ALTERNATIVE APPROACH TO OPTIMIZATION IN MODEL PREDICTIVE CONTROL USING HILL CLIMBING ALGORITHM

Jan Antos, Marek Kubalcik

Department of Process Control, Faculty of Applied Informatics, Tomas Bata University in Zlín, nám. T. G. Masaryka 5555, 760 01 Zlín, Czech Republic

Abstract

The term predictive control designates a class of control methods suitable for control of various kinds of systems. One of the major advantages of predictive control is its ability to do on-line constraints handling in a systematic way. The predictive control is based on the prediction of a system behavior using a model. Based on this prediction, it is possible to optimize the systems behavior by utilization of a cost function. Each of control variables may be limited thus creating a specific subspace within a cost function. This problem is computationally complex and must be solved in each sampling period by optimization algorithms. Various kinds of algorithms may be applied. This contribution is focused on an alternative approach to optimization by implementation of Hill Climbing algorithm. The motivation for this concept is an effort to find algorithms suitable for reduction of computational expenses. These algorithms might be applied for control of systems with faster dynamics.

Keyword: predictive control; optimization; simulation; evolutionary algorithm; hill climbing

This Publication has to be referred as: Antos, J[an] & Kubalcik, M[arek] (2016). Alternative Approach to Optimization in Model Predictive Control Using Hill Climbing Algorithm, Proceedings of the 26th DAAAM International Symposium, pp.0856-0864, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-07-5, ISSN 1726-9679, Vienna, Austria
DOI:10.2507/26th.daaam.proceedings.119

1. Introduction

Model Predictive Control (MPC) or only Predictive Control [1], [2], [3] is one of the control methods which have considerably developed over the past few years. Predictive control is essentially based on the discrete or sampled models of processes. The computation of appropriate control algorithms is then realized especially in the discrete domain.

The basic idea of MPC is to use the model of a controlled process to predict *N* future outputs of the process. The trajectory of future manipulated variables is given by solving an optimization problem incorporating a suitable cost function and constraints. Only the first element of the obtained control sequence is applied. The whole procedure is then repeated in the following sampling period. This principle is known as the receding horizon strategy [4], [5].
The main computational problem in predictive control is solving of the optimization problem. A commonly used approach for the optimization principle in predictive control is the quadratic programming [6] which enables the inclusion of constraints [7] in a controller synthesis. The cost function has a quadratic form with a positively defined Hessian [8].

In the past, the predictive control was mostly applied to control of systems with large time constants. Nowadays it is also possible to apply it to control of systems with faster dynamics because of increasing computational power. The crucial issue of optimization is computational demands; therefore, finding an appropriate algorithm is important. Although it is possible to deploy many algorithms, quadratic programming is still used in most cases. However, there are many other methods which may provide comparable results by adjusting their parameters; moreover, it is possible to fulfil the specific requirements of the control process.

This paper deals with an alternative approach based on the methods of artificial intelligence [9]. However there are many methods of artificial intelligence, Hill Climbing algorithm was applied to the optimization problem because of its simplicity and the possibility of setting of various parameters. This algorithm is easily programmable and it is able to solve this type of functions. The supposed advantage of this concept could be the reduction of the computational expense compared to existing methods.

2. Methods

2.1. Predictive control

Model predictive control is based on a model. There are various kinds of models which can be applied such as impulse response, step response, transfer function, ARMA, a neural network and other functions. One of the most widely used models is the CARIMA (Controller Autoregressive Integrated Moving Average) model which directly includes the difference of a manipulated variable and which can be written in the following form

$$Ay(k) = Bu(k-1) + \frac{C}{\Delta} n(k), \quad \text{where} \quad \begin{aligned} A &= 1 + a_1 z^{-1} + \ldots + a_n z^{-n} \\ B &= b_0 + b_1 z^{-1} + \ldots + b_m z^{-m} \end{aligned} \tag{1}$$

where polynomials *A* and *B* represent a transfer function of the system, $\Delta = 1 - z^{-1}$, *y* is an output variable, *u* is the manipulated variable, *C* is a colouring polynomial and *n* is a nonmeasurable noise which is assumed to have a zero-mean value and a constant covariance. Consequently, the model describes a relation between input and output.
Based on the model, a predictor, which defines the relation between past and future values, can be calculated. The predictor can be divided into two parts: a free response which is determined by the past values and a forced response determined by the future control increments. The predictor is then formulated as

$$\vec{\mathbf{y}} = \mathbf{G}\Delta\vec{\mathbf{u}} + \mathbf{X}\begin{pmatrix} \overleftarrow{\mathbf{y}} \\ \overleftarrow{\mathbf{u}} \end{pmatrix} \tag{2}$$

where $\vec{\mathbf{y}}$ and $\vec{\mathbf{u}}$ are the future values and $\overleftarrow{\mathbf{y}}$ and $\overleftarrow{\mathbf{u}}$ are the past values. Matrices **G** and **X** contain coefficients which are derived from the system of model equations. The size of the matrices is given by horizons which are explained later. The predictor can also be written in the form shown in equation (3)

$$\hat{\mathbf{y}} = \mathbf{G}\tilde{\mathbf{u}} + \mathbf{y_0} \tag{3}$$

where $\mathbf{y_0}$ is the free response of the process.
Matrix **G** involves the values of the step sequence and it can be expressed by

$$\mathbf{G} = \begin{pmatrix} g_0 & 0 & 0 & \cdots & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ g_2 & g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{N_2-1} & g_{N_2-2} & g_{N_2-3} & \cdots & g_0 \end{pmatrix} \tag{4}$$

Another part of the predictive control is a cost function which is defined as the sum of squared control errors and the sum of squared control increments. This function is, therefore, calculated by

$$\begin{aligned} J &= (\hat{\mathbf{y}} - \mathbf{w})^T (\hat{\mathbf{y}} - \mathbf{w}) + \lambda \tilde{\mathbf{u}}^T \tilde{\mathbf{u}} \\ J &= (\mathbf{G}\tilde{\mathbf{u}} + \mathbf{y_0} - \mathbf{w})^T (\mathbf{G}\tilde{\mathbf{u}} + \mathbf{y_0} - \mathbf{w}) + \lambda \tilde{\mathbf{u}}^T \tilde{\mathbf{u}} \\ J &= c_0 + 2\mathbf{g}^T \tilde{\mathbf{u}} + \tilde{\mathbf{u}}^T \mathbf{H} \tilde{\mathbf{u}} \end{aligned} \tag{5}$$

where λ is a weighting factor, which is another degree of freedom, **g** is the gradient of the cost function and **H** is the Hessian matrix.
They can be expressed by the following equations

$$\begin{aligned} \mathbf{g}^T &= \mathbf{G}^T (\mathbf{y_0} - \mathbf{w}) \\ \mathbf{H} &= \mathbf{G}^T \mathbf{G} + \lambda \mathbf{I} \end{aligned} \tag{6}$$

The solution of the optimization problem can be obtained by the derivative of the cost function. Equation (7) considers the vector of the control increments

$$\begin{aligned} \tilde{\mathbf{u}} &= -\mathbf{H}^{-1}\mathbf{g} \\ \tilde{\mathbf{u}} &= \mathbf{K}(\mathbf{w} - \mathbf{y_0}) \end{aligned} \tag{7}$$

Where

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

Despite the fact that the length of the control vector is mostly greater than 1, only the first element is applied to the control process and the whole procedure is then repeated in the next sampling period. This method is called the Receding Horizon concept.
There are two horizons in the predictive control: a prediction horizon and a control horizon. The prediction horizon is specified by two values denoted as N_1 (a minimum prediction horizon) and N_2 (a maximum prediction horizon) which affect the length of the predicted output vector. The length of the vector of the control increments is given by the control horizon called N_u.

2.2. Cost function and constraints

As can be seen in equation (5), the cost function describes the relationship between the future variables: the control errors and the control increments. This equation has a quadratic form which is depicted in Fig. 1.

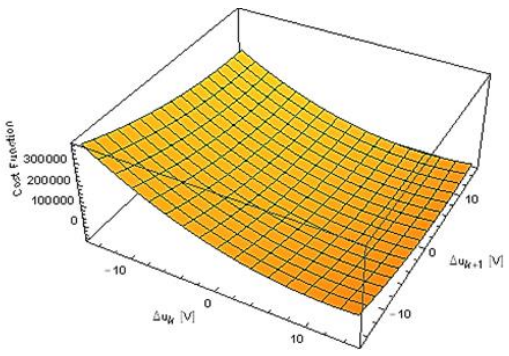


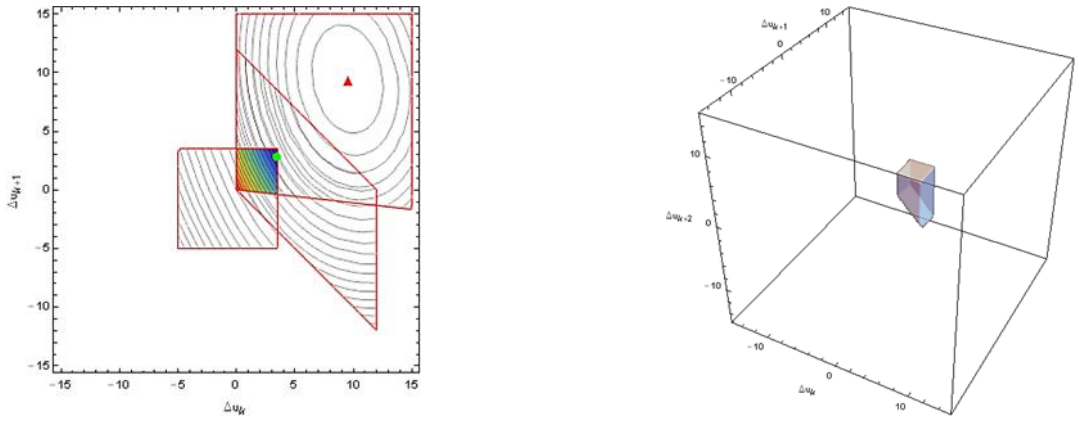
Fig. 1. Cost Function

Though the figure shows the cost function in 3D space, it has the same character in spaces with higher dimensionality. The function is smooth and unimodal (in case of CARIMA model) which means that the minimum can simply be found by the derivative of this function. In this case, the computational demands of the optimization problem are very low.

In a real system, the variables of the control process are constrained. Three types of constrains can be considered. There are usually three types of constraints: the constraints of the control increments, of the manipulated variable and of the output variable. The constraints can be represented by the system of inequalities or in the matrix form (see equation (8)).

$$\begin{aligned} \mathbf{\Delta u}_{\mathbf{min}} \le \mathbf{\Delta u} \le \mathbf{\Delta u}_{\mathbf{max}} \\ \mathbf{u}_{\mathbf{min}} \le \mathbf{u} \le \mathbf{u}_{\mathbf{max}} \\ \mathbf{y}_{\mathbf{min}} \le \mathbf{y} \le \mathbf{y}_{\mathbf{max}} \end{aligned} \quad \Rightarrow \quad \begin{pmatrix} \mathbf{I} \\ -\mathbf{I} \\ \mathbf{T} \\ -\mathbf{T} \\ \mathbf{G} \\ -\mathbf{G} \end{pmatrix} \mathbf{\Delta u} \le \begin{pmatrix} \mathbf{1}\Delta\mathbf{u}_{\mathbf{max}} \\ -\mathbf{1}\Delta\mathbf{u}_{\mathbf{min}} \\ \mathbf{1u}_{\mathbf{max}} - \mathbf{1u}(k-1) \\ -\mathbf{1u}_{\mathbf{min}} + \mathbf{1u}(k-1) \\ \mathbf{1y}_{\mathbf{max}} - \mathbf{y}_{\mathbf{0}} \\ -\mathbf{1y}_{\mathbf{min}} + \mathbf{y}_{\mathbf{0}} \end{pmatrix} \quad \mathbf{A\Delta u} \le \mathbf{b} \tag{8}$$

The shape of an area, created due to the constraints, is then depicted in Fig. 2a.



a) b)

Fig. 2. Constraints a) Nu = 2; b) Nu = 3

In case of the first type of the constraints, the shape is in a form of a hypercube. The shapes in other cases are non-squared; however, they are similar which is given by the way of mapping between input and output of the system. The result is then in the intersection of these areas (hereinafter referred to as a feasible area). According to the result shape and the location, the analytical solution may appear outside of the feasible area. In this case, the computational demands of optimization are higher. The feasible area in higher dimensions has the form of a subspace which is illustrated in Fig. 2b.

The optimization problem is then to find the minimum value of the cost function within the feasible area, which is one dimension lower than the cost function.

2.3. Algorithms, parameters and software

The crucial issue of the optimization problem is a computational time in each sampling period. From many evolutionary algorithms, hill climbing algorithm was chosen because of its simplicity and the possibility of setting various parameters. The basic principle of this algorithm is to generate a population around a leader and then to determine a new best solution (new leader) within this population. This procedure is then repeated until a stop condition is not fulfilled. The initial point within the feasible area is determined from the key points which may be obtained by several methods. The constraints are represented by a test function which returns true if the point is located within the feasible space.

The algorithm was manually programmed in Matlab software due to the specific shape of the feasible area. Data were then displayed in Wolfram Mathematica software which was chosen because of the wide range of features and graphic options.

In the simulation, a minimum phase non-oscillatory system without delay was used.

$$G(s) = \frac{35.88}{(519.29s + 1)(40.80s + 1)} \tag{9}$$

This system was then discretized in order to respect the discrete character of the control process and the coefficients of the CARIMA model (equation 1) were derived. The magnitudes of the horizons were set to $N_1 = 1$, $N_2 = 3$ and $N_u = 2$ in order to have the possibility of displaying the data. The coefficients of matrix ***G*** can be calculated as follows.

$$\mathbf{G} = \begin{pmatrix} b_1 & 0 \\ da_1 b_1 + b_2 & b_1 \\ da_1{}^2 b_1 + da_2 b_1 + da_1 b_2 & da_1 b_1 + b_2 \end{pmatrix} \quad \text{where} \quad \begin{matrix} da_1 = (1 - a_1) \\ da_2 = (a_1 - a_2) \\ da_3 = a_2 \end{matrix} \tag{10}$$

The shape of matrix X is presented in the following equation.

$$\mathbf{X} = \begin{pmatrix} da_1 & da_2 & da_3 & b_2 \\ da_1{}^2 + da_2 & da_1 da_2 + da_3 & da_1 da_3 & da_1 b_2 \\ da_1{}^3 + 2da_1 da_2 + da_3 & da_1{}^2 da_2 + da_2{}^2 + da_1 da_3 & da_1{}^2 da_3 + da_2 da_3 & da_1{}^2 b_2 + da_2 b_2 \end{pmatrix} \tag{11}$$

The values of constraints were determined as follows.

$$\begin{aligned} -5 \le \Delta u \le 3.5 \\ 0 \le u \le 12 \\ 0 \le y \le 310 \end{aligned} \tag{12}$$

The stop condition of Hill climbing algorithm was defined as the distance of two last leaders and it was set to ε = 0.01. The parameters of hill climbing algorithm were variably determined in order to investigate the results of optimization which were measured by the computational time and the accuracy of the control process. The simulations were repeated several times because of the testing the stability of a computational performance. There were also added some preparing cycles and the priority of the computing process was increased in order to keep comparable conditions. For the purpose of comparing, the proposed approach is appropriate due to the fact that only the performance of compared algorithms is important and not their principles of work.

3. Results and discussion

3.1. Simulation

After the simulation, the time responses of the control process were shown which is depicted in Fig. 3.

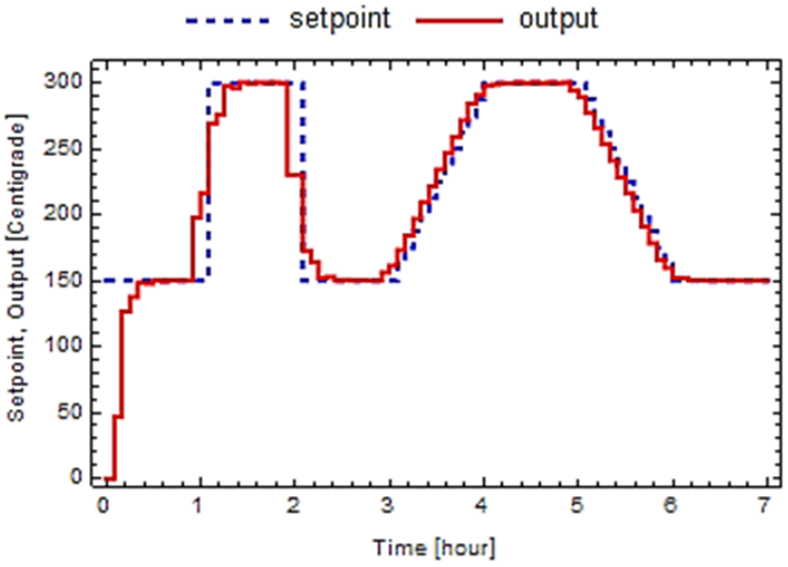
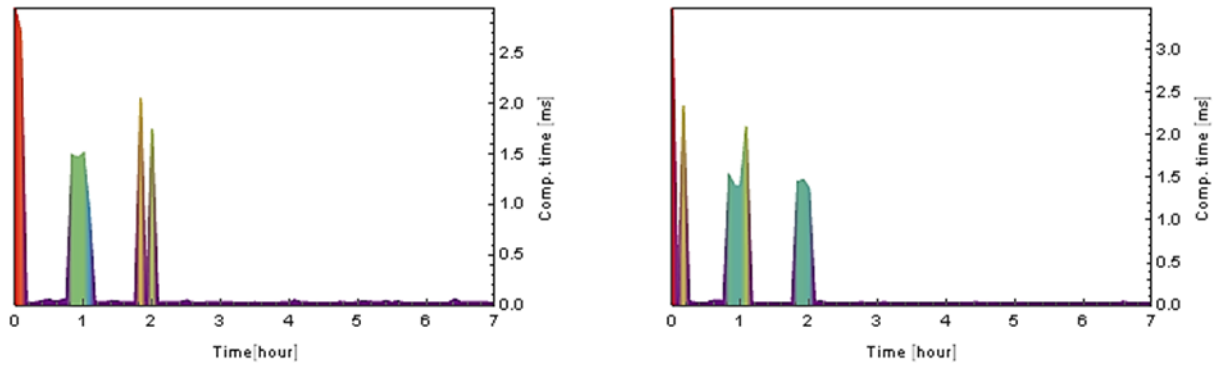


Fig. 3. Time responses of the control process

The set-point involves step functions, ramp functions and constant sections. As can be seen, the highest control errors are situated in the vicinity of the step functions. However, the quality of the control process is acceptable and the output variable traces the set-point.

3.2. Performance and Calculation



a) b)

Fig. 4. Time distribution of the computational time a) hill climbing algorithm; b) quadratic programming

Fig. 4a shows the time distribution of the computational time of one setting of the hill climbing algorithm. As can be seen, the maximum values are situated around the same time points as in the previous case. The substantial change in the set-point may, therefore, cause that the analytical solution is placed outside of the feasible area due to the constraints. In this case, the optimization problem must be solved. In the other cases, the analytical solution may directly be applied. Despite this fact, the maximum value is the most important because of the necessity of solving of the optimization problem within the sampling period. The computational time for Quadratic programming algorithm was also measured in order to compare the computing performance. The time distribution of this performance is presented in Fig. 4b.

The maximum computational time of both sets is similar; in this case, the computing performance of hill climbing is even better.

The range of the settings of hill climbing parameters was determined by displaying the movement of a population which is shown in Fig. 5.

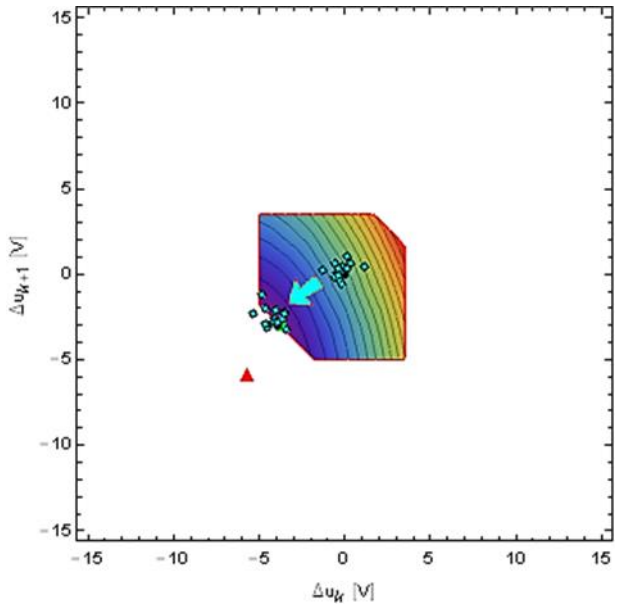


Fig. 5. Movement of population through the feasible area

In this case, the zero point was chosen as the initial point. The optimization process may actually be split into two phase. In the first phase, the feasible area must be found. There are several strategies of finding this area such as testing key points (and its neighborhoods), solving inequalities, random walk method and other approaches. The zero

point is equivalent to the zero change of the manipulated variable which means that this point is a good candidate to be inside the feasible area. If the zero point is located outside of the feasible area, other key points have to be tested or other methods must be applied in order to find this area (mentioned before).

It can also be seen, that some generated points may appear outside of the feasible area; therefore, the population size should be such that at least one point is located within this area. Otherwise, the whole population must be regenerated. There are also some points which are placed in the opposite way of the direction of the solution. This problem might be solved by adjusting the optimization algorithm.

3.3. Comparison

In the simulation, there were applied several settings of Hill climbing algorithm (hereinafter referred to as HillClimb). Each setting was then repeated 40 times in order to monitor the performance stability. In each iteration, the computational time was measured and the maximum value was stored. One of these settings is shown in Fig. 6 and it is compared to Quadratic programming algorithm (hereinafter referred to as QuadProg).

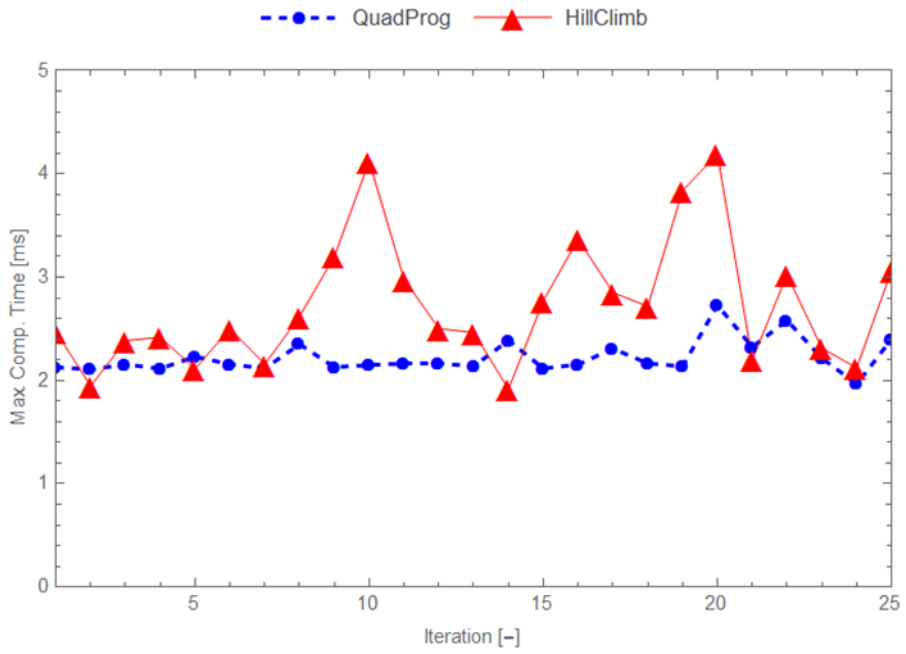


Fig. 6. Maximum computational time of algorithms

As can be seen, HillClimb performance is comparable to QuadProg performance and it may be sometimes better. However, QuadProg is better on average. There are also some values of HillClimb which are almost twice higher than the respective values of QuadProg. This may be caused by several factors such as the regeneration of the whole population, the required accuracy of the manipulated variable or differences in the distribution of the population. Despite these facts, the results are still acceptable, even though there was applied a basic version of HillClimb algorithm.

Apart from this setting, there were collected the computational times of other settings in order to compare their effects to the monitored properties of optimization. The average of these values was then stored and the results are presented in Table 1.

Quad		Variance of population [%]			
2.30		10	20	30	40
Pop. size	8	7.45	4.44	3.19	**2.83**
	12	9.73	5.54	3.97	3.21
	16	10.96	6.28	4.74	3.89
	20	12.48	7.46	6.13	5.65
	24	16.44	9.38	7.14	5.97

Table 1. Computational time of different settings of hill climbing algorithm

The table shows the computational time of different combinations between the population size and the variance of the population. However, this variance (*cov*) is related to the computational interval (Δu_{min}, Δu_{max}). The actual value of the variance of the normal distribution is then calculated by this equation.

$$\sigma = \frac{cov\left(\Delta u_{\max} - \Delta u_{\min}\right)}{600} \tag{13}$$

The value of QuadProg is located in the upper left corner of the table. As can be seen, the minimum value of the computational time was measured for the population size equal to 8 and the variance of the population equal to 40. However, there may be some more stable setting, which means that this computational time is only one of the factors which should be considered during selecting the proper optimization algorithm. For instance, if the size of population is too small, the whole population may appear outside of the feasible space and it must be regenerated. In order to monitor the stability of computational performance, the standard deviation of data sets was determined and the results are shown in Table 2a.

a)

Quad		Variance of population [%]			
0.27		10	20	30	40
Pop. size	8	0.81	0.89	**0.40**	0.71
	12	1.66	0.88	0.43	0.47
	16	1.05	0.61	0.59	0.66
	20	0.69	0.85	1.90	1.53
	24	1.90	1.45	1.18	0.96

b)

Quad		Variance of population [%]			
232.00		10	20	30	40
Pop. size	8	235.39	235.79	235.63	235.70
	12	232.51	233.21	234.82	235.21
	16	232.61	232.89	233.30	234.53
	20	232.69	232.83	233.44	233.79
	24	**232.44**	232.61	233.20	233.60

Table 2. a) Performance stability of different settings of hill climbing algorithm ;b) Accuracy of different settings of hill climbing algorithm

The most stable setting was measured for a lower variance of the population than in the previous case. It shows that the best setting might not be the setting of the best computational time.
Another examined factor of the optimization quality was the accuracy which was calculated by the sum of squared control errors. In order to avoid large numbers, the square roots of these values are presented in Table 2b.

In contrast to the table of the computational time, the best value is recorded for the highest population and the smallest variance of a generated population. Nevertheless, all numbers are similar; therefore, the accuracy of the control process is acceptable for all settings.

Even though QuadProg seems to be better than HillClimb in all examined parameters, the presented values in tables may not be guaranteed and they depend on several factors such as test conditions, available computational resources, the stability of computational performance and other aspects. The results revealed that the basic version of HillClimb algorithm is comparable to inbuilt function and further research is necessary in order to adjust this algorithm or create a new one. However, this is not an easy task as the control algorithms have been developed over decades.

4. Conclusion

The aim of this paper was the implementation and the comparison of alternative methods of optimization in the predictive control. The basic aspects of solving of the optimization problem in the predictive control were introduced. It was shown that in constrained case, the optimization process is computationally demanding and it must be solved by optimization algorithms. These algorithms (and their parameters) must be chosen so that the control requirements are fulfilled. Although there are many algorithms able to solve this problem, Hill climbing algorithm was applied because of its simplicity and the possibility of setting various parameters.

There were executed several simulations in order to monitor the quality of the optimization process. It was revealed that the optimization process is the most computationally demanding in the vicinity of substantial changes of the set-point. Consequently, it is important to reduce the computational time at these points.

The quality of the optimization process may be defined by several properties such as the computational time, the stability or the accuracy. These factors were monitored and it was discovered that the basic version of Hill climbing algorithm is comparable to inbuilt Quadratic programming. However, further research is necessary in order to find the proper optimization algorithms. These algorithms might be then a promising method of solving complex systems such as strongly nonlinear or even time variant systems.

5. Acknowledgements

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

6. 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, pp. 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] D. W. Clarke, C. Mohtadi, P. S. Tuffs, Generalized predictive control, part I: the basic algorithm. Automatica, 23, 1987, pp. 137-148.
[5] D. W. Clarke, C. Mohtadi, P. S. Tuffs, Generalized predictive control, part II: extensions and interpretations. Automatica, 23, 1987, pp. 149-160.
[6] G.M. Lee, N.N. Tam, and N.D. Yen, Quadratic Programming and Affine Variational Inequalities:A Qualitative Study, Springer, 2005.
[7] D.G. Luenberger and Y. Ye, Linear and nonlinear programming, 3rd ed. New York: Springer, 2008.
[8] Z. Dostál, Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities, New York: Springer, 2009.
[9] T. Back, D. B. Fogel, Z. Michalewicz, Handbook of evolutionary algorithms, Oxford: Oxford University Press, 1997