Optimization Algorithm Tailored for Embedded Model Predictive Control

author[1]

***Abstract*— Write an abstract here.**

I. INTRODUCTION

Write the introduction here.

II. RELATED WORK

A. Model predictive control

List some literature with information about model predictive control (MPC).

B. Solving optimization problems

In literature exist many algorithms which use primal dual methods to solve several optimization problems. Most of them are focused on finding solutions with high accuracy. For application as MPC, the optimization often does not necessarily has to yield such exact solutions. Therefore [1] describes a primal barrier method focused on getting fast, sufficiently exact solutions of quadratic programs, by making use of the special MPC problem structure. This paper's contribution is manly based on the algorithm in [1], which has been extended by some numerical robustness approaches. As already proposed in [6] we use regularization to have a more robust algorithm if necessary in combination with iterative refinement to compensate the error of regularization. Similar to [5] we integrate soft constraints in the Fast MPC algorithm of Boyd and focus particularly on the consequences of losing the exact penalty function guarantee. We extend the algorithm to be able to solve more general optimization problems, especially quadratically constrained quadratic programs (QCQPs) and second-order cone programs (SOCPs).
We aim to combine the advantages of the existing algorithms to receive fast solutions and use several approaches to strengthen the robustness of the algorithm. As result, we can compare the solutions for several approaches and estimate the influence of them to the algorithm.

III. PROBLEM STATEMENT

A. Problem Statement and Algorithm summary of Fast MPC

{Introduce the unchanged problem statement of [1]}
The control variables $u(t), \dots, u(t+T-1)$ and the predicted states $x(t+1), \dots, x(t+T)$ up to the planning horizon T are rewritten to one optimization variable

$$z = (u(t), x(t+1), \dots, u(t+T-1), x(t+T)) \in \mathbb{R}^{T(m+n)}.$$

The quadratic program (QP) can be formulated as

$$\begin{aligned} \min_{z} \quad & z^T H z + g^T z \\ \text{s.t.} \quad & Pz \leq h, \quad Cz = b \end{aligned} \tag{1}$$

For the use of an infeasible start primal barrier method a logarithmic barrier function

$$\phi(z) = \sum_{i=1}^{lT+l_f} -\log\left(h_i - p_i^T z\right) \tag{2}$$

is introduced to the cost function to handle the inequality constraints.

B. Quadratically Constrained Quadratic Program

{Vorteile von QCQP nennen} If linear constraints as bounds for state and control variable are not sufficient it can be necessary to use nonlinear constraints such as quadratic constraints

$$z^T \Gamma_i z + \beta_i^T z \leq \alpha_i, \quad i = 1, \dots, p \tag{3}$$

with positive semidefinite matrices $P_1, \dots, P_p \in \mathbb{R}^{n \times n}$ ([7]).

C. Second-Order Cone Program

A more general form of optimization problems are second-order cone programs (SOCP). It is possible to formulate a lot of different nonlinear convex optimization problems as SOCP ([7]). The constraints of dimension k_i of such SOCP can be formulated as the following inequality:

$$\|A_i z + b_i\|_2 \leq c_i^T z + d_i \quad i = 1, \dots, q,$$

where $A_1, \dots, A_q \in \mathbb{R}^{(k_i-1) \times n}$, $b_1, \dots, b_q \in \mathbb{R}^{k_i - 1}$, $c_1, \dots, c_q \in \mathbb{R}^n$ and $d_1, \dots, d_q \in \mathbb{R}$

D. Soft Constraints

To use the algorithm of [1] it is necessary that z always strictly satisfies the inequality constraints. If the system is moving near its constraints, it is not guaranteed that in case of disturbances its state x is remaining in the feasible area. To avoid such feasibility problems and to make the algorithm more robust we additionally introduce soft constraints. Subsequently the algorithm can violate some inequality constraints by some additional penalty in the cost function. There are different ways to introduce soft constraints in an optimization algorithm. [5] proposes a method tailored for the algorithm we use without any typically needed additional slack variable and compares the solutions with the typical procedure. Its method comes along the loss of the exact penalty formulation, what makes it

*This work was not supported by any organization
[1]author hannes.heinemann@st.ovgu.de

possible that the solution is not necessarily the same as in the unmodified case although there is a feasible point that satisfies the constraints.
We now want to focus on the consequences of the loss of the exact penalty formulation.

IV. EXTENDED ALGORITHM

A. Generalized Constraints

In the described primal barrier method the gradient and the Hessian of the logarithmic barrier function are necessary. SOCCs in the above mentioned form are not continuously differentiable. Therefore SOCCs in Generalized form [2]

$$\|A_i z + b_i\|_2^2 \leq \left(c_i^T z + d_i\right)^2 \quad i = 1, \ldots, q \tag{4}$$

can be used.

B. Extended Problem Statement

The algorithm of [1] shell still be used, therefore the general form of the extended problem statement is not changed. We can resolve the norm in (4) and reorder the terms to

$$\left[-\left(c_i^T z + 2d_i\right) c_i^T + \left(z^T A_i^T + 2b_i^T\right) A_i\right] z \leq d_i^2 - b_i^T b_i \tag{5}$$

Only the constant matrix P and the vector h, also constant with respect to the optimization variable z, are expanded with p rows belonging to the p new quadratic constraints (3) and q rows for the conic constraints (5) to

$$\hat{P}(z) = \begin{bmatrix} P \\ \beta_1^T + z^T \Gamma_1 \\ \vdots \\ \beta_p^T + z^T \Gamma_p \\ -\left(c_1^T z + 2d_1\right) c_1^T + \left(z^T A_1^T + 2b_1^T\right) A_1 \\ \vdots \\ -\left(c_q^T z + 2d_q\right) c_q^T + \left(z^T A_q^T + 2b_q^T\right) A_q \end{bmatrix}$$

and

$$\hat{h} = \begin{bmatrix} h \\ \alpha_1^T \\ \vdots \\ \alpha_p^T \\ d_1^2 - b_1^T b_1 \\ \vdots \\ d_q^2 - b_q^T b_q \end{bmatrix}.$$

Different from the extended vector $\hat{h}$, the extended matrix $\hat{P}(z)$ is not constant with respect to z anymore. Despite the dependency on z, expanding P and h does not change the structure of Φ exploited in [1]. Consequently we can still use the algorithm to solve the QCQP or SOCP. With the new inequality constraint $\hat{P}(z)z \leq \hat{h}$ the logarithmic barrier function looks like

$$\phi(z) = \sum_{i=1}^{lT+l_f+p+q} -\log\left(\hat{h}_i - \hat{p}_i^T(z) z\right) \tag{6}$$

where $\hat{p}_i^T(z)$ is the ith rows of $\hat{P}(z)$ depending on z. The gradient of the logarithmic barrier function $\nabla\phi(z)$ is necessary to calculate the residual r. Because of the first order derivatives of the functions associated with the inequality constraints

$$\nabla f_i(z) = \beta_i^T + 2z^T \Gamma_i \tag{7}$$

and

$$\nabla f_i(z) = \left(2\left(-\left(c_i^T z + d_i\right) c_i + \left(z^T A_i^T + b_i z^T\right) A_i\right)\right),$$

the gradient of the logarithmic barrier function is simply derived by forming $\hat{P}$ with argument $2z$ multiplied by $\hat{d}$.

$$\nabla\phi(z) = \hat{P}^T(2z)\hat{d} \tag{8}$$

with

$$\hat{d}_i = \frac{1}{\hat{h}_i - \hat{p}_i^T(z) z} \tag{9}$$

The second order derivatives of the functions associated with the inequality constraints are

$$\nabla^2 f_i(z) = 2\Gamma_i \tag{10}$$

and

$$\nabla^2 f_i(z) = -2\left(c_i c_i^T - A_i^T A_i\right), \tag{11}$$

of which we obtain two additional terms, that have to be added to the Hessian of $\phi(z)$

$$\begin{aligned} \nabla^2\phi(z) = & \hat{P}(2z)\mathrm{diag}(\hat{d})^2 \hat{P}(2z) \\ & + \sum_{i=lT+lf+1}^{lT+lf+p} \left(2\hat{d}_i \Gamma_i\right) \\ & + \sum_{j=lT+lf+p+1}^{lT+lf+p+q} \left(-2\hat{d}_j\left(c_j c_j^T - A_j^T A_j\right)\right), \end{aligned} \tag{12}$$

to form Φ in the resulting system of linear equations.

C. Selecting κ

In [1] the use of a fixed κ is proposed. But it is difficult to find a single κ which provides fast convergence of the algorithm against the optimal solution for all feasible states of a controlled system. It should be considered to calculate a new κ once in every time step of the MPC. To have the effect of the barrier function term to the whole cost function in the same magnitude as the effect of the weighting terms, we can calculate κ similar as in [4] for linear programs. Several tests have shown that a good κ can be estimated by adopting the suggested procedure of [4] for quadratic programs as

$$\kappa = \frac{z^T H z + g^T z}{T(n+m)}$$

D. Regularization

The most important step to solve the system of linear equations is to compute the Cholesky factorization of every block in Φ. For Cholesky factorization the blocks have to be symmetric and positive definite ([2]). In Order to have all blocks positive definite and so ensure the Cholesky factorization of every block in Φ exists, we introduce a regularization term in the system of linear equations. We now solve

$$\begin{bmatrix} \Phi + \epsilon I & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta v \end{bmatrix} = - \begin{bmatrix} r_d \\ r_p \end{bmatrix} \tag{13}$$

and tolerate a small error we make instead of solving the original system of linear equations.

E. Iterative Refinement

For solving the optimization problem as application of a MPC, it is often sufficient to use the solution of the linear equations with regularization term. If it is necessary to get the solution of the original linear equations we can compensate the error of regularization by iterative refinement as described in [6]. Let

$$Kl = r$$

the original system of linear equations and

$$\tilde{K} l^{(0)} = r$$

the system with regularization we can solve easily. Setting $k = 0$ a correction $\delta l^{(k)}$ can be approximated by solving

$$\tilde{K} \delta l^{(k)} = \left(r - K l^{(k)} \right),$$

which takes less effort because we already calculated the operator $\tilde{K}^{-1}$ before. Then one has to update

$$l^{(k+1)} = l^{(k)} + \delta l^{(k)}$$

and iterate k. After repeating both steps until

$$\left\| r - K l^{(k)} \right\|$$

is sufficiently small, $l^{(k)}$ can be used as good approximation for l.

V. RESULTS

Comparison of the speed of the algorithm and how it satisfies the constraints for PCE of the airplane example

condensed formulation (one block)
$< - >$
exploit the structure of MPC optimization problem

A. Test QPs

Results of Solving Test QPs by [3]

B. Description of Airplane PCE

C. Results for Airplane PCE

VI. CONCLUSIONS

What are the conclusions?

ACKNOWLEDGMENT

Acknowledgment.

REFERENCES

[1] Y. Wang and S. Boyd, Fast Model predictive control using online optimization, IEEE Transactions on Control Systems Technology, vol. 18, no. 2, pp. 267-278, March 2010.
[2] S. Boyd and L. Vandenberghe, Convex Optimization
[3] I. Maros and C. Mszros, A Repository of Convex Quadratic ...
[4] A. Marxen, Primal Barrier Methods for Linear Programming...
[5] A. Richards, Fast Model Predictive Control with Soft Constraints
[6] J. Mattingley, S. Boyd, CVXGEN: a code generator for embedded convex optimization
[7] M. S. Lobo and L. Vandenberghe, Application pf second-order cone programming
[8] Weitere Literatur