Bat Algorithm in Model Predictive Control of Dynamic Systems

Nikita Mazurenko[1*], prof. Ing. Milan Hofreiter, CSc[1].

[1] *CTU in Prague, Faculty of Mechanical Engineering, Department of Instrumentation and Control Engineering, Technická 4, 166 07 Prague 6, Czech Republic*

Abstract

The project is devoted to design and realization of model predictive control algorithm for dynamic systems. The control system is based on a Bat algorithm, that is used for optimization and adjustment of the control variable. The algorithm was developed in MATLAB. Functionality of the control algorithm was verified on a real laboratory system.

Keywords; Dynamic System; Model Predictive Control; Bat Algorithm; Maltab

1. Introduction

Nowadays is impossible to imagine a plant or manufacture with no automated processes. That's why development of a new automation methods is not only important, it is necessary. The paper is devoted describing a possible prospective method of system control process.

The idea of this method is to combine two advanced techniques and make them work together in the system control algorithm. These techniques are Model Predictive Control and Bat algorithm. Each part and the way they work together are described in text below.

2. Controller

The base of the control system which is described in this paper is a controller. Because of the computer implementation we decided to use a discontinuous type of system controllers. The main task for designed controller is to set the value of a control variable at which the difference between the actual and the reference process trajectory is minimal.

$$u[k+1] = u[k] + \Delta u[k] \tag{1}$$

At every sample time step a new value of control variable is calculated according to *(11)*. But number of possible values for $\Delta u[k]$ is limited. The control variable can only change to values from this set:

$$\Delta u[k] \in K[-10, -5, -2, -1, 0, 1, 2, 5, 10] \tag{2}$$

This set of values was chosen by analogy with the coins. With these nine values, it is possible to get any necessary integer. The constant K was added to specify changes in control variable. This allows more precise regulation of the controlled process.

This is the way the controller influences inputs of the process. But here becomes the question: how does the controller search for right or optimal changes in control variables? To answer this question in the next chapter is described an advanced control technique - model predictive control.

3. Model Predictive Control

In this chapter is considered model predictive control (MPC). MPC is an important control technique for solving of difficult multivariable control problems. Suppose that it's necessary to control a process while satisfying inequality constraints on the input and output variables [2]. If an accurate dynamic model is available, this model and current measurements can be used to predict behaviour of a system outputs. Then according to these measurements and predictions changes in the input variables can be made. In essence, changes in the input variable are made only when the response of the system to these changes is known due to the dynamic model.

3.1. The Concept

The basic idea of model predictive control is to compute a trajectory of the future system input u to optimize the future behaviour of the system output y. Before implementing this control method, it is necessary to determine [1]:

- Reasonably accurate dynamic model of the process.
- System inputs and outputs.
- Length of prediction horizon T_P - how "far" the future values to be predicted for.
- Criterion for solution assessing.

These items are used for frequently realization of the following steps:

1. At the actual time point k a set of future input sequences $u_p[k+i-1]$ (where $i = 1,2 \ldots T_P$) is generated [4].
2. Dynamic model is used to calculate predicted system outputs $\hat{y}[k+i]$ for each sequence of inputs [4].
3. According to the determined criterion, the optimal sequence is chosen [4].

* Corresponding author: Nikita.Mazurenko@fs.cvut.cz

4. From a sequence of future inputs that matches the optimal solution only the first input value is sent to the controlled process [4].

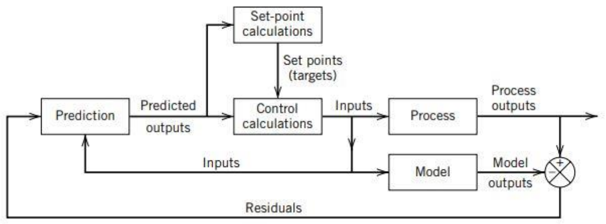


Figure 1. *Block diagram of MPC (Reprinted from [3])*

A block diagram of the system with model predictive control is shown in Fig 1. As stated above, a process model is used to calculate future values of the output variables. The differences between predicted and the current output values (residuals) are sent to *prediction* block and used as the feedback signal.

3.2. Optimization Problem

Now we know that MPC strategy manipulates with plenty of possible output values to evaluate the best value at the current sampling instant. But this manipulation can become a problem when it's necessary to deal with a set of possible values for a longer prediction horizon. Calculation and evaluation of each outputs sequence will take a lot of time. And time is one of the most important thinks in dynamic systems control. It means, that selection process of must be optimized. To solve this optimization problem, we decided to use metaheuristic algorithms.

4. Bat Algorithm

Metaheuristic algorithms are now becoming powerful methods for solving optimization problems. The majority part of these algorithms has been derived from the behaviour of physical or biological systems in nature. A new metaheuristic method, which is proposed in this paper, is based on the echolocation behaviour of bats. That is why this method is named Bat Algorithm.

4.1. The Structure

In 2010, X.-S. Yang proposed a new optimization algorithm by observing the behavior and the characteristics of microbats [2]. The capability of echolocation of microbats is fascinating as these bats can find their prey and discriminate different types of insects even in complete darkness. To construct the basic structure of the algorithm were used three main characteristics of microbats [2]:

1. All bats can use the echolocation to search for prey and to avoid obstacles.
2. Bats use the signal with fixed frequency f_{min}, variable wavelength λ and the loudness A_0 to find a prey.
3. The loudness is varied from a positive value A_0 to a minimum A_{min} constant value.

Then, according to these approximate and idealized rules, the movement of each virtual bat could be simulated by:

$$f_n = f_{min} + (f_{max} - f_{min}) \cdot \beta \tag{3}$$

$$v_n{}^k = v_n{}^{k-1} + (x_n{}^k - x_{best}) \cdot f_n \tag{4}$$

$$x_n{}^k = x_n{}^{k-1} + v_n{}^k \tag{5}$$

where f is a frequency used by bat (suffixes min and max represent the minimum and maxim value of frequency), x_n and v_n stand for the location and velocity of n-th bat. $\beta \in [0,1]$ is a random vector, which is drawn from a uniform distribution. And x_{best} represents the best solution found so far over the population of bats.

But it's not the only way to move bats. In BA is implemented the second local search strategy, namely, random walk. For this strategy the rate of the pulse emission from the bat is also taken to be one of the roles in the process. In every iteration the pulse emission ($r_n \in [0,1]$) is compared with a random number. In case a random number is greater than the current pulse emission rate of the bat, the new solution is generated by:

$$x_n{}^k = x_n{}^{k-1} + \varepsilon A_n^k \tag{6}$$

where $\varepsilon \in [0,1]$ is a random number. After updating the position of the bat, the loudness A_n and the pulse emission rate r_n are also updated:

$$A_n^{k+1} = \alpha \cdot A_n^k \tag{7}$$

$$r_n^{k+1} = r_n^0[1 - e^{-\gamma k}] \tag{8}$$

where α and γ are constants.

When positions of the whole population of bats are updated, the cost function is calculated for every solution. Then, according to the cost function, all solutions can be evaluated, and the best solution can be updated.

4.2. Application in the Project

In this project, bat algorithm was implemented to solve the optimization problem of searching for the best set of values for the control variable. In our case bats position x_n represents the set of possible future changes in process input:

$$x_n = (\Delta u[k+1], \Delta u[k+2], \ldots, \Delta u[k+T_p]) \tag{9}$$

As we defined in 2. chapter, Δu can only take a value from the set of nine elements (see (2*)*. But the distribution of elements in the vector is uneven. Because of this, the probability of usage is not the same for different elements. To avoid this problem our algorithm will deal with the indices of these elements instead of their values.

$$x_m[k+1] = \boldsymbol{A}x_m[k] + \boldsymbol{B}u[k] \tag{10}$$

$$y[k] = \boldsymbol{C}x_m[k] + \boldsymbol{D}u[k] \qquad (11)$$

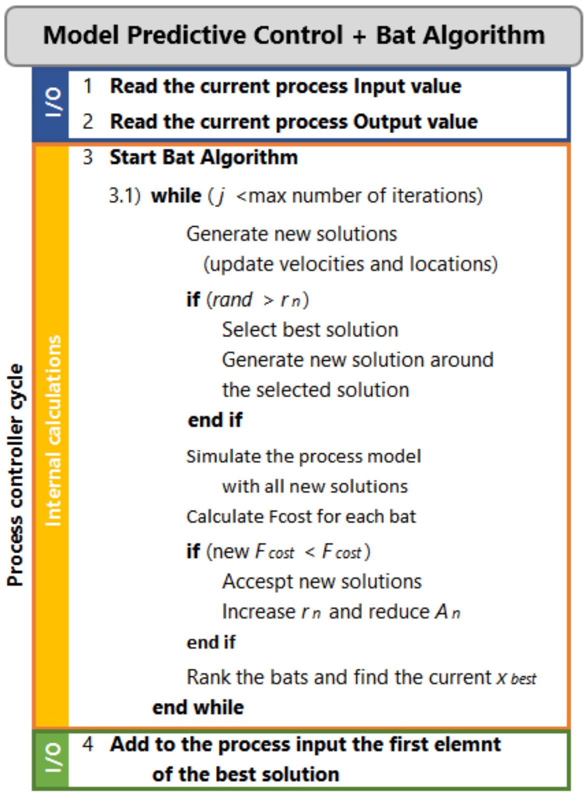
Evaluation of solutions for the controller consists of two parts. At the first part it's necessary to get information, how the dynamic model of the process reacts to each possible solution. As a mathematical model of the process, discrete state-space model is used (see 10 and 11). Simulating the process model with the set of inputs values algorithm generate the predict trajectory $\hat{y}$ of system output. The second part of solutions processing is an evaluation of the predict trajectory using the cost function. In this step algorithm compare reference trajectory of the process output with calculated trajectory. At every sample step the difference between predicted value and reference value is computing by:

$$F_{cost} = \sum_{i=1}^{T_p} (w[k+i] - \hat{y}[k+i])^2 \qquad (12)$$

After calculating of the cost function for every possible solution, bat algorithm finds the optimal sequence of changes in process input.

5. MPC with BA

All separate parts of predictive controller were described in previous chapters. And when all these parts were combined the discontinuous predictive controller with bat algorithm is designed. The working cycle of the controller is shown in Fig. 2:



```
1   Read the current process Input value
2   Read the current process Output value
3   Start Bat Algorithm
    3.1) while ($j$ <max number of iterations)
            Generate new solutions
              (update velocities and locations)
            if ($rand > r_n$)
                Select best solution
                Generate new solution around
                the selected solution
            end if
            Simulate the process model
                with all new solutions
            Calculate Fcost for each bat
            if (new $F_{cost} < F_{cost}$)
                Accept new solutions
                Increase $r_n$ and reduce $A_n$
            end if
            Rank the bats and find the current $x_{best}$
        end while
4   Add to the process input the first elemnt
        of the best solution
```

Figure 2. *Process controller cycle*

5.1. Controller Testing

5.1.1. Simulation tests

At the first stage of verification, designed controller was tested with simulated data. The tests were done with different process model. As an example, was chosen a linear model of the SISO system. The system matrices are shown in Table 1:

Table 1. *The system matrices for simulation model.*

A	B	C	D
$\begin{bmatrix} 1{,}215 & -0{,}736 \\ 0{,}5 & 0 \end{bmatrix}$	$\begin{bmatrix} 0{,}5 \\ 0 \end{bmatrix}$	$[0{,}476 \quad 0{,}473]$	0

For this test into the system was also added white noise with normal distribution.

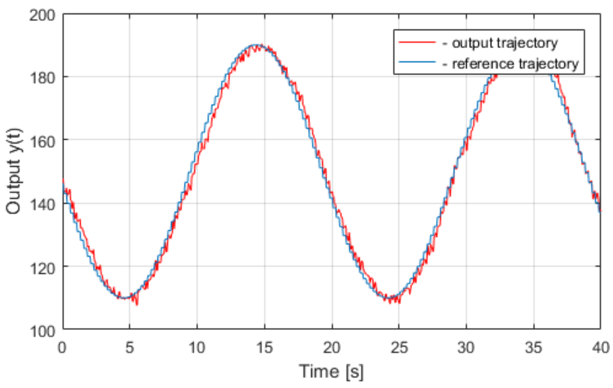


Figure 3. *Control process – simulation model*

The graph with control process of the simulation model is shown in Fig. 3. We can see, that difference between the reference and actual trajectory is minimal. On the basis the simulation tests, it can be stated, that the designed control algorithm works well. This conclusion allowed us to proceed to tests with a real process.

5.1.2. Real Process tests

The second stage of control system verification was to test the controller on a real process. For this test was chosen the laboratory system of water levitation. As in the previous case SISO system was chosen.

In preparation for testing, it was necessary to identify the dynamic model of the process. For this, the process output was measured with different trajectories and values of the input. Then, to process the measured data was used special tool System Identification Toolbox in MATLAB. This toolbox provided the state-space model with system matrices which are shown in Table 2. Also, the model of this process had the transport delay - $T_d = 0{,}21\ [s]$.

Table 2. *The system matrices for real process model.*

A	B	C	D
$\begin{bmatrix} 1{,}835 & -0{,}84 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0{,}5 \\ 0 \end{bmatrix}$	$[0{,}322 \quad 0{,}304]$	0

For verification, Simulink file was downloaded to the PC, which was communicated with the laboratory system.

Test with a real process model were made with two types of reference trajectory. The first trajectory was a sine-graph and the second was a sequence of constant pulses.

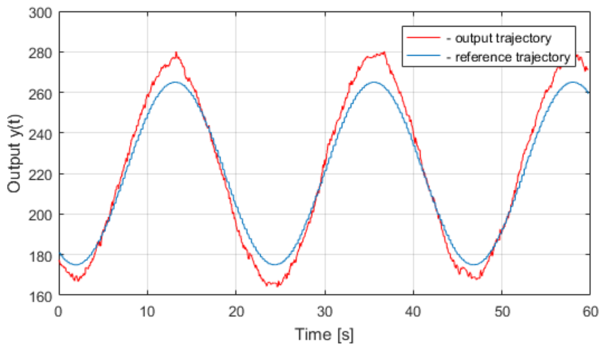


Figure 4. *Control process – sine-graph trajectory*

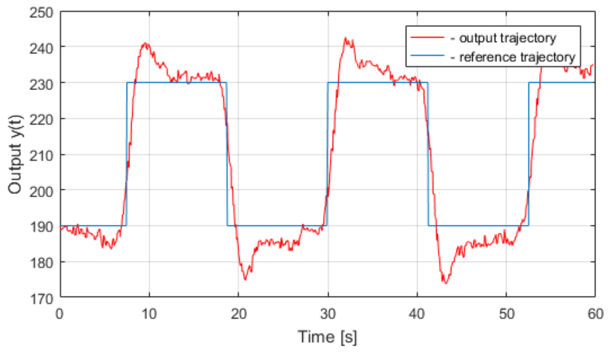


Figure 5. *Control process – pulse trajectory*

The results of the tests with laboratory system of water levitation are shown in Fig. 4 and Fig. 5. From both these graphs it’s clear, that the difference between the process output and reference trajectory is more expressive than this difference in simulation test. The truth is, that during the tests with laboratory system there were several negative factors, which had influence on control process.

One of the most important factors is the dynamic model of the process. Chances are, that this model is not as much qualitative as it’s necessary. To solve this problem more advanced techniques of system identification could be used.

Another reason could be the processing speed of Simulink and PC. Faster calculations could help improve the quality of regulation process.

6. Conclusion

The aim of this project is the study the Bat Algorithm, design the algorithm for Model Predictive Control based on the BA and testing of the algorithm with a linear Single-Input-Single-Output system.

In the first part of the paper were described the basic structures of model predictive control and bat algorithm. These two techniques were combined and were implemented in the process control system.

The designed algorithm for the system controller was vitrificated with simulation tests and with the real process system tests. Based on the tests results, we can conclude that the design of algorithm was made well.

The next stage of the project is implementation of the algorithm in faster equipment such as PLC.

Acknowledgement

I would like to thank my supervisor prof. Ing. Milan Hofreiter, CSc for his guidance during the work with this project.

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS19/158/OHK2/3T/12

Symbols

A	loudness (dB)
A_o	original value of loudness (dB)
A_{min}	minimal value of loudness (dB)
F_{cost}	cost function (-)
f	pulse frequency (Hz)
f_{max}	maximum value of pulse frequency (Hz)
f_{min}	minimal value of pulse frequency (Hz)
n	number of bats (-)
T_p	prediction horizon (-)
r	rate of pulse emission (-)
u	input of the system (-)
u_p	predicted input of the system (-)
$\hat{y}$	predicted output of the system (-)
y	output of the system (-)
λ	wavelength (m)

References

[1] CAMACHO E.F., BORDONS C., “*Model Predictive Control. 2nd ed.,*” 2007. [Online]. Available: http://een.iust.ac.ir/profs/Shamaghdari/MPC/Resources. [Accessed 27 2 2019].

[2] YANG Xin-She, *Nature-inspired metaheuristic algorithms. 2nd ed.*, Frome: Luniver Press, 2010.

[3] D.E. SEBORG, “*Process Dynamic and Control, Second Edition”* , 2004. [Online] Available: https://www.academia.edu/25474988/ [Accessed 12 1 2019]..

[4] L. WANG, *Model Predictive Control System Design and Implementation Using MATLAB,* London: Springer-Verlag London Limited, 2009.