FAST OPTIMIZATION ALGORITHM FOR CONSTRAINED MODEL PREDICTIVE CONTROL

P. Škrabánek, D. Honc

University of Pardubice, Faculty of Electrical Engineering and Informatics
Department of Process Control

Abstract

Model Predictive Control (MPC) is a modern powerful control strategy which reached wide popularity in industry but also academic sphere. MPC allows adding constraints in input changes calculation in principle but a QP problem has to be solved in this case. The paper describes a fast QP algorithm. The fastness of the algorithm is compared with QP algorithm from MATLAB Optimization Toolbox.

1 Principle of Model Predictive Control

By Camacho [CAMACHO, 2002] Model Predictive Control doesn't designate a specific control strategy but a very ample range of control methods. The common ideas appearing in greater or lesser degree in all the predictive control family are basically:

- Explicit use of a model to predict the process output at future time instants.
- Receding strategy, so that at each instant the horizon is displaced towards the future, which involves the application of the first control signal of the sequence calculated at each step.
- Calculation of a control sequence minimizing an objective function

The minimizing of the objective function is the article's point of interest. The objective function differs at particular MPC algorithms but the general aim is that the future output $\mathbf{y}$ on the considered horizon should follow a determined reference signal $\mathbf{w}$ and, at the same time, the control effort $\Delta\mathbf{u}$ necessary for doing so should be penalized. The expression of such an objective function for Multi Input Multi Output case is

$$J(N_1, N_2, N_u) = (\hat{\mathbf{y}} - \mathbf{w})^{\mathrm{T}}.\overline{\mathbf{R}}.(\hat{\mathbf{y}} - \mathbf{w}) + \Delta\mathbf{u}^{\mathrm{T}}.\overline{\mathbf{Q}}.\Delta\mathbf{u} \tag{1}$$

where

N_1 (N_2)	prediction horizon minimum (maximum)
N_u	length of control horizon
$\hat{\mathbf{y}}$	vector of outputs prediction on prediction horizon
$\mathbf{w}$	vector of desired outputs values on prediction horizon
$\Delta\mathbf{u}$	vector of control efforts on control horizon
$\overline{\mathbf{R}}, \overline{\mathbf{Q}}$	positive semi-definite weighting matrices

This objective function can be transformed to

$$\text{minimize } J(\Delta\mathbf{u}) = \tfrac{1}{2}.\Delta\mathbf{u}^{\mathrm{T}}.\mathbf{G}.\Delta\mathbf{u} + \Delta\mathbf{u}^{\mathrm{T}}.\mathbf{g} \tag{2}$$

and in case that constraints are considered

$$\begin{aligned} &\text{minimize } J(\Delta\mathbf{u}) = \tfrac{1}{2}.\Delta\mathbf{u}^{\mathrm{T}}.\mathbf{G}.\Delta\mathbf{u} + \Delta\mathbf{u}^{\mathrm{T}}.\mathbf{g} \\ &\text{subject to} \quad \mathbf{a}_i^{\mathrm{T}}.\Delta\mathbf{u} \geq b_i \end{aligned} \tag{3}$$

The equation (3) is a standard Quadratic Programming (QP) problem with constraints defined by set of inequalities.

2 Quadratic Programming

Exist many ways how to solve QP problem with constraints. A QP algorithm and its theory are described in the chapter 2.1. This algorithm needs an initial feasible point on begin, hence an algorithm for its calculation is described in the chapter 2.2. Both chapters draw from [FLETCHER, 2006].

2.1 The algorithm

The algorithm is based on primal active set method where certain constraints, indexed by active set $\mathcal{A}$, are regarded as equalities whilst the rest are temporarily disregarded. The method adjusts this set in order to identify the correct active constraints at the solution to (3). On iteration k a feasible point $\Delta\mathbf{u}^{(k)}$ is known which satisfies the active constraints as equalities, that is

$$\mathbf{a}_i^{\mathrm{T}}.\Delta\mathbf{u}^{(k)} = b_i,\ i \in \mathcal{A} \tag{4}$$

so each iteration attempts to locate the solution to an equality problem (EP) in which only the active constraints occur. This is most conveniently done by shifting origin to $\Delta\mathbf{u}^{(k)}$ and looking for a correction $\boldsymbol{\delta}^{(k)}$. The active constraints can be written in matrix form

$$\mathbf{A}^{\mathrm{T}}.\Delta\mathbf{u}^{(k)} = \mathbf{b} \tag{5}$$

where $\mathbf{A}$ is n x m matrix and collects the column vectors $\mathbf{a}_i$. $\mathbf{b}$ is a column vector of the length m composed by b_i. The original QP problem is then transform to the form

$$\begin{aligned} &\text{minimize } J(\boldsymbol{\delta}) = \tfrac{1}{2}.\boldsymbol{\delta}^{\mathrm{T}}.\mathbf{G}.\boldsymbol{\delta} + \boldsymbol{\delta}^{\mathrm{T}}.\mathbf{g}^{(k)} \\ &\text{subject to } \ \mathbf{A}^{\mathrm{T}}.\boldsymbol{\delta} = \mathbf{0} \end{aligned} \tag{6}$$

where

$$\mathbf{g}^{(k)} = \mathbf{g} + \mathbf{G}.\Delta\mathbf{u}^{(k)} \tag{7}$$

The objective function (6) is solved by the method of Lagrange multipliers. The Lagrangian function is

$$\mathcal{L}(\boldsymbol{\delta}, \boldsymbol{\lambda}) = \tfrac{1}{2}\boldsymbol{\delta}^{\mathrm{T}}.\mathbf{G}.\boldsymbol{\delta} + \boldsymbol{\delta}^{\mathrm{T}}.\mathbf{g}^{(k)} - \boldsymbol{\lambda}^{\mathrm{T}}.\mathbf{A}^{\mathrm{T}}.\boldsymbol{\delta} \tag{8}$$

The solution of the function is given by equations

$$\boldsymbol{\delta}^{(k)} = -\mathbf{H}.\mathbf{g}^{(k)} \tag{9}$$

$$\boldsymbol{\lambda}^{(k)} = \mathbf{T}^{\mathrm{T}}.\mathbf{g}^{(k)} \tag{10}$$

where explicit expressions for $\mathbf{H}$ and $\mathbf{T}$ are

$$\begin{aligned} \mathbf{H} &= \mathbf{G}^{-1} - \mathbf{G}^{-1}.\mathbf{A}.\left(\mathbf{A}^{\mathrm{T}}.\mathbf{G}^{-1}.\mathbf{A}\right)^{-1}.\mathbf{A}^{\mathbf{T}}.\mathbf{G}^{-1} \\ \mathbf{T} &= \mathbf{G}^{-1}.\mathbf{A}.\left(\mathbf{A}^{\mathrm{T}}.\mathbf{G}^{-1}.\mathbf{A}\right)^{-1} \end{aligned} \tag{12}$$

If $\boldsymbol{\delta}^{(k)}$ from (9) is feasible with regard to the constraints not in $\mathcal{A}$, then next iterate is taken as

$$\Delta\mathbf{u}^{(k+1)} = \Delta\mathbf{u}^{(k)} + \boldsymbol{\delta}^{(k)} \tag{13}$$

If not then a line search is made in the direction of $\boldsymbol{\delta}^{(k)}$ to find the best feasible point. Then the step $\alpha^{(k)}$ is

$$\alpha^{(k)} = \min\left(1, \left[\frac{-\mathbf{a}_i^{\mathrm{T}}.\Delta\mathbf{u}^{(k)}}{\mathbf{a}_i^{\mathrm{T}}.\boldsymbol{\delta}^{(k)}}\right]\right) \quad \text{for } i \notin \mathcal{A} \text{ and } \mathbf{a}_i^{\mathrm{T}}.\boldsymbol{\delta}^{(k)} < 0 \tag{14}$$

If $\alpha^{(k)} < 1$ then a new constraint becomes active and is moved to the active set.

If $\boldsymbol{\delta} = \mathbf{0}$, then it is possible to compute multipliers $\boldsymbol{\lambda}^{(k)}$ by equation (10). If all $\lambda_i \geq 0$ then $\Delta\mathbf{u}^{(k)}$ is the global solution $\Delta\mathbf{u}^{*}$ otherwise there exists an index q, $q \in \mathcal{A}^{(k)}$, such that $\lambda_q^{(k)} < 0$. In this case it is possible to reduce (3) by allowing the constraint q to become inactive. Thus q is removed from $\mathcal{A}$ and the algorithm continues as before by solving the resulting EP (6). If there is more than one index for which $\lambda_i^{(k)} < 0$ then q is selected by

$$\lambda_q = \min \lambda_i^{(k)},\ \ i \in \mathcal{A} \tag{15}$$

The primal active set method algorithm can be summarized as follows:

1. Given $\Delta\mathbf{u}^{(1)}$ and $\mathcal{A}^{(1)}$, set k=1.
2. Solve (9). If $\boldsymbol{\delta}^{(k)} \neq \mathbf{0}$ go to 4.
3. Compute Lagrange multipliers $\boldsymbol{\lambda}^{(k)}$ by (10) and solve (15). If $\lambda_q^{(k)} \geq 0$ terminate with $\Delta\mathbf{u}^{*} = \Delta\mathbf{u}^{(k)}$, otherwise remove constraint q from $\mathcal{A}$.
4. Get $\boldsymbol{\delta}^{(k)}$ from (9).
5. Find $\alpha^{(k)}$ from (14).
6. Set $\Delta\mathbf{u}^{(k+1)} = \Delta\mathbf{u}^{(k)} + \alpha^{(k)}.\boldsymbol{\delta}^{(k)}$.
7. If $\alpha^{(k)} < 1$, add p to $\mathcal{A}$.

8. Set $k = k + 1$ and go to 2.

2.2 Initial feasible point

The algorithm described in 2.1 requires an initial feasible point $\Delta\mathbf{u}^{(1)}$ and an appropriate active set $\mathcal{A}^{(1)}$. These can be obtained by solving an auxiliary problem

$$\begin{aligned} &\text{minimize } \textstyle\sum_{i\in V^{(k)}}\left(\mathrm{b}_i - \mathbf{a}_i^{\mathrm{T}}.\Delta\mathbf{u}\right) \\ &\text{subject to} \qquad \mathbf{a}_i^{\mathrm{T}}.\Delta\mathbf{u} \geq b_i \end{aligned} \tag{16}$$

where $V^{(k)}$ is the set of violated (infeasible) constraints at $\Delta\mathbf{u}^{(k)}$. This objective function is solved in similar way as linear programming problem.

The iteration commences by creating the initial vertex $\Delta\mathbf{u}^{(1)} = \mathbf{0}$. Also pseudoconstraints $\Delta u_i = 0,\ i = 1,..,n$ are added to the problem, if they are not already present, and form the initial active set $\mathcal{A}_{aux}$. Once a pseudoconstraint becomes inactive it is removed from the problem.

At $\Delta\mathbf{u}^{(k)}$ the gradient of the cost function is $-\sum_{i\in V^{(k)}}\mathbf{a}_i$ and the Lagrange multiplier vector

$$\boldsymbol{\lambda} = -\mathbf{A}^{-1}.\left(\textstyle\sum_{i\in V^{(k)}}\mathbf{a}_i\right) \tag{17}$$

is evaluated, where $\mathbf{A}$ now denotes the n x n matrix with columns $\mathbf{a}_i,\ i \in \mathcal{A}_{aux}$. Let the columns of $\mathbf{A}^{-\mathrm{T}}$ be written $\mathbf{s}_i$ then vectors $\mathbf{s}_i$ are the directions of all feasible edges at $\Delta\mathbf{u}^{(k)}$, and the multipliers λ_i are the slopes along these edges. Thus if

$$\lambda_i \geq 0,\ i \in \mathcal{A}_{aux} \tag{18}$$

then no feasible descent direction exists, so $\Delta\mathbf{u}^{(k)}$ is optimal and the iteration terminates. Otherwise the most negative λ_i (λ_q say) is chosen, and a search is made along the downhill edge

$$\Delta\mathbf{u}^{(k+1)} = \Delta\mathbf{u}^{(k)} + \alpha^{(k)}.\mathbf{s}_q^{(k)} \tag{19}$$

The search is terminated by the first inactive constraint (p say) to become active. Candidates for p are therefore indices $i \notin \mathcal{A}_{aux}$ with $\mathbf{a}_i^{\mathrm{T}}.\mathbf{s}_q \neq 0$ and such constraint function becomes zero when

$$\alpha_i = \frac{b_i - \mathbf{a}_i^{\mathrm{T}}.\Delta\mathbf{u}^{(k)}}{\mathbf{a}_i^{\mathrm{T}}.\mathbf{s}_q} \tag{20}$$

Thus the index p and the corresponding value of α are defined by

$$\alpha = \min_{\substack{i:i\notin\mathcal{A}_{aux} \\ \mathbf{a}_i^{\mathrm{T}}.\mathbf{s}_q\neq 0}} \frac{b_i - \mathbf{a}_i^{\mathrm{T}}.\Delta\mathbf{u}^{(k)}}{\mathbf{a}_i^{\mathrm{T}}.\mathbf{s}_q} \tag{21}$$

The new vertex is defined by (19) and the new active set $\mathcal{A}_{aux}$ is obtained by replacing q by p. The iteration is then repeated from this new vertex until $\Delta\mathbf{u}^{(k)}$ is found to be a feasible point. All added pseudoconstraints are removed from final active set $\mathcal{A}_{aux}$ on the end.

3 Practical realization

3.1 Algorithm description

The realization of the theoretical part consists from two separated algorithms. The first one is used as initial feasible point generator based on theory from chapter 2.2. It has been created as a function with two inputs and five outputs and it is called "*initial_condition*". Its inputs are the matrix **A** and vector **b**. Outputs are an feasible point $\mathbf{x}_0$, active constraints described by a matrix $\mathbf{A}_c$ and a column vector $\mathbf{a}_c$ and inactive constraints described by a matrix $\mathbf{N}_c$ and a column vector $\mathbf{n}_c$.

Second algorithm is the realization of described QP algorithm from chapter 2.1. It is also realized as a function and is called "*qpas*". The function has seven inputs parameters and one output parameter. Its inputs are an initial feasible point $\mathbf{x}_0$, matrix **G** and vector **g** describing the objective function, and active and inactive constrains described by $\mathbf{A}_c$, $\mathbf{a}_c$, $\mathbf{N}_c$ and $\mathbf{n}_c$. The output is then the solution of the QP problem **x**. The created functions are attached on the CD.

The first algorithm has not to be used implicitly of course but the second algorithm requires an initial feasible point as function input[1].

3.2 Algorithm testing

Although the algorithm was created for a MPC controller, its implementation to the controller isn't aim of this paper. Only its testing and speed comparison is described here.

The testing problem is described by equation (3) where

$$\mathbf{G} = \begin{bmatrix} 5 & 1 & 0 & 3 \\ 1 & 6 & 0 & 0.3 \\ 0 & 0 & 5 & 0.1 \\ 3 & 0.3 & 0.1 & 2 \end{bmatrix} \qquad \mathbf{g} = \begin{bmatrix} 10 \\ 7.8 \\ -4 \\ 7 \end{bmatrix} \tag{22}$$

The constraints are given by

$$x_1 \in \langle 3,5 \rangle \qquad x_2 \in \langle -3,3 \rangle \qquad x_3 \in \langle -0.5,-0.1 \rangle \qquad x_4 \leq 3 \tag{23}$$

which can be written by

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 3 \\ -5 \\ -3 \\ -3 \\ -0.5 \\ 0.1 \\ -3 \end{bmatrix} \tag{24}$$

The comparison of the algorithm with the standard QP algorithm from MATLAB Optimization Toolbox has been done for two variants. The time needed to reaching of the solution was observed in both cases. In the first was the initial feasible point calculated separately and then was relayed to both compared algorithms as input. Only the time necessary to QP problem solving was measured. The results are given in table 1.

Table 1: TIME NEEDED TO THE QP PROBLEM SOLVING IN CASE INITIAL FEASIBLE POINT IS CALCULATED SEPARATELY

	Described QP	**QP from MATLAB opt. toolbox**
1	0.000822	0.003126
2	0.000602	0.005045
3	0.000720	0.005052
4	0.000608	0.003513
Average	0.000688	0.004184

In the second case was measured time needed for the whole calculation (QP algorithm execution and initial feasible point calculation). In this case wasn't the initial feasible point handed over to the QP algorithm from MATLAB optimization toolbox. The results follow in table 2.

[1] Some MPC algorithms don't find an initial feasible point in every QP algorithm calling because result from the last QP algorithm is used as input for the following calling and thus is feasible. It doesn't have to hold every time because constraints can be change in every sample time.

Table 2: TIME NEEDED TO THE QP PROBLEM SOLVING IN CASE CALCULATION OF INITIAL FEASIBLE POINT IS INCLUDED TO THE MEASURED TIME

	Described QP	**QP from MATLAB opt. toolbox**
1	0.001009	0.005248
2	0.001010	0.004657
3	0.001162	0.004126
4	0.001004	0.004637
Average	0.001046	0.004667

The realized QP algorithm testing was conducted during the algorithm comparing, because both algorithms give the same results. The code used by the QP algorithms comparing is attached also on CD.

4 Conclusion

The algorithm described in this paper doesn't offer so many variants of using compare to QP algorithm from MATLAB optimization toolbox, but it can faster solve a QP problem, which can be in practice important. The described algorithm approximately six times faster was in the case the initial feasible point calculation time wasn't included to the measuring. In the second case was more than four times faster.

References

CAMACHO, E. F.; BORDONS, C. 2002. *Model Predictive Control 3rd*. Great Britain : Springer – Verlag London. 208 p. ISBN 3-540-76241-8.

FLETCHER, R. 2006. *Practical Methods of Optimization 2nd*. Great Britain : Springer – Verlag. 436 p. ISBN 0-471-49463-1.

Ing. Pavel Škrabánek
Department of Process Control
Faculty of Electrical Engineering and Informatics University of Pardubice
Email: pavel.skrabanek@upce.cz
Phone: (+420) 466 037 124

Ing. Daniel Honc, Ph.D.
Department of Process Control
Faculty of Electrical Engineering and Informatics University of Pardubice
Email: daniel.honc@upce.cz
Phone: (+420) 466 037 107