

Fast Analytical Model Predictive Controllers and Their Implementation for STM32 ARM Microcontroller

Patryk Chaber  and Maciej Ławryńczuk 

Abstract—This paper describes fast versions of model predictive control (MPC) algorithms and their practical implementation using the STM32 ARM microcontroller. Two MPC algorithms are considered: Dynamic Matrix Control (DMC) and Generalized Predictive Control (GPC). Computationally efficient Fast DMC (FDMC) and Fast GPC (FGPC) algorithms are derived in which the values of the manipulated variables are calculated from uncomplicated explicit formulas. To demonstrate effectiveness of the discussed algorithms, two emulated benchmark processes and a laboratory servo are considered. The influence of some tuning parameters on algorithms' calculation time is studied. It is shown that for short sampling periods the classical DMC and GPC algorithms fail to work since calculations last longer than the sampling period, which is unacceptable. Conversely, the FDMC and FGPC algorithms require only hundreds or tens of microseconds and single microseconds, respectively. Hence, proposed algorithms may be used for very fast processes, with very short sampling time.

Index Terms—Microcontrollers, model predictive control.

I. INTRODUCTION

MODEL predictive control (MPC) refers to a family of advanced control algorithms in which the values of process manipulated variables are repeatedly calculated online, at each sampling instant, in such a way that future control errors are minimized [1], [2]. Due to such an objective, the MPC algorithms give very good control quality, in particular for multiple-input multiple-output (MIMO) systems. As a result, various types of MPC algorithms have been used for years in practice in large-scale industrial applications, e.g., in chemical and petrochemical engineering, food processing, and paper industry [3]. Typically, in such applications time constants are relatively long, which implies that the sampling time of MPC may be long, typically of the order of seconds or tens of seconds. Programmable logic controllers are used for implementation of MPC algorithms in process control.

Manuscript received July 25, 2018; revised October 8, 2018 and November 29, 2018; accepted December 30, 2018. Date of publication January 14, 2019; date of current version August 1, 2019. Paper no. TII-18-1926. (Corresponding author: Patryk Chaber.)

The authors are with the Institute of Control and Computation Engineering, Warsaw University of Technology, Warsaw 00-665, Poland (e-mail: P.Chaber@ia.pw.edu.pl; M.Lawrynczuk@ia.pw.edu.pl).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TII.2019.2893122

Thanks to the availability of fast microcontrollers, it is possible to develop MPC algorithms for fast dynamic systems (embedded systems). Unlike classical process control applications, they require short sampling time, shorter than one second, typically of millisecond order. Example applications of fast MPC include: cranes [4], helicopter models [5], car models [6], active vibration attenuation [7], and combustion engines [8]. MPC have been also used in fast electrical engineering systems: e.g., power converters [9], synchronous machines in automotive electrical traction drives [10], induction motors [11], active rectifiers [12], and inverters [13].

In many embedded applications of MPC online optimization methods are used, the same as those used in slow industrial process control systems, i.e., quadratic optimization [5]–[7], nonlinear gradient optimization [4], [14], and heuristic optimization [8]. The mentioned approaches are computationally demanding: the vector of decision variables is not calculated analytically from a given formula (but in numerous iterations) and calculations may last long. The literature also reports implementations of explicit (analytical) MPC algorithms based on linear models, typically state-space ones and difference equations, step response models are not considered. In such cases online optimization is not necessary, the decision variables are calculated from a formula, but usually excessive decision variables are calculated [15]–[23]. That is why very short prediction and control horizons must be used, e.g., 5 and 2, respectively, in [15] or 7 and 2, respectively, in [16].

Many publications do not analyze computational time necessary to perform online calculations, it is only shown that the MPC algorithm works for chosen parameters (horizons), e.g., [18], [20]–[22], and [24]. Some works report calculation times, but only the classical versions of MPC are considered and only for short horizons [15], [16]. Unfortunately, the influence of the sampling period on computational burden is not studied [15], [19], [23], [24], but this issue is important since shortening of the sampling time requires lengthening of horizons. Typically, quite moderate sampling periods are used, e.g., 100 ms in [15], [16], and [21], 20 ms in [22] (in the latter case the horizons are 4 and 1, respectively).

This paper is concerned with MPC algorithms developed with maximal computational efficiency in mind. Two MPC algorithms are considered: Dynamic Matrix Control (DMC) and Generalized Predictive Control (GPC). Both use linear

models. A step response model and a difference equation are used for prediction in DMC and GPC, respectively. At first computationally efficient Fast DMC (FDMC) and Fast GPC (FGPC) algorithms are derived in which the values of the manipulated variables are calculated from uncomplicated explicit formulas. Next, practical implementation of the FDMC and FGPC algorithms using the STM32 ARM microcontroller is discussed. Results of experiments for two emulated benchmark processes and a laboratory servo are reported. In particular, the influence of some tuning parameters on MPC algorithms' calculation time is studied. In contrast to many works mentioned in the previous paragraphs earlier, calculation time of the fast FDMC and FGPC algorithms are compared with those of the classical DMC and GPC control schemes, for very broad changes of parameters.

II. DESCRIPTION OF MPC ALGORITHMS

A. MPC Problem Formulation

MIMO processes with n_u manipulated variables (process inputs) and n_y controlled variables (process outputs) are considered. In this paper, two notation methods are used: vectors $u = [u_1 \dots u_{n_u}]^T$, $y = [y_1 \dots y_{n_y}]^T$ and scalars u_n , where $n = 1, \dots, n_u$, and y_m , where $m = 1, \dots, n_y$. At each sampling instant, $k = 0, 1, \dots$, the MPC algorithm calculates online the increments of the future manipulated variables [1], [2] (a vector of length $n_u N_u$) as follows:

$$\Delta u(k) = \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k + N_u - 1|k) \end{bmatrix} \quad (1)$$

where N_u is the control horizon. The increments for the future sampling instant $k + p$ calculated at the current instant k are denoted by $\Delta u(k + p|k)$. They are defined as

$$\Delta u(k + p|k) = \begin{cases} u(k|k) - u(k-1) & \text{for } p = 0 \\ u(k + p|k) - u(k + p - 1|k) & \text{for } p \geq 1 \end{cases} \quad (2)$$

There are N_u future changes of the manipulated variables and next they are constant, i.e., $\Delta u(k + p|k) = 0$ for $p \geq N_u$. The objective of MPC is to find the future increments (1) which lead to the best possible control quality defined as the minimal difference between the set-points of the controlled variables and their future predicted values (calculated using a dynamic model), taken into account over the prediction horizon N . Let the vectors of set-points and predicted values for the future sampling instant $k + p$ known at the instant k be $y^{sp}(k + p|k)$ and $\hat{y}(k + p|k)$, respectively, $p = 1, \dots, N$. MPC may also minimize excessive changes of the manipulated variables. Hence, the minimized MPC cost-function is

$$J(k) = \sum_{p=1}^N \|y^{sp}(k + p|k) - \hat{y}(k + p|k)\|_{M_0}^2 + \sum_{p=0}^{N_u-1} \|\Delta u(k + p|k)\|_{\Lambda_0}^2 \quad (3)$$

where $M_0 = \text{diag}(\mu_1, \dots, \mu_{n_y})$ and $\Lambda_0 = \text{diag}(\lambda_1, \dots, \lambda_{n_u})$ are weighting matrices of dimensionality $n_y \times n_y$ and $n_u \times n_u$, respectively. The parameters $\mu_n \geq 0$ prioritize the predicted control errors of the consecutive controlled variables, $n = 1, \dots, n_y$. The weights $\lambda_n > 0$ penalize excessive increments of the consecutive manipulated variables, $n = 1, \dots, n_u$. Although at each sampling instant k as many as $n_u N_u$ future increments of the manipulated variables (1) are calculated, only the increments for the current sampling instant are applied to the process, i.e., $u(k) = \Delta u(k|k) + u(k-1)$. At the next sampling instant, $k + 1$, the whole procedure is repeated.

The MPC cost-function (3) may be compactly expressed as

$$J(k) = \|y^{sp}(k) - \hat{y}(k)\|_M^2 + \|\Delta u(k)\|_{\Lambda}^2 \quad (4)$$

where the set-point and predicted trajectories

$$y^{sp}(k) = \begin{bmatrix} y^{sp}(k + 1|k) \\ \vdots \\ y^{sp}(k + N|k) \end{bmatrix}, \quad \hat{y}(k) = \begin{bmatrix} \hat{y}(k + 1|k) \\ \vdots \\ \hat{y}(k + N|k) \end{bmatrix} \quad (5)$$

are vectors of length $n_y N$. In typical process control tasks the set-points are constant over the prediction horizon, i.e., $y^{sp}(k) = y^{sp}(k + 1|k) = \dots = y^{sp}(k + N|k)$. The weighting matrices $M = \text{diag}(M_0, \dots, M_0)$ and $\Lambda = \text{diag}(\Lambda_0, \dots, \Lambda_0)$ are of dimensionality $n_y N \times n_y N$ and $n_u N_u \times n_u N_u$, respectively.

B. Classical DMC Algorithm

A discrete-time step response model is used in the DMC algorithm [25]. The model shows reactions of the consecutive process outputs to unit steps applied to the consecutive process inputs. Assuming that the process is stable, with no integration, the process outputs stabilize after D sampling instants, which is called the horizon of process dynamics. The step response model is represented by D matrices of dimensionality $n_y \times n_u$

$$S_p = \begin{bmatrix} s_p^{1,1} & \dots & s_p^{1,n_u} \\ \vdots & \ddots & \vdots \\ s_p^{n_y,1} & \dots & s_p^{n_y,n_u} \end{bmatrix} \quad (6)$$

for $p = 1, \dots, D$. The step response coefficient $s_p^{m,n}$ is a measured value of the m th process output caused by a unit step of the n th input occurring at the sampling instant $k = 0$ [2]. The measurements are recorded over the whole horizon of dynamics, i.e., for $p = 1, \dots, D$. The procedure is repeated for all input-output channels, i.e., for $m = 1, \dots, n_y$, $n = 1, \dots, n_u$. If a unit step is impossible, the measurements must be scaled. It may be proven [2] that the vector of predicted controlled variables is

$$\hat{y}(k) = G \Delta u(k) + y(k) + G^p(k) \Delta u^p(k) \quad (7)$$

where the vectors

$$y(k) = \begin{bmatrix} y(k) \\ \vdots \\ y(k) \end{bmatrix}, \quad \Delta u^p(k) = \begin{bmatrix} \Delta u(k-1) \\ \vdots \\ \Delta u(k-(D-1)) \end{bmatrix} \quad (8)$$

are of length $n_y N$ and $n_u(D-1)$, respectively. The matrices

$$\mathbf{G} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_N & \mathbf{S}_{N-1} & \dots & \mathbf{S}_{N-N_u+1} \end{bmatrix} \quad (9)$$

and

$$\mathbf{G}^p = \begin{bmatrix} \mathbf{S}_2 - \mathbf{S}_1 & \dots & \mathbf{S}_D - \mathbf{S}_{D-1} \\ \vdots & \ddots & \vdots \\ \mathbf{S}_{N+1} - \mathbf{S}_1 & \dots & \mathbf{S}_{N+D-1} - \mathbf{S}_{D-1} \end{bmatrix} \quad (10)$$

are of dimensionality $n_y N \times n_u N_u$ and $n_y N \times n_u(D-1)$, respectively. Using the DMC prediction equation (7), the minimized cost-function (4) becomes

$$J(k) = \|\mathbf{y}^{\text{sp}}(k) - \mathbf{G}\Delta\mathbf{u}(k) - \mathbf{y}(k) - \mathbf{G}^p(k)\Delta\mathbf{u}^p(k)\|_M^2 + \|\Delta\mathbf{u}(k)\|_\Lambda^2. \quad (11)$$

Since the cost-function (11) is quadratic with respect to $\Delta\mathbf{u}(k)$, the optimal solution vector is

$$\Delta\mathbf{u}(k) = \mathbf{K}(\mathbf{y}^{\text{sp}}(k) - \mathbf{y}(k) - \mathbf{G}^p\Delta\mathbf{u}^p(k)) \quad (12)$$

where a matrix of dimensionality $n_u N_u \times n_y N$ is

$$\mathbf{K} = (\mathbf{G}^T \mathbf{M} \mathbf{G} + \Lambda)^{-1} \mathbf{G}^T \mathbf{M}. \quad (13)$$

C. Fast DMC Algorithm (FDMC)

In the classical DMC approach all n_u manipulated variables over the control horizon, N_u , are calculated from (12), i.e., the whole vector $\Delta\mathbf{u}(k)$ (1). It is also possible to calculate n_u values of the manipulated variables only for the current sampling instant, k , i.e., the vector $\Delta\mathbf{u}(k|k)$. Let the consecutive entries of the matrix $\mathbf{K}$ (13) be denoted by $K_{i,j}$, for $i = 1, \dots, n_u N_u$, $j = 1, \dots, n_y N$. Let the consecutive entries of the matrix $\mathbf{G}^p$ (10) be denoted by $G_{i,j}^p$, for $i = 1, \dots, n_y N$, $j = 1, \dots, n_u(D-1)$. Assuming that the set-points are constant over the prediction horizon, from (12) and (13) the analytical FDMC control law is derived as

$$\Delta u_r(k|k) = \sum_{m=1}^{n_y} K_{r,m}^e (y_m^{\text{sp}}(k) - y_m(k)) + \sum_{n=1}^{n_u} \sum_{j=1}^{D-1} K_{r,n}^u(j) \Delta u_n(k-j) \quad (14)$$

for $r = 1, \dots, n_u$. The coefficients are

$$K_{r,m}^e = \sum_{p=1}^N K_{r,(p-1)n_y+m} \quad (15)$$

for $r = 1, \dots, n_u$, $m = 1, \dots, n_y$ and

$$K_{r,n}^u(j) = - \sum_{p=1}^{n_y N} K_{r,p} G_{p,(j-1)n_u+n}^p \quad (16)$$

for $r = 1, \dots, n_u$, $n = 1, \dots, n_u$, $j = 1, \dots, D-1$.

The coefficients $K_{r,m}^e$ and $K_{r,n}^u(j)$ are calculated offline for the chosen tuning parameters $\mu_1, \dots, \mu_{n_y}$ and $\lambda_1, \dots, \lambda_{n_u}$. The

calculated increments $\Delta u_r(k|k)$ may be projected onto the admissible set determined by the constraints.

Complexity of the computationally efficient FDMC control law used online (14) depends only on the horizon of dynamics, D , and the number of process inputs and outputs, n_u and n_y , whereas it is independent of the prediction and control horizons, N and N_u . The horizons are taken into account during offline calculation of the control law coefficients.

D. Classical GPC Algorithm

The rudimentary GPC algorithm [26] uses Diophantine equations for prediction calculation. In this paper, an original, much simpler formulation is presented, which leads to a computationally efficient control law, similar to that of the DMC algorithm characterized by (14). The process model is

$$\mathbf{A}(q^{-1})\mathbf{y}(k) = \mathbf{B}(q^{-1})\mathbf{u}(k) \quad (17)$$

where the entries of the matrices

$$\mathbf{A}(q^{-1}) = \begin{bmatrix} A_{1,1}(q^{-1}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & A_{n_y,n_y}(q^{-1}) \end{bmatrix} \quad (18)$$

$$\mathbf{B}(q^{-1}) = \begin{bmatrix} B_{1,1}(q^{-1}) & \dots & B_{1,n_u}(q^{-1}) \\ \vdots & \ddots & \vdots \\ B_{n_y,1}(q^{-1}) & \dots & B_{n_y,n_u}(q^{-1}) \end{bmatrix} \quad (19)$$

are the polynomials in the backward shift operator q^{-1}

$$A_{m,m}(q^{-1}) = 1 + a_1^m q^{-1} + \dots + a_{n_A}^m q^{-n_A} \quad (20)$$

for $m = 1, \dots, n_y$ and

$$B_{m,n}(q^{-1}) = b_1^{m,n} q^{-1} + \dots + b_{n_B}^{m,n} q^{-n_B} \quad (21)$$

for $m = 1, \dots, n_y$, $n = 1, \dots, n_u$. It may be proven [2] that the vector of predicted controlled variables is

$$\hat{\mathbf{y}}(k) = \mathbf{G}\Delta\mathbf{u}(k) + \mathbf{y}^0(k) \quad (22)$$

where the matrix $\mathbf{G}$ is the same as in the DMC algorithm (9). The step response coefficients may be easily calculated from the model (17). The vector of length $n_y N$

$$\mathbf{y}^0(k) = \begin{bmatrix} y^0(k+1|k) \\ \vdots \\ y^0(k+N|k) \end{bmatrix} \quad (23)$$

represents the free trajectory, i.e., the reaction of the model to the past. Using the GPC prediction equation (22), minimization of the MPC cost-function (11) yields the increments

$$\Delta\mathbf{u}(k) = \mathbf{K}(\mathbf{y}^{\text{sp}}(k) - \mathbf{y}^0(k)) \quad (24)$$

where the matrix $\mathbf{K}$ is defined by (13). The only difference between DMC and GPC control laws, (12) and (24), respectively, is the way the free trajectory is described.

E. Fast GPC Algorithm (FGPC)

Derivation of the computationally efficient control law for the FGPC algorithm is more complicated than in the case of

the FDMC algorithm. For the model defined by (17)–(21), the predicted values of the controlled variables for the sampling instant $k + p$ are calculated at the instant k as the sum of the model output, $y_m(k + p|k)$, and an estimation of the disturbance acting on the process output, $d_m(k)$, which leads to

$$\hat{y}_m(k + p|k) = y_m(k + p|k) + d_m(k) \quad (25)$$

for all process outputs, $m = 1, \dots, n_y$. The estimated disturbances are found as the differences between the currently measured process outputs and their values calculated from the model [2]. The consecutive predictions are

$$\begin{aligned} \hat{y}_m(k + p|k) = & \sum_{n=1}^N \left[\sum_{i=1}^{p-1} g_{1-p+i}^{m,n}(p) \Delta u_n(k + p - 1 - i|k) \right. \\ & \left. + \sum_{i=1}^{n_B-1} g_i^{m,n}(p) \Delta u_n(k - i) \right] \\ & + \sum_{i=0}^{n_A} f_i^m(p) y_m(k - i) \end{aligned} \quad (26)$$

for all $p = 1, \dots, N$, $m = 1, \dots, n_y$. The constant coefficients are calculated from model parameters. For $p = 1$, $g_i^{m,n}(1) = b_{i+1}^{m,n}$ for all $i = 0, \dots, n_B - 1$ whereas for $p = 2, \dots, N$

$$g_i^{m,n}(p) = \begin{cases} g_{i+1}^{m,n}(p-1), & i = 1 - p, \dots, -1 \\ f_0^m(p-1) g_i^{m,n}(1) \\ \quad + g_{i+1}^{m,n}(p-1), & i = 0, \dots, n_B - 2 \\ f_0^m(p-1) g_i^{m,n}(1), & i = n_B - 1 \end{cases} \quad (27)$$

For $p = 1$ one has

$$f_i^m(1) = \begin{cases} 1 - a_1^m, & i = 0 \\ a_i^m - a_{i+1}^m, & i = 1, \dots, n_A - 1 \\ a_i^m, & i = n_A \end{cases} \quad (28)$$

whereas for $p = 2, \dots, N$

$$f_i^m(p) = \begin{cases} f_0^m(p-1) f_i^m(1) \\ \quad + f_{i+1}^m(p-1), & i = 0, \dots, n_A - 1 \\ f_0^m(p-1) f_i^m(1), & i = n_A \end{cases} \quad (29)$$

The free trajectory is derived from the GPC prediction equation (26), assuming only the reaction of the model to the past, i.e., all increments $\Delta u(k + p|k) = 0$ for $p \geq 0$, which gives

$$\begin{aligned} y_m^0(k + p|k) = & \sum_{n=1}^N \sum_{i=1}^{n_B} e_i^{m,n}(p) u_n(k - i|k) \\ & + \sum_{i=0}^{n_A} f_i^m(p) y_m(k - i) \end{aligned} \quad (30)$$

where the constant coefficients are

$$e_i^{m,n}(p) = \begin{cases} 0, & i = 1 \text{ and } n_B = 1 \\ g_i^{m,n}(p), & i = 1 \text{ and } n_B > 1 \\ g_i^{m,n}(p) - g_{i-1}^{m,n}(p), & i = 2, \dots, n_B - 1 \\ \quad \text{and } i < n_B \text{ and } n_B > 1 \\ -g_{i-1}^{m,n}(p), & i = n_B \text{ and } n_B > 1 \end{cases} \quad (31)$$

The values of the manipulated variables for the current sampling instant, k , are derived from (24) and (30), which results in the analytical FGPC control law

$$\begin{aligned} \Delta u_r(k|k) = & \sum_{m=1}^{n_y} K_{r,m}^e (y_m^{sp}(k) - y_m(k)) \\ & + \sum_{n=1}^{n_u} \sum_{j=1}^{n_B} K_{r,n}^u(j) u_n(k - j) + \sum_{m=1}^{n_y} \sum_{j=1}^{n_A} K_{r,m}^y(j) y_m(k - j) \end{aligned} \quad (32)$$

for $r = 1, \dots, n_u$. The coefficients are

$$K_{r,m}^e = \sum_{p=1}^N K_{r,(p-1)n_y+m} - \sum_{l=1}^{n_y} K_{r,l}^y(0) \quad (33)$$

for $r = 1, \dots, n_u$, $m = 1, \dots, n_y$

$$K_{r,n}^u(j) = - \sum_{p=1}^N \sum_{m=1}^{n_y} K_{r,(p-1)n_y+m} e_j^{m,n}(p) \quad (34)$$

for $r = 1, \dots, n_u$, $n = 1, \dots, n_u$, $j = 1, \dots, n_B$; and

$$K_{r,m}^y(j) = - \sum_{p=1}^N K_{r,(p-1)n_y+m} f_j^m(p) \quad (35)$$

for $r = 1, \dots, n_u$, $m = 1, \dots, n_y$, $j = 1, \dots, n_A$.

The coefficients $K_{r,m}^e$, $K_{r,n}^u(j)$, and $K_{r,m}^y(j)$ are calculated offline for the chosen tuning parameters $\mu_1, \dots, \mu_{n_y}$ and $\lambda_1, \dots, \lambda_{n_u}$. The calculated increments $\Delta u_r(k|k)$ may be projected onto the admissible set determined by the constraints.

Complexity of the computationally efficient FGPC control law (32) depends only on the model order of dynamics, defined by n_A and n_B , and the number of process inputs and outputs, n_u and n_y , respectively, whereas it is independent of the prediction and control horizons, N and N_u .

III. IMPLEMENTATION OF MPC ALGORITHMS USING STM32 MICROCONTROLLER

All MPC algorithms have been implemented using the STM32F746IGT6 microcontroller installed on the Open746I-C development board from Waveshare. The microcontroller is based on the high-performance ARM Cortex-M7 32-bit RISC core. It operates at 216 MHz, which combined with a hardware floating point coprocessor, renders it as a reasonably powerful computation device. The microcontroller also incorporates high-speed embedded memories with a Flash memory of 1 MB, 320 kB of SRAM (including 64 kB of Data TCM RAM for critical real-time data), 1 kB of instruction TCM RAM (for critical

real-time routines). The microcontroller offers numerous peripherals, e.g.,; three 12-bit analogue-digital converters (ADC), two 12-bit digital-analogue converters (DAC), and 15 general purpose timers, communication interfaces (USART, I²C, SPI, and Ethernet). The output and input signals of the controlled processes are represented as voltages, which means that there are constraints on these variables, from 0 to 3.3 V.

The program is implemented in C language. To assure that the sampling period is always constant, it is necessary to precisely measure time. Therefore, the reconfigurable general purpose timer peripheral is used. Every time the timer raises an interrupt, a new discrete time instant starts. Thus, in each consecutive time instant the following scheme is executed:

- 1) the manipulated variables calculated at the previous time instant are applied to the process;
- 2) the measured process output signals are stored in the memory of the microcontroller;
- 3) the new values of the manipulated variables are calculated from (14) or (32);
- 4) the microcontroller waits for a new discrete-time instant and returns to Step 1).

There is a unit delay introduced in this algorithm to compensate for the time of execution of the MPC algorithm, which may vary depending on the number of model parameter. The MPC algorithm (executed in Step 3) itself is very compact, and it consists of only a few lines of code. For example, for the FGPC algorithm the code is

```
for (r=0; r<nu; ++r) {           // r=1:nu
    du[r] = 0.0f;                 //
    for (m=0; m<ny; ++m)         // m=1:ny
        du[r] += Ke[r][m] * (ysp[m][k] - y[m][k]);
    for (n=0; n<nu; ++n)         // n=1:nu
        for (j=0; j<nB; ++j)     // j=1:nB
            du[r] += Ku[r][n][j] * u[n][k-j-1];
    for (m=0; m<ny; ++m)         // m=1:ny
        for (j=0; j<nA; ++j)     // j=1:nA
            du[r] += Ky[r][m][j] * y[m][k-j];
}
```

The above implementation uses the control law defined by (32). Minor differences in indexing results from the fact that arrays in C language are indexed starting from 0, whereas most indices in the formulation of MPC problem starts at 1.

Measurements are obtained constantly using the mechanism of direct memory access, which allows to perform memory transfers without the use of microprocessor. When all measurements are obtained, the microprocessor is used to store it in the array $\underline{y}$. On the other hand, the control signal $\underline{u}$ is applied to the process using the DAC. This is performed once at the beginning of each time instant, and it takes fractions of micro seconds, therefore no additional mechanism is used to reduce microprocessor's load. Additionally, some RC filters are applied to the lines connecting process' and controllers' ADC and DAC. This allows to reduce the measurement noise.

All necessary vectors and matrices are automatically generated offline from a process model and transferred to the microcontroller. Example code for the FGPC algorithm is

which defines: the type of algorithm, the number of inputs and outputs, the model order, and coefficients of the control law.

```
#define ALGORITHM FGPC
#define nu 1
#define ny 1
#define nA 1
#define nB 2
float Ke[nu][ny] = {5.8964375e+00};
float Ku[nu][nu][nB] = {-3.100340e-01, +3.100340e-01};
float Ky[nu][ny][nA] = {+4.187304e+00};
```

To measure calculation time at every sampling instant, another general purpose timer is used. It is reset with the start of new time instant and stopped when calculations are finished. The resolution of measurement is approx. 9.26 ns. All process measurements may be sent to an external PC.

IV. RESULTS OF EXPERIMENTS

There are two objectives of the experiments presented next. First, it is demonstrated that the discussed analytical FDMC and FGPC algorithms may be applied to fast processes, with short sampling periods. Second, the influence of some parameters of the MPC algorithms on the maximal calculation time carried out at a single sampling instant, $T_c^{\max}$, is studied. For DMC and FDMC the influence of the number of process inputs and outputs as well as the length of the horizon of dynamics, D , is considered. For GPC and FGPC the influence of process dimensionality and model order is analyzed. The parameters, which do not have any impact on online computational burden of FDMC and FGPC are: the horizons $N = 50$, $N_u = 50$, all the weights $\mu_n = 1$, and the weights λ_n , which are adjusted to obtain good control quality. T_p denotes the sampling period. In the experimental results presented next the shortest possible sampling periods are used.

Continuous-time models discussed next do not contain the aforementioned artificial delay, which is introduced to compensate for the calculation time of the MPC algorithm. Due to the discrete nature of this delay, it is only considered at the implementation stage, where the discretized models are used—the manipulated and controlled signals are then delayed accordingly without influencing the model itself.

A. MPC of Emulated SISO Process

At first, the FDMC and FGPC algorithms are applied to an emulated (using second STM32F746IGT6 microcontroller board) single-input single-output (SISO) process. It is described by the continuous-time first-order transfer function $G(s) = \frac{1}{0.014s+1}$ (the time constant in seconds). The discrete-time version of this transfer function, discretized with the sampling period T_p , is used to derive the step response model used in the FDMC algorithm and the model used in the FGPC scheme. For process emulation, the same transfer function is also used, but with the sampling instant 100 μ s. The models used in FDMC and FGPC algorithms are theoretically perfectly accurate, i.e., both controller and emulator models are derived from the same continuous transfer function. Unfortunately, measurement noise, bias of measurement modules, and numerical inaccuracies make this model less accurate than it would appear.

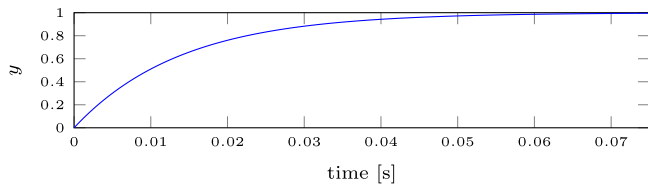


Fig. 1. Step response of the model of emulated SISO process.

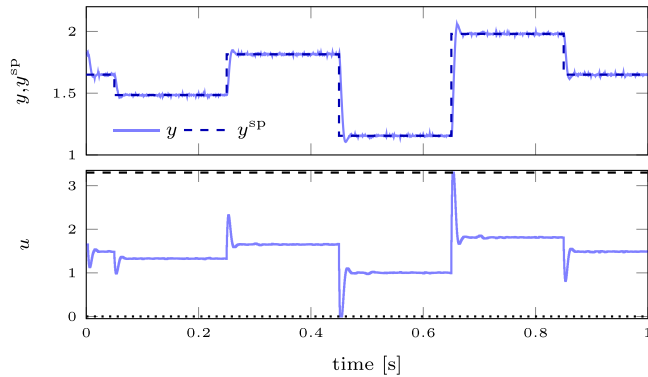


Fig. 2. Emulated SISO process: Results of the experiment using FDMC with $\lambda_1 = 1$ and $T_p = 500 \mu\text{s}$ ($T_c^{\text{max}} = 145.80 \mu\text{s}$).

The emulation configuration allows to minimize influence of external measurement units and actuators since only on-board DAC and ADC are used. Since they are integrated with the core of the microcontroller, time of measurement can be as low as a fraction of a microsecond, and the conversion of digital value to its analogue representation takes single microseconds.

The step response of the model of process is depicted in Fig. 1. It is necessary to notice that the recorded step response is given in a continuous time domain. Based on this representation, discrete time step response coefficients are derived for the chosen sampling time. Fig. 2 shows the obtained results for the FDMC algorithm for an assumed set-point trajectory, the horizon of dynamics is $D = 100$. Even with very short sampling time, $T_p = 500 \mu\text{s}$, the algorithm gives good control quality. The calculation time is $145.8 \mu\text{s}$. It is interesting to note that at the beginning of the experiment the process is not in the expected operating point. The FDMC algorithm overcomes such inaccuracies.

For sampling periods shorter than $500 \mu\text{s}$ it is impossible to finish calculations. For example, the sampling period of $200 \mu\text{s}$ requires a longer horizon of dynamics and increased calculation time, which finally renders this controller too slow for the given sampling period. Assuming that the time of output signal stabilization is approximately 0.05 s , then the horizon can be determined as $D = 50000/T_p$, where T_p is a sampling period expressed in microseconds. This information allows to determine how fast the controller can be, and therefore maximize model quality.

Fig. 3 shows the obtained results for the FGPC algorithm, and the first-order model is used ($n_A = n_B = 1$). The sampling period is set to $100 \mu\text{s}$. The calculation time is $2.17 \mu\text{s}$ compared to $145.8 \mu\text{s}$ in the FDMC scheme. The reason is that typically

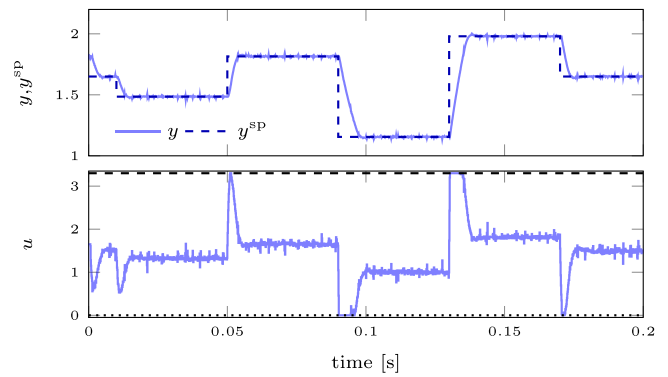


Fig. 3. Emulated SISO process: Results of the experiment using FGPC with $\lambda_1 = 1$ and $T_p = 100 \mu\text{s}$ ($T_c^{\text{max}} = 2.17 \mu\text{s}$).

the horizon of process dynamics, D , is much longer than the order of the dynamic model, characterized by the numbers n_A and n_B . Hence, the FDMC scheme requires as many as D multiplications and summations (14), whereas the FGPC one only $n_A + n_B + 1$ such operations (32).

It is obvious that control quality obtained by the FGPC algorithm is lower than that of the FDMC one. There are a few factors that lead to this phenomenon. First, the sampling period is very low compared to the capabilities of DAC. This is the main cause of low slope rate—rising and falling speed of the output signals is low. Fig. 3 shows that even using maximum allowed control signal, the rising speed of the output signal is constant, even though no constraints other than on maximal and minimal value of control signal are imposed. When $T_p = 500 \mu\text{s}$, the resulting trajectory is almost the same as in the FDMC scheme.

Unfortunately, in the case of the FGPC algorithm the manipulated variable trajectory is not smooth. The reason is that the values of the manipulated variable in the FGPC law (32) depend on $n_A + 1$ past values of the process output, whereas in the FDMC control law (14) only measurements for the current sampling instant are used. Therefore, in FGPC all inaccuracies of the model are further amplified and transferred into the trajectory of the manipulated variable.

It is interesting to compare robustness of FDMC and FGPC. In practice the mismatch between the process and its model used in MPC is inevitable as the properties of the process may change in time. Typically, in order to check robustness, gain of the process is increased as it may result in overshoot, oscillations, and even instability when changes are big. Figs. 4 and 5 depict the obtained results for FDMC and FGPC, respectively. For the sake of comparison, the same parameters ($\lambda_1 = 1$ and $T_p = 500 \mu\text{s}$) are used in both algorithms. Three scenarios are considered for both algorithms: the process gain is increased by 100%, 200%, and 300%. Additionally, the process is affected by an unmeasured output step disturbance of amplitude 0.003 . Both algorithms turn out to be robust to the considered change of process gain and disturbances. Both algorithms are able to control the process in such a way that the changes of the set-point are followed without any steady-state errors. However, there are some differences between the algorithms. There is a noticeable overshoot for FDMC when the process gain is increased by

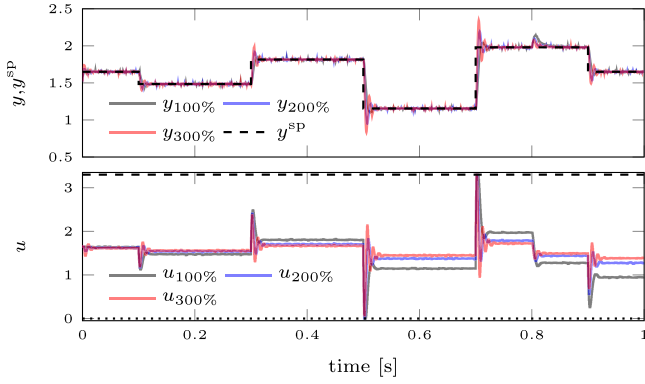


Fig. 4. Emulated SISO process: Results of the experiments using FDMC with $\lambda_1 = 1$ and $T_p = 500 \mu s$ when the process gain is increased by 100%, 200%, and 300%, an additional step disturbance occurs at 0.8 s.

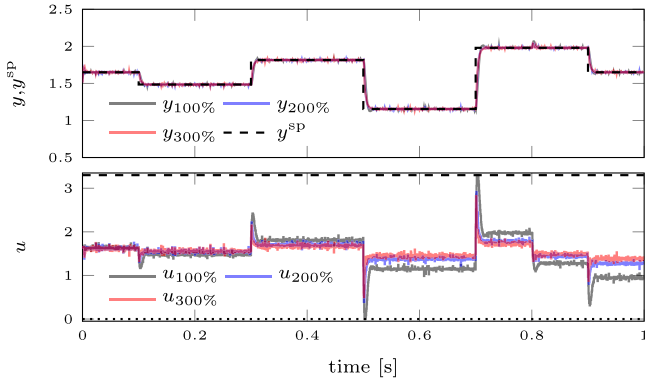


Fig. 5. Emulated SISO process: Results of the experiments using FGPC with $\lambda_1 = 1$ and $T_p = 500 \mu s$ when the process gain is increased by 100%, 200%, and 300%, an additional step disturbance occurs at 0.8 s.

200% and 300% in relation to the model gain. Control quality is much better for FGPC, practically with no overshoot, and compensation of the unmeasured disturbance is faster. It is because the FGPC control law (32), in comparison with the FDMC control law (14), is characterized by much complex dependence on measured past values of the process output signal. Undamped oscillations are obtained when the gain of the process is increased by 700% and 600% for FDMC and FGPC, respectively, in comparison with the model gain. Such huge modeling errors are very unlikely in practice.

B. MPC of Emulated MIMO Process

The second emulated process has two inputs and two outputs. Its continuous-time transfer function matrix is

$$G(s) = \begin{bmatrix} \frac{1}{0.014s+1} & \frac{5}{0.006s+1} \\ \frac{1}{0.01s+1} & \frac{2}{0.008s+1} \end{bmatrix}. \quad (36)$$

The time constants are given in seconds. This model is discretized with the sampling period T_p . For process emulation, the sampling instant is $100 \mu s$.

The step response of the model of process is depicted in Fig. 6. Fig. 7 shows the results of the FDMC algorithm, the horizon of

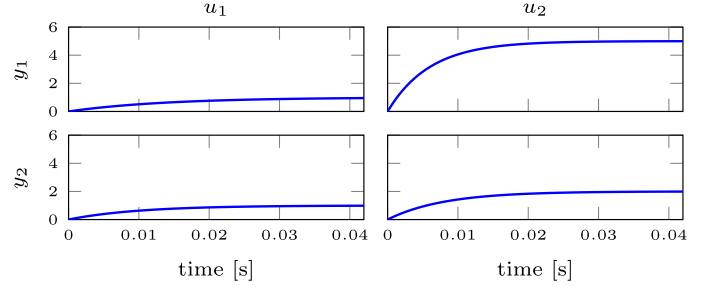


Fig. 6. Step response of the model of emulated MIMO processes.

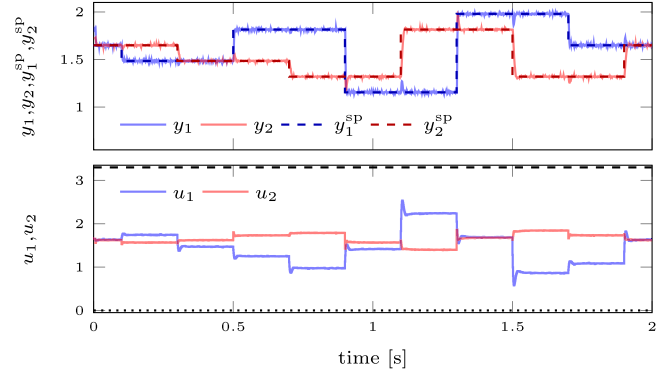


Fig. 7. Emulated MIMO process: Results of the experiment using FDMC with $\lambda_1 = \lambda_2 = 1$ and $T_p = 1000 \mu s$ ($T_c^{\max} = 332.00 \mu s$).

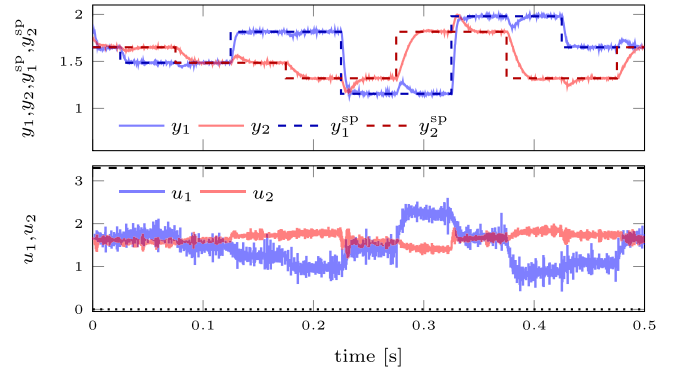


Fig. 8. Emulated MIMO process: Results of the experiment using FGPC with $\lambda_1 = \lambda_2 = 100$ and $T_p = 250 \mu s$ ($T_c^{\max} = 7.17 \mu s$).

dynamics is $D = 50$. The sampling period $T_p = 1 \text{ ms}$ is used, which is the shortest possible. Control quality is good, both process outputs quickly reach their set-points. The manipulated variables are smooth and no significant noise is visible. The calculation time is $332.00 \mu s$. Shorter sampling periods lead to an increase of the calculation time, which is longer than the sampling period.

Fig. 8 shows the results of the FGPC algorithm, the second-order model is used ($n_A = n_B = 2$). The sampling period is $T_p = 250 \mu s$, which is the shortest possible. The calculation time is $7.17 \mu s$ (compared to $332.00 \mu s$ for the FDMC scheme). The trajectories are almost as fast as the FGPC algorithm for the SISO process. However, the trajectories of the manipulated variables are not as smooth as in the case of the MIMO FDMC

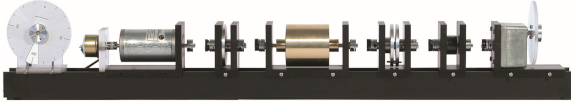


Fig. 9. Side view of modular servo by INTECO.

scheme or the SISO FDMC one. High rate of changes in the manipulated variables is correlated with measurement noise. Its influence is significant since for lower values of the tuning coefficients λ_n , the controller may be even unstable. The increase of these coefficients also slows down the time at which the output signal reaches the set-point. It is also one of the major factors that limit the sampling period of this controller (it could be much shorter taking into account only the calculation time).

C. MPC of Laboratory Servo

Finally, the FDMC and FGPC algorithms are used to control a laboratory modular servo produced by the INTECO company (<http://www.inteco.com.pl>) shown in Fig. 9. Different classical controllers have been developed for the servo, e.g., PI controllers, fuzzy PI controllers, sliding mode controllers [27], LQG controllers [28], and fractional order controllers [29]. Its main parts are: a potentiometer, which allows to set new set-point value; a tachometer, used to determine the speed of the servo shaft; the servo itself; a damping module; and an inertia module (additional mass). The input signal of the process is the width of the pulsewidth modulation (PWM) signal. It is therefore limited to values from 0% to 100% (0% stops the servo, 100% gives full speed). The output value of this process is the voltage measured by the tachometer. This value is proportional to the speed of the servo, but because of its low signal to noise ratio, the values are given in volts. The voltage range of tachometer is from 0 to 10 V, assuming it spins in only one direction. This process is used in the basic configuration, i.e., the tachometer and the servo only, with no additional inertia module (to test fast MPC algorithms). The speed of the shaft may be measured by a tachometer or calculated based on data from rotary encoders. Although the second approach provides far less noisy results, the resolution of encodes is not high enough for controllers with so low sampling period. Hence, the tachometer is used, even though it yields noisy measurements, the frequency of which depends on the speed of the servo and is problematic to filter.

To connect the servo and the ST microcontroller an additional module is required for converting the voltage levels between them. The ST microcontroller uses the range 0–3.3 V, the servo's digital signals (like its input PWM signal) are in the range 0–24 V, and analogue signals (like output signal from tachometer) are in the –10 to 10 V range. To cope with this inconsistency, a voltage divider using three resistors is used to convert analogue signals from –10 to 10 V range to 0–3.3 V, and for the conversion of digital signal going from the controller to the servo, the amplifier based on MOSFET transistors is used. The digital signal fully propagates through the circuit, maintaining its shape, and which is the most important, the width of the PWM signal. The RC filters are used to smooth the measured

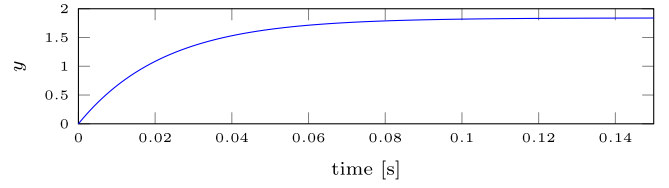


Fig. 10. Step response of the model of servo.

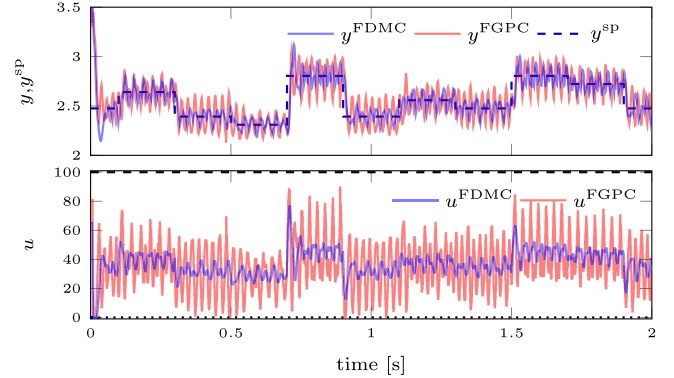


Fig. 11. Servo: Results of the experiment using FDMC and FGPC, both with $\lambda_1 = 10$ and $T_p = 1000 \mu s$ ($T_c^{max} = 584.20 \mu s$ for FDMC and $T_c^{max} = 11.83 \mu s$ for FGPC).

signal from the tachometer. The continuous-time transfer function of the experimentally obtained model is $G(s) = \frac{1.841}{0.02247s+1}$ (the time constant in seconds). This model is discretized with the sampling period T_p . The constraints imposed on the range of the PWM manipulated variable are: 0%–100%.

The step response of the model of process is depicted in Fig. 10. Fig. 11 shows the results of the FDMC and FGPC algorithms, in both cases $\lambda_1 = 10$. In FDMC, the horizon of dynamics is $D = 150$. Visible oscillations are generated by the tachometer, which even after filtration generates noise (its frequency depends on the servo speed). The minimal possible sampling period is $T_p = 1$ ms. Over half of this time is required for calculations. The calculation time is $584.20 \mu s$.

In the FGPC algorithm, the first-order model is used ($n_A = n_B = 1$). The shortest possible sampling period is $T_p = 1$ ms. For shorter ones, the servo becomes nonlinear and the FGPC may be unstable. Control quality is lower compared to the FDMC, although it still tracks the set-point trajectory with reasonable precision. Again, the tachometers inaccuracies are clearly visible. The calculation time is $11.83 \mu s$ (compared with $584.20 \mu s$ in the FDMC algorithms).

It is worth underlining that even if the general model of the servo is accurately denoted using a single inertia, lowering the sampling period introduces nonlinear behaviour of the process. Therefore, the FDMC and FGPC algorithms, which use for prediction linear models may yield low control quality.

D. Calculation Time

At first, calculation time of the FDMC and classical DMC algorithms is studied for the MIMO emulated process. Table I gives the influence of the control horizon, N_u , and the

TABLE I
INFLUENCE OF THE CONTROL HORIZON N_u AND THE SAMPLING TIME T_p ON CALCULATION TIME [μ s] OF FDMC AND DMC ALGORITHMS ($N = N_u$) APPLIED TO THE EMULATED MIMO PROCESS

T_p [μ s]	D	FDMC	DMC								
		$N_u = 50$	$N_u = 1$	$N_u = 5$	$N_u = 10$	$N_u = 15$	$N_u = 20$	$N_u = 25$	$N_u = 30$	$N_u = 40$	$N_u = 50$
50	1000	5771	5900	23970	47460	71180	—	—	—	—	—
100	500	2605	3253	11980	25120	41010	51190	65470	77780	—	—
500	100	656	552	3153	5409	8447	14200	18480	23050	33080	44290
1000	50	332	280	1367	2944	4765	8265	11060	12420	21240	29490
5000	10	64	62	443	1029	2196	3516	5131	7041	11750	15560
10000	5	36	35	292	873	1750	2922	4389	6152	10560	16160
25000	2	17	18	170	578	1482	2566	3944	5619	9853	15270

When the calculation time is longer than the sampling time, the results are emphasized; “—” means that the compiler fails to compile the code due to high memory requirements.

TABLE II
INFLUENCE OF THE CONTROL HORIZON N_u AND THE SAMPLING TIME T_p ON CALCULATION TIME [μ s] OF FGPC AND GPC ALGORITHMS ($N = N_u$) APPLIED TO THE EMULATED MIMO PROCESS

T_p [μ s]	FGPC	GPC								
	$N_u = 50$	$N_u = 1$	$N_u = 5$	$N_u = 10$	$N_u = 15$	$N_u = 20$	$N_u = 25$	$N_u = 30$	$N_u = 40$	$N_u = 50$
50	7.13	20.26	163	552	1182	2044	3142	4476	7854	12180
100	7.13	20.26	163	552	1182	2044	3142	4476	7854	12180
500	7.13	20.26	163	552	1182	2044	3142	4476	7854	12180
1000	7.13	20.26	163	552	1182	2044	3142	4476	7854	12180
5000	7.13	20.26	163	552	1182	2044	3142	4476	7854	12180
10000	7.13	20.26	163	552	1182	2044	3142	4476	7854	12180
25000	7.13	20.26	163	552	1182	2044	3142	4476	7854	12180

When the calculation time is longer than the sampling time, the results are emphasized.

sampling time, T_p , on calculation time. The prediction and control horizons are equal ($N = N_u$), the horizon of dynamics is set to $D = 5 \times 10^{-6}/T_p$ since shortening of the sampling period needs lengthening of the step response to capture the process dynamics. One can observe the following.

- 1) For $T_p = 50 \mu$ s, $N_u \geq 20$ and for $T_p = 100 \mu$ s, $N_u \geq 40$, the compiler fails to compile the code of the classical DMC algorithm (table entries denoted by “—”) because of memory shortage. It is because the horizon of dynamics is long and a great number of model parameters are necessary.
- 2) The FDMC and DMC algorithms may be used for sampling times not shorter than 1000μ s. Again, it is because for shorter sampling times the horizon of dynamics is long, which means that the resulting calculation time is longer than the possible sampling period.
- 3) Calculation time of the FDMC algorithm is much shorter than that of the classical DMC scheme. Moreover, calculation time of FDMC only depends on the horizon of dynamics, D , whereas in the case of DMC it also depends on the control and prediction horizons. It means that for long horizons, calculation time of DMC is longer than the sampling period. For example, for $T_p = 1000 \mu$ s, the classical DMC algorithm may be used only with the horizons $N_u = N = 1$, whereas in the FDMC the horizons are not limited.

Next, calculation time of the FGPC and classical GPC algorithms is studied for the MIMO emulated process. Table II gives the influence of the control horizon, N_u , and the sampling time, T_p , on calculation time. The prediction and control horizons are equal ($N_u = N$). One can observe the following.

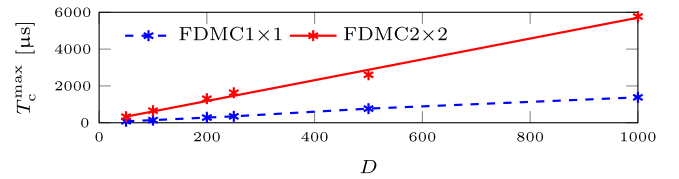


Fig. 12. Emulated SISO and MIMO processes: Influence of the horizon of dynamics, D , on calculation time of the FDMC algorithm; lines represent approximations whereas measurements are marked with asterisks.

- 1) No compiling problems (memory shortages) are encountered. It is because the FGPC and GPC algorithms use very compact models.
- 2) Because compact models are used, the FGPC and GPC algorithms are characterized by much shorter calculation times than the FDMC and DMC ones, respectively.
- 3) Calculation time of the FGPC algorithm is much shorter than that of the classical GPC scheme. Moreover, for FGPC it depends only on the model order, whereas in the case of GPC it also depends on the prediction and control horizons. Calculation time for both FGPC and GPC is independent on the sampling time.
- 4) For short sampling periods the GPC algorithm may be used only with short horizons. For example, for $T_p = 50 \mu$ s, the classical GPC algorithm may be used only with the horizons $N = N_u = 1$ (for longer horizons calculation time is longer than the sampling period), whereas in the FGPC the horizons are not limited.

For the emulated SISO and MIMO processes, Fig. 12 shows the influence of the horizon of dynamics, D , on calculation time of the FDMC algorithm. The obtained relations are clearly

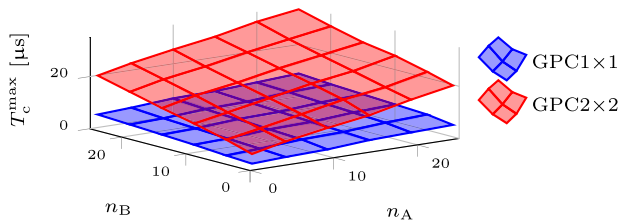


Fig. 13. Emulated SISO and MIMO processes: Influence of the model order (n_A , n_B) on calculation time of the FGPC algorithm.

linear, the compiler's optimization leads to small differences. The approximation of the calculation time, $T_c^{\max}$, (in microseconds) for the FDMC algorithm is: $1.39D + 14.24$ for the emulated SISO process, $3.69D + 130.85$ for the servo, and $5.66D + 49.30$ for the emulated MIMO process.

Fig. 13 depicts the influence of the order of model dynamics (n_A , n_B) on calculation time of the FGPC algorithm. Additional experiments have been performed to take into account emulated processes with high dynamic orders (up to $n_A = n_B = 25$). The relations are linear. Even for a complicated difference equations with the 25th order of dynamics, the calculation time is shorter than $10 \mu s$ for the SISO process, and shorter than $40 \mu s$ for the MIMO process with two inputs and two outputs. The linear approximations of the calculation time, $T_c^{\max}$, (in microseconds) for the FGPC algorithms are: $0.15n_A + 0.15n_B + 1.18$ for the emulated SISO process and $0.57n_A + 0.59n_B + 4.60$ for the emulated MIMO process. The influence of the coefficients n_A and n_B is almost the same for SISO and MIMO cases, respectively. It is not surprising taking into account the FGPC control law (32).

V. CONCLUSION

The described fast versions of the DMC and GPC algorithms (FDMC and FGPC) may be used for controlling very fast dynamic systems, characterized by very short sampling periods. The FDMC and FGPC algorithms need much shorter calculation time than the classical DMC and GPC schemes. In particular, the discussed algorithms work correctly when the classical ones are unable to finish calculations during the time period determined by the sampling time. As the FDMC algorithm uses a nonparametric step response model, typical calculation time is of the order of hundreds or tens of microseconds. Since the FGPC algorithm uses a compact difference equation model, calculation time may be reduced to single microseconds. Calculation time of the discussed FDMC algorithm linearly depends on the horizon of dynamics and on the number of process inputs and outputs. Calculation time of the discussed FGPC algorithm linearly depends on model order and on the number of process inputs and outputs. In contrast to the classical DMC and GPC schemes, calculation time is independent of the prediction and control horizons.

REFERENCES

- [1] J. Maciejowski, *Predictive Control With Constraints*. Harlow, Poland: Prentice-Hall, 2001.
- [2] P. Tatjewski, *Advanced Control of Industrial Processes: Structures and Algorithms*. London, U.K.: Springer, 2007.
- [3] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Eng. Pract.*, vol. 11, pp. 733–764, 2003.
- [4] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 60, pp. 2279–2285, 2011.
- [5] K. Kunz, S. M. Huck, and T. H. Summers, "Fast model predictive control of miniature helicopters," in *Proc. Eur. Control Conf.*, Zurich, Switzerland, 2013, pp. 1377–1382.
- [6] L. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Appl. Methods*, vol. 36, pp. 628–647, 2015.
- [7] G. Takács, P. Zometa, R. Findeisen, and B. Rohal'-Ilkiv, "Efficiency and performance of embedded model predictive control for active vibration attenuation," in *Proc. Eur. Control Conf.*, Aalborg, Denmark, 2016, pp. 1334–1340.
- [8] F. Xu, H. Chen, and X. Gong, "Fast nonlinear model predictive control on FPGA using particle swarm optimization," *IEEE Trans. Ind. Electron.*, vol. 63, no. 1, pp. 310–321, Jan. 2016.
- [9] C. Xia, T. Liu, T. Shi, and Z. Song, "A simplified finite-control-set model-predictive control for power converters," *IEEE Trans. Ind. Inform.*, vol. 10, no. 2, pp. 991–1002, May 2014.
- [10] S.-C. Carpiuc and C. Lazar, "Real-time multi-rate predictive cascade speed control of synchronous machines in automotive electrical traction drives," *IEEE Trans. Ind. Electron.*, vol. 63, no. 8, pp. 5133–5142, Aug. 2016.
- [11] M. Habibullah, D. D.-C. Lu, D. Xiao, J. E. Fletcher, and M. F. Rahman, "Predictive torque control of induction motor sensorless drive fed by a 3L-NPC inverter," *IEEE Trans. Ind. Inform.*, vol. 13, no. 1, pp. 60–70, Feb. 2017.
- [12] S. Kwak and J.-C. Park, "Model-predictive direct power control with vector preselection technique for highly efficient active rectifiers," *IEEE Trans. Ind. Inform.*, vol. 11, no. 1, pp. 44–52, Feb. 2015.
- [13] Y. Liu, B. Ge, H. Abu-Rub, H. Sun, F. Z. Peng, and Y. Xue, "Model predictive direct power control for active power decoupled single-phase quasi-Z-source inverter," *IEEE Trans. Ind. Inform.*, vol. 12, no. 4, pp. 1550–1559, Aug. 2016.
- [14] M. B. Kane, J. Scruggs, and J. P. Lynch, "Model-predictive control techniques for hydronic systems implemented on wireless sensor and actuator networks," in *Proc. Amer. Control Conf.*, Portland, OR, USA, 2014, pp. 3542–3547.
- [15] A. A. Kheriji, F. Bouani, M. Ksouri, and M. B. Ahmed, "A microcontroller implementation of model predictive control," *World Acad. Sci., Eng. Technol.*, vol. 53, pp. 68–74, 2011.
- [16] L. Bitjoka, M. Ndjé, A. T. Boum, and J. Song-Manguelle, "Implementation of quadratic dynamic matrix control on arduino due ARM cortex-M3 microcontroller board," *J. Eng. Technol.*, vol. 6, pp. 682–695, Jul. 2017.
- [17] P. Chaber and M. Ławryńczuk, "Implementation of analytical generalized predictive controller for very fast applications using microcontrollers: Preliminary results," in *Trends in Advanced Intelligent Control, Optimization and Automation*, W. Mitkowski, J. Kacprzyk, K. Oprzedkiewicz, and P. Skruch, Eds. Cham, Switzerland: Springer, 2017, pp. 378–387.
- [18] R. Kouki, H. Salhi, and F. Bouani, "Application of model predictive control for a thermal process using STM32 microcontroller," in *Proc. Int. Conf. Control, Automat. Diagnosis*, Hammamet, Tunisia, 2017, pp. 146–151.
- [19] I. Malouche, A. K. Abbes, and F. Bouani, "Automatic model predictive control implementation in a high-performance microcontroller," in *Proc. IEEE 12th Int. Multi-Conf. Syst., Signals Devices*, Mahdia, Tunisia, 2015, pp. 1–6.
- [20] B. Nguyen, Y. Ho, Z. Wu, and C.-K. Chui, "Implementation of model predictive control with modified minimal model on low-power RISC microcontrollers," in *Proc. 3rd Symp. Inf. Commun. Technol.*, Halong City, Vietnam, 2012, pp. 165–171.
- [21] M. Short and F. Abugchem, "A microcontroller-based adaptive model predictive control platform for process control applications," *Electronics*, vol. 6, no. 4, 2017, Art. no. 88.
- [22] A. Smeti, W. Chagra, and M. Ksouri, "Implementation of a predictive controller on the STM32 board," in *Proc. Int. Conf. Control, Decis. Inf. Technol.*, Hammamet, Tunisia, 2013, pp. 250–253.
- [23] X. Jiao, X. Zhu, M. Xu, and L. Di, "A simplified dynamic matrix control for application of embedded real-time MPC," in *Proc. Int. Conf. Inf. Sci. Technol.*, Nanjing, China, 2011, pp. 1062–1065.
- [24] Z. Chen, L. Yu, and E. Dong, "A fast generalized predictive control algorithm based on toeplitz matrix," in *Proc. Int. Conf. Comput. Intell. Softw. Eng.*, Wuhan, China, 2010, pp. 1–4.

- [25] C. Cutler and B. Ramaker, "Dynamic matrix control—A computer control algorithm," in *Proc. AIChE Nat. Meeting*, 1980.
- [26] D. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive control," *Automatica*, vol. 23, pp. 137–160, 1987.
- [27] A. Stinean, S. Preitl, R. Precup, C. Dragos, E. Petriu, and M. Radac, "Choosing a proper control structure for a mechatronic system with variable parameters," in *Proc. 2nd IFAC Workshop Convergence Inf. Technol. Control Methods Power Syst.*, Cluj-Napoca, Romania, 2013, pp. 26–31.
- [28] A. Kozakova and M. Hypiusova, "LQG/LTR based reference tracking for a modular servo," *J. Elect. Syst. Inf. Technol.*, vol. 2, pp. 347–357, 2015.
- [29] R. Barbosa, J. Tenreiro Machado, and I. Jesus, "Effect of fractional orders in the velocity control of a servo system," *Comput. Math. Appl.*, vol. 59, pp. 1679–1686, 2010.



Patryk Chaber was born in Warsaw, Poland, in 1990. He received the B.Sc. and M.Sc. degrees, in 2012 and 2014, respectively, both in computer science, and the Ph.D. degree, in 2018, in automatic control and robotics, all from Faculty of Electronics and Information Technology, Warsaw University of Technology, Warsaw, Poland.

He is currently a Teaching Assistant with the Institute of Control and Computation Engineering, Warsaw University of Technology. He is the author or coauthor of more than ten publications, including journal articles, conference papers, and reports. His research interests include advanced control algorithms, in particular MPC algorithms, microcontrollers, real-time process control, and modeling.



Maciej Ławryńczuk was born in Warsaw, Poland, in 1972. He received the M.Sc., Ph.D., and D.Sc. degrees, in 1998, 2003, and 2013, respectively, all in automatic control, from the Faculty of Electronics and Information Technology, Warsaw University of Technology, Warsaw, Poland.

He is currently an Associate Professor with the Institute of Control and Computation Engineering, Warsaw University of Technology. He is the author or co-author of 6 books and more than 100 other publications, including more than 30 journal articles. His research interests include advanced control algorithms, in particular MPC algorithms, set-point optimization algorithms, soft computing methods, in particular neural networks, modeling, and simulation.