detailed description of the proposed firmware development is presented in which the different source files and functions are illustrated. In section V, the effectiveness of the proposed code is outlined through two examples in which a performance study of this proposed MPC software and a comparison with the PID anti windup controller are done. The last section is dedicated to conclude this paper.

II. Theoretical background: review of the constrained MPC and the interior point method

A. Review of the constrained MPC

We consider the following discrete state space model:

$$x(k+1) = Fx(k) + G\Delta u(k) \tag{1a}$$

$$y(k) = Hx(k) \tag{1b}$$

in which $x(k) \in \mathbb{R}^n$ is the state of the system, $\Delta u(k)$ is the control increment, $y(k)$ is the measured output and the operator $\Delta = 1-z^{-1}$ denotes the integral action which ensures static error elimination.

Consequently, we can obtain using equation 1 the following state at time $k+j$:

$$x(k+j|k) = F^j x(k) + \sum_{i=0}^{j-1} F^{j-1-i} G\Delta u(k+i) \tag{2}$$

Furthermore, by using equations 1 and 2, the j-step ahead output predictor value is written as follows:

$$\widehat{y}(k+j|k) = HF^j x(k) + \sum_{i=0}^{j-1} HF^{j-1-i} G\Delta u(k+i) \tag{3}$$

The constrained MPC problem to be minimized is a quadratic one given by:

$$J_1 = \sum_{j=1}^{H_p} (\widehat{y}(k+j|k) - w(k+j))^2 + \lambda_{11} \sum_{j=1}^{H_c} \Delta u(k+j-1)^2 \tag{4}$$

subject to linear inequality constraints on the system inputs:

$$u_{min} \leq u(k+i-1) \leq u_{max} \quad , \quad i = 1, \cdots, H_c \tag{5a}$$

$$\Delta u_{min} \leq \Delta u(k+i-1) \leq \Delta u_{max} \ , \ i = 1, \cdots, H_c \tag{5b}$$

Here H_p is the prediction horizon, H_c is the control horizon, λ_{11} is the weighting factor, $\widehat{y}(k+j|k)$ is described in equation 3, $w(k+j)$ denotes the set-point at time $k+j$, u_{min} and u_{max} are respectively the low and the high levels of the control action and Δu_{min} and Δu_{max} are are respectively the low and the high levels of the control increments.

It is easier to use the matrix form. Therefore, the output sequence on H_p prediction horizon can be written as follows:

$$Y = L\Delta U + Mx(k) \tag{6}$$

in which:

$$Y = [\widehat{y}(k+1|k), \widehat{y}(k+2|k), \ldots, \widehat{y}(k+H_p|k)]^T,$$
$$\Delta U = [\Delta u(k), \Delta u(k+1), \ldots, \Delta u(k+H_c-1)]^T,$$

It is assumed that there is no control action after time $k+H_c-1$, i.e. $\Delta u(k+i) = 0$ for $i > H_c - 1$. Since the MPC is a receding horizon approach, only the first computed control increment $\Delta u(k)$ is implemented.

The L matrix with the (H_p, H_c) dimension and M which is an (H_p, n) dimensional matrix are given by:

$$L = \begin{bmatrix} HG & 0\ldots & 0 & \\ HFG & HG & 0 & \\ \vdots & \vdots & \ddots & \\ HF^{H_p-1}G & HF^{H_p-2}G & \ldots & HF^{H_p-H_c}G \end{bmatrix},$$

$$M = \begin{pmatrix} HF \\ HF^2 \\ \vdots \\ HF^{H_p} \end{pmatrix}.$$

The objective function can be expressed as:

$$J_1 = (Y-W)^T(Y-W) + \lambda_{11}\Delta U^T \Delta U \tag{7}$$

in which:

$$W = [w(k+1), \ldots, w(k+H_p)]^T$$

In this way, the constrained MPC problem is formulated as a compact Quadratique Program (QP) problem:

$$\min(\frac{1}{2}\Delta U^T Q \Delta U + c^T \Delta U) \tag{8}$$

s.t:

$$J\Delta U \leq g \tag{9}$$

where :

$$Q = 2(L^T L + \Lambda) \ ,$$
$$\Lambda_1 = \lambda_{11} I_c \quad ,$$
$$c = (L^T M x - L^T W).$$

$I_c \in \mathbb{R}^{H_c \times H_c}$ is the identity matrix, $Q \in \mathbb{R}^{H_c \times H_c}$, $J \in \mathbb{R}^{m_c \times H_c}$ and $g \in \mathbb{R}^{H_c \times 1}$ are computed using equations 5 and

where m_c is the number of constraints (in our case $m_c = 4H_c$ since only the constraints on the control and the control increment are considered).

B. The interior point method

The connections between optimization and control theory have been explored by many researchers and optimization algorithms have been applied with success to optimal control [21].

In this paper, we apply the unfeasible interior point method in the proposed constrained MPC software microcontroller implementation. The special case of the optimality conditions that must be satisfied by solutions of linear program problems of equations 8 and 9 are knwown as the Karush Kuhn Tucker (KKT) conditions [20]:

$$\begin{aligned} Qv + J^T\lambda &= -c \\ -Jv + g &= t \\ \lambda \geq 0\ ,\ t \geq 0\ ,\ t^T\lambda &= 0. \end{aligned} \tag{10}$$

where $v = \Delta U$.

Moreover, the unfeasibilities and the duality gap μ defined by:

$$\mu^k = \frac{(t^k)^T\lambda^k}{m_c} \tag{11}$$

are gradually reduced to zero as $k \longrightarrow \infty$, where k is the number of iteration sequence.

The mixed Linear Complementary is defined such that:

$$\begin{bmatrix} Q & J^T \\ -J & 0 \end{bmatrix} \begin{bmatrix} v \\ \lambda \end{bmatrix} + \begin{bmatrix} c \\ g \end{bmatrix} = \begin{bmatrix} 0 \\ t \end{bmatrix} \tag{12}$$

The steps of the unfeasible interior point method for solution of equations 8 and 9 are listed in algorithm 1.

	Algorithm 1: the unfeasible Interior Point method
Step 1	Given (v^0, λ^0, t^0) with $(\lambda^0, t^0) > 0$.
Step 2	At the k-th iteration step, solve the increments $(\Delta v^k, \Delta\lambda^k, \Delta t^k)$ as follows: $\begin{bmatrix} Q & J^T \\ J & -(\Lambda^k)T^k) \end{bmatrix} \begin{bmatrix} \Delta v^k \\ \Delta\lambda^k \end{bmatrix} = \begin{bmatrix} r_1^k \\ r_2^k \end{bmatrix}$ and $\Delta t^k = -t^k + (\Lambda^k)^{-1}(\sigma^k\mu^k e - T^k\Delta\lambda^k)$, where: $r_1^k = -Qv^k - J^T\lambda^k - c$, $r_2^k = -Jv^k + g - \sigma^k\mu_k(\Lambda_k)^{-1}e$, $\Lambda^k = \begin{bmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_{m_c}^k \end{bmatrix}$, $e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$. $T^k = \begin{bmatrix} t_1^k & & \\ & \ddots & \\ & & t_{m_c}^k \end{bmatrix}$, for some $\sigma_k \in (0,1)$ which is often confined to the range $[10^{-3}, 0.8]$. *Remark:* The increments $(\Delta v^k, \Delta\lambda^k)$ can also be solved using these equations: $\Delta\lambda^k = (\Gamma - JQ^{-1}J^T)^{-1}(r_2^k - JQ^{-1}r_1^k)$, $\Delta z^k = Q^{-1}r_1^k - Q^{-1}J^T\Delta\lambda^k$,
Step 3	Compute the new variables: $(v^{k+1}, \lambda^{k+1}, t^{k+1}) = (v^k, \lambda^k, t^k) + \alpha^k(\Delta v^k, \Delta\lambda^k, \Delta t^k)$ for some $\alpha^k \in (0,1]$, this step length is computed as follows : first, find the maximum value of α^k such that: $(v^k, \lambda^k, t^k) + \alpha_{max}(\Delta v^k, \Delta\lambda^k, \Delta t^k) > 0$, then, we set: $\alpha_k = \min(1, 0.995\alpha_{max})$
Step 4	If the iteration converges stop the process and the optimal value v^{k+1} is obtained; otherwise go back to step 2 with the new values obtained of $(v^{k+1}, \lambda^{k+1}, t^{k+1})$ and continue the iteration process.

III. STM32 starter kit and Keil development tool

In this section, an overview of the hardware and the software development tools is presented.

A. STM32F103RB microcontroller

First of all, we have to clarify why the choice of a micro-controller? Most of the proposed works of the implementation

of the predictive control method use only the FPGA or the PLC.

May be this is due to the complexity of use of the microcontroller and the huge amount of time needed to implement such method above all when incorporating constraints. However, by proposing an optimized algorithm with a reducing size of code and by choosing a low cost microcontroller with a low power consumption, one can takes advantages of such miniaturized device.

Moreover, the choice to adopt STM32 is based on trade offs between price (since the STM32 discovery costs few dollars), performance and low power consumption. Indeed, the STM32F103RB performance line family incorporates the high-performance ARM Cortex-M3 32-bit RISC core operating at a maximum of 72 MHz frequency, high-speed embedded memories (Flash memory of 128 Kbytes size and SRAM of 20 Kbytes size), and an extensive range of enhanced I/Os and peripherals connected to two APB buses [22]. Furthermore, it has 12-bit ADCs, three general purpose 16-bit timers plus one PWM timer, as well as standard and advanced communication interfaces: two I2Cs and SPIs, three Universal synchronous asynchronous receiver transmitter USARTs, an USB and a CAN peripherals. In addition, the STM32F103RB has configurable and flexible power management features. The power consumption or hardware can be managed to match the system's requirements. Power management is provided via clock control to the CPU and individual peripherals. This device supports the following three global power control modes.

The STM32F10xxx devices feature three low-power modes: sleep mode (CPU clock off, all peripherals including Cortex-M3 core peripherals like NVIC, SysTick, etc. are kept running), stop mode (all clocks are stopped) and the standby mode (1.8V domain powered-off). In addition, the power consumption in run mode can be reduced by one of the following means: slowing down the system clocks, gating the clocks to the APB and AHB peripherals when they are unused. This is obviously an attractive property to industry as saving power and CPU high frequency can be a very costly affair indeed.

B. STM32 starter kit

The STM32 starter kit presented in Fig. 1 was used to implement the MPC program. It is composed of:

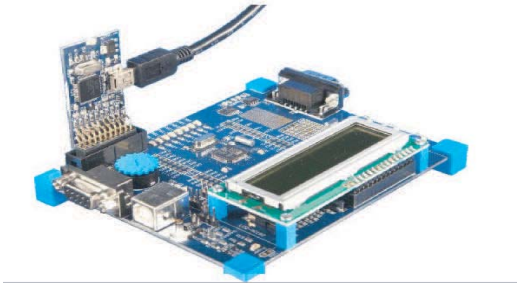


Fig. 1. STM32 starter kit

- MCBSTM32 Evaluation Board which includes:
  - STM32F103 with 72 MHz maximum Cortex-M3 processor based MCU with 128KB Flash, 20KB RAM, and 49 GPIO.
  - USART, CAN, USB Interfaces and SD/MMC card slot.
  - 16x2 LCD panel, 8 LED's, 3 push buttons, and scratchpad area.
- A JTAG interface supporting Cortex-M3 Serial Wire Debugger (SWD) and Serial Wire Viewer (SWV) modes.

In order to load the program into the STM32 device, the ULINK-ME is used. Moreover, test data and test results can be transferred between STM32 and the PC through the RS232 serial link via the USART communication protocol.

C. Keil development tools

Keil is a software developement tools. It makes C/C++ compilers, debuggers, integrated environments, middleware, real-time kernels, simulation models, and evaluation boards for ARM, Cortex-M processor families. The used version is the μVision 4. The μVision 4 screen provides a menu bar for command entry, a tool bar where can select command buttons, and windows for source files, dialog boxes, and information displays. This version has two operating modes:

- *Build Mode:* Allows to translate all the application files and to generate executable programs.
- *Debug Mode:* Provides a debugger for testing the application.

This tool has the ability to communicate information to the serial port of the PC monitor.

IV. Firmware Development

In the proposed algorithm, the constrained Model Predictive Control is implemented (see Fig. 2).

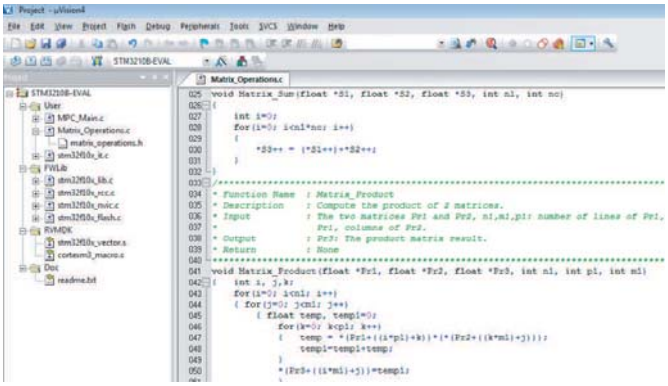


Fig. 2. Structure of MPC algorithm in Keil μVision4 development tool

A. Software description of the MPC controller

This subsection describes the MPC controller software and gives details about the related functions. It is important to notice that automatically generating matlab algorithms into C/C++ environment is time demanding. Therefore in this work

we choose to implement all the predictive control algorithm in C/C++ code. In spite of its complexity, this method will be more optimized.

In order to use the MPC controller software, some steps should be followed:

1) define the number of states, the number of iterations, the prediction horizon, the control horizon and the weighting factor λ_1,
2) define the state matrix, the input matrix and the output matrix of the model,
3) fix the initial values of the control signal and the state,
4) solve the minimization problem of equations 8 and 9 and find the optimal control action using the unfeasible interior point method,
5) Test the convergence of the interior point algorithm, if it converges inject the control action in the plant in order to find the state and the output actions, otherwise go to step 4 with the new computed variables $(v^{k+1}, \lambda^{k+1}, t^{k+1})$.

Below, we present the different functions developed in this firmware, their description, their inputs and their outputs which allow finally to compute the optimal control action.

- MPC_Main: it presents the main program of the MPC controller. A call of routines of the matrix operations is made when needed to compute the optimal control action using the interior point method. In order to be executed, this main program needs the declaration of the model, the prediction horizon, the control horizon, the weighting factor and the number of iterations. It requires also the initialization of the control signal and the initialization of the interior point variables such as λ^0, t^0 and v^0. Moreover, the system clock frequency is configured using RCC_Configuration function described in Table I:

TABLE I
RCC_CONFIGURATION FUNCTION DESCRIPTION

Function name	RCC_Configuration
Function prototype	void RCC_Configuration(void)
Behavior description	Configures the different system clocks
Input	None
Output	None

- Matrix_Operations: it contains all the matrix manipulation functions described below:

1) Zero_Matrix: it is described in Table II and allows to initialize all the matrices at zeros.

TABLE II
ZERO_MATRIX FUNCTION DESCRIPTION

Function name	Zero_Matrix
Function prototype	void Zero_matrix(float *T, int n_z, int m_z)
Behavior description	Matrix initialization
Input	T: the output matrix, n_z: the number of lines of T, m_z: the number of columns of T.
Output	None

2) Matrix_Product: it is described in Table III and allows the multiplication of two matrices.
3) The Matrix_Sum: it is described in Table IV and make the sum of two matrices.

TABLE III
MATRIX_PRODUCT FUNCTION DESCRIPTION

Function name	Matrix_Product
Function prototype	void Matrix_Product(float *P_1, float *P_2, float *P_3, int n_1, int p_1, int m_1)
Behavior description	Compute the product of two matrices
Input	P_1: the first left matrix, P_2: the second right matrix, P_3: the product matrix, n_1: the number of lines of P_1, p_1: the number of lines of P_2, m_1: the number of columns of P_2.
Output	None

TABLE IV
MATRIX_SUM FUNCTION DESCRIPTION

Function name	Matrix_Sum
Function prototype	void Matrix_Sum(float *S_1, float *S_2, float *S_3, int n_l, int n_c)
Behavior description	Compute the sum of two matrices
Input	S_1: the first matrix, S_2: the second matrix, S_3: the sum matrix , n_l: the number of lines of S_1, n_c: the number of columns of S_1.
Output	None

4) The Matrix_Trans: it is described in Table V and return the transpose of a matrix.

TABLE V
MATRIX_TRANS FUNCTION DESCRIPTION

Function name	Matrix_Trans
Function prototype	void Matrix_Transp(float* M, float* M_t, int n , int m)
Behavior description	Compute the transpose of a matrix
Input	M: the matrix, M_t: the transpose matrix, n: the number of lines of M, m: the number of columns of M.
Output	None

5) The Prod_Vect_Sca : it is described in Table VI and it allows the multiplication of a vector with a scalar.

TABLE VI
PROD_VECT_SCA FUNCTION DESCRIPTION

Function name	Prod Prod_Vect_Sca
Function prototype	void Prod_Vect_Sca(float *V, float *S, float *VS, int n_{11}, int m_{11})
Behavior description	Compute the product of a vector with a scalar
Input	V: the vector, S: the scalar, VS: the result , n_{11}: the number of lines of V, m_{11}: the number of columns of V.
Output	None

6) Matrix_Inverse : This routine returns the resulting inverted matrix using augmented matrix with the Gauss Jordan algorithm. Its parameters are described in table VII.

V. SIMULATION EXAMPLES

The main features of the proposed constrained MPC controller software are illustrated through two examples. All these examples were tested in run mode and executed from the

TABLE VII
MATRIX_INVERSE FUNCTION DESCRIPTION

Function name	Matrix_Inverse
Function prototype	void Matrix_Inverse(float *M, float *Inver, int nlines)
Behavior description	Compute the inverse of a matrix
Input	M: the input matrix, $Inver$: the inverted matrix, nlines: the number of lines of M.
Output	None

flash memory. In order to allow the implementation of such a computationally expensive controller on chip, we propose reducing the frequency of the STM32 CPU to 24MHz while maintaining good performance. This value is the maximum frequency of STM32 discovery which is characterized by a very low price and additional peripherals comparing with the proposed STM32 Keil board (for example: it contains a DAC: Digital to Analog Converter). All the simulations examples were performed on a Core 2 Duo CPU 2.2 GHz/3.00 Go RAM.

Moreover, besides the constrained MPC firmware, the anti windup PID controller was implemented in the STM32 on the Keil tool, using the Takahashi method anti saturation of the integral term [23], [24]. Therefore, the control signal is computed as follows:

$$u(k) = K_p\varepsilon(k) + u_i(k) + u_d(k)$$

where:

$$u_i(k) = \begin{cases} u_{max} - (K_p\varepsilon(k) + u_d(k)) & \text{if } u_0(k) > u_{max} \\ u_{min} - (K_p\varepsilon(k) + u_d(k)) & \text{if } u_0(k) \leq u_{min} \\ u_i(k-1) + K_p\frac{T_s}{T_i}\varepsilon(k-1) & \text{otherwise} \end{cases}$$

in which:

$$u_d(k) = \frac{1}{1+K_dT_s}u_d(k-1) - \frac{K_pN}{1+K_dN},$$
$$u_0(k) = K_p\varepsilon(k) + u_i(k) + K_p\frac{T_s}{T_i}\varepsilon(k-1) + u_d(k),$$
$$\varepsilon(k) = w(k) - y(k),$$
$$T_i = \frac{T_s}{K_i} \quad , \quad K_d = \frac{T_d}{T_s},$$

and $N \geq 5$ and T_s is the sampling time.

In this work, we found that solving the QP problem as in 8 and 9 as it is commented in [14] sometimes gave incorrect results. In fact, in order to overcome this limitation, we re-scaled the QP problem as follows:

$$\min(\frac{1}{2}\Delta U^T\widetilde{Q}\Delta U + \widetilde{c}^T\Delta U) \tag{13}$$

s.t:

$$\widetilde{J}\Delta U \leq \widetilde{g} \tag{14}$$

where : $\widetilde{Q} = \alpha Q$, $\widetilde{c} = \alpha c$, $\widetilde{J} = \beta J$ and $\widetilde{g} = \beta g$, in which α and β are scalar constants. It is observed that the scaling of Q, c, J and g matrices to a range of ± 1 do not change the solution of the original QP problem, however it allows obtaining accurate solutions.

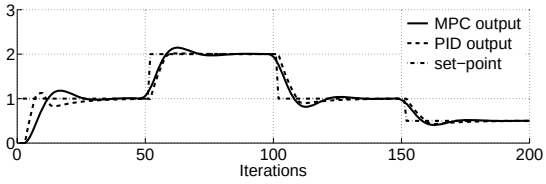
A. A First order plant

The first example is a simple first order system which has the following discrete time model:

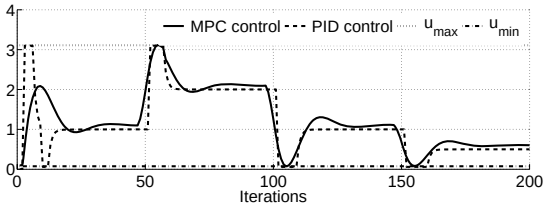
$$y(k) = \frac{0.09516z^{-1}}{1 - 0.9048z^{-1}}u(k)$$

The closed loop simulation results are obtained with the following constrained MPC and anti windup PID parameters: $H_p = 5$, $H_c = 1$, $\lambda_1 = 6$, $K_p = 5.7$, $K_i = 0.1$ and $K_d = 0.25$, starting from: $u(0) = u(1) = 0.1$ and $x_0 = [0\,0]^T$. The PID gains are computed using the Takahashi method. In addition, the following constraints are considered : $0.07 \leq u(k) \leq 3.2$ for the MPC and the PID and $-2 \leq \Delta u(k) \leq 2$ for the MPC.

The Fig. 3 presents the closed loop simulation results of the first order system.



(a)



(b)

Fig. 3. Closed-loop simulation results for the first order plant

B. A second order plant

In this second example, we consider a classical angular positioning system. The system is consisted of a rotating antenna at the origin of the plane, driven by an electric motor. The control problem is to use the input voltage to the motor to rotate the antenna so that it is always points in the direction of a moving object in the plane [3].

The motion of the antenna can be described by the following discrete-time equation obtained from their continuous time counterparts by discretization, using a sampling time of $0.1s$ and Euler's first order approximation for the derivative and based on the observer canonical form:

$$x(k+1) = \begin{bmatrix} \theta(k+1) \\ \dot{\theta}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ 0 & 0.9 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0.0787 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k)$$

The closed loop simulation results are obtained with the following constrained MPC and anti windup PID parameters:

$H_p = 6$, $H_c = 1$, $\lambda_1 = 0.1$, $K_p = 7.58$, $K_i = 0.1$ and $K_d = 0.12$ starting from: $u(0) = u(1) = 0.1$ and $x_0 = [0\ 0]^T$.

In Fig. 4, the closed loop simulations of the angular positioning process using the constrained MPC and the PID anti windup methods is presented where the following constraints are considered: $0 \leq u(k) \leq 30$ for both methods and $-2 \leq \Delta u(k) \leq 2$ for the MPC method.

However, Fig. 5 shows the closed loop simulations of the angular positioning process using the constrained MPC and the PID anti windup methods where only constraints on the control deviation signal are considered as follows: $-0.7 \leq \Delta u(k) \leq 0.7$ for the first simulation (output1, control1 and $\Delta u1$) and $-0.35 \leq \Delta u(k) \leq 0.35$ for the second simulation (output2, control2 and $\Delta u2$).

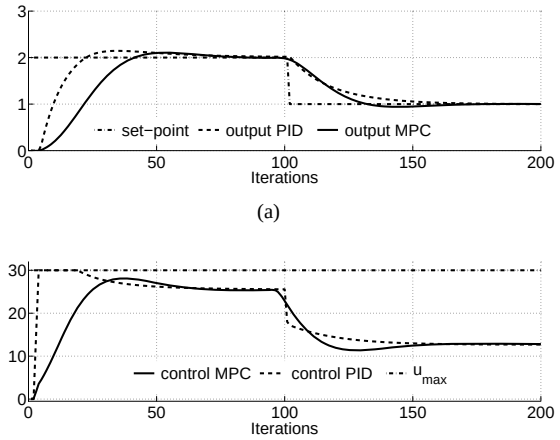


Fig. 4. Closed-loop simulation results for angular positioning process: constraints on control

VI. Interpretations and Results

Figs. 3, 4 and 5 show that the proposed software constrained MPC contoller and the anti windup PID based on STM32 successfully control both processes with a good set-point tracking.

TABLE VIII
Performance Comparison between PID and MPC controller

	PID anti windup		MPC controller	
	1^{st} example	2^{nd} example	1^{st} example	2^{nd} example
The average execution time per sample (μs)	3.5	13.214	316.554	785
The flash memory code size (KBytes)	5.956	7.34	8.858	8.884

Although based on table VIII, the execution time of the on-line algorithm of the anti windup PID is less than that of the MPC, the last method has more parameters such as λ_1, H_c and H_p which allow controlling the performances of the closed loop response. Indeed, from the simulation results of Figs. 3 and 4, we notice that the constrained MPC has the advantage

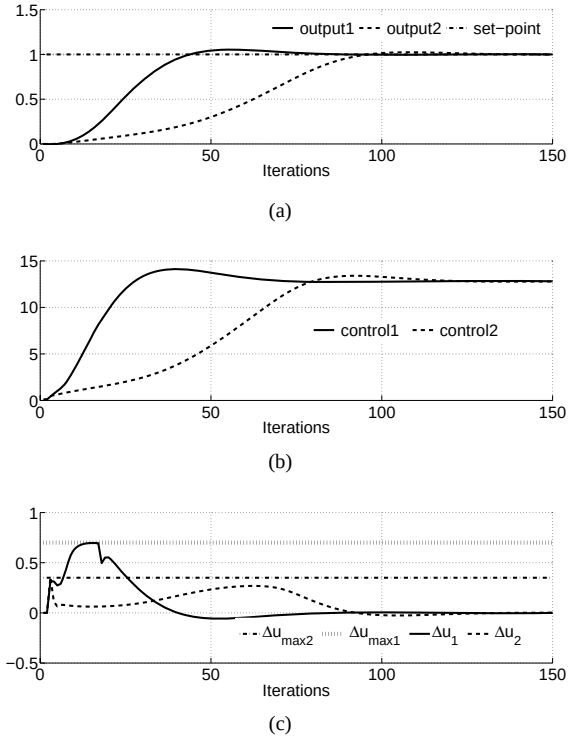


Fig. 5. Closed-loop simulation results for angular positioning process: constraints on control deviation

of predicting the behavior of the output with respect to the set-point changes. In addition Fig. 5, shows that when we reduce the control deviation constraints, the response of the MPC is more sluggish.

Tables IX and X list the time performance of the proposed constrained MPC software. In the first table the control horizon is fixed at 1. However, for the second one the prediction horizon is equal to 4.

TABLE IX
Performance study for Hc=1

The prediction horizon (Hp)	2	4	6	8
The average execution time per sample (ms)	0.766	0.776	0.785	0.795

TABLE X
Performance study for Hp=4

The control horizon (Hc)	1	2	3	4
The average execution time per sample (ms)	0.766	3.626	10.586	234.468

Based on these tables, it can be seen that when the prediction horizon grows higher, the computation time of the on-line MPC algorithm increases slightly. However, this increase is considerably clear when the control horizon rises. This can be explained by the computation each iteration of some matrices in which their dimensions depend on the control horizon such as: J, Λ and T.

VII. Conclusion

Two frameworks for embedding constrained model predictive control and PID anti windup controller on a performed STM32 microcontroller have been provided. The STM32 Keil starter kit was used to implement both firmwares. Two examples were used to test the proposed software performances. Hence, an efficient implementation of this code yields a low computational burden with a high speed. Indeed based on the simulation results, we noticed that the proposed softwares control successfully the processes with a good set-point tracking. A comparison between both softwares has also been done. Although the low computational burden of the PID anti windup software comparing to this of the constrained MPC, the last one gives better control performances such as output prediction and less control signal oscillation. This comparison results should allow the use of MPC to be pioneered in an increasingly wide range of process industries where the computational load has been considered too great and encourage to implement such control method in microcontrollers. There are still much detailed analysis and tests to be done, which should handle multivariable systems and a power consumption study in all possible low power modes has to be investigated.

References

[1] E. Camacho and C. Bordons, *Model Predictive Control*. London: Springer, 2004.
[2] P. Campo and M. Morari, "Robust model predictive control," in *Proceedings of American Control Conference*, 1987, pp. 1021–1026.
[3] M. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.
[4] Z. Wan and M. Kothare, "Robust output feedback model predictive control using linear matrix inequalities," *Journal of Process Control*, vol. 12, pp. 763–774, 2001.
[5] Y. Wang and J. Rawlings, "A new robust model predictive control method 1: theory and computation," *Journal of Process Control*, vol. 14, pp. 231–247, 2002.
[6] A. Casavola, D. Famularo, and G. Franze, "Robust constrained predictive control of uncertain norm-bounded linear systems," *Automatica*, vol. 40, pp. 1865–1876, 2004.
[7] G. Pannochia, "Robust model predictive control with guaranteed set point tracking," *Journal of Process Control*, vol. 14, pp. 927–937, 2004.
[8] T. Alamo, D. Ramirez, and E. Camacho, "Efficient implementation of constrained min-max model predictive control with bounded uncertainties: a vertex rejection approach," *Journal of Process Control*, vol. 15, pp. 149–158, 2005.
[9] A. Kheriji, F. Bouani, and M. Ksouri, "Efficient implementation of constraind robust model predictive control using a state space model," in *Proceedings of International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2010.
[10] A. Kheriji, F. Bouani, and M.Ksouri, "Ggp approach to solve non convex min-max robust model predictive controller for a class constrained mimo systems," in *Proceedings of the International Workshop on Symbolic and Numerical Methods, Modeling and applications to circuit design, (SM2ACD) CEDA competition*, 2010.
[11] A. Kheriji, F. Bouani, and M. Ksour, "A ggp approach to solve non convex min-max predictive controller for a class of constrained mimo systems described by state-space models," *International Journal of Control Automations and Systems (IJCAS)*, vol. 9, no. 3, 2011, will appear in June.
[12] S. Karacan, H. Hapoglu, and M. Alpbaz, "Generalized predictive control to a packed distillation column for regulatory problems," in *European Symposium on Computer Aided Process Engineering*, 1998.
[13] G. D. Nicolao, L. Magni, and R. Scattolini, "Robust predictive control of systems with uncertain impulse response," *Automatica*, vol. 32, no. 10, pp. 1475–1479, 1996.
[14] K. Ling, S. Yue, and J. Maciejowski, "A fpga implementation of model predictive control," in *Proceedings of American Control Conference*, 2006.
[15] K. Ling, B. Wu, and J. Maciejowski, "Embedded model predictive control (mpc) using a fpga," in *Proceedings of the 17th World Congress: The International Federation of Automatic Control, Seoul, Korea*, 2008.
[16] U. R. Y. Jayaraman, "Fpga implementation of predictive control strategy for power factor correction," in *Proceedings of World Academy of Science, Engineering and Technology*, 2008.
[17] G. Palomo, K. Hilton, and J. Rossiter, "Predictive control implementation in a plc using the iec 1131.3 programming standard," in *Proceedings of American Control Conference*, 2009.
[18] A. K. Abbes, F. Bouani, and M. Ksouri, "A microcontroller implementation of model predictive control," in *Proceedings of International Conference on Computer, Electrical, and Systems Sciences, and Engineering, WASET*, 2011.
[19] P. Lascaux and R. Thodor, *Analyse numerique matricielle appliquee l'art de l'ingenieur, tome 1*. Dunod, 2004.
[20] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of Optimization Theory and Applications*, vol. 99, no. 3, pp. 723–757, 1998.
[21] S. Wright, "Applying new optimization algorithms to model predictive control," in *Fifth International Conference on Chemical Process Control*, 1997.
[22] STMicroelectronics, "Datasheet stm32f103x8, stm32f103xb," 2009. [Online]. Available: http://www.st.com/internet/mcu/family/141.jsp
[23] A. Voda and S. Gentil, *Regulateur PID analogique et numeriques*. Technique de l'ingenieur, 1999.
[24] S. Sung, J. Lee, and I. Lee, *Process Identification and PID Control*. John Wiley and Sons (Asia), 2009.