

Optimisation-Based Tuning of Dynamic Matrix Control Algorithm for Multiple-Input Multiple-Output Processes

Jakub Sawulski and Maciej Ławryńczuk

Institute of Control and Computation Engineering, Warsaw University of Technology
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland

Email: J.Sawulski@ia.pw.edu.pl, M.Lawrynczuk@ia.pw.edu.pl

Abstract—This work discusses tuning of Model Predictive Control (MPC) algorithms by means of some global optimisation methods. For test purposes the Dynamic Matrix Control (DMC) algorithm applied to a Multiple-Input Multiple-Output (MIMO) process with 4 manipulated and 3 controlled variables is considered. The tuned parameters include prediction and control horizons as well as the weights of the minimised MPC cost-function. Four global optimisation methods are considered: the Particle Swarm Optimisation method, the Firefly Algorithm, the Grey Wolf Optimiser and the Jaya algorithm. They are compared in terms of the ability to find the best solution and convergence. The obtained results show that global optimisation methods can be successfully used in this type of tasks.

I. INTRODUCTION

Model Predictive Control (MPC) is one of techniques known as advanced process control [11]. In contrast to classical controllers, which compute the values of process manipulated (process input) variables as a function of actual and past values of process variables, MPC algorithms use model-based prediction of the controlled variables (process outputs) at finite time horizon to optimise the current and future values of the manipulated variables. This feature gives possibility to obtain much better control quality, impossible in the several typical cases for the PID algorithm. In particular, for delayed processes or when the numbers of process inputs and outputs are unequal the MPC gives very good results whereas control quality offered by the PID controller may be highly unsatisfactory [1], [6], [11].

One of the widely used MPC techniques is a Dynamic Matrix Control (DMC) algorithm [2]. There are a lot of reported successful applications of the DMC, mainly in the chemical, petrochemical, processing and paper industry [9]. The primary advantage of the DMC scheme is fact that it uses a model of the controlled process in the form of a set of step responses. Such a model may be very easily obtained in practice and it has a straightforward interpretation. The way of operation in the specific implementation is determined by several algorithm parameters: lengths of the prediction and control horizons and the tuning coefficients of the DMC minimised cost-function. In the typical approach those values are tuned experimentally.

The aim of this work is to find the values of the DMC parameters which result in the best possible control quality.

For this purpose the following global optimisation algorithms are considered: Particle Swarm Optimisation (PSO) [3], Firefly Algorithm (FA) [12], Grey Wolf Optimiser (GWO) [7] and Jaya algorithm [10]. A dynamic process with 4 manipulated and 3 controlled variables is considered as a benchmark. Formulation the optimisation task, implementation of the computational procedure and results of experiments are discussed. Finally, some possibilities of extending this methodology to wider application class are proposed.

II. MULTIPLE-INPUT MULTIPLE-OUTPUT MPC PROBLEM FORMULATION

In this work Multiple-Input Multiple-Output (MIMO) processes are considered. Let n_u denote the number of process inputs and let n_y be the number of process outputs. Hence, the process input vector is $u = [u_1 \dots u_{n_u}]^T$ and the output vector is $y = [y_1 \dots y_{n_y}]^T$.

At each sampling instant, $k = 0, 1, 2, \dots$, the MPC algorithm finds repeatedly on-line a set of the optimal increments of the manipulated variables

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

where N_u is the control horizon and the increments are

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

It is assumed that $\Delta u(k + p|k) = 0$ for $p \geq N_u$. The decision variables of MPC (1) are determined from an optimisation problem in which the predicted control errors over the prediction horizon N are minimised. They are defined as the sum of squared differences between the set-point trajectory $y^{sp}(k + p|k)$ and the predicted output values $\hat{y}(k + p|k)$ for $p = 1, \dots, N$, i.e.

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

where

$$\mathbf{M}_0 = \begin{bmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_{n_y} \end{bmatrix}, \quad \mathbf{\Lambda}_0 = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_{n_u} \end{bmatrix} \quad (3)$$

are weighting matrices of dimensionality $n_y \times n_y$ and $n_u \times n_u$, respectively. The tuning parameters are: $\mu_n \geq 0$ ($n = 1, \dots, n_y$) and $\lambda_n > 0$ ($n = 1, \dots, n_u$). The first of them prioritise the predicted control errors of consecutive controlled variables, the second ones penalise excessive increments of the consecutive manipulated variables. Although at each sampling instant 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 measurements of the process outputs are updated and the whole procedure is repeated.

III. MULTIPLE-INPUT MULTIPLE-OUTPUT DMC ALGORITHM

A distinctive feature of the DMC algorithm is the fact that a discrete-time step-response model is used to calculate on-line the predicted values of the process controlled variables, i.e. the vector $\hat{y}(k+p|k)$ for all $p = 1, \dots, N$. It is assumed that the process is stable, with no integration, i.e. after a step of one of the manipulated variables all the controlled variables stabilise on some constant values. The step-response model is represented by D matrices of dimensionality $n_y \times n_u$

$$\mathbf{S}_p = \begin{bmatrix} s_p^{1,1} & \dots & s_p^{1,n_u} \\ \vdots & \ddots & \vdots \\ s_p^{n_y,1} & \dots & s_p^{n_y,n_u} \end{bmatrix} \quad (4)$$

for $p = 1, \dots, D$, where D is a horizon of process dynamics.

Assuming that there are no constraints imposed on process inputs and outputs, optimisation of the DMC cost-function (2) subject to the future increments of the manipulated variables (1) gives the optimal control increments

$$\Delta \mathbf{u}(k) = \mathbf{K}(\mathbf{y}^{\text{sp}}(k) - \mathbf{y}(k) - \mathbf{G}^{\text{p}} \Delta \mathbf{u}^{\text{p}}(k)) \quad (5)$$

The matrix of dimensionality $n_u N_u \times n_y N$ is

$$\mathbf{K} = (\mathbf{G}^{\text{T}} \mathbf{M} \mathbf{G} + \mathbf{\Lambda})^{-1} \mathbf{G}^{\text{T}} \mathbf{M} \quad (6)$$

where the matrices composed of the tuning coefficients of the algorithm, $\mu_n \geq 0$ and $\lambda_n > 0$

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_0 & & \\ & \ddots & \\ & & \mathbf{M}_0 \end{bmatrix}, \quad \mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_0 & & \\ & \ddots & \\ & & \mathbf{\Lambda}_0 \end{bmatrix} \quad (7)$$

are of dimensionality $n_y N \times n_y N$ and $n_u N_u \times n_u N_u$, respectively. The vector $\mathbf{y}(k)$, of length $n_y N$, represents the measurements of all process outputs for the current sampling instant and the vector

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

is of length $n_u(D-1)$ and it represents increments of all the manipulated variables applied to the process in the past. The matrices $\mathbf{G}$ and $\mathbf{G}^{\text{p}}$, of dimensionality $n_y N \times n_u N_u$ and $n_y N \times n_u(D-1)$, respectively, are comprised of step-response coefficients of the model [1], [2], [11]. In the DMC algorithm it is also possible to find the increments of the manipulated variables for the current sampling instant, $\Delta u(k|k)$, as a linear function of the current step-point values, the measured values of the current process outputs and some past values of the process inputs

$$\Delta u(k|k) = K^e(y^{\text{sp}}(k) - y(k)) - \sum_{i=1}^{D-1} K_i^u \Delta u(k-i) \quad (9)$$

The matrices K^e and K_i^u are calculated off-line from the step-response model and the weighting coefficients μ_n and λ_n [11].

IV. OPTIMISATION-BASED TUNING OF DMC ALGORITHM

A. Tuning Objective

During tuning the DMC algorithm the following parameters are found: the prediction and control horizons, N and N_u , respectively (positive integer numbers) as well as the coefficients μ_n and λ_n (real numbers) that comprise the matrices $\mathbf{M}_0$ and $\mathbf{\Lambda}_0$ (3). The objective function for optimisation-based tuning is defined as the sum of squared differences (SSE) between the set-points, $y^{\text{sp}}(k)$, and the measured values of process output variables, $y(k)$, along the simulation horizon T

$$\text{SSE} = \sum_{k=1}^T \|y^{\text{sp}}(k) - y(k)\|^2 \quad (10)$$

The resulting Mixed Integer Nonlinear Programming (MINLP) optimisation problem is

$$\min_{N, N_u, \mu_1, \dots, \mu_{n_y}, \lambda_1, \dots, \lambda_{n_u}} \text{SSE} \quad (11)$$

subject to

$$N, N_u \in \mathbb{N} \setminus \{0\} \quad (12)$$

$$N \geq N_u \quad (13)$$

$$\mu_1, \dots, \mu_{n_y} \geq 0 \quad (14)$$

$$\lambda_1, \dots, \lambda_{n_u} > 0 \quad (15)$$

The above optimisation problem is solved only off-line.

B. Compared Optimisation Methods

The formulated optimisation problem (11) cannot be solved by classical algorithms since the cost-function is likely to have many shallow local minima. Therefore, global optimisation methods are used. The majority of them have a similar work flow. At the beginning, a fixed number of random vectors are generated. For each of them (called “particles”, “individuals”, etc.) the value of the minimised cost-function is evaluated. Next, the coordinates of vectors are updated according to some algorithm-specific policy. The consecutive iterations of the algorithm (called epochs or generations) which consist of this two actions are repeated until a termination criterion is reached. In this work 4 optimisation methods are used.

1) *Particle Swarm Optimization*: PSO is a non-gradient evolutionary algorithm which is used to wide class optimisation problems [3]. The change of position (originally named “velocity”, v_i) for each particle in the i -th generation is obtained as weighted sum of “velocity” from the previous generation (v_{i-1}) as well as disturbed (uniformly distributed random vectors r with elements from $[0; 1]$) differences between the actual position relative to the best known position of particle (p) and the best known position achieved by whole swarm (b)

$$v_i = c_0 v_{i-1} + c_1 r_1 (p_i - x_{i-1}) + c_2 r_2 (b_i - x_{i-1}) \quad (16)$$

$$x_i = x_{i-1} + v_i \quad (17)$$

The weights c , according to [8], must satisfy the conditions

$$0 < (c_1 + c_2) < 4, \frac{c_1 + c_2}{2} - 1 < c_0 < 1 \quad (18)$$

in order to make algorithm converge to a stable equilibrium point.

2) *Firefly Algorithm*: FA is an evolutionary strategy inspired by behaviour of fireflies [12]. The change of position of every particle is based on a function called “brightness” (which can be obtained for example as the objective function) and the distance between individuals (d_{xy}). It is also randomly noised (by the vector r from unit interval of adequate dimension). In every generation, the positions are updated according to

$$x_i = x_{i-1} + c_0 \exp(-c_1 d_{xy}^2) (x_{i-1} - y_{i-1}) + c_2 r \quad (19)$$

in each pair of the fireflies, denoted as x and y , only if the brightness function value for x is better than for y . There is no fixed formula to determine the coefficients c . In [12] it is recommended to set $c_0 = 1$, $c_1 \in [0.1; 10]$ and $c_2 \in [0; 1]$ to get satisfactory convergence rate for typical applications.

3) *Grey Wolf Optimizer*: The GWO algorithm is based on behaviour of the grey wolves’ pack [7]. A major impact for specifying the position of each individual have three of them, marked as α , β and δ . They are chosen from the whole population as these which provide three best values of the cost-function. This choice is updated in every iteration of the algorithm.

The position of each particle in the following generations is determined as result of a complex formula involving positions of three best individuals, the position between the actual “wolf” in relation to them and the random factors. A noteworthy behaviour is that the influence of the last component is linearly reduced to zero during the optimisation process.

For the initial iterations particles tend to explore searched space and finally converge to a sub-optimal point which an approximate position is revealed by α , β and δ .

4) *Jaya Algorithm*: Jaya is a relatively simple algorithm which accomplishes rule that a single particles in each generation should be moved toward the best solution and evade from the worst [10]. The best and the worst individuals are b and w , respectively. The formula for updating positions of particles is

$$x_{i+1} = x_i + r_1 (b_i - x_i) - r_2 (w_i - x_i) \quad (20)$$

where r are random vectors with the coordinates from $[0; 1]$. A characteristic feature of the Jaya, unlike the previous considered algorithms, is that it does not require any parameters, which need be provided for a proper operation.

V. SIMULATION RESULTS

A. Implementation Issues

All four considered global optimisation algorithms have been implemented in the C language. The separated objective function allows to use them as an optimisation engine for almost any computational problem which can be formulated as minimisation of a scalar function. In particular, through the use of standard library functions, for instance, file operations or invoked external tools can be used. This feature leads to high flexibility of a designed software. In case of the typical engineering problems (minimisation of errors in control systems, optimisation of mechanical structures, etc.), developed solution can act as a higher level optimisation engine while an actual value of goal function would be computed by a highly specialised software (e.g. simulator, numerical solver).

In this work a few matrix operations must be implemented. Consequently, one of prerequisites to perform this task is to write or use a ready-made matrix computational engine. The commonly used in control system engineering MATLAB software has been chosen. Since it offers simple to use the C API, it allows to made both simple and more complicated operations to evaluate the value of the objective function. The final value can be obtained, for example, as a variable loaded directly from the MATLAB workspace.

The designed C toolkit allows to apply simple cubic constraints defined on the optimisation decision variables. An unfeasible candidate, depending on the user choice, can be brought back to feasible area by limiting candidate element by element or not tested as potential solution while value of objective function is set as $+\infty$. This approach is called as the “death penalty”[4] and is also used when inequality constraints enforced on linear combination (or more sophisticated function) of candidate’s elements are not fulfilled. These set of the sub-routines allows to meet all required constraints (12)–(15).

The sum of squared errors between the set-point trajectories and the output values of the simulated DMC-controlled process (10) is used as the cost-function. Both the process and the controller are simulated in discrete-time domain as a batch job for the MATLAB engine.

B. Process Description

The considered MIMO process has four inputs and three outputs. It is described in the continuous-time domain by the following transfer function matrix

$$G(s) = \begin{bmatrix} \frac{2.5}{3s+1} & \frac{1.5}{12s+1} & \frac{0.1}{3.5s+1} & \frac{0.9}{2.5s+1} \\ \frac{1.9}{1.2s+1} & \frac{0.8}{20.2s+1} & \frac{1.2}{4s+1} & \frac{0.25}{3.9s+1} \\ \frac{0.25}{1.8s+1} & \frac{1.1}{2.5s+1} & \frac{1.7}{15.5s+1} & \frac{0.85}{3.1s+1} \end{bmatrix} \quad (21)$$

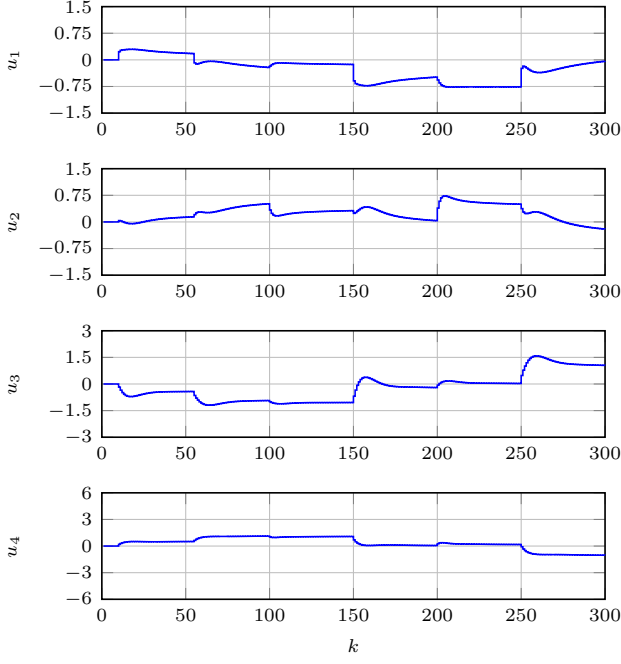


Fig. 1. Process trajectories obtained when the parameters of the DMC algorithm are tuned experimentally

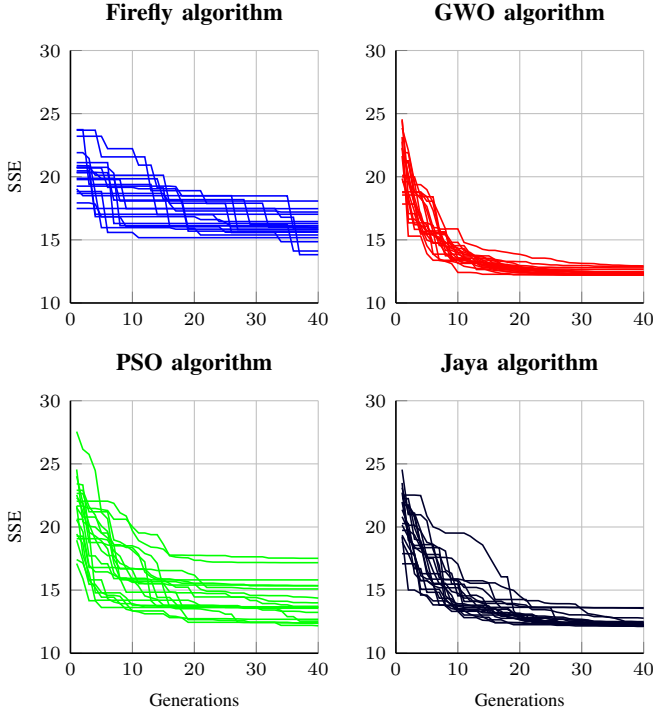
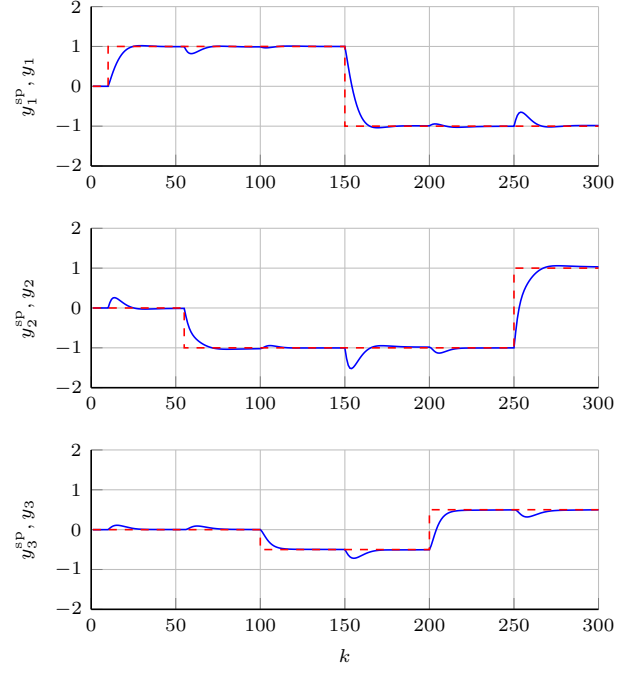


Fig. 2. Convergence of four compared optimisation algorithms

Using the zero-order holder with the sampling time $T_s = 0.5$ s, the discrete-time transfer function matrix is obtained

$$G(z) = \begin{bmatrix} \frac{0.3838}{z-0.8465} & \frac{0.0612}{z-0.9592} & \frac{0.0133}{z-0.8669} & \frac{0.1631}{z-0.8187} \\ \frac{0.6474}{z-0.6592} & \frac{0.0196}{z-0.9756} & \frac{0.1410}{z-0.8825} & \frac{0.0301}{z-0.8797} \\ \frac{0.0606}{z-0.7575} & \frac{0.1994}{z-0.8187} & \frac{0.0540}{z-0.9683} & \frac{0.1266}{z-0.8510} \end{bmatrix} \quad (22)$$

Furthermore, it is assumed that limits of the input variables are

$$[-1.5; -1.5; -3; -6]^T \leq \mathbf{u} \leq [1.5; 1.5; 3; 6]^T \quad (23)$$

When the controller tries to apply manipulated variables which do not belong to the allowed range, they are bounded by the process according to those limits.

For all considered DMC controllers the horizon of process dynamics, D , is constant, set to 150 steps.

C. Controller Performance Before Tuning

To determine the reference solution, an initial simulation has been performed with preselected parameters of the controller. The prediction and control horizons have been set to $N = 10$ and $N_u = 1$, respectively. The weighting coefficients have been set to $\lambda_1 = \dots = \lambda_4 = 10$ and $\mu_1 = \mu_2 = \mu_3 = 1$. The resulting process trajectories are presented in Fig. 1. Unfortunately, the obtained quality of control is unsatisfactory. Considerable differences between the set-points and the values of the process controlled variables are obtained when the set-points are changed. Furthermore, since the process has very strong cross-couplings, such control errors are present for all combinations of input-output channels. The controller is unable to compensate fast for such big errors. Considering the trajectories of the manipulated variables one may clearly find it out that their values calculated by the experimentally tuned controller do not reach the limits defined by Eq. (23).

D. Controller Performance After Tuning

In order to solve the optimisation problem (10) subject to the constraints (12)-(15), four global optimisation methods

TABLE I
THE BEST RESULT OBTAINED BY CONSIDERED OPTIMISATION ALGORITHMS

Tuning method	N	N_u	Λ_0	M_0	SSE
Experimental tuning	10	1	diag(10, 10, 10, 10)	diag(1, 1, 1)	39.77
FA optimisation	114	53	diag(1.206×10^1 , 1.336×10^1 , 8.324×10^{-1} , 3.281×10^{-2})	diag(52.32, 66.45, 83.36)	13.83
GWO optimisation	60	39	diag(8.713×10^{-1} , 2.891×10^0 , 8.676×10^{-1} , 5.236×10^{-3})	diag(100, 92.60, 100)	12.21
Jaya optimisation	72	58	diag(1.565×10^0 , 2.200×10^{-1} , 5.500×10^{-1} , 10^{-4})	diag(100, 99.55, 99.96)	12.12
PSO optimisation	14	13	diag(10^{-4} , 1.394×10^0 , 3.463×10^{-1} , 10^{-4})	diag(100, 100, 100)	12.17

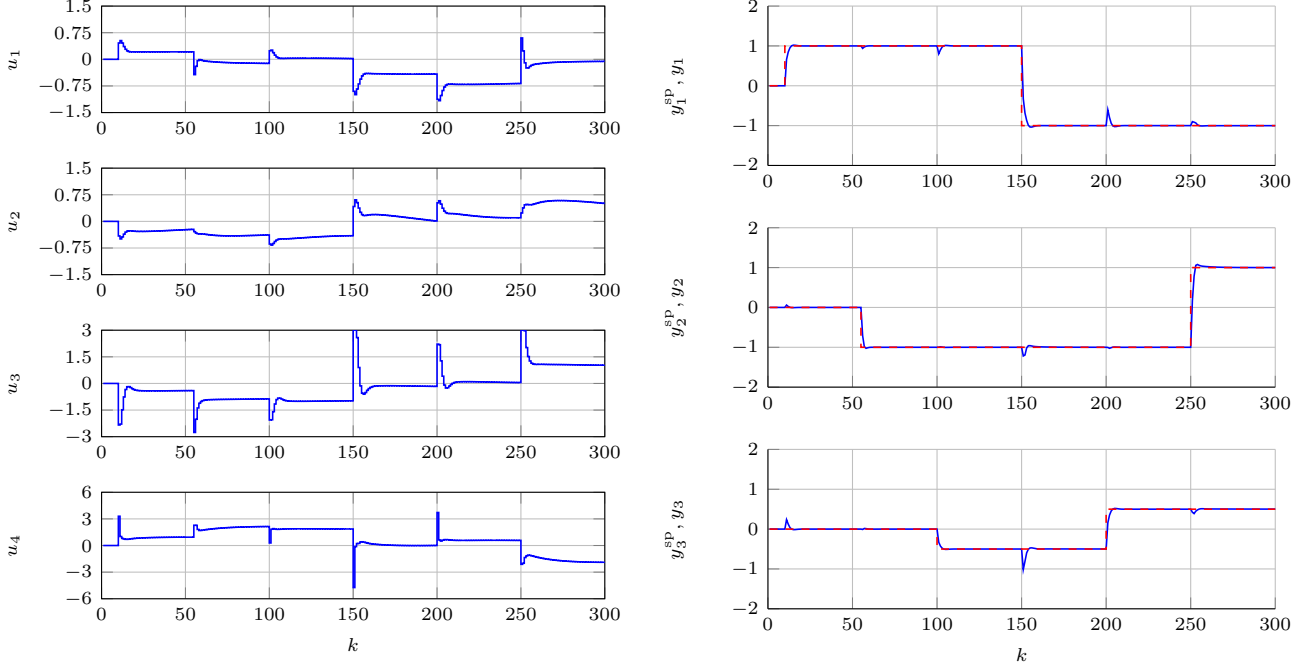


Fig. 3. Process trajectories obtained when the parameters of the DMC algorithm are tuned by the Firefly Algorithm (SSE=13.83).

shortly characterised in Section IV.B are used. Additionally, to reasonably bound searched space the constraints are used

$$3 \leq N \leq 150 \quad (24)$$

$$1 \leq N_u \leq 70 \quad (25)$$

$$0 \leq \mu_i \leq 100 \quad (26)$$

$$10^{-4} \leq \lambda_i \leq 100 \quad (27)$$

For each algorithm optimisation is repeated 20 times (from different initial solutions). For each algorithm 40 generations and 10 individuals are set which results in 400 objective function evaluations per one optimisation process. The rate of convergence are presented in Fig. 2. The solutions obtained by the Firefly Algorithm and the PSO strategy differ from each other in single attempts. The solutions reached by the Jaya and the GWO algorithms are much closer (in terms of the objective function value) for different initial points. Nevertheless, almost all results given by the global optimisation algorithms are characterised by at least twice smaller value of SSE in comparison with experimental tuning. The best solutions obtained by the all compared algorithms are listed in Table I. It can be

observed that solutions given by all algorithms return greater μ and lower λ values than in case of experimental tuning. Especially, for the Jaya and the PSO algorithms which give the best SSE values, the final parameters values are close or equal to their constraints. Such a behaviour can be explained that the algorithm strives to reduce the “penalty factor” caused by fast changes of input signals which is entangled by the ratio of the μ and λ coefficients. It is interesting to note that the control and prediction horizon are longer than in the case of experimental tuning. This tendency causes improved control quality but also leads to more affordable computation process by determining smaller matrices which are used to execute the DMC algorithm.

Process trajectories obtained when the parameters of the DMC algorithm are tuned by the Firefly and Jaya algorithms are depicted in Figs. 3 and 4, respectively. In comparison with the trajectories obtained by means of experimental tuning (Fig. 1), the responses of the controlled process are considerable faster and overshoots are smaller.

The fast process output trajectories are directly related to variability of inputs variables. Unlike experimental tuning,

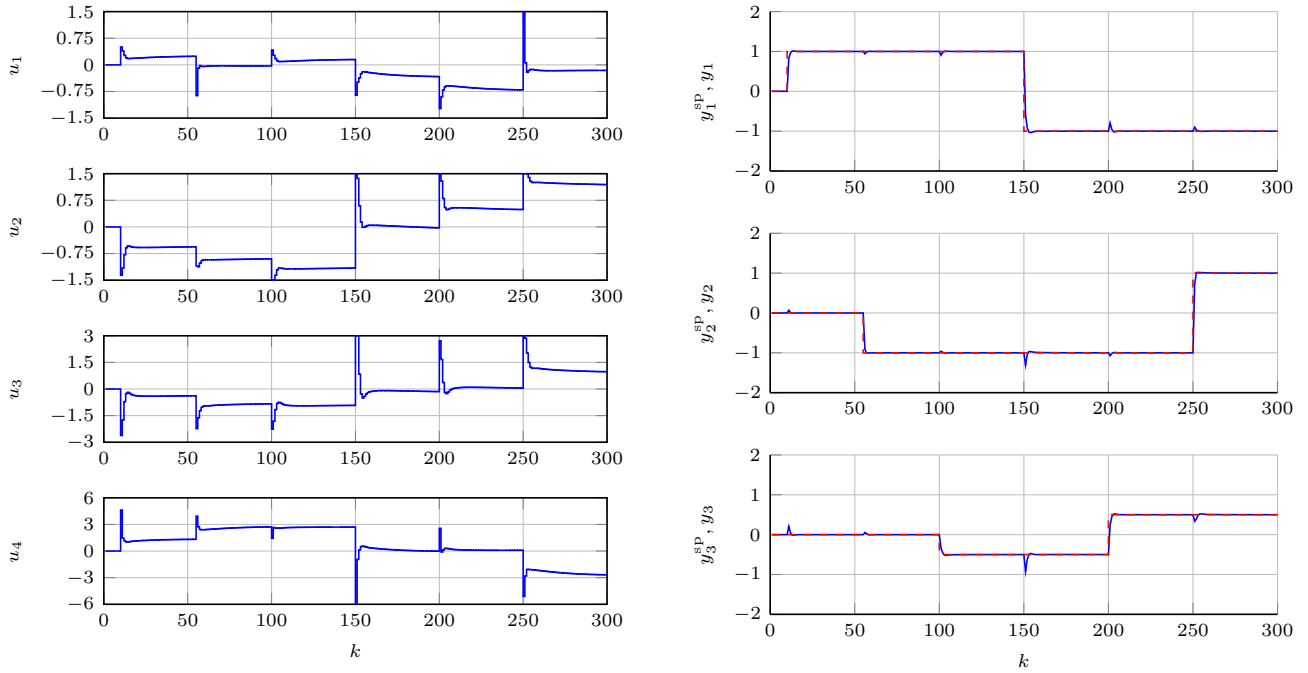


Fig. 4. Process trajectories obtained when the parameters of the DMC algorithm are tuned by the Jaya algorithm (SSE=12.12).

when the controller is tuned by the optimisation algorithms, significant peaks are present when the set-points change. In spite of that, the values of input variables never reach levels of process limits (23), although these are not directly “visible” for the optimisation procedure in contrast to the constraints defined by Eqs. (24)-(27). One may guess that if the constraints imposed on the rate of input change of the manipulated variables are present, the optimisation algorithms are able to find the λ and μ which could lead to the required quality of control by generating inputs in bounds tolerated by the process. A possible disadvantage of the obtained solution is the fact that the manipulated variables change fast, although the limits (24)-(27) are satisfied. It is because no rate constraints have been taken into account.

VI. CONCLUSION

In this work four global optimisation algorithms have been used to tune parameters of the DMC algorithms applied to a MIMO process. It must be stressed that the experimental tuning in the MIMO case is not straightforward. The four optimisation methods have been compared in terms of the ability to find the best solution and convergence. In general, the best solutions obtained by the following three methods: the Grey Wolf Optimiser, the Particle Swarm Optimisation and the Jaya algorithm give very similar control quality in terms of the value of the minimised cost-function whereas the Firefly algorithm gives somehow worse result. It must be stressed, however, that the Grey Wolf Optimiser and the Jaya algorithm are characterised by very good convergence. From different initial points they converge fast, the solutions are quite close.

Although in this work the values and the rate of change of the manipulated variables are not taken into account as the constraints during optimisation, the magnitude constraints are satisfied by selecting proper ranges of optimisation decision variables. Nevertheless, inclusion of all the constraints in the optimisation problem is possible and may be considered in the future research. The approach discussed in this work is not limited to the DMC algorithm, it may be also used in the case of more advanced nonlinear MPC algorithms [5].

REFERENCES

- [1] E. F. Camacho, C. Bordons, Model Predictive Control. Springer, London, 1999.
- [2] C. R. Cutler, B. L. Ramaker, Dynamic matrix control—a computer control algorithm. Proceedings of the AIChE National Meeting, Houston, 1979.
- [3] J. Kennedy, R. Eberhart, Particle swarm optimization. IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942–1948.
- [4] O. Kramer, A review of constraint-handling techniques for evolution strategies. Applied Computational Intelligence and Soft Computing, vol. 2010, Article ID 185063, 2010.
- [5] M. Ławryńczuk, Computationally Efficient Model Predictive Control Algorithms: A Neural Network Approach, Studies in Systems, Decision and Control, vol. 3, Springer, Heidelberg, 2014.
- [6] J. M. Maciejowski, Predictive Control with Constraints. Prentice Hall, Harlow, 2002.
- [7] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer. Advances in Engineering Software, 69, pp. 46–61, 2014.
- [8] R. E. Perez, K. Behdinan, Particle swarm approach for structural design optimization. Computers & Structures, 85, pp. 1579–1588, 2007.
- [9] S. J. Qin, T. A. Badgwell, A survey of industrial model predictive control technology. Control Engineering Practice, 11, pp. 733–764, 2003.
- [10] R. Venkata Rao, Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, 7, pp. 19–34, 2016.
- [11] P. Tatjewski, Advanced Control of Industrial Processes, Structures and Algorithms. Springer, London, 2007.
- [12] X. S. Yang, Nature-inspired Metaheuristic Algorithms. Luniver Press, Frome, 2008.