Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 100 (2015) 1649 – 1656

Procedia Engineering

www.elsevier.com/locate/procedia

25th DAAAM International Symposium on Intelligent Manufacturing and Automation, DAAAM 2014

Implementation of Mixed-Integer Programming on Embedded System

Jakub Novak*, Petr Chalupa

Faculty of Applied Informatics, Tomas Bata University, nam. TGM 5555, Zlín 76001, Czech Republic

Abstract

One of the most widespread modern control strategies is the discrete-time Model Predictive Control (MPC) method which requires the solution of the quadratic programming problem. For systems with binary input variables the quadratic problem is replaced by more challenging Mixed-Integer Quadratic Programming (MIQP) problem. The objective of this work is the implementation of MIQP problem solver in a low power embedded computing platform with limited computational power and limited memory. The MIQP problem is solved using branch-and-bound method and the solution of the relaxed original quadratic problems with equality and inequality constraints solved in the nodes of a binary tree is found with interior-point algorithm. A simulation study of the reserve constrained economic dispatch problem for power generators with prohibited zones is presented. Simulation results show the applicability of the proposed solver for small size MIQP problems.

Peer-review under responsibility of DAAAM International Vienna

Keywords: Embedded System; Mixed-Integer Quadratic Programming; Branch-and-Bound; Economic Dispatch; Optimization

1. Introduction

Model predictive control (MPC) has gained a lot of interest of both academia and industry in the recent years. The main reason for the wide-scale adoption of MPC is its ability to handle constraints on inputs and states that arise in most applications. Besides, MPC problem formulation enables direct inclusion of predictive information, allowing the controller to react to future changes in reference signal. MPC naturally handles processes with multiple

* Corresponding author. Tel.: +42-057-603-5316; fax: +42-057-603-5279.
E-mail address: jnovak@fai.utb.cz

Peer-review under responsibility of DAAAM International Vienna
doi:10.1016/j.proeng.2015.01.539

inputs or outputs and its concept can be used with dynamic models of any dimension. To avoid online optimization the solution of the control problem for different states can be pre-computed off-line. This explicit solution represent a piece-wise affine map over a partition of the state-space and can be stored efficiently in the form of a look-up table [1]. The explicit MPC offers reduction in online evaluation time but the primary limitation is that the complexity can grow quickly with the problem size, thus limiting the applicability of explicit MPC to small and medium-sized control problems. Recently, the interest of using MPC for controlling systems with both continuous dynamics and additional integer variables has arisen. The resulting optimization problem is no longer quadratic programming (QP) problem but a Mixed Integer Quadratic Programming (MIQP) problem. The inclusion of integer variables turns the easily solved QP problem, into an *NP*-hard problem [2]. A special case of MIQP is when the integer variables are constrained to be 0 or 1. Several methods exist for solution of the MIQP problem, however branch-and-bound method is superior to other methods such as decomposition method or logic-based method [3]. A review of different methods of solving MIQP problems can be found in [4]. Commercial optimization software (CPLEX, GUROBI) is able to solve the MIQP problems and there are also several toolboxes for MATLAB exist (OPTI Toolbox which uses SCIP solver [5], Hybrid toolbox [6]).

With respect to the efficient online solution of quadratic problem in predictive control problems many tailored approaches exist by now. An online MPC strategy with a good balance between computational speed and memory demand based that uses a fast gradient method was developed in [7]. Many real-life problems can be represented as MLD (Mixed Logic Dynamical) systems which are hybrid systems [3] and whose MPC control requires solution of the MIQP problem. A fast implementation of Interior point method is can be found in [7] and its applicability is demonstrated in simulation studies. The particular structure of the MPC problem is beneficially used and considerably reduces the computation time of control action. Different methods for hybrid optimal control problem solution were evaluated in [8]. Currie, Prince-Pike and Wilson developed a MATLAB framework for generating fast model predictive controllers for embedded targets such as ARM processors and tested it on inverted pendulum in [8]. Implementation aspects of the MPC on Embedded System are also discussed in [10] and [11]. The increase in computational power such as ARM Cortex processors and advances in optimization algorithms has opened a new trend which brings MPC capabilities also to complex and fast systems. With the development of cheap multi-core CPU in microcontrollers, the parallel computation might be the promising way for further decrease of computation time.

In this work we focus on the implementation aspects of MIQP solver on a low cost embedded system where the problem is solved using branch-and-bound method and the relaxed quadratic programming problem is solved with interior point method. The paper is structured as follows: Section 2 briefly repeats the MIQP formulation and the solution algorithm. A description of the example system is given in Section 3. Section 4 contains the results of the implementation of the solver on embedded system. Finally, the main conclusions are summarized in the last section.

2. Mixed-integer quadratic problem

Mixed Integer Quadratic Program (MIQP) is a non-convex optimization problem where the objective function is quadratic and constraints are linear. The non-convexity stems from the fact that the optimized variables x_i belong to the binary set. In particular,

$$
\begin{aligned}
&\min 0.5\left[\boldsymbol{x}_c^T, \boldsymbol{x}_i^T\right]\boldsymbol{H}\begin{bmatrix}\boldsymbol{x}_c^T\\ \boldsymbol{x}_i^T\end{bmatrix} + f^T\begin{bmatrix}\boldsymbol{x}_c^T\\ \boldsymbol{x}_i^T\end{bmatrix}\\
&x_c \in R^{n_c}, x_i \in N^{n_i} : \boldsymbol{a}_j^T\begin{bmatrix}\boldsymbol{x}_c^T\\ \boldsymbol{x}_i^T\end{bmatrix} + b_j = 0, \quad j = 1,\dots,m_{ec}\\
&\qquad\qquad\qquad\quad \boldsymbol{c}_j^T\begin{bmatrix}\boldsymbol{x}_c^T\\ \boldsymbol{x}_i^T\end{bmatrix} + d_j \geq 0, \quad j = 1,\dots,m_{ic}
\end{aligned}
\tag{1}
$$

where $\boldsymbol{H}$ is a positive definite *n x n* matrix ($n = n_i + n_c$), $\boldsymbol{f}$ is the n-dimensional vector. The n-dimensional vectors $\boldsymbol{a}_j$ and $\boldsymbol{c}_j$ and vectors $\boldsymbol{b}$ and $\boldsymbol{d}$ are used to set up the constraints. The numbers of equality and inequality constraints are specified with m_{ec} and m_{ic}, respectively. The equality and inequality constraints define a feasible region in which the solution to the problem must be located in order for the constraints to be satisfied. The only difference comparing to convex QP is a presence of binary variables x_i and this is also a reason why the set is non-convex. Fortunately, if the binary variable is fixed or relaxed, a convex set is obtained and the problem can be solved using methods for convex optimization. A constrained QP can be solved either using an interior point method or an active set method.

Branch-and-bound has been the most used tool for solving large scale *NP*-hard combinatorial optimization problems [12]. During the solution process, the status of the solution is described by a pool of yet unexplored subset of the solution space and the best solution found so far. The nodes in a dynamically generated search tree, which initially only contains the root, and each iteration of a classical branch-and-bound algorithm processes one such node represent unexplored subspaces. The iteration has two main components: selection of the node to process and branching strategy. The nodes created are then stored together with the bound of the processed node. The search stops when the pool of unexplored subset is empty and the optimal solution is then the one recorded as "current best".

Branching on a variable involves choosing the branching variable of the current optimal solution of the relaxed problem and then adding a constraint to it. We apply the maximum fractional branching strategy that chooses the variable with the highest fractional part. There are two common node selection strategies for selection of the node to proceed in the next iteration. The first one is *best-first-search*, where the next node is always the one with the lowest dual bound. This method however requires a large amount of storage. The second class of node selection strategies *depth-first-search* where warm-starting can be successfully applied due to the similarity of the subproblems and also number of unexplored nodes is low, which significantly reduces the storage requirements. The *depth-first-search* strategy is used in the example.

A constrained QP can be solved either using an interior point method or an active set method. Interior-point methods solve problems iteratively with each iteration being computationally expensive but can make significant progress towards the solution [13]. The solver uses the interior point method for solution of the relaxed problems:

$$\begin{aligned} &\min\ 0.5\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x} \\ &\quad x \in R^n : \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} = 0 \\ &\qquad\qquad \boldsymbol{C}\boldsymbol{x} + \boldsymbol{d} \ge 0 \end{aligned} \tag{2}$$

where **A** is an m_{ec} x n matrix describing the equality constraints and **C** is an m_{ic} x n matrix describing the inequality constraints. $\boldsymbol{b}$ and $\boldsymbol{d}$ are m_{ec} x n and m_{ic} x n vectors respectively. The Lagrangian *L(x,y,z)* with vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ containing the Lagrange multipliers is defined as:

$$L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x} - \boldsymbol{y}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) - \boldsymbol{z}^T(\boldsymbol{C}^T - \boldsymbol{d}) \tag{3}$$

The following optimality conditions can be obtained with the introduction of the slack vector $\boldsymbol{s} = \boldsymbol{C}\boldsymbol{x} - \boldsymbol{d} \ge 0$

$$\begin{aligned} &\boldsymbol{H}x + \boldsymbol{f} - \boldsymbol{A}^T - \boldsymbol{C}\boldsymbol{z} = 0 \\ &\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = 0 \\ &\boldsymbol{s} - \boldsymbol{C}\boldsymbol{x} + \boldsymbol{d} = 0 \\ &(\boldsymbol{z},\boldsymbol{s}) \ge 0 \\ &\mathrm{s}_i\, z_i = 0 \end{aligned} \tag{4}$$

Defining the function ***F(x,y,z,s)*** such that the roots of this function are solutions to the first four optimality conditions we obtain set of linear equation:

$$\begin{bmatrix} \boldsymbol{G} & -\boldsymbol{A}^T & -\boldsymbol{C}^T & 0 \\ -\boldsymbol{A} & 0 & 0 & 0 \\ -\boldsymbol{C} & 0 & 0 & \boldsymbol{I} \\ 0 & 0 & \boldsymbol{S} & \boldsymbol{Z} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \boldsymbol{H}x + \boldsymbol{f} - \boldsymbol{A}^T \boldsymbol{y} - \boldsymbol{C}^T \boldsymbol{z} \\ \boldsymbol{Ax} - \boldsymbol{b} \\ \boldsymbol{s} - \boldsymbol{Cx} + \boldsymbol{d} \\ (\mathrm{s}^* \mathrm{z}) \end{bmatrix} \tag{5}$$

where * is the element-wise multiplication of vectors. For solution of this set of equation predictor-corrector method proposed by Mehrotra is used. The full algorithm is given in Appendix A. As a stopping criterion the following criterions are used:

$$\begin{aligned} &\left\| \boldsymbol{H}x + \boldsymbol{f} - \boldsymbol{A}^T \boldsymbol{y} - \boldsymbol{C}^T \boldsymbol{z} \right\| \le \varepsilon \\ &\left\| \boldsymbol{Ax} - \boldsymbol{b} \right\| \le \varepsilon \\ &\left\| \boldsymbol{s} - \boldsymbol{Cx} + \boldsymbol{d} \right\| \le \varepsilon \\ &\left\| \mu \right\| \le \varepsilon \end{aligned} \tag{6}$$

and also maximum number of iterations k_{max} is specified. For the solution of the set of linear equations ***Ax=b*** from (5) the $\mathrm{LDL}^{\mathrm{T}}$ factorization is used.

$$\boldsymbol{PAP}^T = \boldsymbol{LDL}^T \tag{7}$$

where ***P*** is a permutation matrix, ***L*** is a unit lower triangular matrix and ***D*** is a block diagonal matrix with 1x1 and 2x2 blocks. Once a factorization has been computed, the solution to the linear system $\boldsymbol{Ax} = \boldsymbol{b}$ can be computed at comparably low cost by solving a sequence of equations:

$$\begin{aligned} &\boldsymbol{Lu} = \boldsymbol{Pb} \\ &\boldsymbol{Dv} = \boldsymbol{u} \\ &\boldsymbol{L}^T \boldsymbol{w} = \boldsymbol{v} \\ &\boldsymbol{x} = \boldsymbol{P}^T \boldsymbol{w} \end{aligned} \tag{8}$$

with intermediate vectors ***u,v,w***. The cost of the solve procedure (8) is most of the time negligible with respect to the cost of computing the factorization (7). The LDL factorization implemented in the LAPACK library [14] exploits the partial pivoting based on the Bunch-Kaufmann method [15].

3. Example problem

The memory requirements and computational performance of the solver was evaluated using a numerical example. The economic dispatch of generators is a key element in the optimal operation of power generation systems. The main goal is the generation of a given amount of electricity at the lowest cost possible. Although the basic objective is straightforward, the problem is typically extended with a number of constraints. Method for solution of the economic dispatch of power systems using Lagrange multiplier method is described in [16]. One specific case is the consideration of generators which have prohibited zones of operation within their overall domain of operation. The example is taken from [17]. The total fuel cost F is the objective function to be minimized:

$$\min F = \sum_{i=1}^{N} F_i(\mathrm{P}_i) \tag{9}$$

where the fuel cost function of each generating unit is described by a quadratic function of the power output P_i :

$$F_i(P_i) = a_i + b_i P_i + c_i P_i^2 \tag{10}$$

There are several constraint on the problem. The sum of power outputs must equal the total network demand P_D:

$$\sum_{i=1}^{N} P_i = P_D \tag{11}$$

The power output of each source is limited

$$P_i^{\min} \le P_i \le P_i^{\max} \tag{12}$$

To capture the disjoint operating regions the following variables are introduced:
Y_{ik} 1 if unit *i* operates in power output range *k*; 0 otherwise
Θ_{ik} power output of unit *i* if operating in range *k* (i.e. if Y_{ik}=1 then $\Theta_{ik}=P_i$); 0 otherwise.
Each unit i with prohibited operating zones can operate within only one of the allowed set of ranges.

$$P_i^{\min} \le P_i \le P_i^{\max} \tag{13}$$

If unit *i* operates in range *k*, then the corresponding Θ_{ik} variable should be equal to P_i otherwise it is forced to the value of zero. This can be achieved by the following two constraints:

$$P_i = \sum_{k=1}^{K} \Theta_{ik} \tag{14}$$

and

$$P_{ik}^L Y_{ik} \le \Theta_{ik} \le P_{ik}^U Y_{ik}, \forall i \in \omega, k = 1,...,K \tag{15}$$

The example involves four on-line units with two units with 2 prohibited zones and three allowed zones (*k=3*) and two units that use the entire power output range. The parameters are provided in Tables 1 and 2:

Table 1. Parameters of units.

Parameter	Value
a	500
b	1E5
c	1E-6
P^{min}	0.100 GW
P^{max}	0.450 GW
P_D	1.375 GW

Table 2. Prohibited zones.

Unit	Zone 1 (MW)	Zone 2(MW)
1	[200-250]	[300-350]
2	[210-260]	[310-360]

The mathematical formulation of the economic dispatch problem (9) with constraints (11,12,13,14,15) corresponds to an MIQP model with 10 continuous variables (P_i, Θ_{ik}) and 6 binary variables (Y_{ik}), 5 equality constraints and 28 inequality constraints. Twelve of the 28 inequality constraints are necessary to restrict the 6 binary values from 0 to 1.

4. Implementation on selected hardware platform

The proposed MIQP problem solver was implemented on The Stellaris® LM4F120 board which is a low-cost evaluation platform for 32-bit ARM® Cortex™-M4F-based microcontrollers from Texas Instruments. The microcontroller runs at 80 MHz. The board has 32KB of SRAM memory, 256KB of flash memory and 2KB EEPROM. For implementation of the solver the requirements for memory and evaluation speed must be considered. The board has only 32KB of RAM however system parameters and constraints can be stored in flash memory as they are fixed and only read during the solution of the problem.

We have developed a simple implementation of branch-and-bound algorithm with interior point method for computation of the relaxed problem, written in C, using the LAPACK and BLAS libraries to carry out the numerical linear algebra computations such as matrix-vector multiplication, LDL^T decomposition and solution of system of algebraic equations. The solver is implemented using double precision floating-point arithmetic.

The matrices $H \in R^{n*n}, C \in R^{n_{ic}*n}, A \in R^{n_{ec}*n}$ and vectors *f, b, d* for definition of constraints and cost function of the MIQP problem are stored in flash memory. The branch-and-bound method requires a pool for storing the nodes during the solution process. The memory requirements in bytes are given by two matrices for storage the additional constraints of the size $n_{pool} * n_i$, a double vector of the size n_{pool} to hold the bounds for each node in a pool and integer vector to store the priority of the nodes in the pool.

Interior point method requires allocation of vectors $x, \Delta x, y, \Delta y, z, \Delta z, s, \Delta s$, the matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ of the system $\boldsymbol{Ax=b}$ in the memory. The matrix $\boldsymbol{A}$ is symmetric so only lower triangular part of the matrix $\boldsymbol{A}$ is stored. The number of elements of lower triangular matrix is given as:

$$\frac{1}{2}(\mathrm{n}+\mathrm{n}_{\mathrm{ic}}+\mathrm{n}_{ec})(\mathrm{n}+\mathrm{n}_{\mathrm{i}c}+\mathrm{n}_{ec}+1) \tag{16}$$

The optimal solution has objective function value 16,223.2125 with 0 optimality gap (i.e. the solution obtained is the global optimum). The Table 3 shows the time of evaluation of optimal solution for different value of stopping criterion. No warm-starting was used and the initial solution was set to a vector of zeros.

Table 3. Prohibited zones.

Parameter	$\varepsilon = 1E-3$	$\varepsilon = 1E-5$	$\varepsilon = 1E-7$
P_1	0.332478 GW	0.332499 GW	0.332500 GW
P_2	0.332478 GW	0.332499 GW	0.332500 GW
P_3	0.350033 GW	0.350001 GW	0.350000 GW
P_4	0.360011 GW	0.360001 GW	0.360000 GW
time	5.03s	6.69s	9.43s

Conclusion

The results show that solution of the MIQP problem 16 variables and 33 constraints is manageable on the low power platform. The simulation study showed that the memory as well as the computational demand of an MIQP solver implementation is decisive for real-time use on low-cost embedded systems. The proposed mixed-integer solver led to satisfactory results which can be further improved in many ways, for example by adopting some kind of cutting planes strategy. The ***A*** matrix in the set of linear equations ***Ax=b*** is very sparse and 80 percent of the computation time is spent in the computation of LDL^T factorization. Implementation of the algorithms for sparse LDL^T decomposition and solution of the equation set would improve the performance of the solver. Further extensions of the proposed solver will be for mixed-integer model predictive control scheme with binary signals.

Acknowledgements

The work was performed with financial support of research project NPU I No. MSMT-7778/2014 by the Ministry of Education of the Czech Republic and also by the European Regional Development Fund under the Project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

Appendix A. Interior-point predictor-corrector algorithm

1. Initialize $\mathbf{x_0, y_0, z_0, s_0}$
2. Compute the complementary measure μ

$$\mu = \frac{s^T z}{m_{ic} + m_{ec}} \tag{17}$$

3. Start while loop (terminate if stopping criteria are satisfied)
4. Predictor step: Compute affine scaling direction $\Delta x^{aff}, \Delta y^{aff}, \Delta z^{aff}, \Delta s^{aff}$ by solution

$$\begin{bmatrix} \boldsymbol{G} & -\boldsymbol{A}^T & -\boldsymbol{C}^T & 0 \\ -\boldsymbol{A} & 0 & 0 & 0 \\ -\boldsymbol{C} & 0 & 0 & \boldsymbol{I} \\ 0 & 0 & \boldsymbol{S} & \boldsymbol{Z} \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta y^{aff} \\ \Delta z^{aff} \\ \Delta s^{aff} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{Hx} + \boldsymbol{f} - \boldsymbol{A}^T \boldsymbol{y} - \boldsymbol{C}^T \boldsymbol{z} \\ \boldsymbol{Ax} - \boldsymbol{b} \\ \boldsymbol{s} - \boldsymbol{Cx} + \boldsymbol{d} \\ \boldsymbol{s} * \boldsymbol{z} \end{bmatrix} \tag{18}$$

5. Compute the maximum allowable α^{aff} satisfying the following conditions

$$\begin{aligned} \boldsymbol{z} + \alpha^{aff} \Delta z^{aff} &\geq 0 \\ \boldsymbol{s} + \alpha^{aff} \Delta s^{aff} &\geq 0 \end{aligned} \tag{19}$$

6. Compute affine complementary measure

$$\mu^{aff} = \frac{(\mathrm{s} + \alpha^{aff} \Delta \mathrm{s}^{\mathrm{aff}})(\mathrm{z} + \alpha^{aff} \Delta z^{\mathrm{aff}})}{m_{ic} + m_{ec}} \tag{20}$$

7. Compute the centering parameter σ

$$\sigma = \left(\frac{\mu^{aff}}{\mu} \right)^3 \tag{21}$$

8. Corrector step: Compute the search direction ($\Delta x, \Delta y, \Delta z, \Delta s$) by solution

$$\begin{bmatrix} \boldsymbol{G} & -\boldsymbol{A}^T & -\boldsymbol{C}^T & 0 \\ -\boldsymbol{A} & 0 & 0 & 0 \\ -\boldsymbol{C} & 0 & 0 & \boldsymbol{I} \\ 0 & 0 & \boldsymbol{S} & \boldsymbol{Z} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \boldsymbol{Hx} + \boldsymbol{f} - \boldsymbol{A}^T \boldsymbol{y} - \boldsymbol{C}^T \boldsymbol{z} \\ \boldsymbol{Ax} - \boldsymbol{b} \\ \boldsymbol{s} - \boldsymbol{Cx} + \boldsymbol{d} \\ \boldsymbol{s} * \boldsymbol{z} + \Delta s^{aff} * \Delta z^{aff} - \sigma \mu e \end{bmatrix} \tag{22}$$

9. Compute the maximum allowable α satisfying the following condition

$$\begin{aligned} z + \alpha \Delta z \geq 0 \\ s + \alpha \Delta s \geq 0 \end{aligned} \tag{23}$$

10. Update $(\boldsymbol{x,y,z,s})$

11. Update the complementary measure

$$\mu = \frac{\boldsymbol{s}^T \boldsymbol{z}}{m_{ic} + m_{ec}} \tag{24}$$

12. End while loop

References

[1] M. Vašak, N. Peric, Model predictive control – a recent advance in control theory, Proceedings of the International Conference "Business System Management – UPS 2004" (2004) 263-267.
[2] L. A. Wolsey, Integer programming, John Wiley & Sons, Inc., 1998.
[3] A. Bemporad, M. Morari, Control of systems integrating logic, dynamics and constraints, Automatica 35(1999) 407-427.
[4] O. V. Volkovich, V. A. Roshchin, and I. V. Sergienko. Models and methods of solution of quadratic integer programming problems. Cybernetics 23(1987) 289 – 305.
[5] T. Achterberg, SCIP: Solving constraint integer programs, Mathematical Programming Computation 1(2009) 1-41.
[6] A. Bemporad, Hybrid Toolbox - User's Guide,2004. http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox.
[7] Y. Wang, S. Boyd, Fast model predictive control using online optimization, IEEE Transactions on Control Systems Technology 18(2010), 267-278.
[8] R.T. Meyer, M. Žefran, R. A. DeCarlo, A comparison of the embedding method with multiparametric programming, mixed-integer programming, gradient-descent, and hybrid minimum principle-based methods, IEEE Transactions on Control Systems Technology 22(2014), 1784-1800
[9] J. Currie, A. Prince-Pike, D.I. Wilson, Auto-code generation for fast embedded model predictive controllers, Proceedings of the International Conference on Mechatronics and Machine Vision in Practice (2012) 122-128.
[10] P. Zometa, M. Kogel, T. Faulwasser, R. Findeisen, Implementation aspects of model predictive control for embedded systems, Proceeding of the American Control Conference (2012) 1205-1210.
[11] B. Huyck, H. J. Ferreau, M. Diehl, J. De Brabanter, J. F. M. Van Impe, B. De Moor, F. Logist, Towards Online Model Predictive Control on a Programmable Logic Controller: Practical Considerations, Mathematical Problems in Engineering , Volume 2012 (2012).
[12] J. Clausen, Branch and Bound Algorithms - Principles and Examples, 1999: http://janders.eecg.toronto.edu/1387/readings/b_and_b.pdf
[13] J. Nocedal, S. Wright, Numerical Optimization, Springer, 2006.
[14] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide. Warrendale, PA: SIAM,1999.
[15] J.R. Bunch, L. Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. Math. Comp. 31 (1977) 163–179.
[16] M.Z. Djurovic, A. Milancic, M. Krsulja, A simplified model of quadratic cost function for thermal generators. Proceedings of the 23rd DAAAM Symposium (2012) 25-25.
[17] F. N. Lee, A. M. Breipohl, Reserve constrained economic dispatch with prohibited operating zones. IEEE Transactions on Power Systems 8(1993) 246-254.