Particle Swarm Optimisation for Model Predictive Control Adaptation

Květoslav Belda[1] Lenka Kuklišová Pavelková[2]
department of Adaptive Systems
The Czech Academy of Sciences, Institute of Information Theory and Automation
Pod Vodárenskou věží 4, 182 00 Prague 8, Czech Republic
[1] belda@utia.cas.cz, https://orcid.org/0000-0002-1299-7704
[2] pavelkov@utia.cas.cz, https://orcid.org/0000-0001-5290-2389

***Abstract*—This paper is focused on parameter identification for Model Predictive Control (MPC). Two identification techniques for parameters of Auto Regressive model with eXogenous input (ARX model) are considered: namely the identification based on Particle Swarm Optimisation (PSO) and Least Square (LS) method. PSO is investigated and LS is presented in square-root form as a reference method for comparison, respectively. The following points are elaborated and discussed: i) parameters' estimation of ARX model; ii) design of PSO and LS procedures; iii) design of data-driven MPC algorithm in square-root form; iv) concept of possible use of PSO for semiautomatic fine tuning or retuning of MPC parameters. The proposed theoretical procedures are demonstrated using simply reproducible simulation experiments. Application possibilities are discussed towards robotics and mechatronics.**

***Index Terms*—data-driven modelling, parameter estimation, particle swarm optimisation, predictive control**

I. Introduction

The introduction of modern control approaches into practice requires a detailed knowledge of controlled system. Despite the existence of various identification methods, the model composition and its possible adaptation still represents a challenge. For many systems such as mechanical ones, it is sufficient to define the relevant model of the controlled system at the initial phase and consider it to be constant during the control process. Such a model can be obtained by a thoroughgoing mathematical and physical analysis.

To take into account time-varying model parameters, it is possible to implement some sophisticated identification method into the control algorithm, but this can increase the algorithm complexity (more advanced mathematical operations such as matrix inversion) and demands on the used target processor platforms. Thus, it is useful to have a simpler or sub-optimal identification algorithm that can run either fully in parallel or just occasionally to improve the model used in control design.

One of such algorithms is Particle Swarm Optimisation (PSO). It is a promising optimisation algorithm due to a simple implementation, only three setting parameters and flexibility in combination with other optimisation algorithms [1].

This work was supported by The Czech Academy of Sciences, Institute of Information Theory and Automation under project No. 23-04676J of the Czech Science Foundation: Controllable gripping mechanics: Modelling, control and experiments.

In [2], Model Predictive Control (MPC) with PSO for ARX model with varying parameters is proposed for greenhouse applications such as optimisation of air temperature control. The paper [3] presents a method for tuning a MPC-based quadrotor trajectory using cooperative PSO, where two swarms exchange information to more efficiently explore the search space and find tuning parameter values that improve trajectory tracking performance. Applications in photovoltaic and smart systems are investigated in [4], [5]. These examples represent specific potential for real-world applications in addition to the theoretical PSO utilisation such as in [6], [7].

In this paper, we investigate PSO principle within MPC to detect changes in model parameters. In contrast to the works cited above, such a task represents an online computation in fast MPC sampling. PSO can also be used to identify isolated parameters. For instance, if some parameter of a physically based model cannot be simply determined by a measurement such as mechanical parameters (e.g. moments of inertia, the positions of centres of gravity) or electro-mechanical parameters (e.g. electromagnetic constant), then some auxiliary identification is needful. The proposed PSO algorithm is compared with the standard Least Square (LS) method, presented in a computationally optimised square-root form [8].

The paper is organised as follows. Firstly, a model used for control design is defined in Section II. Then, the principle of PSO procedure is introduced in Section III. The comparative LS procedure is outlined in Section IV. A design of MPC is presented in Section V. Finally, Section VI and Section VII summarise and demonstrate introduced theoretical results, respectively. The paper concludes with a summary and outline of possible future work in Section VIII.

II. Preliminaries

Mathematical models used in control design are either based on physical analysis (rational, theoretically based, so-called white box models) or on targeted measurements (data-driven approach, black box models). The former, 'analytical' models have usually a form of differential equations. They are subsequently discretised and transformed in state-space forms with determined physical relations. In contrast, the latter, 'empirical' models use outer description expressed by difference equations with link to discrete state-space models.

For purposes of parameter identification or parameter change detection, the Auto Regressive model with eXogenous input (ARX model) is considered as representative data-driven model of the controlled system in the following form:

$$\begin{aligned} y_k &= \sum_{i=0}^{n} b_i\, u_{k-i} - \sum_{i=1}^{n} a_i\, y_{k-i} + e_k \\ &= \vartheta_k\, f_k + e_k \\ \vartheta_k &= [\, b_0 \;\; b_1 \;\; \cdots \;\; b_n \;\; -a_1 \;\; -a_2 \;\; \cdots \;\; -a_n \,] \\ f_k &= [\, u_k \;\; u_{k-1} \;\; \cdots \;\; u_{k-n} \;\; y_{k-1} \;\; \cdots \;\; u_{k-n} \,]^T \end{aligned} \tag{1}$$

where n is the order of the system, y and u are its outputs and inputs and e is an error - noise influencing the output y.

III. PSO Algorithm

In PSO, a swarm of S particles moves in a D-dimensional search space and search for an optimal solution. Each particle represents a candidate solution to the solved optimisation problem and keeps a current velocity vector v_m and a current position vector x_m, where index $m = 1, 2, \cdots, S$.

The PSO process starts by randomly initialising all vectors v_m and x_m. In every iteration, the best solution of each particle as well as the best solution of the swarm is updated based on the value of some fitness function f. The PSO process stops when reaching a global optimum or maximum of iterations.

In this paper, we use a basic PSO variant [9] that consists in the following evolution of m-th velocity and particle position at the i-th iteration:

$$\begin{aligned} v_{m,\,i+1} = \;& w_i\, v_{m,\,i} + c_{1,i}\, r_1 (p_{b\,m,\,i} - x_{m,\,i}) \\ & + c_{2,i}\, r_2\, (g_{b\,i} - x_{m,\,i}) \end{aligned} \tag{2}$$

$$x_{m,\,i+1} = \; x_{m,\,i} + v_{m,\,i+1} \tag{3}$$

where i denotes iteration number, w represents inertia weight, c_1 is a cognitive acceleration parameter, c_2 is a social acceleration parameter, $r_1, r_2 \in \langle 0, 1 \rangle$ are random numbers, $p_{b\,m,i}$ is the best location found by the m-th particle and $g_{b\,i}$ is the global best location of all the particles at the i-th iteration.

$$p_{b\,m,\,i} = \begin{cases} p_{b\,m,\,i-1}, & \text{if } f(x_{m,\,i}) \geq f(p_{b\,m,\,i-1}) \\ x_{m,\,i}, & \text{if } f(x_{m,\,i}) < f(p_{b\,m,\,i-1}) \end{cases} \tag{4}$$

$$g_{b\,i} = \arg\min_{x_{m,\,i}} f(x_{m,\,i}), \;\; 1 \leq m \leq S \tag{5}$$

The tuning parameters of PSO algorithm, i.e. c_1, c_2 and w maintain the balance between global discovery and local detection. We have used the following approved setting [10]:

- time-varying acceleration coefficients

$$c_{1,i} = 2.5 - 2\, i/nit \tag{6}$$

$$c_{2,i} = 0.5 + 2\, i/nit \tag{7}$$

  where nit denotes the total number of iterations.
- linearly decreasing inertia weights

$$w_i = w_{\max} - (w_{\max} - w_{\min})\, i/nit \tag{8}$$

  where $w_{\max}$ and $w_{\min}$ denote the maximal and minimal inertia weight, respectively.

Algorithm summary - steps 1) - 7):

1) Initialise the PSO algorithm parameters, i.e., the swarm size S, inertia weight w, acceleration coefficients c_1, c_2, maximum number of iterations nit, and maximum velocity V_{max}.
2) Set a swarm that has S particles.
3) Initialise the position $x_{m,1}$ and velocity $v_{m,1}$, and $p_{b\,m,1}$ of each particle (m = 1, 2, $\cdots$, S); and initialise $g_{b\,1}$ of the swarm.
4) Calculate each particle's fitness value.
5) Update the $p_{b\,m,i}$ of each particle and $g_{b\,i}$ of the swarm.
6) Update the velocity $v_{m,i}$ and the position $x_{m,i}$ of each particle given by (2) and (3).
7) If maximum iterations are met or fitness value reaches the threshold then algorithm is in **end**, else it **go**es to the step 4).

The algorithm itself provides iteration on a specific set of data and selects the best one individuals for specific generated parameters. The individuals form the basis of searched model parameters.

IV. LS Algorithm

For comparison, we have chosen LS algorithm in square-root form [8]. Let us proceed from (1) with the reversed order of data vector f_k and model parameters ϑ_k:

$$y_k = f_k^T\, \vartheta_k^T + e_k \tag{9}$$

A relevant set of equations corresponding to the number of parameters is composed as:

$$\mathrm{y}_k = F_k\, \vartheta_k^T + \mathrm{e}_k \tag{10}$$

$$\mathrm{e}_k = \mathrm{y}_k - F_k\, \vartheta_k^T = [\, F_k \;\; \mathrm{y}_k \,]\, [\, -\vartheta_k \;\; 1 \,]^T \tag{11}$$

$$\begin{aligned} F_k &= [\, f_k^T \;\; f_{k-1}^T \;\; \cdots \;\; f_{k-2\,n}^T \,]^T \\ \mathrm{y}_k &= [\, y_k \;\; y_{k-1} \;\; \cdots \;\; y_{k-2\,n} \,]^T \\ \mathrm{e}_k &= [\, e_k \;\; e_{k-1} \;\; \cdots \;\; e_{k-2\,n} \,]^T \end{aligned}$$

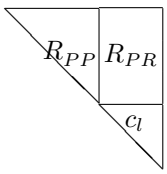
Then, the cost function can be defined as follows

$$J_k = \mathrm{e}_k^T\, \mathrm{e}_k \to \min_{\vartheta_k^T} J_k = ||\bar{J}_k||_2^2 = ||e_k||_2^2 \tag{12}$$

The criterion (12) can be minimised over searched parameters ϑ_k using orthogonal-triangular decomposition as follows:

$$\bar{J}_k = Q\, [\, F_k \;\; \mathrm{y}_k \,]\, [\, -\vartheta_k \;\; 1 \,]^T = [\, 0 \;\; c_l \,]^T \tag{13}$$

$$Q\, [\, F_k \;\; \mathrm{y}_k \,] = R = \tag{14}$$



$$-R_{PP}\, \vartheta_k + R_{PR} = 0 \tag{15}$$

where individual sub-matrices are defined as follows:

$$R_{PP} = R_{1:2\,n,\,1:2n} \tag{16}$$
$$R_{PR} = R_{1:2\,n,\,2\,n+1} \tag{17}$$
$$c_l = R_{2\,n+1,\,2\,n+1} \tag{18}$$

Then, the parameter estimates can be obtained solving (15).

Initialisation of LS algorithm is realised as follows:

$$R = k_0\, I_{2\,n+1} \tag{19}$$

where k_0 is initial diagonal element and I is identity matrix of order $2\,n+1$, with specific set of initial values of parameters.

V. MPC Adaptive Algorithm

This section introduces specific model reorganisation and its application in a structure of equations of predictions. Let us consider the deterministic part of (1) with $b_0 = 0$ for usual cases of mechanical systems in predictive form:

$$\hat{y}_{k+1} = \sum_{i=1}^{n} b_i\, u_{k-i+1} - \sum_{i=1}^{n} a_i\, y_{k-i+1} \tag{20}$$

It can be rearranged into a specific observable canonical form of a state-space model:

$$\begin{bmatrix} y_{k-n+2} \\ \vdots \\ y_k \\ \hat{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \\ 0 & & & 1 \\ -a_n & & \cdots & -a_1 \end{bmatrix} \begin{bmatrix} y_{k-n+1} \\ \vdots \\ y_{k-1} \\ y_k \end{bmatrix} + \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \\ b_n & \cdots & b_1 \end{bmatrix} \begin{bmatrix} u_{k-n+1} \\ \vdots \\ u_{k-1} \\ u_k \end{bmatrix} \Rightarrow$$

$$x_{k+1} = A\, x_k \;+\; B_0\, u_k \tag{21}$$
$$y_k = [\,0\ \cdots\ 0\ 1\,]\, x_k \quad \Rightarrow \quad y_k = C\, x_k \tag{22}$$

This rearrangement is provided in every time step according to parameters obtained from identification procedure.

A. Structure of Equations of Predictions

Structure of equations of predictions follows from (21) and (22), on which recursive substitution and expansion over prediction horizon N is applied:

$$\begin{bmatrix} \hat{y}_{k+1} \\ \vdots \\ \hat{y}_{k+N} \end{bmatrix} = \begin{bmatrix} C\,A \\ \vdots \\ C\,A^N \end{bmatrix} \begin{bmatrix} y_{k-n+1} \\ \vdots \\ y_k \end{bmatrix} + \begin{bmatrix} C\,B_0 & \cdots & 0 \\ \ddots & & \vdots \\ C\,B_{N-1} & & \end{bmatrix} \begin{bmatrix} u_{k-n+1} \\ \vdots \\ u_{k+N-1} \end{bmatrix} \tag{23}$$

$$\hat{y} = \bar{f} + \bar{G}\, u_{k-n+1\,:\,k+N-1} \tag{24}$$
$$\hat{y} = \bar{f} + G_p\, u_{k-n+1\,:\,k-1} + G\, u_{k\,:\,k+N-1} \tag{25}$$
$$\hat{y} = f + G\, u_{k\,:\,k+N-1} \tag{26}$$

where initial matrix $\bar{G}$ represents forced response to mixture of control actions both known past and unknown future.

Rows of $\bar{G}$ or matrices $B_0,\, B_1,\, \cdots, B_N$ are constructed as:

$$B_1 = [\,A\,B_0\ \ 0\,] + [\,0\ \ B_0\,] \tag{27}$$
$$B_2 = [\,A\,B_1\ \ 0\,] + [\,0\ \ 0\ \ B_0\,] \tag{28}$$
$$\vdots = \quad \vdots$$
$$B_{N-1} = [\,A\,B_{N-2}\ \ 0\,] + [\,0_{n,N-1}\ \ B_0\,] \tag{29}$$

Therefore, this matrix can be decomposed into two sub-matrices: G_p (for known past) and G (for unknown future). Then, G_p with known path control actions $u_{k-n+1\,:\,k-1}$ can be connected to the known free response $\bar{f}$ to obtain the known past f separated from unknown forced response $G\, u_{k\,:\,k+N-1}$.

B. Minimisation of the Cost Function

Let the criterion with square-root decomposed cost function be defined as follows:

$$\mathcal{J}_k = \min_{u_{k:k+N-1}} J^T\, J \tag{30}$$

$$J = \begin{bmatrix} Q_{yw} & 0 \\ 0 & Q_u \end{bmatrix} \begin{bmatrix} \hat{y} - w \\ u \end{bmatrix} = \underbrace{\begin{bmatrix} Q_{yw}\, G & Q_{yw}\,(w - f) \\ Q_u & 0 \end{bmatrix}}_{\mathcal{A} \qquad\quad b} \begin{bmatrix} u \\ -1 \end{bmatrix} \tag{31}$$

Then, square-root J (31) can be minimised using orthogonal-triangular decomposition similarly to LS (14):

$$Q\,[\,\mathcal{A}\ \ b\,] = R = \begin{array}{|ll} R_1 & c_1 \\ & c_z \end{array} \tag{32}$$

$$-R_1\, u + c_1 = 0 \tag{33}$$

where the vector c_z represents a loss vector. Its Euclidean norm $\|c_z\|_2$ corresponds to the square-root of the minimum of the cost function (30), i.e., $\mathcal{J}_k = c_z^T c_z$.

Note that only the first element u_k from the computed vector u (33) is used for control. Furthermore, note that the presented MPC adaptive algorithm with varying parameters $\vartheta_k = [\,b_0\ b_1\ \cdots\ b_n\ -a_1\ -a_2\ \cdots\ -a_n\,]$ can be rearranged into incremental form in control actions point of view. In such case, the penalisation of control is applied only to increments of control actions. It is indicated in the corresponding cost function:

$$J_k = \begin{bmatrix} Q_{yw} & 0 \\ 0 & Q_{\Delta u} \end{bmatrix} \begin{bmatrix} \hat{y} - w \\ \Delta u \end{bmatrix} \tag{34}$$

The incremental property, in view of control actions, can solve overshoots and constraints in general. The incremental algorithm concept is discussed in [11], [12] and [13].

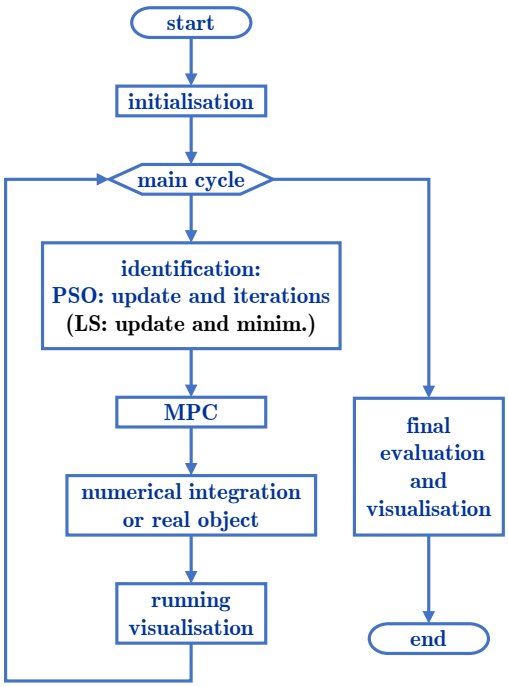


Fig. 1. Flow chart of used procedures.

VI. Main Results

The paper presents a PSO procedure applied to parameter estimation implemented in an MPC design as shown in the flowchart in Fig. 1. Block initialisation represents the setup of necessary parameters for identification and control as well, see Table I. There is also an initialisation of visualisation and other supporting subroutines.

The main cycle loop represents one discrete time step k of simulation, where true time is $t = k\,T_s$. The identification block represents the same position for both PSO (Section III) and comparative LS (Section IV) algorithms. PSO and LS algorithms are interchangeable. They contain a sliding data window utilised in an appropriate identification procedure. PSO can be executed in specific times only or can run continuously like LS.

Then, MPC block follows (Section V). This block includes computation of control actions (33) with equations of predictions (26) based on data-driven model (20). The application of MPC actions to the controlled system is simulated using numerical integration (MATLAB ode45 function: Dormand-Prince method [14]) or this position represents a real object.

The overview flowchart (Fig. 1) contains also supporting blocks for running and final visualisations and for evaluation. The corresponding MATLAB code of the main PSO cycle is in Alg. 1, where parameter setup is on lines 3 and 4, update of velocity and particle position at the i-th iteration on lines 6-10, data window shift on lines 14 and 15, prediction error on line 17, computation of fitness function is on line 18, evaluation over one iteration is on lines 21-28 and the best global values are selected on line 31.

Alg. 1. MATLAB code of main PSO cycle.

```
1  for i = 1:max_iter % main PSO cycle
2   for j = 1:S
3    r1 = rand(D,1); r2 = rand(D,1);
4    w = 0.0005*(ones(D,1)+rand(D,1)); % or eq. (6)
5    % evaluate v_m_id(j+1) using eq. (2)
6    v_m(1:D,j) = w.*v_m(1:D,j) ...
7        + c1*r1.*(Pbest(1:D,j) - x_m(1:D,j)) ...
8        + c2*r2.*(Gbest(1:D) - x_m(1:D,j));
9    % update   x_m_id(i+1) using eq. (3)
10   x_m(1:D,i) = x_m(1:D,j) + v_m(1:D,j);
11   x_m_ext = [x_m(1:2,j); 0; x_m(3:4,j)];
12   fval = 0; % prediction error => fitness function
13   for jit = 3:ndat
14    psi = [DATA(2,jit-1:-1:jit-2)'; ...
15           DATA(1,jit:-1:jit-2)'];
16    y = DATA(2,jit);
17    yp = x_m_ext'*psi;        % theta_est = x_m_ext;
18    fval = fval + abs(y-yp); % sum of absolute errors
19   end
20
21   % evaluation of whole window - 1 it.
22   if fval < Pfun(j)
23      Pbest(:,j) = X(:,j); Pfun(j) = fval;
24   end
25   % selection of the best from window - 1 it.
26   if Pfun(j) < Gfun
27      Gbest = Pbest(:,j); Gfun = Pfun(j);
28   end
29  end
30  % the best candidates from window and iterations
31  Gbest_all(:,i) = Gbest; Gfun_all(i) = Gfun;
32 end
```

VII. Experiments

In this paper, data-driven model type is considered. To provide easily repeatable experiments, the following single-input single-output second order stable system is considered

$$G_s(s) = \frac{1}{s^2 + 2s + 1} \tag{35}$$

For control design, transfer function (35) can be transformed into discrete time domain, i.e. for $T_s = 0.1$ s is as follows:

$$G_s(z^{-1}) = \frac{0.0047\,z^{-1} + 0.0044\,z^{-2}}{1 - 1.8097\,z^{-1} + 0.8187\,z^{-2}} \tag{36}$$

and

$$\begin{aligned} y_k = {} & 0.0047\,u_{k-1} \;+\; 0.0044\,u_{k-2} \\ & +\,1.8097\,y_{k-1} \;-\; 0.8187\,y_{k-2} \end{aligned} \tag{37}$$

where the correspondence between values above and the model parameters is as follows:

$$G_s(z^{-1}) = \frac{b_1\,z^{-1} + b_2\,z^{-2}}{a_1\,z^{-1} + a_2\,z^{-2}} \tag{38}$$

and $$y_k = b_1\,u_{k-1} + b_2\,u_{k-2} \;-\; a_1\,y_{k-1} \;-\; a_2\,y_{k-2} \tag{39}$$

The model parameters $\vartheta = [\,b_1 \;\; b_2 \;\; -a_1 \;\; -a_2\,]$ are estimated by PSO and LS procedures. The algorithm setup is characterised by parameters in Table I. There are realised two types of experiments with PSO:
i) one-shot estimation, see Fig. 2 and ii) online estimation in combination with MPC algorithm, see Fig. 3 and Fig. 4. The selected experiments are run under normal disturbances with standard deviation $\sigma = 0.01$.

TABLE I
ADJUSTABLE PARAMETERS: PSO, LS AND MPC

	description	parameter	considered range
PSO	swarm size	S	$50-100$
	swarm dimension	D	$\equiv$ no. of model prms
	no. of iterations	nit	$50-500$
	no. of data in data set	$ndat$	$20-100$
	cognitive accel. coef.	c_1	$\varepsilon-5,\ \varepsilon>0$
	social acceleration coef.	c_2	$\varepsilon-5,\ \varepsilon>0$
	inertia weight	w	$\varepsilon-2,\ \varepsilon>0$
LS	forgetting factor	f_i	$0.9-1$
	initial diag. coef.	k_0	$10^{-4}-10^{-8}$
MPC	prediction horizon	N	$>n$ (system order), 10
	penalty factor	q_u	0 (hard) -1 (soft)

The one-shot experiment (Fig. 2) shows profiles of parameter searching convergence over PSO iterations for one specific data window. The last sub-figure in the column shows the decreasing trend of prediction deviation 'Gfun' as the accuracy of the model parameters increases, i.e. trend of fitness function.

The run of MPC with PSO versus MPC with LS is documented by the resulting values of identified parameters in Table II and by the above-mentioned Fig. 3 and Fig. 4. Fig. 3 shows controlled system behaviour along sin and rectangle shape reference signal under random disturbance. Fig. 4 demonstrates PSO convergence over the whole simulation time interval. The convergence speed depends on the excitation signal and amount of information contained in it.

The ranges of the individual parameters in the sub-figures are specially adapted/zoomed so that the gradual evolution of the parameter values could be recognisable. From the experiments for available training data sequences, it is clear that PSO quickly converges to specific ideal values and stabilises.

TABLE II
COMPARISON OF PARAMETERS: IDEAL, PSO AND LS ALGORITHMS

	b_1	b_2	a_1	a_2
Ideal	0.0046788	0.0043771	-1.8097	0.81873
PSO	0.0047914	0.0043842	-1.8057	0.81498
LS	0.0046788	0.0043766	-1.8097	0.81873

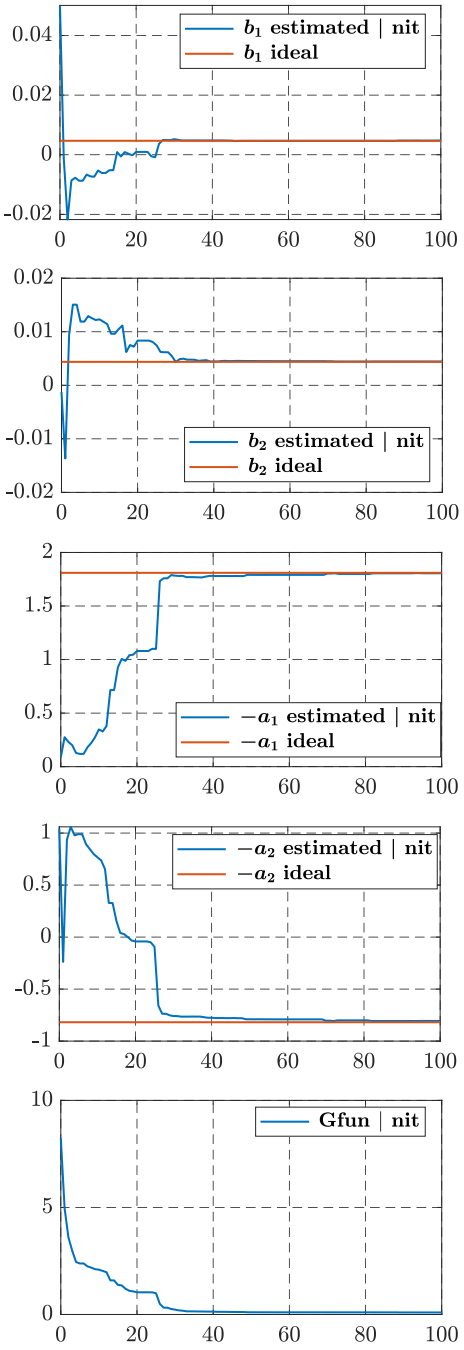


Fig. 2. Time histories of b_i, a_i and Gfun by PSO.

VIII. CONCLUSION

The paper presents utilisation of PSO algorithm within MPC to track changes in the parameters of the ARX model. PSO does not represent an optimal solution and gives too cautious parameter estimation in comparison with standard identification methods. However, PSO uses only simple mathematical operations, namely addition and multiplication and only trends of acceleration coefficients or magnitudes of computed velocities. The magnitudes should be proportionally in correspondence with usual parameter changes otherwise a swarm is too volatile and searched parameters can not converge to their true values. PSO can be used to monitor changes of parameters and consistency with their expected trends.

Future work will focus on PSO for the detection of isolated parameter changes and prediction of future parameter values with the use in industrial robotics and its dynamic models.

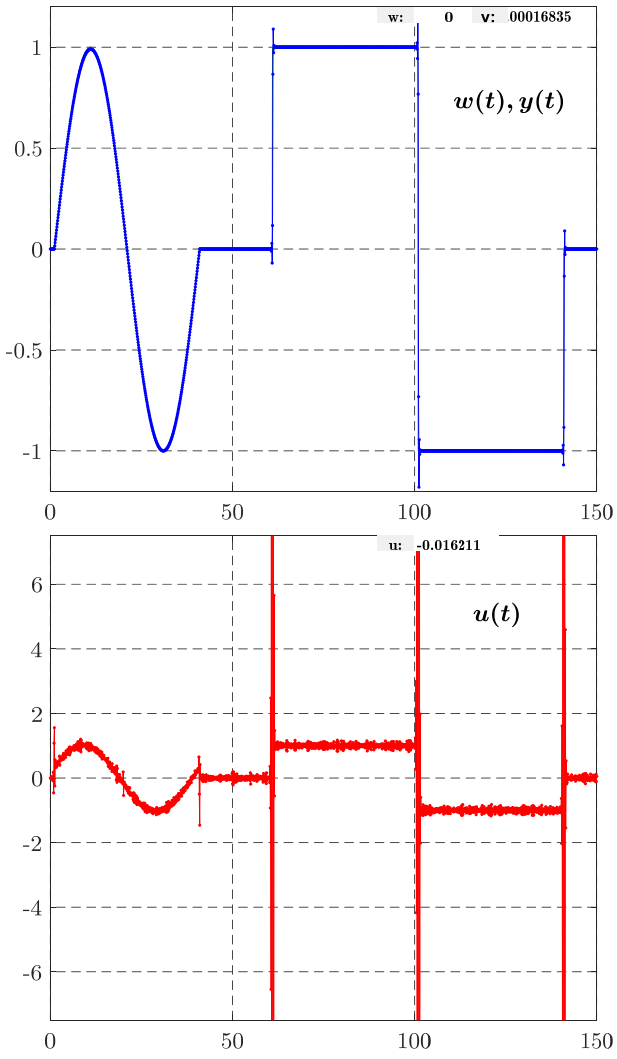


Fig. 3. Time histories of system outputs under MPC with PSO.

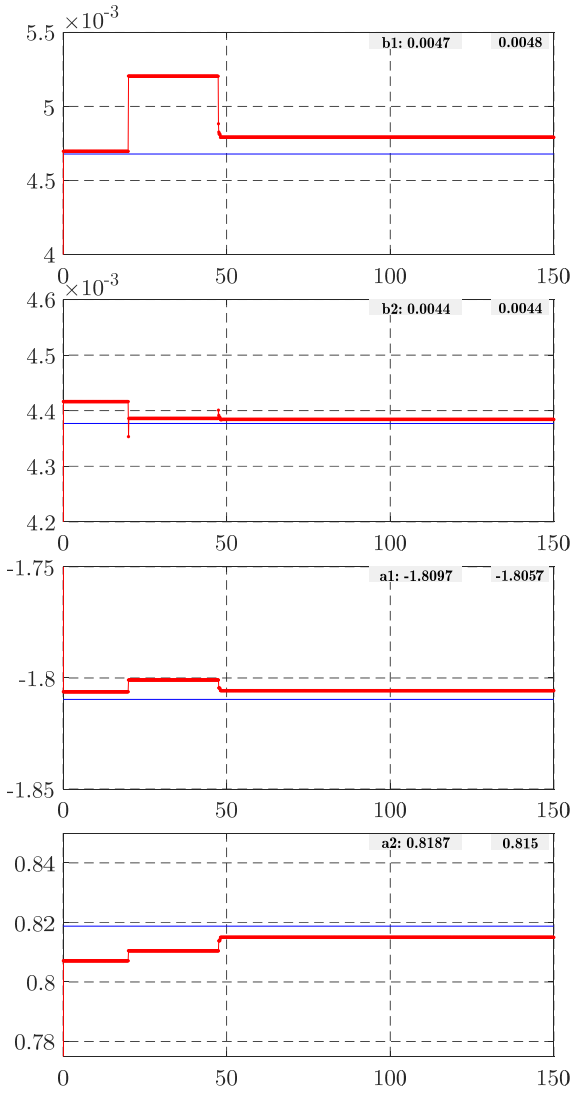


Fig. 4. Time histories of individual system parameters determined by PSO.

REFERENCES

[1] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," *IEEE Access*, vol. 10, pp. 10 031–10 061, 2022.

[2] J. Coelho, de Moura O.P., and J. Cunha, "Greenhouse air temperature predictive control using the particle swarm optimisation algorithm," *Computers electron. in agriculture*, vol. 49, no. 3, pp. 330–344, 2005.

[3] A. Kapnopoulos and A. Alexandridis, "A cooperative particle swarm optimization approach for tuning an mpc-based quadrotor trajectory tracking scheme," *Aerospace Science and Technology*, vol. 127, p. 107725, 2022.

[4] A. Elgammal and T. Ramlal, "Optimal model predictive frequency control management of grid integration pv/wind/fc/storage battery based smart grid using multi objective particle swarm optimization MOPSO," *WSEAS Trans. Electron*, vol. 12, pp. 46–54, 2021.

[5] M. Brahmi, C. B. Regaya, H. Hamdi, and A. Zaafouri, "Comparative study of p&o and pso particle swarm optimization mppt controllers for photovoltaic systems," in *Int. Conf. Control, Decision and Information Technologies (CoDIT)*, vol. 1. IEEE, 2022, pp. 1608–1613.

[6] S. Pervaiz, Z. Ul-Qayyum, W. H. Bangyal, L. Gao, and J. Ahmad, "A systematic literature review on particle swarm optimisation techniques for medical diseases detection," *Computational and Mathematical Methods in Medicine*, vol. 2021, pp. 1–10, 2021.

[7] A. G. Gad, "Particle swarm optimization algorithm and its applications: a systematic review," *Archives of computational methods in engineering*, vol. 29, no. 5, pp. 2531–2561, 2022.

[8] K. Belda and J. Böhm, "Adaptive predictive control for simple mechatronic systems," in *Proc. WSEAS CSCC Int. Conf.*, 2006, pp. 307–312.

[9] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *IEEE Int. Conf. on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence*, 1998, pp. 69–73.

[10] J. Fang, W. Liu, L. Chen, S. Lauria, A. Miron, and X. Liu, "A survey of algorithms, applications and trends for particle swarm optimization," *International Journal of Network Dynamics and Intelligence*, vol. 2, no. 1, p. 24–50, 2023. [Online]. Available: https://www.sciltp.com/journals/ijndi/article/view/176

[11] K. Belda and V. Záda, "Predictive control for offset-free motion of industrial articulated robots," in *2017 22nd Int. Conf. Methods and Models in Autom. and Robotics (MMAR)*, 2017, pp. 688–693.

[12] L. Kuklišová Pavelková and K. Belda, "Output-feedback MPC for robotic systems under bounded noise," in *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2021, pp. 574–582.

[13] K. Belda and P. Píša, "Explicit model predictive control of pmsm drives," in *IEEE 30th Int. Symposium on Indus. Electronics (ISIE)*, 2021, pp. 1–6.

[14] J. Dormand and P. Prince, "A family of embedded runge-kutta formulae," *J. Comp. Appl. Math.*, vol. 6, pp. 19–26, 1980.