Combination of Evolutionary and Gradient Optimization Techniques in Model Predictive Control

Jan Antos and Marek Kubalcik

Abstract**—**Model predictive control (MPC) designates a control method based on the model. This method is suitable for controlling of various kinds of systems. The basic principle is to calculate the future behaviour of a system and to use this prediction for the optimization of a control process. The optimization problem must be then solved in each sampling period. One of the advantages of MPC is its ability to do online constraints handling systematically. These constraints may, however, cause that the optimization problem is more complex. In this case, some iterative algorithms must be applied in order to solve this problem effectively. This paper is focus on the combination of the optimization techniques. The basic idea is to combine the advantages of gradient and evolutionary algorithms.

Keywords**—**model predictive control, evolutionary algorithms, gradients algorithms, optimization, control process

I. Introduction

MODEL predictive control (MPC) [1], [2] [3] is one of the control method which have been developed over the past few years. The predictive control [4], [5], [6] is a technique based on a discrete model which was mostly applied in a process industry. The crucial issue in MPC is the computational complexity due to the computational demands. With developing computational power and appropriate algorithms, it is possible to apply this approach to systems with faster dynamics such as electrical systems.

The basic idea of MPC is to use the model of a system to predict the future behaviour over the specific horizon. Based on this prediction, the optimization problem represented by a cost function must be solved. This optimization problem may be computationally complex due to the constraints and it must be solved in each sampling period. Although a control sequence is calculated over a control horizon, only the first element is applied and the whole procedure is then repeated in the following sampling period. This is known as a receding horizon strategy [7], [8].

The computational problem may be solved by several types of algorithms. One of the common methods is quadratic programming [9] enabling the inclusion of constraints [10] in a controller synthesis. Besides this algorithm, many other methods may be applied which may have comparable performance and they may fulfil the specific requirements.

This contribution deals with an alternative method which is based on the combination of gradient and evolutionary methods [11]. The motivation for this concept is the effort to reduce a computational expense.

The rest of this paper is organized as follows. Section II is devoted to the basic idea and mathematical background of MPC. Hill climbing algorithm and proposed evolutionary-gradient algorithm are described in section III and section IV respectively. Some results and discussion are presented in section V and the paper is concluded in section VI.

Jan Antos is with the Tomas Bata University in Zlín, Faculty of Applied Informatics, Nám. T. G. Masaryka 5555, 760 05 Zlín (e-mail: antos@ fai.utb.cz).

Marek Kubalcik is with the Tomas Bata University in Zlín, Faculty of Applied Informatics, Nám. T. G. Masaryka 5555, 760 05 Zlín (corresponding author to provide phone: +420 57-603-5198; e-mail: kubalcik@ fai.utb.cz).

II. Model Predictive Control

The model represents a controlled system and it describes the relation between input and output. There are various models which may be used such as transfer function, linear models [12], state-space models [13], neural networks [14], [15], [16] and others. A widely used model is the CARIMA model which directly contains a control increment. The model can be written in the following form

$$Ay(k) = Bu(k-1) + \frac{C}{\Delta}n(k) \tag{1}$$

where polynomials A and B represent a transfer function of the system, integrator Δ = 1 – z-1, C is a colouring polynomial and y, u and n are output, input and a nonmeasurable noise respectively.

Based on this model, the predictor describing the relation between past and future values can be calculated

$$\begin{aligned} &\vec{\boldsymbol{y}} = \boldsymbol{G}\Delta\vec{\boldsymbol{u}} + \boldsymbol{X}\begin{pmatrix} \overleftarrow{\boldsymbol{y}} \\ \overleftarrow{\boldsymbol{u}} \end{pmatrix} \qquad && \tilde{\boldsymbol{u}} = \Delta\vec{\boldsymbol{u}} \\ &\hat{\boldsymbol{y}} = \boldsymbol{G}\tilde{\boldsymbol{u}} + \boldsymbol{y_0} && \hat{\boldsymbol{y}} = \vec{\boldsymbol{y}} \end{aligned} \tag{2}$$

where $\vec{\boldsymbol{y}}$, $\vec{\boldsymbol{u}}$, $\overleftarrow{\boldsymbol{y}}$ and $\overleftarrow{\boldsymbol{u}}$ are future output, future input, past output and past input respectively. Matrices G and X contains coefficients which can be derived from the system of the model equations. The dimension of these matrices depends on horizons (explained later). The term consists of two parts: a forced response and a free response y0. The forced respond is determined by the future control increments while the free response depends on the past values and it is the response of the system to the constant control action.

Another part of MPC is the cost function which represents the quality of the control process for specific control action. This is usually defined as the sum of squared control errors and the sum of squared control increments.

$$\begin{aligned} \boldsymbol{J} &= (\hat{\boldsymbol{y}} - \boldsymbol{w})^{\boldsymbol{T}} (\hat{\boldsymbol{y}} - \boldsymbol{w}) + \lambda \tilde{\boldsymbol{u}}^{\boldsymbol{T}} \tilde{\boldsymbol{u}} \\ \boldsymbol{J} &= (\boldsymbol{G}\tilde{\boldsymbol{u}} + \boldsymbol{y_0} - \boldsymbol{w})^{\boldsymbol{T}} (\boldsymbol{G}\tilde{\boldsymbol{u}} + \boldsymbol{y_0} - \boldsymbol{w}) + \lambda \tilde{\boldsymbol{u}}^{\boldsymbol{T}} \tilde{\boldsymbol{u}} \end{aligned} \tag{3}$$

Parameter λ is a weighting factor which may be used for adjusting the behaviour of the control process. After some calculation, the cost function may be expressed in the following form

$$\boldsymbol{J} = c_0 + 2\boldsymbol{g}^{\boldsymbol{T}} \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{u}}^{\boldsymbol{T}} \boldsymbol{H} \tilde{\boldsymbol{u}} \tag{4}$$

where g is the gradient of the cost function and H is the Hessian matrix [17] which are calculated by

$$\begin{aligned} \boldsymbol{g}^{\boldsymbol{T}} &= \boldsymbol{G}^{\boldsymbol{T}} (\boldsymbol{y_0} - \boldsymbol{w}) \\ \boldsymbol{H} &= \boldsymbol{G}^{\boldsymbol{T}} \boldsymbol{G} + \lambda \boldsymbol{I} \end{aligned} \tag{5}$$

The aim of the optimization process is to find the optimal control action. The cost function is defined such that the best solution is located in the minimum of this function. The term of the cost function involves the setpoint, the output of the process and the control increments. Hence, the shape of this function depends on the model. Fig. 1 shows the shape of this function for the CARIMA model and the control horizon equal to 2.

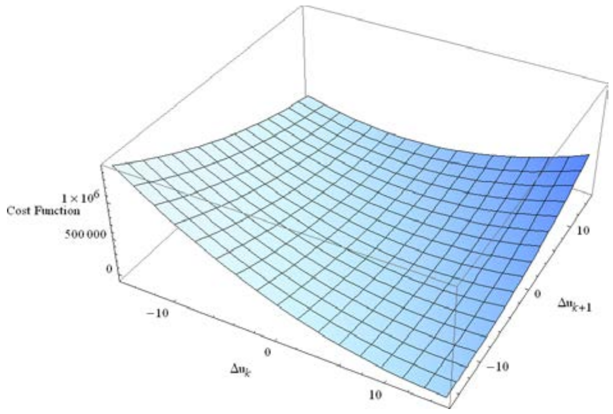


Fig. 1. Cost function

It is clear that the cost function is unimodal with one local minimum. This minimum can be obtained by derivative of this function and setting to zero. The analytical solution may be then expressed by the following equation.

$$\begin{aligned} \tilde{\boldsymbol{u}} &= -\boldsymbol{H}^{-1} \boldsymbol{g} \\ \tilde{\boldsymbol{u}} &= \boldsymbol{K}(\boldsymbol{w} - \boldsymbol{y_0}) \end{aligned} \tag{6}$$

where $\boldsymbol{K} = \left(\boldsymbol{G}^{\boldsymbol{T}} \boldsymbol{G} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{G}^{\boldsymbol{T}}$.

In MPC, there are two horizons which define the size of the predicted behaviour. The first horizon is denoted as prediction horizon (or output horizon) which is specified by a minimum horizon N_1 and a maximum horizon N_u. This horizon determines the length of the predicted output and of the setpoint which means the number of the future control errors. The dimensionality of the optimization problem and the size of the control vector are given by the control horizon N_u. The following values of the control increments are assumed to be equal to zero. Although the control horizon is usually higher than 1, only the first element of the control vector is applied and the whole procedure is then repeated in the following sampling period. This technique is called the receding horizon concept.

In practice, the variables are usually constrained which may cause the optimization problem more complex. There are usually three types of constraints: the constraints of the control increments, of the manipulated variable and of the output variable. In the case of state-space models, there are the constraints of the states as well. However, we consider only input-output model; therefore, the constraints of the states are not included in this paper.

$$\begin{aligned} \boldsymbol{\Delta u_{min}} &\le \boldsymbol{\Delta u} \le \boldsymbol{\Delta u_{max}} \\ \boldsymbol{u_{min}} &\le \boldsymbol{u} \le \boldsymbol{u_{max}} \\ \boldsymbol{y_{min}} &\le \boldsymbol{y} \le \boldsymbol{y_{max}} \end{aligned} \tag{7}$$

The constraints are then represented as the system of inequalities

$$\begin{pmatrix} \boldsymbol{I} \\ -\boldsymbol{I} \\ \boldsymbol{T} \\ -\boldsymbol{T} \\ \boldsymbol{G} \\ -\boldsymbol{G} \end{pmatrix} \boldsymbol{\Delta u} \ge \begin{pmatrix} \boldsymbol{1\Delta u_{min}} \\ -\boldsymbol{1\Delta u_{max}} \\ \boldsymbol{1u_{min}} - \boldsymbol{1u}(\boldsymbol{k}-\boldsymbol{1}) \\ -\boldsymbol{1u_{max}} + \boldsymbol{1u}(\boldsymbol{k}-\boldsymbol{1}) \\ \boldsymbol{1y_{min}} - \boldsymbol{y_0} \\ -\boldsymbol{1y_{max}} + \boldsymbol{y_0} \end{pmatrix} \tag{8}$$

and they can be expressed by the condensed matrix form as it is shown in eq. 9.

$$\boldsymbol{A\Delta u} \ge \boldsymbol{b} \tag{9}$$

Fig. 2 shows the feasible area within the cost function created due to the constraints. This area is given by the combination of the constraints.

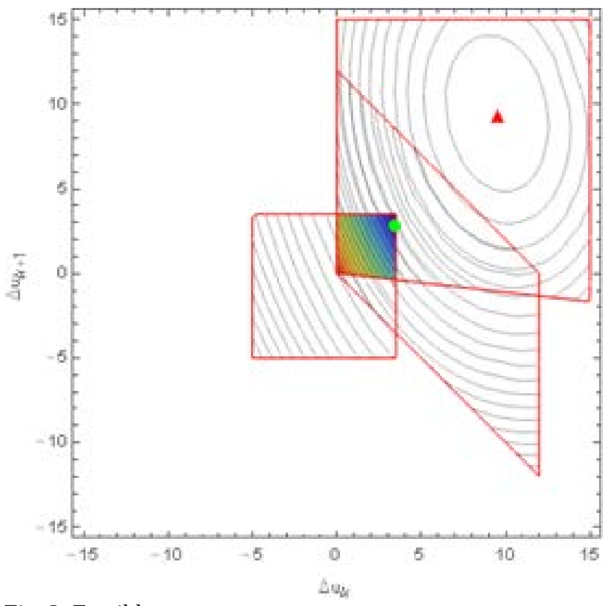


Fig. 2. Feasible area

The dimensionality of the feasible area is determined by the control horizon. In general, the feasible area is a subspace (Fig. 3) which is one dimensional lower than the cost function. The result shape is in the form of polytope and the optimization problem is to find the best solution within this area.

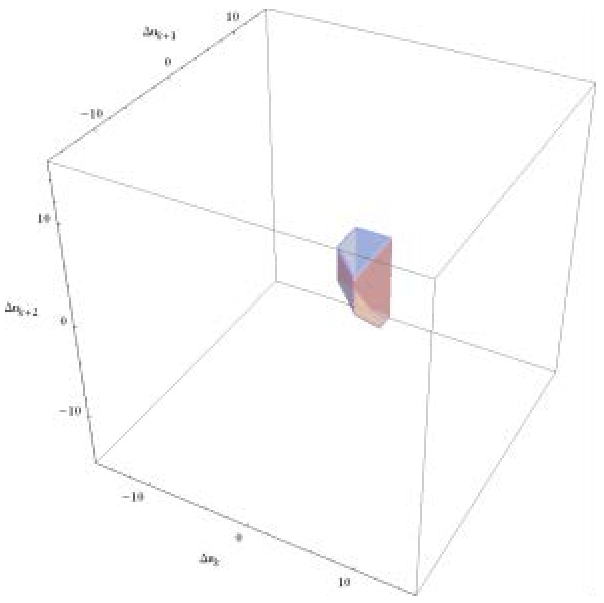


Fig. 3. Feasible area (3D case)

As can be seen, the constraints may cause that the analytical solution is located outside of the feasible area, thus the proper algorithm should be applied to solve this problem.

III. HillClimbing Algorithm

Hill climbing algorithm (HC) is a type of evolutionary algorithms which are characterized by their deterministic approach with a certain rate of probability. For the sake of convenience, table 1 shows the list of terms used for the explanation of the basic principle of HC.

Table 1 List of terms of hill climbing algorithm

Term	Meaning
individual	specific values of the control vector $\boldsymbol{\Delta u}$ (one point within the feasible area)
population	a group of individuals
Leader	the best solution within a population

The basic idea of hill climbing algorithm is simple and it can be described by the following steps.

1) determine the initial individual (first leader)
2) generate the population around the previous leader
3) the best solution within the population becomes new leader
4) go back to step 2 and repeat the procedure until the stop conditions are not fulfilled

In the original algorithm, the initial individual must be chosen manually or it must be generated randomly. In the constrained case, the feasible area is created within the cost function; therefore, the initial individual should be chosen such that this point is located within the feasible area. There are several ways of obtaining this point such as testing the key points, solving the system of inequalities, the random walk method or other techniques. The last one is not suitable for high dimensional cases due to the fact that the ratio between the space occupied by the feasible area and the total space given by the range of finding rapidly decrease in higher dimensions. Because of this fact and factors some key points were tested first.

The first key point is the analytical solution. If this point is located within the feasible area, it can be directly applied in the control process. The second point is the zero point which is the equivalent of a constant control action. Therefore, there is a high probability that this point is located within the feasible area. The next possibility is to use the previous solution. If the key points are located outside of the feasible area, their neighbourhoods can also be tested.

In step 2, the generation around the previous leader is generated. These points are usually generated with the normal distribution. Due to the constraints, some points may appear outside of the feasible area; hence, a test function or a corrective function should be defined. This function is given by the system of inequalities of the constraints which means that every generated solution is tested whether it fulfils the inequalities. If the generated individual is located outside of the feasible area, it can be dropped, moved to the boundary (if possible) or regenerated. The first option is the least time-consuming; however in this case, the size of the population should be such that at least one of the generated points lies within the feasible area. In contrast, higher size of population implies higher computational demands. The computational expense is also affected by the variance of the normal distribution.

In the third step, all of the individuals are evaluated and the best solution is selected. If the previous leader is involved in the evaluation, the algorithm exhibits local search behaviour. This approach is appropriate for the solving of unimodal functions due to the gradient characteristics. On the other hand, if the previous leader is not included to the evaluated population, it may accept worse solutions; therefore, it may skip local minima and it is possible to solve multimodal functions. However, it may exhibit a cycling behaviour and as a result of this, some advanced techniques have been developed such as tabu search, simulated annealing and other algorithms.

In the last step, the stop conditions must be defined in order to finish the optimization process within the finite period. The stop conditions in the original algorithms are defined as the maximum number of iterations and the minimum difference between the cost functions of the previous and of the actual leader. In our case, the last condition was replaced by the minimum distance between these points. This can cover the requirement of the accuracy of the control increment Δu. The algorithm may also stop if there is no new best solution or the population is empty.

Fig. 4 depicts the application of the hill climbing algorithm in the model predictive control.

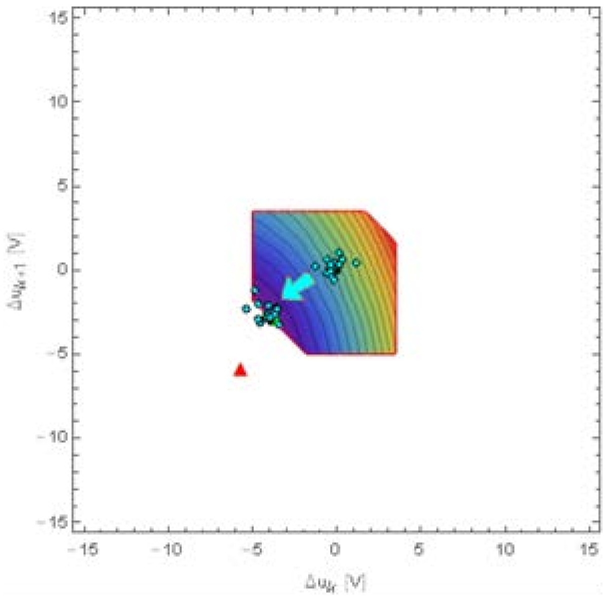


Fig. 4. Hill climbing algorithm

As you can see, some individuals of the final population are located outside of the feasible area; consequently, they are not considered during the evaluation process.

IV. Evolutionary-Gradient Algorithm

The basic idea of the evolutionary-gradient algorithm (EG) is to combine the advantages of gradient algorithms and of evolutionary algorithms. The proposed algorithm is based on the principle of the hill climbing algorithm explained above. However, as you can see in Fig. 4, some of the individuals are located in the opposite direction than the searched minimum. Therefore, the objective of the EG algorithm is to generate the population in the way of the negative gradient which may lead to the faster convergence in high dimensional case.

The gradient of an actual position can be derived from equations (4) and (6) and the result term is presented in eq. (10).

$$\begin{aligned} \frac{\partial \boldsymbol{J}}{\partial \Delta \boldsymbol{u}} &= 2\left(\boldsymbol{G}^{\boldsymbol{T}}\boldsymbol{G} + \lambda \boldsymbol{I}\right)\Delta \boldsymbol{u} + 2\boldsymbol{G}^{\boldsymbol{T}}\left(\boldsymbol{y_0} - \boldsymbol{w}\right) \\ \frac{\partial \boldsymbol{J}}{\partial \Delta \boldsymbol{u}} &= 2\boldsymbol{H}\Delta \boldsymbol{u} + 2\boldsymbol{g}^{\boldsymbol{T}} \end{aligned} \tag{10}$$

Although the absolute value of the gradient can be calculated, only the direction of the negative gradient is important.

$$\boldsymbol{ngrad} = -\boldsymbol{H}\Delta \boldsymbol{u} - \boldsymbol{g}^{\boldsymbol{T}} \tag{11}$$

In order to generate the population with the determined variation of the normal distribution, the magnitude of the negative gradient is adjusted so that the maximum value of this vector is equal to 1.

$$\boldsymbol{dir} = \frac{\boldsymbol{ngrad}}{\max\left(\left|\boldsymbol{ngrad}\right|\right)} \tag{12}$$

The population is then generated in the way of the obtained direction. The specific individual is then generated by the following term.

$$\boldsymbol{ind} = \boldsymbol{dir} \cdot \left|N\left(0, \sigma_g\right)\right| + \boldsymbol{leader} \tag{13}$$

The basic principle of the proposed algorithm is depicted in figure 5.

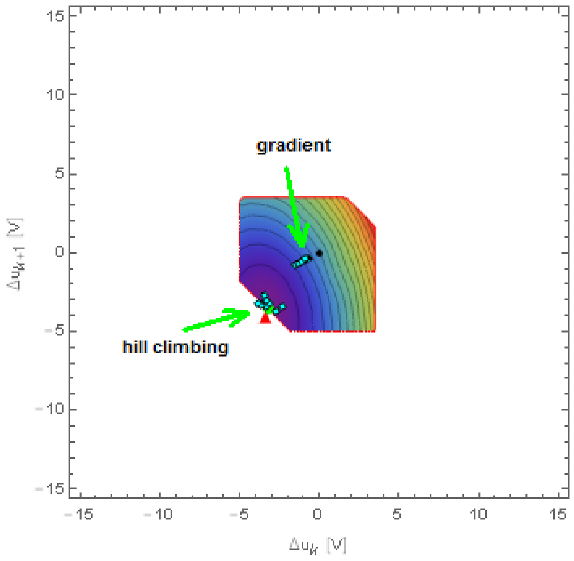


Fig. 5. Evolutionary-gradient algorithm

As can be seen, the evolutionary-gradient algorithm is applied until the boundary of the feasible area is not reached. Then, the algorithm is switched to the hill climbing mode as it is described in the previous section. This is due to the possibility of finding a misleading solution on the boundary of the feasible area.

V. Results and DiscussionHelpful Hints

A. Simulation

For the purpose of the simulation, the second order system was chosen and identified.

$$G(s)=\frac{k}{(T_1\ s+1)(T_2\ s+1)} \tag{14}$$

The continuous system was then discretized in order to respect the discrete character of the control process. The polynomials of the CARIMA model (eq. 1) are then in the following form

$$\begin{aligned}A&=1+a_1\ z^{-1}+a_2\ z^{-2}\\ B&=b_0+b_1\ z^{-1}+b_2\ z^{-2}\end{aligned} \tag{15}$$

and polynomial C was chosen to be equal to 1. After the multiplication of polynomial A by Δ, the following equation is obtained

$$\Delta A=1+(a_1-1)z^{-1}+(a_2-a_1)z^{-2}+a_2z^{-3} \tag{16}$$

and the differential equation of the controlled system can be expressed by the following form.

$$\begin{aligned}y(k)&=-(a_1-1)y(k-1)-(a_2-a_1)y(k-2)\\ &-a_2y(k-3)+b_1\,\Delta u(k-1)+b_2\,\Delta u(k-2)\end{aligned} \tag{17}$$

The future outputs can be defined as the system of equations (predictor) and by the substitutions and splitting of the terms, the coefficients of matrices G and X (eq. 2) can be derived. Matrix G is then in the form

$$\mathbf{G}=\begin{pmatrix} g_0 & 0\\ g_1 & g_0\\ g_2 & g_1\end{pmatrix} \tag{18}$$

where

$$\begin{aligned}g_0&=b_1\\ g_1&=(1-a_1)b_1+b_2\\ g_2&=(1-a_1)^{\,2}b_1+(a_1-a_2)b_1+(1-a_1)b_2\end{aligned} \tag{19}$$

Matrix X is expressed by the following form

$$\mathbf{X}=\begin{pmatrix}\mathbf{X_1} & \mathbf{X}_2 & \mathbf{X}_3 & \mathbf{X}_4\end{pmatrix}$$

$$\mathbf{X_1}=\begin{pmatrix}(1-a_1)\\ (1-a_1)^2+(a_1-a_2)\\ (1-a_1)^3+2(1-a_1)(a_1-a_2)+a_2\end{pmatrix}$$

$$\mathbf{X}_2=\begin{pmatrix}(a_1-a_2)\\ (1-a_1)(a_1-a_2)+a_2\\ (1-a_1)^2(a_1-a_2)+(a_1-a_2)^2+(1-a_1)a_2\end{pmatrix} \tag{20}$$

$$\mathbf{X}_3=\begin{pmatrix}a_2\\ (1-a_1)a_2\\ (1-a_1)^2a_2+(a_1-a_2)a_2\end{pmatrix}$$

$$\mathbf{X}_4=\begin{pmatrix}b_2\\ (1-a_1)b_2\\ (1-a_1)^2b_2+(a_1-a_2)b_2\end{pmatrix}$$

These shapes of the matrices depend on the model and the size of horizons. In the case of higher horizons, the coefficients can be calculated by a recurrent procedure.

The horizons were set such that the graphs are possible to be depicted. All of the parameters are shown in table 2.

Table 2 Parameters of simulation

System parameters		
gain	k	35.88
time constant 1	T_1	519.29
time constant 2	T_2	40.80
MPC parameters		
weighting factor	λ	1
minimum horizon	N_1	1
maximum horizon	N_2	3
control horizon	N_u	2
Constraints		
range of control increment	$\langle \Delta u_{\min}, \Delta u_{\max} \rangle$	$\langle -5,3.5 \rangle$
range of manipulated value	$\langle u_{\min}, u_{\max} \rangle$	$\langle 0,12 \rangle$
range of output	$\langle y_{\min}, y_{\max} \rangle$	$\langle 0,310 \rangle$

There are also some parameters of the HC and the EG algorithms. The basic idea of both is to generate the population with the normal distribution. Since the constraints determine the range of the finding, the variance of this distribution is related to the range of the control increments it is then calculated by this equation.

$$\sigma = \frac{cov(\Delta u_{\max} - \Delta u_{\min})}{600} \qquad (14)$$

The parameters of the algorithms are presented in table 3.

Table 3 Parameters of algorithms

Hill climbing		
distance of leaders	ε	0.01
rate of coverage	cov_{HC}	40
size of population	$popSize_{HC}$	8
max. number of iterations	NI	100
Evolutionary-gradient		
rate of coverage	cov_{EG}	60
size of population	$popSize_{EG}$	8

The time response of control process is presented in figure 6.

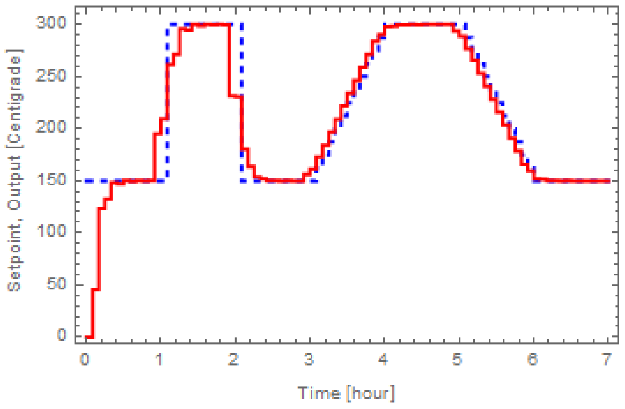
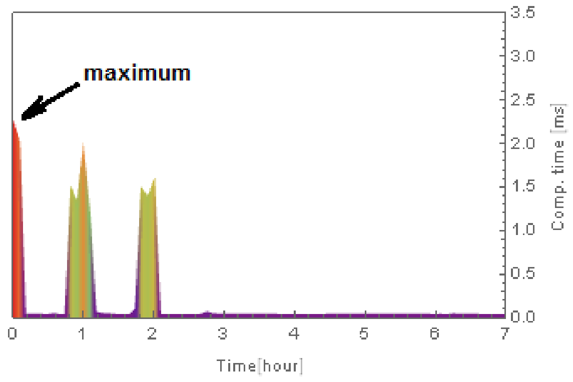


Fig. 6 Output and setpoint

As can be seen, the output tracks the setpoint. The highest control error is located in the vicinity of substantial changes of the setpoint.

Figure7 shows the time distribution of the computational expense measured by the computational time. As you can see, the highest computational times are located at the same time points as in the previous case. It is due to the constraints and the saturation of the signals.

a)



b)

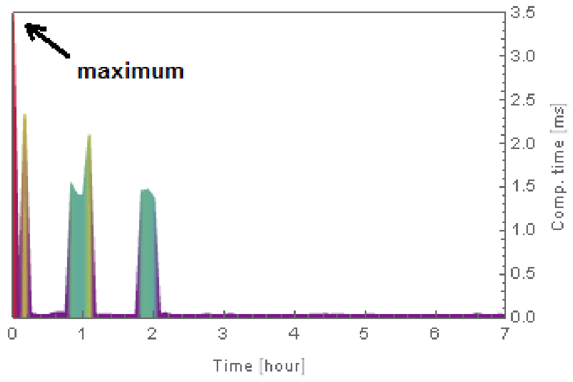


Fig. 7. Computational time of a) EG algorithm; b) QP algorithm.

In this particular case, the EG algorithm exhibits better performance than the quadratic programing (QP) algorithm. However, the results are strongly dependent on the setting of the algorithm parameters. Moreover, the computation time is only one of the parameters which should be examined. For this reason, other characteristics such as the performance stability and the control accuracy have been tested. During the testing process, the maximum effort of keeping comparable conditions has been made such as adding the preparing cycles, calculating during the same time period or increasing the priority of the calculating process. The results are presented in table 4 (lower value means better result).

Table 4 Performance results

	comp. time	perfor. stability	accuracy
EG	2.10	0.39	234.67
HC	2.28	0.29	238.29
QP	2.01	0.15	232.00

As can be seen from the table, all of the numbers are similar. However, quadratic programming seems to be better in all examined characteristics. Despite this fact, the proposed algorithm is at least comparable and, as you see in Fig. 7, it may be sometimes better. There is also a little improvement compared to the HC algorithm. Besides, the results of algorithms are strongly dependent on setting which may be another optimization problem.

The evolutionary-gradient and quadratic programming algorithm were also compared by computational time in different dimensions. Both horizons were increased and the results are presented in figure 8.

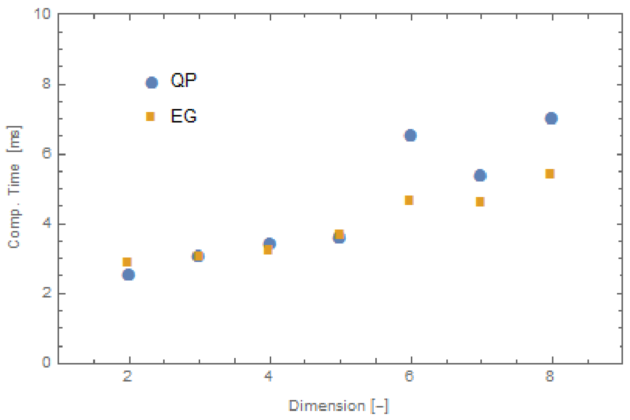


Fig. 8 Computational expense in different dimensions

As can be seen, the evolutionary-gradient algorithms exhibits a little improvement compared to QP algorithm. Moreover, the setting of algorithm may be adjusted in order to fulfil the specific requirements of the control process.

The proposed algorithm has fulfilled the requirements; however, the dimensionality of the problem was relatively low. It is assumed that in high dimensional case, the improvement may be much higher. This is due to the fact that the size of the population of the HC depends on the dimensionality. On the other hand, the EG generates the population within 1-dimensional line regardless of the dimensionality of the optimization; therefore, the EG algorithm is the promising method of the solving of high dimensional problems. This will be the objective of future research.

It should be also noted that the shape of the cost function is relatively simple due to the linearity of the model. It means that evolutionary techniques may be the proper method of the solving of strongly nonlinear even time-variant systems. This is the secondary aim of future research.

VI. Conclusion

The aim of this paper was to introduce the basic aspects of the evolutionary-gradient method and to develop the algorithm. The basic idea of this concept is to combine the advantages of evolutionary and gradient techniques in order to achieve better computational performance. First, the principle of the hill climbing algorithm has been described. Then, the ideas of evolutionary-gradient algorithm, which is based on the previous method, have been presented. Finally, some simulations were executed in order to validate the proposed technique and it has been also compared with other methods by means of the investigation of the performance characteristics.

It has been revealed, that the results depends on the setting of the parameters. However, there is a little improvement compared to the HC algorithm. Due to the basic principle of the proposed algorithm, it is assumed that there may be significant improvement in higher dimensional problem which will be the aim of future research. Moreover, the evolutionary techniques may be the promising methods in case of nonlinear and time-variant systems. This is the secondary aim of future research.

Acknowledgment

Authors are thankful to Internal Grant Agency (IGA/CebiaTech/2015/026) of Tomas Bata University in Zlín, Czech Republic for financial support.

References

[1] E. F. Camacho, C. Bordons, *Model Predictive Control*, Springer-Verlag, London, 2004.

[2] M. Morari, J. H. Lee, Model predictive control: past, present and future. *Computers and Chemical Engineering*, 23, 1999, 667-682.

[3] R. R. Bitmead, M. Gevers, V. Hertz, *Adaptive Optimal Control. The Thinking Man's GPC*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[4] P. Thitiyasook, P. Kittisupakorn, "Model Predictive Control of a Batch Reactor with Membrane – Based Separation," in Proc. 7th WSEAS Int. Conf. on Signal Processing, Robotics and Automation (ISPRA '08), University of Cambridge, UK, 2008, pp. 88-92.

[5] R. Balan, O. Hancu, S. Stan, C. Lapusan, R. Donca, "Application of a Model Based Predictive Control Algorithm for Building Temperature Control," in Proc. 3rd WSEAS Int. Conf. on Energy Planning, Energy Saving, Environmental Education (EPESE '09), Tenerife, Spain, 2009, pp. 97-101.

[6] Z. Ju, W. Wanliang, "Synthesis of Explicit Model Predictive Control System with Feasible Region Shrinking," in Proc. 8th WSEAS Int. Conf. on Robotics, Control and Manufacturing Technology (ROCOM '08), Hangzhon, China, 2008, pp. 80-85.

[7] D. W. Clarke, C. Mohtadi, P. S. Tuffs, Generalized predictive control, part I: the basic algorithm. *Automatica*, 23, 1987, 137-148.

[8] D. W. Clarke, C. Mohtadi, P. S. Tuffs, Generalized predictive control, part II: extensions and interpretations. *Automatica*, 23, 1987, 149-160.

[9] G.M. Lee, N.N. Tam, and N.D. Yen, *Quadratic Programming and Affine Variational Inequalities:A Qualitative Study*, Springer, 2005.

[10] D.G. Luenberger and Y. Ye, *Linear and nonlinear programming*, 3rd ed. New York: Springer, 2008.
[11] T. Back, D. B. Fogel, Z. Michalewicz, *Handbook of evolutionary algorithms*, Oxford: Oxford University Press, 1997
[12] Z. Muhammad, Z.M. Yusoff, M.H.F. Rahiman, M.N. Taib, Steam temperature control for steam distillation pot using model predictive control, *Signal Processing and its Applications (CSPA), 2012 IEEE 8th International Colloquium on*, 2012, 474-479.
[13] M. Rau, D. Schroder, Model predictive control with nonlinear state space models, Advanced *Motion Control, 2002. 7th International Workshop on*, 2002, 136-141.
[14] G.I. Suárez, O.A. Ortiz, P.M. Aballay, N.H. Aros, Adaptive neural model predictive control for the grape juice concentration process, *Industrial Technology (ICIT), 2010 IEEE International Conference on*, 2010, 57-63
[15] G. Tan, H. Hao, Y. Wang, Real time turning flow estimation based on model predictive control, *Information Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International*, 1, 2011, 356-360.
[16] H.G. Han; X.L. Wu; J.F. Qiao, Real-Time Model Predictive Control Using a Self-Organizing Neural Network, *Neural Networks and Learning Systems, IEEE Transactions on*, 24, 9, 2013, 1425-1436.
[17] Z. Dostál, Optimal *Quadratic Programming Algorithms: With Applications to Variational Inequalitie*s, New York: Springer, 2009.