

Dynamic Matrix Control Algorithm Implementation on ARM Cortex-R5 MCU: Performance Analysis

Kamil Czerwiński * Maciej Ławryńczuk **

* *Institute of Control and Computation Engineering,
 Warsaw University of Technology, ul. Nowowiejska 15/19,
 00-665 Warsaw, Poland, Email: kczerwi2@mion.elka.pw.edu.pl*

** *Institute of Control and Computation Engineering,
 Warsaw University of Technology, ul. Nowowiejska 15/19,
 00-665 Warsaw, Poland, Email: M.Lawrynczuk@ia.pw.edu.pl*

Abstract: This paper describes implementation of the Dynamic Matrix Control (DMC) algorithm using a microcontroller with the ARM Cortex R5 core. Two versions of the DMC algorithm are considered: the classical one and the simplified one. It is shown that the latter one needs much shorter calculation time, but give the same control quality. Moreover, the influence of tuning parameters of the DMC algorithm on the calculation time is studied experimentally. The obtained implementation of the DMC algorithm can be used in very fast embedded system, with short sampling periods. Results of real experiment are demonstrated for Hercules RM57L843 LaunchPad board from Texas Instruments.

© 2018, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Embedded systems, microprocessors and microcontrollers; control theory; Dynamic Matrix Control; Model Predictive Control; real-time embedded systems

1. INTRODUCTION

For the last decades Model-based Predictive Controllers (MPC) (Camacho and Bordons (1999); Maciejowski (2002); Tatjewski (2007)) have been extensively used mainly for large-scale industrial systems, e.g. in chemical engineering, refineries, power plants, waste water treatment facilities etc. In such applications typically the multilayer control system structure (Tatjewski (2007)) is used, i.e. the set-points for MPC are calculated on-line by an optimisation layer, and MPC algorithms are implemented using high quality, very reliable industrial controllers (Qin and Bangwell (2006)). A unique feature of numerous applications in the mentioned industries is relatively long sampling period, typically of order of seconds or even minutes. Dynamic Matrix Control (DMC) algorithm is very frequently used in industry due to very basic procedure of model identification as for process description a simple step-response model is used, which may be obtained in practice very easily.

Although MPC algorithms are very successful in industry, currently new possible applications of advanced control schemes emerge. Rapid and huge advancements in electronics, information technology and communication create a possibility to use advanced control algorithms in a embedded systems, i.e. in processes we use or will be using every day, e.g. autonomous vehicles, robots etc. (Lucia et al. (2016)). In contrast to industrial process control, in embedded systems it is usually necessary to use very short sampling periods, typically of the order of fractions of second or even milliseconds. For such fast applications advanced control algorithms are implemented using minia-

ture electronic equipment, e.g. microcontrollers (Chaber (2017)) or Field Programmable Gate Arrays (FPGAs) (Wojtulewicz (2017)).

Implementation of complex control algorithms is always a compromise between available resources of memory and CPU load utilization (Plamowski (2017)). In particular in embedded systems, resources (e.g. memory) may be limited and the necessity of using very short sampling periods encourages to minimise the time necessary for algorithms' calculations. This paper studies calculation time of the classical and simplified versions of the DMC algorithm for different setting parameters. The algorithm is implemented in an embedded system controlled by a high-performance microcontroller ARM Cortex-R5F with floating-point CPU. Results of real laboratory experiments are presented and discussed.

2. DMC ALGORITHM

2.1 Preliminaries

We consider a dynamic system with one input (manipulated variable) u and one output (controlled variable) y . At each discrete sampling instants, $k = 0, 1, 2, \dots$, the DMC algorithm calculates on-line set of N_u future control increments

$$\Delta \mathbf{u}(k) = [\Delta u(k|k) \ \dots \ \Delta u(k + N_u - 1|k)]^T \quad (1)$$

where N_u is called a control horizon. The increments are defined as

$$\Delta u(k|k) = u(k|k) - u(k-1) \quad (2)$$

and

$$\Delta u(k+p|k) = u(k+p|k) - u(k+p-1|k) \quad (3)$$

for $p = 1, \dots, N_u - 1$. It is assumed that $\Delta u(k+p|k) = 0$ for $p \geq N_u$. The future increments of the manipulated variables (1) are calculated in such a way that the differences between the set-point trajectory $y^{\text{sp}}(k+p|k)$ and the predicted output values $\hat{y}(k+p|k)$ are minimised over the prediction horizon $N \geq N_u$, i.e. for $p = 1, \dots, N$. Hence, for optimisation of the future control policy (1) the following quadratic cost function is usually used

$$J(k) = \sum_{p=1}^N \|y^{\text{sp}}(k+p|k) - \hat{y}(k+p|k)\|^2 + \lambda \sum_{p=0}^{N_u-1} \|\Delta u(k+p|k)\|^2 \quad (4)$$

where $\lambda > 0$ is a tuning coefficient (the bigger its value, the slower the set-point tracking, but it makes it possible to penalise the excessive increments of the manipulated variable). In spite of the fact that as many as N_u increments (1) are calculated, only its first element is actually applied to the process, i.e.

$$u(k) = \Delta u(k|k) + u(k-1) \quad (5)$$

At the next sampling instant, $k+1$, the measurement of the process output is updated, the prediction is shifted one step forward and the whole calculation procedure is repeated.

In order to calculate the predictions of the controlled variable over the prediction horizon, i.e. the quantities $\hat{y}(k+1|k), \dots, \hat{y}(k+N|k)$, a step-response model of the controlled process is used. The model shows the reaction of the process output when it is excited by the input step occurring at the sampling instant $k=0$. The model consists of the values of process output, which are denoted $s_1, \dots, s_D$ and named step-response coefficients. Since the DMC algorithm assumes that the process is stable with no integration, it is only sufficient to record D step-response coefficients, where D is named the horizon of dynamics.

2.2 Classical DMC Formulation

The DMC cost-function (4) may be expressed in the compact vector-matrix-notation

$$J(k) = \|\mathbf{y}^{\text{sp}}(k) - \hat{\mathbf{y}}(k)\|^2 + \|\Delta \mathbf{u}(k)\|_{\mathbf{\Lambda}}^2 \quad (6)$$

where the vectors of length N are

$$\mathbf{y}^{\text{sp}}(k) = [y(k+1|k) \dots y(k+N|k)]^T \quad (7)$$

$$\hat{\mathbf{y}}^{\text{sp}}(k) = [\hat{y}(k+1|k) \dots \hat{y}(k+N|k)]^T \quad (8)$$

and $\mathbf{\Lambda} = \text{diag}(\lambda, \dots, \lambda)$ is a matrix of dimensionality $N_u \times N_u$. Since the DMC cost-function (6) is quadratic in terms of the decision vector $\Delta \mathbf{u}(k)$, the optimal control increments are

$$\Delta \mathbf{u}(k) = \mathbf{K}(\mathbf{y}^{\text{sp}}(k) - \mathbf{y}(k) - \mathbf{G}^{\text{p}} \Delta \mathbf{u}^{\text{p}}(k)) \quad (9)$$

The matrix of dimensionality $N_u \times N$ is

$$\mathbf{K} = (\mathbf{G}^{\text{T}} \mathbf{G} + \mathbf{\Lambda})^{-1} \mathbf{G}^{\text{T}} \quad (10)$$

The step-response matrices

$$\mathbf{G} = \begin{bmatrix} s_1 & 0 & \dots & 0 \\ s_2 & s_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s_N & s_{N-1} & \dots & s_{N-N_u+1} \end{bmatrix} \quad (11)$$

and

$$\mathbf{G}^{\text{p}} = \begin{bmatrix} s_2 - s_1 & \dots & s_D - s_{D-1} \\ s_3 - s_1 & \dots & s_{D+1} - s_{D-1} \\ \vdots & \ddots & \vdots \\ s_{N+1} - s_1 & \dots & s_{N+D-1} - s_{D-1} \end{bmatrix} \quad (12)$$

are of dimensionality $N \times N_u$ and $N \times n_u(D-1)$, respectively. The vectors

$$\mathbf{y}(k) = [y(k) \dots y(k)]^T \quad (13)$$

$$\Delta \mathbf{u}^{\text{p}}(k) = [\Delta u(k-1) \dots \Delta u(k-(D-1))]^T \quad (14)$$

are of length N and $D-1$, respectively.

2.3 Simplified DMC Control Law

Let us note the fact that when the classical formulation of the DMC algorithm is used at each sampling instant as many as N_u future control increments $\Delta \mathbf{u}(k)$ are calculated from Eq. (9), although only its first element is actually used for control (Eq. (5)). Hence, it is reasonable, in particular in fast applications with short sampling periods, to use an explicit version of the control law. From Eq. (9) and taking into account the structure of the matrix $\mathbf{K}$ (Eq. (10)) which is

$$\mathbf{K} = \begin{bmatrix} \bar{\mathbf{K}}_{1,1} \\ \vdots \\ \bar{\mathbf{K}}_{N_u,1} \end{bmatrix} = \begin{bmatrix} K_{1,1} & \dots & K_{1,N} \\ \vdots & \ddots & \vdots \\ K_{N_u,1} & \dots & K_{N_u,N} \end{bmatrix} \quad (15)$$

we obtain the explicit control law

$$\Delta u(k|k) = K^e(y^{\text{sp}}(k) - y(k)) - \sum_{i=1}^{D-1} K_i^u \Delta u(k-i) \quad (16)$$

where

$$K^e = \sum_{p=1}^N K_{1,p} \quad (17)$$

$$K_i^u = \bar{\mathbf{K}}_1 \mathbf{G}_i^{\text{p}}, \quad i = 1, \dots, D-1 \quad (18)$$

and $\mathbf{G}_i^{\text{p}}$ denotes the consecutive columns of the matrix $\mathbf{G}^{\text{p}}$, i.e. $\mathbf{G}^{\text{p}} = [\mathbf{G}_1^{\text{p}} \dots \mathbf{G}_{D-1}^{\text{p}}]^T$. In the above simplified formulation (16) we assume that the set-point variable is constant over the prediction horizon, i.e.

$$y^{\text{sp}}(k+1|k) = \dots = y^{\text{sp}}(k+N|k) \quad (19)$$

which is typical in practical applications, because changes of set-point values are usually unknown or not considered during the time of calculation of the manipulated value. Let us emphasise the fact that the coefficients K^e and K_i^u are calculated from the step-response model of the controlled process completely off-line, before running the DMC algorithm.

3. EXPERIMENTAL SET-UP

For implementation of the MPC algorithm the Hercules RM57L843 development board is used (Texas Instruments (2017)). This is high-performance microcontroller for safety critical applications. The RM57L843 device integrates two ARM Cortex-R5F floating-point CPUs, operating in lockstep and offers 1.66 DMIPS/MHz, and can run at the speed of up to 330MHz (the default speed is 300MHz). This device has 4MB of integrated flash memory and 512KB of RAM with single-bit error correction and

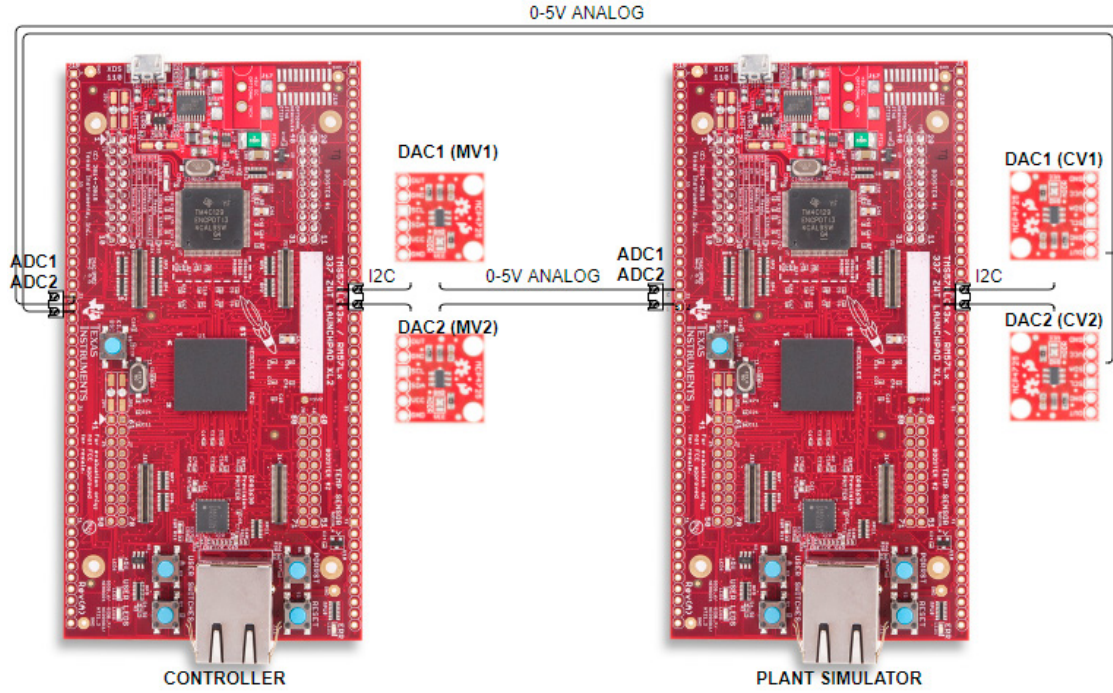


Fig. 1. Experimental system connection schema: the first board serves as the DMC controller, the second board is used as a plant simulator

double-bit error detection. The RM57L843 device features peripherals for real-time control-based applications, including two Next Generation High-End Timer (N2HET) timing coprocessors with up to 64 total I/O terminals. The considered development board is dedicated for real-time industrial safety applications and are certified with Safety Integrity Level 3 (SIL3).

The connection schema of the experimental system is presented in Fig 1. The experimental set-up uses two RM57843 boards. The first one is responsible for control algorithm implementation (*the controller*) and the second one for a configurable simulated process (*the process*). Communication between boards uses simple 0-5V analog signals. For signal acquisition and conversion 12-bit Analog to Digital converters (MibADCs), 12-bit Digital to Analog converters (MCP4725) and I2C interfaces are used. Of course, this is only experimental work, the objective of which is to study performance of the controller whereas in real applications the board used for process simulation is replaced by a real process.

Time measurement is implemented based on real-time interrupt module (RTI), which provides timer functionality for operating systems and for benchmarking code. This module can incorporate several counters that define the time bases needed for scheduling an operating system. The timers also give functionality for benchmark certain areas of code by reading the values of the counters at the begin and the end of the desired code range and calculating the difference between the values. Values can be accessed with high 64-bits precision. Counter reading for code benchmarking of this module is presented in the following listing:

```
uint64_t getRunTimeCounterValue64 (void)
{
```

```
    register uint32_t c1, c1_, c2;
    c1 = rtiREG1->CNT[1U].FRCx;
    c2 = rtiREG1->CNT[1U].UCx;
    c1_ = rtiREG1->CNT[1U].FRCx;
    if (c1_ != c1) {
        c1 = c1_;
        c2 = rtiREG1->CNT[1U].UCx;
    }
    return ((uint64_t) c1) << 32 | c2;
}
```

Coefficients of the DMC control laws are precalculated off-line. In the classical implementation (Eq. (9)) it is necessary to find from the step-response model the matrices K and G^P . In the simplified implementation (16) the coefficients K^e and K^u are used. Let us note the fact that these matrices and coefficients are calculated from the step response model of the controlled process. In the first version of the DMC algorithm we use Eqs. (10), (11) and (12) whereas in the second one Eqs. (17) and (18) are necessary. Since vector-matrix calculations require a lot of dynamic memory allocation and the mentioned vectors and matrices are constant, they are determined off-line (the DMC controller initialisation step), not in the implementation of the DMC algorithm running on-line. Implementation of DMC law calculation in classical and simplified versions are presented on the listing below:

```
double classical_DMC_1x1(DMC_1x1_t* reg
                        ,double sp_k
                        ,double pv_k)
{
    calculate_Y0(reg, pv_k);
    calculate_DU(reg, sp_k);
    reg->U_k = reg->DU[0] + reg->U_k-1;

    for(int i=1; i<reg->Hd-1;i++){
```

```

        reg->DUP[i] = reg->DUP[i+1];
    }
    reg->DUP[0] = reg->DU[0];
    reg->U_k_1 = reg->U_k;
    return reg->U_k;
}

double simplified_DMC_1x1(DMC_1x1_t* reg
                        ,double sp_k
                        ,double pv_k)
{
    for (int i=0; i < reg->Hd-1; i++){
        ku += reg->Ku[i] * reg->DUP[i];
    }

    reg->U_k = reg->Ke*(sp_k - pv_k)-ku;

    for (i=1; i<reg->Hd-1; i++){
        reg->DUP[i] = reg->DUP[i+1];
    }

    reg->DUP[0] = reg->U_k;
    reg->U_k += reg->U_k_1;
    reg->U_k_1 = reg->U_k;
    return reg->U_k;
}

```

The following steps are carried out during experiments with the considered laboratory set-up:

- Implementation of the process model on the second RM57843 board (*the process*).
- Off-line DMC controller initialisation for specific values of the horizons D , N , N_u and the penalty factor λ . It is possible to find off-line a great number of control law coefficients, for different tuning parameters. Each configuration is stored in scripts, used by embedded controller application
- Initialisation of the embedded controller from script, execution of experiments. All results of experiments are stored in output files, which may be easily used for analysis of the DMC algorithm.

Controller application is synchronised by RTI timer ticks generated in interrupt mode, every 20 ms. In our experimental set-up the sampling time of process plant is set to 10 ms. Controller application is responsible for measurement of the controlled variable, calculation DMC algorithm, processing manipulation variables and calculation time factors. Digital to Analog converters operates using I2C interface with SCL equals 400kHz. To benchmark the code, two time ranges are monitored. The first one is the time measured at the beginning and at the end of each control loop (data acquisition and algorithm calculation), the second one is only the DMC algorithm execution time. The main controller application is presented in the following listing:

```

// Initialize System and peripherals
SystemInit();

// Controller initialization
Controller reg;
reg.type=DMC_1x1;

// Create controller from script
reg.object=create_controller_from_script
(script , reg.type);

```

```

while(simulationActive)
{
    if(newScanTick)
    {
        //New control loop
        start_loop=getRunTimeCounterValue64();

        //Measurements
        GetAnalogInputs(ProcessData);

        //Convert inputs from pin to internal
        ConvertInputs(ProcessData);

        //Calculate DMC
        start_calc=getRunTimeCounterValue64();
        Calculate((DMC_1x1_t *)reg.object ,
                ProcessData);
        end_calc=getRunTimeCounterValue64();

        //Convert outputs from internal to pin
        ConvertOutputs(ProcessData);

        //Set manipulated variables
        SetAnalogOutputs(ProcessData , i2cREG1);

        //End of control loop
        end_loop=getRunTimeCounterValue64();

        //Calculate Performance Factors
        loop_time=((double)
                (end_loop - start_loop))/RTIFREQ;
        calc_time=((double)
                (end_calc - start_calc))/RTIFREQ;

        //Save result
        StoreTimeStamps(loop_time , calc_time);
    }
}

```

4. RESULTS OF EXPERIMENTS

In this paper it is assumed that the controlled process is simulated on the second board shown in Fig. 1 (*the process*). The simulated process is described as a first-order discrete-time transfer function

$$G(z) = \frac{0.2855}{z - 0.9048} \quad (20)$$

The sampling period of the DMC algorithms is 20 ms. The results of the experiments presented next have the objective to find the relation between the horizons of two versions of the DMC algorithm on the time necessary to perform calculations in one sampling instant. The horizon of dynamics (D), the prediction horizon (N) and the control one (N_u) are taken into account. In all the experiments carried out the penalty factor λ is set to 2.5, its choice does not influence the calculation time. The reader interested in the influence of these tuning parameters on possible quality of control is referred to the classical textbooks concerned with MPC (Camacho and Bordons (1999); Maciejowski (2002); Tatjewski (2007)), this issue is not considered in this work. Fig. 2 depicts process input and output trajectories for different control horizons, N_u , the horizon of dynamics and the prediction horizon are constant: $D = 40$, $N = 30$.

Table 1. Calculation time of two versions of the DMC algorithm for different horizons (D , N and N_u), the communication time necessary for signal transitions is constant in all cases is $484\mu s$

Horizons			Classical implementation (μs)			Simplified implementation (μs)		
D	N	N_u	Min.	Max.	Mean.	Min.	Max.	Mean
20	10	9	5.18×10^1	5.21×10^1	5.18×10^1	4.81×10^0	5.39×10^0	4.82×10^0
30	20	17	1.62×10^2	1.63×10^2	1.63×10^2	7.03×10^0	7.71×10^0	7.03×10^0
40	30	17	2.88×10^2	2.95×10^1	2.89×10^2	9.19×10^0	9.75×10^1	9.19×10^0
40	30	25	3.35×10^2	3.47×10^1	3.35×10^2	9.19×10^0	1.01×10^1	9.19×10^0
50	40	17	4.47×10^2	4.76×10^2	4.47×10^2	1.14×10^1	1.28×10^1	1.14×10^1
50	40	25	5.10×10^2	5.44×10^2	5.11×10^2	1.14×10^1	1.25×10^1	1.14×10^1
60	50	17	6.43×10^2	6.82×10^2	6.44×10^2	1.35×10^1	1.51×10^1	1.35×10^1
60	50	25	7.26×10^2	7.64×10^2	7.28×10^2	1.35×10^1	1.48×10^1	1.35×10^1
70	60	17	8.83×10^2	9.17×10^3	8.87×10^2	1.57×10^1	1.76×10^1	1.57×10^1
70	60	25	9.84×10^2	1.02×10^3	9.89×10^2	1.57×10^1	1.72×10^1	1.57×10^1
80	70	17	1.16×10^3	1.19×10^3	1.16×10^3	1.79×10^1	1.99×10^1	1.79×10^1
80	70	25	1.28×10^3	1.30×10^3	1.28×10^3	1.79×10^1	1.99×10^1	1.79×10^1
90	80	17	1.47×10^3	1.59×10^3	1.47×10^3	2.03×10^1	2.23×10^1	2.04×10^1
90	80	25	1.60×10^3	1.62×10^3	1.60×10^3	2.00×10^1	2.22×10^1	2.00×10^1

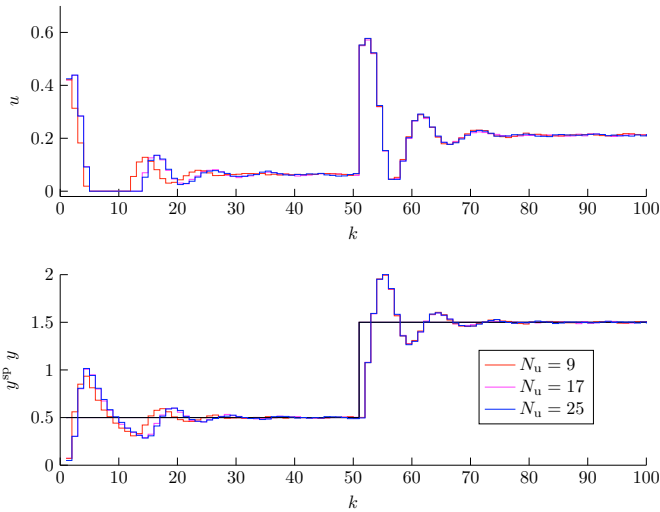


Fig. 2. The influence of the control horizon (N_u) on control quality, the horizon of dynamics and the prediction horizon are constant: $D = 40$ and $N = 30$ (in both versions of the DMC algorithm the trajectories are the same)

Let us emphasise the fact that the trajectories shown in Fig. 2 are exactly the same in two compared versions of the DMC algorithm, the only difference is the calculation time necessary to find the control action at each of the consecutive sampling instants k . It is because in the classical implementation the whole vector $\Delta \mathbf{u}(k)$ is determined at each sampling instant from Eq. (9) whereas in the simplified implementation it is only necessary to find the increment current of the manipulated variable $\Delta u(k)$ from Eq. (16). That is why in the classical version calculation time T is a function of all horizons, i.e. $T(D, N, N_u)$ whereas in the simplified one the time T depends only on the horizon of dynamics, i.e. $T(D)$. Table 1 shows the obtained calculation times for different choices of the horizons (D , N and N_u). Fig. 3 shows calculation time of the classical implementation of the DMC algorithm as a function of the prediction and control horizons, the horizon of dynamics

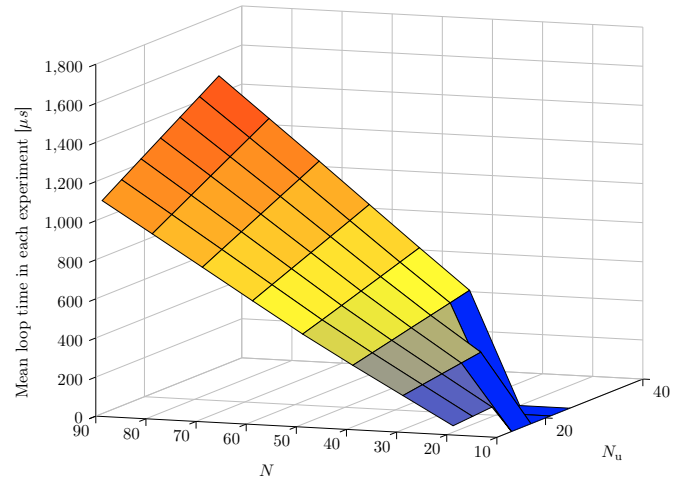


Fig. 3. Calculation time of the classical implementation of the DMC algorithm: the influence of the prediction and control horizons (N and N_u), the horizon of dynamics is constant ($D = 90$)

is constant. Fig. 4 shows calculation time of the simplified implementation of the DMC algorithm as a function of the horizon of dynamics. It is necessary to emphasize the fact that the second version of the DMC algorithm is many times faster than the first one. Using linear regression (least-squares), one may easily find the average calculation time in the second version of the algorithm is

$$T(D) = 0.22D + 0.5 \quad (21)$$

Let us notice the fact that calculation time in the second version of the DMC algorithm is much shorter than time necessary for data acquisition (Table 1), which may turn out to be an important difficulty in implementation for fast processes. In such cases it is necessary to use much faster converters than those used in this study. Maximum calculation time has been registered for the first control loop execution, and next cycles has been close to constant time. It could be related with caches memory utilization which can reduce read/write memory operation time.

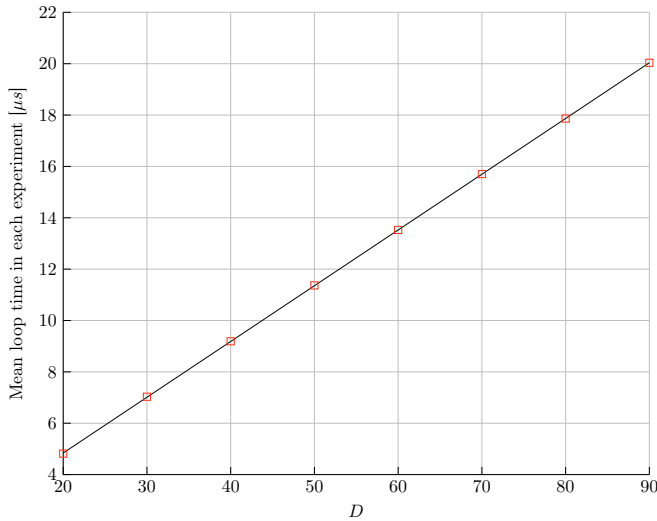


Fig. 4. Calculation time of the simplified implementation of the DMC algorithm as a function of the horizon of dynamics (D)

5. CONCLUSIONS

Two versions of the DMC algorithm have been implemented using a microcontroller with the ARM Cortex R5 core. The relation of calculation time necessary at each sampling instant is determined experimentally. In the classical implementation it depends on all the horizons (the horizon of dynamics as well as the prediction and control horizons) whereas in the second implementation that time depends only on the horizon of dynamics.

The second version of the DMC algorithm is very computationally efficient, the calculation time is very short, i.e. the algorithm needs only a few microseconds at each sampling instant. Hence, it may be used for controlling very fast dynamic systems, which require short sampling times, of a millisecond order. It is important to stress the fact that one may easily assess the time necessary to perform on-line calculations from (Eqs. (21)). In both versions of the algorithm the quality of control is exactly the same. For fast dynamic systems it is necessary to use external fast converters since data acquisition performed using the internal converters may take much longer than the DMC algorithm itself.

In future works it is planned to implement on the described hardware platform some MPC algorithm (not only the DMC one) with on-line optimisation, in which the constraints may be imposed on manipulated and controlled variables (Maciejowski (2002); Tatjewski (2007)). Furthermore, it is planned to implement some nonlinear MPC algorithms, in which nonlinear models are used to predict future behaviour of the process. In particular, computationally efficient nonlinear MPC algorithms with on-line linearisation will be considered (Ławryńczuk (2014)).

REFERENCES

- E.F. Camacho, C. Bordons. *Model Predictive Control*. Springer, London, 1999.
- P. Chaber, M. Ławryńczuk. Automatic Code Generation of MIMO Model Predictive Control Algorithms using Transcompiler. In W. Mitkowski, J. Kacprzyk, K. Oprzedkiewicz, P. Skruch, editor, *Trends in Advanced Intelligent Control, Optimization and Automation. KKA 2017. Advances in Intelligent Systems and Computing*, volume 577, pages 315–324. Springer, Cham, 2017.
- C.R. Cutler, B.L. Ramaker. Dynamic matrix control – a computer control algorithm. *Proceedings of the Joint Automatic Control Conference, San Francisco, USA*, 1979.
- S. Lucia, M. Kögel, P. Zometa, D.E. Quevedo, R. Findelsen. Predictive control, embedded cyberphysical systems and systems of systems – A perspective. *Annual Reviews in Control*, 41:193–207, 2016.
- M. Ławryńczuk. *Computationally Efficient Model Predictive Control Algorithms: A Neural Network Approach*. Studies in Systems, Decision and Control, vol. 3, Springer, Cham, 2014.
- J. M. Maciejowski. *Predictive control with constraints*. Prentice Hall, Harlow, 2002.
- S. Plamowski. Implementation of DMC algorithm in embedded controller - resources, memory and numerical modifications. In W. Mitkowski, J. Kacprzyk, K. Oprzedkiewicz, P. Skruch, editor, *Trends in Advanced Intelligent Control, Optimization and Automation. KKA 2017. Advances in Intelligent Systems and Computing*, volume 577, pages 18–21. Springer, Cham, 2017.
- S.J. Qin, T.A. Bangwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.
- P. Tatjewski. *Advanced Control of Industrial Processes, Structures and Algorithms*. Springer, London, 2007.
- Texas Instruments. Hercules RM57Lx LaunchPad Development Kit. <http://www.ti.com/tool/launchxl2-rm57l>, 2017.
- A. Wojtulewicz. Implementation of Dynamic Matrix Control Algorithm Using Field Programmable Gate Array: Preliminary Results. In W. Mitkowski, J. Kacprzyk, K. Oprzedkiewicz, P. Skruch, editor, *Trends in Advanced Intelligent Control, Optimization and Automation. KKA 2017. Advances in Intelligent Systems and Computing*, volume 577, pages 325–334. Springer, Cham, 2017.