24th COBEM - 2017

Associação Brasileira de Engenharia e Ciências Mecânicas

24th ABCM International Congress of Mechanical Engineering
December 3-8, 2017, Curitiba, PR, Brazil

COBEM-2017-2851
BIO-INSPIRED OPTIMATION APPLIED TO THE TUNING OF MODEL PREDICTIVE CONTROL PARAMETERS

Eduardo de Mendonça Mesquita
Carlos Humberto Llanos
Renato Coral Sampaio
Universidade de Brasília, Brasília, DF, Brazil
eduardo.dmendonca@gmail.com
llanos@unb.br
renatocoral@unb.br

Abstract. *Model Predictive Control (MPC) is a control strategy that uses a system dynamic model to predict its behaviour over a time horizon and has become a widely used tool in the industry. Different systems require different control settings and the choice of parameters is not an easy task. This paper proposes a tuning application of MPC parameters using bio-inspired algorithms: Particle Swarm Optimization (PSO) and Salp Swarm Algorithm (SSA). PSO is an algorithm proposed in 1995 and has been applied in several optimization problems. SSA is one of the newest proposed bio-inspired algorithm. The plant is a Triple Integrator and performance metric is the quadratic error between system output and the reference. The convergence error is faster with PSO than SSA, but the latter converges to the target at the end of the iterations. The results show SSA as efficient as PSO in minimization problems.*

Keywords: *optimation, MPC, bio-inspired algorithm, PSO, SSA.*

1. INTRODUCTION

Model Predictive Control (MPC) is a control strategy that uses a system's dynamic model to predict its behaviour over a time horizon and calculates the best set of control actions to track the reference. MPC became a widely used tool in industry (Kawai *et al.*, 2007). Initially, it was proposed for single input single output systems (SISO), however was extended to multiple input multiple output systems (MIMO) making it efficient in multi-variable models as well as in nonlinear systems (Alamir, 2013).

MPC works with constraints, as is the physical behaviour of real systems. Thus, several MPC applications can be found in current literature, such as aerospace control (Hartley *et al.*, 2012, 2014; Ling *et al.*, 2008) and chemical processes (Yamashita *et al.*, 2016; Kawai *et al.*, 2007). These constraints are assigned to physical actuators, security constraints as pressure and temperature variables, obligations (product quality - thresholds specification), set by law (such as maximum emission of pollutants), time constraints, etc.

Despite being applicable to a wide variety of problems the MPC's behaviour (for each of them) depends on the correct tuning of an objective function, which demands high effort for designer because there is an extensive range of values. In order to make this search easier, optimization techniques can be used to achieve the *parameter tuning*. In literature some tuning applications can be found such as using fuzzy logic (Ali, 2001), stochastic optimization (Liu and Wang, 2000) and biological evolution (SAŁAT *et al.*, 2013). One example applications of the Particle Swarm Optimization (PSO) is found for SVM parameter tuning in Rossi and de Carvalho (2008).

The PSO algorithm was proposed in 1995 by Kennedy and Eberhart (1995) based on Reynold's work by bird flocking (Reynolds, 1987) to food search. Since then it has been modified and presented in several applications. On the other hand the SSA is one of the latest bio-inspired algorithms, presented by Mirjalili *et al.* (2017). This work uses bio-inspired algorithms based on swarms of birds (PSO) and salps (SSA) to performe this optimizations.

2. MODEL PREDICTIVE CONTROL

The MPC method focuses on predicting a system's behaviour from a set of input actions over a *prediction horizon* H_p. The MPC cost function quantifies the system response by a value, called fitness, which has to be minimized. The choice of the input value sequence that results in lower fitness is performed by an internal algorithm called *solver*. Several algorithms can be used as solvers, such as gradient-based methods that works best on mono-model problems (Hinojosa *et al.*, 2017; Jamshidnejad *et al.*, 2016). Bio-inspired algorithm can also be used as solvers as the PSO (Yousuf *et al.*, 2009; Mercieca and Fabri, 2012) and ABC, based on bee colony (Sahed *et al.*, 2015).

The MPC cost function depends on the dynamic model of the system and can be classified in two types: convex and non-convex. These characteristics are described in the next section.

2.1 MPC Cost Function

The MPC cost function is shown in Eq. (1), excerpted from Alamir (2013).

$$J = \sum_{i=1}^{H_p-1} ||Q_y(y_{ref}(i) - y(i)||^2 + Q_u|(u(i))|^2 \tag{1}$$

where Q_y is the penalty matrix of the output value, Q_u is the penalty matrix of the actuator value, $y_{ref}(i)$ is the reference, *y(i)* is the system output and *u(i)* is the output value of actuator.

The MPC solver addresses the minimization problem such as indicated by Eq. (2)

$$P = u_{min} J(u) \tag{2}$$

There are two approaches for this problem depending on the system model, one for $\mathbf{P}$ convex and another for $\mathbf{P}$ non-convex as shown in Fig. 1. A convex function has a valley and a single local minimum that also corresponds a global minimum. For a non-convex function there are several local minima which can be misinterpreted as global minimum.

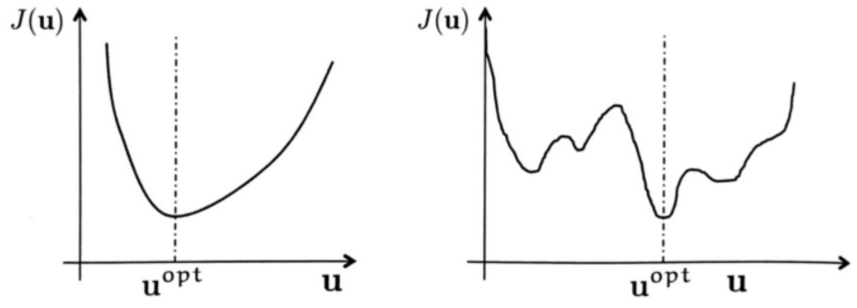

Figure 1. Convex function (left), Non-convex function (right). Modified from (Alamir, 2013)

The system treated in this work presents a convex function model and the solver is an algorithm called Gradient Expansion (GE), shown in Fig. 2.

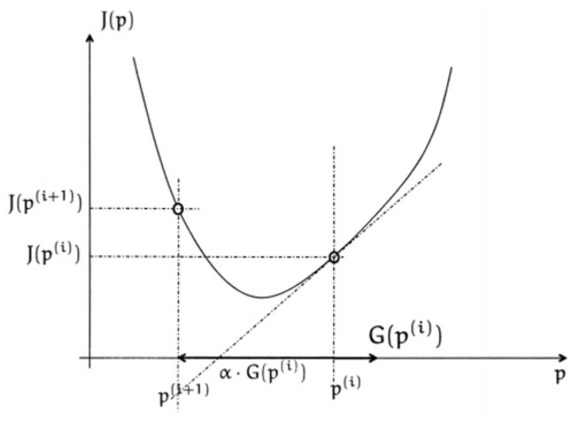

Figure 2. Gradient Expansion Algorithm - GE. Extracted from (Alamir, 2013)

The GE algorithm performs the optimal value search over iterations mutating the p value (future u_{opt}), as well as the search region delimited by α. The α is defined by Eq. 3.

$$\alpha = \frac{\gamma}{h_{max}} \tag{3}$$

where γ is determined by Eq. (6) and h_{max} is a upper bound in search region. At each iteration the p value is then modified by Eq. 4.

$$p_{\gamma}^{i+1} = p^i - \frac{\gamma}{h_{max}} G(p^i) \tag{4}$$

where *G(p)* represents the *p* constraints, treated on next section. The search region is restricted throughout the iterations, therefore the current value of the function is compared to the value to be assigned. Briefly, the following commands are executed:

$$\begin{array}{ll} \textbf{if} & (J(p_\gamma^{i+1}) < J(p^i)) \\ \quad \gamma & \leftarrow max(\gamma_{min}, \beta^- * \gamma) \\ \textbf{then} & \\ \quad \gamma & \leftarrow \beta^+ * \gamma \end{array} \tag{5}$$

Based on this concepts the following steps are performed by the MPC solver:

1: At iteration k, calculate the sequence of future actions that minimize the cost function (Eq. 1);
2: Apply the first control action of the optimal sequence;
3: Calculate the state(s) variable(s) and restart the whole process.

2.2 MPC Constraints

As previously mentioned, the MPC constraints are required to solve real problems. The constraints addressed in this paper are as described in Eq. 6, Eq. 7 and Eq. 8.

$$y_{min} \leq y(k+i) \leq y_{max}, i = 0...H_{p-1} \tag{6}$$

where y_{min} and y_{max} are under and upper bound of output system, respectively.

$$u_{min} \leq u(k+i) \leq u_{max}, i = 0...H_{p-1} \tag{7}$$

where u_{min} and u_{max} are under and upper bound of control action, respectively.

$$\Delta u_{min} \leq \Delta u(k+i) \leq \Delta u_{max}, i = 0...H_{p-1} \tag{8}$$

where Δu_{min} and Δu_{max} are difference of under and upper bound of the control action at consecutive iteration, respectively.

3. BIO-INSPIRED ALGORITHMS

A lot of technology was developed based on nature, the velcro (based on plants texture) and the sonar (based on bats) are some examples. Bio-inspired algorithms are the result of incorporating nature concepts into computer systems. Swarm intelligence is a term used to designate systems with artificial intelligence whose interaction of individuals with little intelligence generates coherent solutions or search patterns to solve tasks (Poli *et al.*, 2007).

The 'swarm' term is used generically to refer to any structured collection of agents capable with ability to interact with each other. Therefore it can be said that the collective interactions of all agents into the system often lead to some kind of collective intelligence. This artificial intelligence includes any attempt to design algorithms or distributed problem-solving devices without centralized control, inspired by the collective behaviour of social agents (Kawai *et al.*, 2007). In this work two bio-inspired algorithms are used, based on flocking birds (PSO) and salp swarm (SSA).

3.1 PSO - Particle Swarm Optimization

In the 80's, the birds behaviour stimulated some researches. Biologist Frank Heppner studied the flock of birds (Fig. 3). James Kennedy and Russel Eberhart, in 1995, inspired by studies of Heppner, developed an optimization technique that came to be called Particle Swarm Optimization - PSO (Kennedy and Eberhart, 1995).

The PSO optimizes a problem iteratively by trying to improve the candidate solution with respect to a given quality measure. Using an analogy, the term particle was adopted to symbolize the birds and represent the possible solutions to the problem to be solved. The area overflown by birds is equivalent to the search space and the place with food, or the nest, corresponds to finding the optimal solution.

In order for the birds flock search the target, the performance of the particles is evaluated (fitness). To reach the target, birds use their experiences and the experience of the flock. The individual experience of each particle, that is, its life history, is called **pbest**. The one responsible to representing complete swarm knowledge is **gbest**.

At each iteration *k* a particle's velocity is update using Eq.9

$$v_i(k+1) = v_i(k) + c_1 * r_1(\mathbf{pbest} - p_i(k)) + c_2 * r_2(\mathbf{gbest} - p_i(k)) \tag{9}$$

Figure 3. Flocking birds. Extracted from (www.en.wikipedia.org/wiki/Flock_(birds)#).

where $v_i(k+1)$ is the new velocity of i^{th} particle, c_1 and c_2 are constant variable, r_1 and r_2 are uniformly random numbers in $[0, 1]$.

The particle's position is updated using Eq.(10)

$$p_i(k+1) = v_i(k+1) + p_i(k) \tag{10}$$

The PSO pseudo code can be seen below:

Algorithm 1 - PSO PseudoCode

```
 1: Initialize the particle's position $x_i$ (i=1,2,...,n) considering ub and lb
 2: repeat
 3:     Calculate the fitness of each search agent (particle)
 4:     if the fitness value is better than the best one (pbest) in history then
 5:         Set the current value as as the new pbest
 6:     end if
 7:     Choose the best fitness as the gbest
 8:     for each particle ($x_i$) do
 9:         Calculates new velocity in accordance with Eq. (9)
10:         Update particle position in accordance with Eq. (10)
11:     end for
12: until (end condition is not satisfied)
return: gbest
```

3.2 SSA - Salp Swarm Algorithm

Salps (Fig. 4) are Salpidae family and have transparent barrel-shaped body. Its locomotion mechanism is similar to the jellyfish. Water is pumped through the body, which also helps it to feed by ingesting plankton.

Figure 4. Salp. Extracted from (www.cmarz.org/species_images/salps/salpa_cylindrica_solitary_2b_edited_.jpg)

Studies of Salp's behaviour is difficult because they live in deep ocean and it is really hard to keep them in laboratory environments. But some researches have already discovered an important peculiarity of Salps. Their collective behaviour. Salps often form a swarm called salp chain (Fig. 5), the main reason of this behaviour is not very clear, but some researchers believe that this is done for achieving better locomotion using rapid coordinated changes and foraging (Mirjalili *et al.*, 2017). The first salp of chain is called leader.

This behaviour inspired the creation of the Salp Swarm Algorithm, proposed by Mirjalili *et al.* (2017). The mathematical model of SSA considers two individuals groups, the leader and the followers. The leader is the individual with the best fitness and that attracts all followers to its position. Similar to PSO, the individuals in SSA are distributed in a

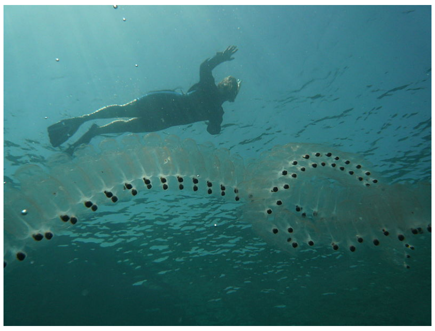

Figure 5. Salps chain. Extracted from (www.pt.wikipedia.org/wiki/Salpa#)

search space of dimension *n* which is the number of problem's variables. In addition, the position of the salps are stored in arrays called *x*. The leader's position update occurs according to Eq. (11).

$$x_j^1 = \begin{cases} F_j + c_1((ub_j + lb_j)c_2 + lb_j), c_3 \geq 0 \\ F_j - c_1((ub_j + lb_j)c_2 + lb_j), c_3 < 0 \end{cases} \tag{11}$$

where x_j^1 is the leader's position in the j^{th} dimension, F_j is the position of food source in the j^{th} dimension, lb_j and ub_j are the lower and upper bounds of j^{th} dimension, c_1, c_2 and c_3 are random numbers.

The coefficient c_1 is the most important parameter of SSA because it balances exploration and exploitation according by Eq. (12).

$$c_1 = 2e^{-(\frac{4l}{L})^2} \tag{12}$$

where l is the current iteration and L is the maximum number of iterations.

The parameters c_2 and c_3 are random number uniformly generated in the interval of $[0, 1]$ and dictate in which direction the leader will follow in the next move.

The followers update their positions according to Newton's law of motion, Eq. (13).

$$x_j^i = \frac{1}{2}at^2 + v_0 t \tag{13}$$

where $i \geq 2$, x_j^i is the position of i^{th} follower in j^{th} dimension, *t* is time, v_0 is initial velocity. *a* and *v* are obtained as described in Eq. (14).

$$a = \frac{v_{final}}{v_0} where\ v = \frac{x - x_0}{t} \tag{14}$$

Considering $v_0 = 0$ and discretizing time so each step is 1, it arrives on Eq. (15).

$$x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1}) \tag{15}$$

Equation. (15) represents the salps chain because each follower updates his position according to the follower in front of him.

SSA pseudo code can be seen in **Algorithm 2**.

4. METHODOLOGY

The MPC case study is applied to a Triple Integrator that was taken from Alamir (2013), whose model in state space is:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \qquad y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{16}$$

where x_1, x_2 e x_3 are state variables of system.

The tuning system proposed is shown in Fig. 6

The quality measure used by bio-inspired algorithms is the quadratic error between reference and system output, at each iteration, as shown in Eq. (17).

$$error = \sqrt{\sum_{i=1}^{ite_{max}} (y_{ref}(i) - y(i))^2} \tag{17}$$

Algorithm 2 - SSA PseudoCode

```
1:  Initialize the salp's population $x_i$ (i=1,2,...,n) considering ub and lb
2:  repeat
3:      Calculate the fitness of each search agent (salp)
4:      $F$ = the best search agent
5:      Update $c_1$ by Eq. (12)
6:      for each salp ($x_i$) do
7:          if ($i == 1$) then
8:              Update the position of the leading salp by Eq. (11)
9:          else
10:             Update the position of the follower salp by Eq. (15)
11:         end if
12:     end for
13:     Amend the salps based on the upper and lower bounds of variables
14: until (end condition is not satisfied)
return: $F$
```

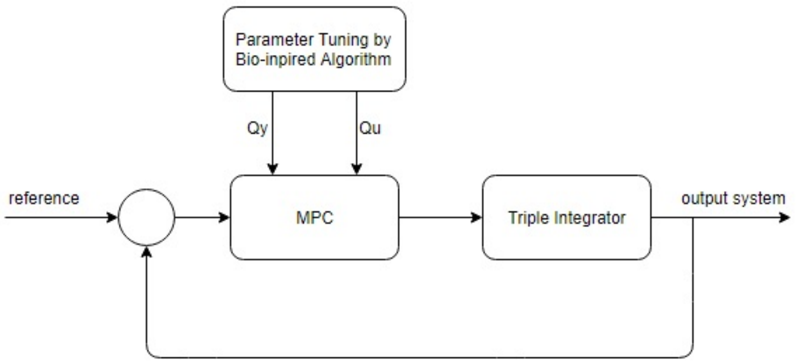

Figure 6. Tuning system proposed.

The constraints were taken from Alamir (2013). They are the following: $-2 \leq y(k+i) \leq 2$, $-30 \leq u(k+i) \leq 30$ and $-50 \leq \Delta u(k+i) \leq 50$, $i = 0, ..., H_p - 1$. Each algorithm was run 50 times by tuning of MPC weights (Q_y and Q_u). The data analysis is presented on a statistical basis in these simulations.

5. SIMULATION RESULTS

The tuning results of the MPC parameters are now presented. The graph of weights distribution by the error calculated in Eq. (17) is shown in Fig. 7 and presents a minimization problem where the global minimum is the algorithm's target. This curve is characterized by the dynamic model of the plant and input reference. So for different problems the graph can vary.

The algorithms are configured with $S = 10$ search agents at a maximum of 200 iterations. The MPC solver configuration found in Alamir (2013) was used: $\gamma_{min} = 1.2$, $\beta^- = 0.2$ and $\beta^+ = 1.8$. Algorithms performance at each iteration can be seen in Fig. 8, this result represents the median from 50 simulations.

The convergence is faster in PSO but SSA is also able to reach the target. Statistically the PSO and SSA performance are given in Tab. 1. This indicates that use of the bio-inspired algorithms result in a better system control than no weights tuning found in Alamir (2013) whose error was 7.866.

Table 1. Experimental results of error for 50 simulations.

ALGORITHM	AVERAGE	MEDIAN	MINIMUM	STANDARD DEVIATION
PSO	7.8067	7.8075	7.7977	3.8661e-10
SSA	7.8086	7.8082	7.7990	3.8490e-10

In general, for the Triple Integrator the bio-inspired algorithms settled the control objective. The system response with tuned weights by PSO and SSA can be observed in Fig. 9 and Fig. 10, respectively.

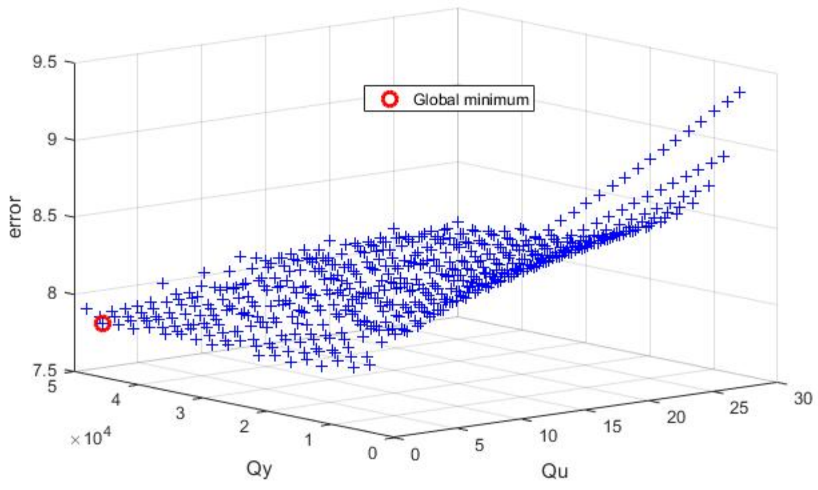

Figure 7. The minimization problem.

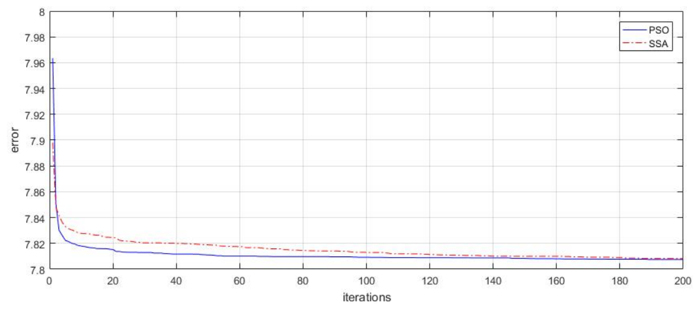

Figure 8. Best value at each iteration step.

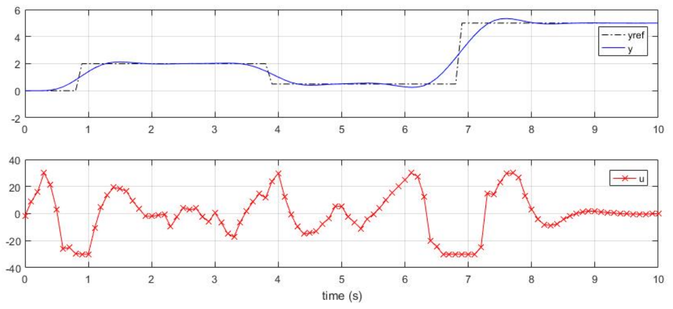

Figure 9. System response with parameter tuning by PSO.

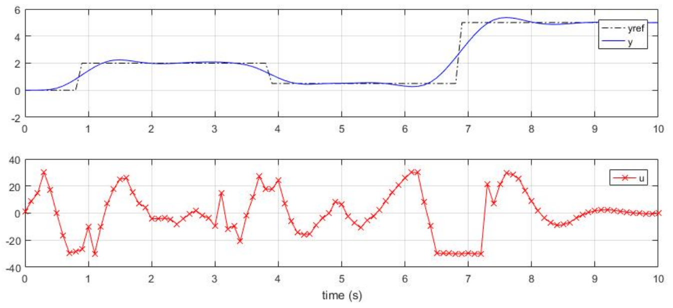

Figure 10. System response with parameter tuning by SSA.

6. CONCLUSIONS

A parameters tuning method using bio-inspired algorithm is presented in this paper. The error convergence is faster in PSO but SSA is also able to reach the target at 200 iterations. Respecting the plant's limitation, the control result expressed the importance of a tuning method regarding the choice of MPC weights. The system's performance was better (error of 7.80) than with no tuning (7.86, weights found in Alamir (2013)). In summary, bio-inspired algorithms, specifically PSO and SSA, are useful tools in minimization problems.

Future work may address tuning in control of non-linear systems. Tests with other MPC solvers can also be performed. The GE algorithm has a high computational cost and can be replaced by faster algorithms that do not use gradient calculations.

7. ACKNOWLEDGEMENTS

This work was supported by CNPq (National Counsel of Technological and Scientific Development) and FAPDF (Fundação de Apoio à Pesquisa do DF).

8. REFERENCES

Alamir, M., 2013. *A Pragmatic Story of Model Predictive Control: Self-Contained Algorithms and Case-Studies*. Create Space Independent Publishing Platform.

Ali, E., 2001. "Automatic tuning of model predictive controllers based on fuzzy logic". *optimization*, Vol. 2, No. 2, p. 1.

Hartley, E.N., Jerez, J.L., Suardi, A., Maciejowski, J.M., Kerrigan, E.C. and Constantinides, G.A., 2012. "Predictive control of a boeing 747 aircraft using an fpga". *IFAC Proceedings Volumes*, Vol. 45, No. 17, pp. 80–85.

Hartley, E.N., Jerez, J.L., Suardi, A., Maciejowski, J.M., Kerrigan, E.C. and Constantinides, G.A., 2014. "Predictive control using an fpga with application to aircraft control". *IEEE Transactions on Control Systems Technology*, Vol. 22, No. 3, pp. 1006–1017.

Hinojosa, A.I., Ferramosca, A., González, A.H. and Odloak, D., 2017. "One-layer gradient-based mpc+ rto of a propylene/propane splitter". *Computers & Chemical Engineering*, Vol. 106, pp. 160–170.

Jamshidnejad, A., Papamichail, I., Hellendoorn, H., Papageorgiou, M. and De Schutter, B., 2016. "Gradient-based model-predictive control for green urban mobility in traffic networks". In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, pp. 1077–1082.

Kawai, F., Ito, H., Nakazawa, C., Matsui, T., Fukuyama, Y., Suzuki, R. and Aiyoshi, E., 2007. "Automatic tuning for model predictive control: Can particle swarm optimization find a better parameter?" In *Intelligent Control, 2007. ISIC 2007. IEEE 22nd International Symposium on*. IEEE, pp. 646–651.

Kennedy, J. and Eberhart, R., 1995. "Particle swarm optimization". In *Conf. on Neural Networks*. Vol. 4.

Ling, K.V., Wu, B.F. and Maciejowski, J., 2008. "Embedded model predictive control (mpc) using a fpga". *IFAC Proceedings Volumes*, Vol. 41, No. 2, pp. 15250–15255.

Liu, W. and Wang, G., 2000. "Auto-tuning procedure for model-based predictive controller". In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*. IEEE, Vol. 5, pp. 3421–3426.

Mercieca, J. and Fabri, S.G., 2012. "A metaheuristic particle swarm optimization approach to nonlinear model predictive control". *International Journal on Advances in Intelligent Systems*.

Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H. and Mirjalili, S.M., 2017. "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems". *Advances in Engineering Software*.

Poli, R., Kennedy, J. and Blackwell, T., 2007. "Particle swarm optimization". *Swarm intelligence*, Vol. 1, No. 1, pp. 33–57.

Reynolds, C.W., 1987. "Flocks, herds and schools: A distributed behavioral model". *ACM SIGGRAPH computer graphics*, Vol. 21, No. 4, pp. 25–34.

Rossi, A.L.D. and de Carvalho, A.C., 2008. "Bio-inspired optimization techniques for svm parameter tuning". In *Neural Networks, 2008. SBRN'08. 10th Brazilian Symposium on*. IEEE, pp. 57–62.

Sahed, O.A., Kara, K. and Benyoucef, A., 2015. "Artificial bee colony-based predictive control for non-linear systems". *Transactions of the Institute of Measurement and Control*, Vol. 37, No. 6, pp. 780–792.

SAŁAT, R., AWTONIUK, M. and KORPYSZ, K., 2013. "Black-box system identification by means of support vector regression and imperialist competitive algorithm". *Przegląd Elektrotechniczny*, Vol. 89, No. 9, pp. 223–226.

Yamashita, A., Zanin, A. and Odloak, D., 2016. "Tuning of model predictive control with multi-objective optimization". *Brazilian Journal of Chemical Engineering*, Vol. 33, No. 2, pp. 333–346.

Yousuf, M., Al-Duwaish, H. and Al-Hamouz, Z., 2009. "Pso based nonlinear predictive control of single area load frequency control". In *IFAC Workshop on Control Applications of Optimization*.

9. RESPONSIBILITY NOTICE