Sequential l1 quadratic programming for nonlinear model predictive control

Boiroux, Dimitri; Jørgensen, John Bagterp

Published in:
IFAC-PapersOnLine

Link to article, DOI:
10.1016/j.ifacol.2019.06.107

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Sequential \( \ell_1 \) Quadratic Programming for Nonlinear Model Predictive Control

Dimitri Boiroux, John Bagterp Jørgensen

Department of Applied Mathematics and Computer Science, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark.

Abstract: In this paper, we present and describe a computationally efficient sequential \( \ell_1 \) quadratic programming (S\( \ell_1 \)QP) algorithm for Nonlinear Model Predictive Control (NMPC). We use a tailored trust region sequential quadratic programming for the solution of the optimal control problem (OCP) involved in the NMPC algorithm. We use a multiple shooting approach for numerical integration and sensitivity computation. A second order correction ensures a faster convergence of the SQP algorithm. We exploit the structure of the OCP by using an efficient primal-dual interior point algorithm based on Riccati factorizations and a block diagonal BFGS update of the Hessian matrix. The complexity scales linearly with the prediction horizon length. We numerically evaluate and compare the performance of our algorithm on a numerical example.

Keywords: Nonlinear model predictive control, trust region algorithm, sequential quadratic programming

1. INTRODUCTION

Nonlinear model predictive control (NMPC) refers to model-based control algorithms where the model consists of nonlinear differential equations (Allgöwer et al. (1999)). The review from Qin and Badgwell (2000) mentions at least 88 industrial applications of NMPC. Some examples of research topics using NMPC include finance (Chong et al. (2015)), oil recovery (Capolei et al. (2012)), industrial processes (Wang et al. (2016)), automotive applications (Liu et al. (2015)), or biomedical engineering (Boiroux et al. (2010); Zavitsanos et al. (2016)). At each time sample, NMPC-based algorithms require the solution of a nonlinear optimal control problem (OCP). For some applications, e.g. in biomedical engineering, it is important that the control algorithm can be implemented on a portable system. Therefore, the algorithms must be designed in a computationally efficient and robust way that can be implemented on an embedded system.

Sequential quadratic programming (SQP) refers to a set of iterative methods for the numerical solution of NLPs. At each iteration, the NLP is approximated by a quadratic program (QP). The equality and inequality constraints are linearized at each iteration. The objective function of the QP sub-problem is a quadratic approximation of the Lagrangian. SQP algorithms can be subdivided between local SQP methods and trust region algorithms.

In local SQP methods, the optimization algorithm computes a search direction at each iteration using a Newton or quasi-Newton method. A line search computes a scaling factor to ensure global convergence. Under some assumptions, local SQP algorithms converge globally with a superlinear rate of convergence (Nocedal and Wright (2006)).

In trust-region methods, the length of the search direction step is bounded to be within a certain region. The size of the trust region is updated at each SQP iteration to ensure a fast convergence. One of the challenges of trust region methods is to ensure feasibility of the QP sub-problem. To circumvent the possible infeasibility of the sub-problem, a popular solution is to use soft constraint formulations of the equality and/or inequality constraints. Plantenga (1994) provides a review of approaches to handle inconsistent constraints in trust region algorithms, and Teny et al. (2004) describe a trust region SQP algorithm for NMPC. Software for the numerical solutions of NLPs have been developed, e.g. SNOPT (Gill et al. (2005)). These software are suited for general SQPs, but do not exploit the sparse and specific structure of OCPs.

This paper presents a tailored trust region algorithm based on sequential \( \ell_1 \) quadratic programming (S\( \ell_1 \)QP) to solve OCPs. S\( \ell_1 \)QP algorithms have been introduced by Fletcher (1987). They ensure feasibility of the QP sub-problem by adding \( \ell_1 \) penalties on equality and inequality constraint violations. In practice, we implement the \( \ell_1 \) penalties by using slack variables. When solving the OCP, one computational bottleneck is the computation of the search direction arising in the QP sub-problem. Works from Wang and Boyd (2010) and Domahidi et al. (2012) show computationally efficient methods for computing the search direction. In this paper, we utilize Riccati recursions (Rao et al. (1998); Jørgensen et al. (2012); Frison and Jørgensen (2013); Hågdrup et al. (2017)).

Section 2 describes the OCP formulation. Section 3 presents the different steps of the trust region algorithm. Section 4 illustrates and discusses the algorithm on the Van der Pol oscillator. Section 5 summarizes the contributions of this paper.
2. PROBLEM FORMULATION

We assume a zero-order hold parametrization of the input vector, \( u(t) \). At each iteration, the NMPC algorithm solves an optimal control problem (OCP) in the Bolza form

\[
\min_{\{x_{k}, u_{k}\}^{N-1}_{k=0}} \phi = \sum_{k=0}^{N-1} G_k(x_k, u_k) + h(x_N),
\]

\[
\text{s.t.} \quad x_0 = \tilde{x}_{0},
\]

\[
b_k := F_k(x_k, u_k, d_k) - x_{k+1} = 0,
\]

\[
u_{\min} \leq u_k \leq u_{\max}.
\]

Remark 1. In this formulation, we assume that \( x_0 \) is fixed and is not an optimization variable.

The function describing the state dynamics is

\[
F_k = \{ x(t_{k+1}) : \dot{x}(t) = f(x(t), u_k, d_k), x(t_k) = x_k \}. \quad (2)
\]

The objective function (1a) consists of discrete stage costs, \( G_k \), and a terminal cost, \( h(x_N) \). The discrete time stage cost is

\[
G_k = \int_{t_k}^{t_{k+1}} g(x(t), u_k) dt. \quad (3)
\]

To have a compact notation, we rewrite the decision variables, \( \tilde{x} \), as

\[
\tilde{x} = [u'_0 \ x'_1 \ u'_1 \ x'_2 \ \ldots \ x'_{N-1} \ u'_{N-1} \ x'_{N}]', \quad (4)
\]

and we compute the residuals as

\[
b(\tilde{x}) = \begin{bmatrix}
F_0(x_0, u_0, d_0) - x_1 \\
F_1(x_1, u_1, d_1) - x_2 \\
\vdots \\
F_{N-1}(x_{N-1}, u_{N-1}, d_{N-1}) - x_N
\end{bmatrix} \quad (5)
\]

The objective function, \( \phi(\tilde{x}) \), is

\[
\phi(\tilde{x}) = \sum_{k=0}^{N-1} G_k(x_k, u_k, d) + H(x_N). \quad (6)
\]

Using these notations, we reformulate the discrete-time OCP (1) as a constrained optimization problem in the form

\[
\min_{\tilde{x}} \phi(\tilde{x}),
\]

\[
\text{s.t.} \quad b(\tilde{x}) = 0,
\]

\[
c(\tilde{x}) \geq 0.
\]

The first order KKT conditions of the constrained nonlinear optimization problem (7) are

\[
\nabla_{\tilde{x}} L(\tilde{x}, y, z) = \nabla_{\tilde{x}} \phi(\tilde{x}) - \nabla_{\tilde{x}} b(\tilde{x}) y - \nabla_{\tilde{x}} c(\tilde{x}) z = 0,
\]

\[
b(\tilde{x}) = 0,
\]

\[
c(\tilde{x}) \geq 0,
\]

\[
z \geq 0,
\]

\[
c_i(\tilde{x}) z_i = 0 \quad \forall i \in I,
\]

where \( y \) and \( z \) are the Lagrange multipliers associated to the equality (7b) and the inequality constraints (7c), respectively. We use these conditions to check the convergence of the algorithm.

3. TRUST REGION ALGORITHM

In trust region algorithms, we ensure that the step, \( p \), lies in a region where the QP approximation of the initial problem (7) is sufficiently accurate, i.e. we add the following constraint

\[
\|p\| \leq \Delta, \quad (9)
\]

where \( \Delta \) is updated at each SQP iteration. Usually, it is not possible to ensure that the QP sub-problem with the additional constraint (9) is feasible. To solve the nonlinear program (7), we use a sequential \( \ell_1 \) quadratic program (SQP) algorithm. We use a multiple-shooting algorithm for the numerical integration of the objective function and the sensitivity computation (Bock and Plitt (1984); Diehl et al. (2009); Boiroux et al. (2010)). Usually, the \( \ell_1 \) QP sub-problem is

\[
\min_{p,v,w,t} \frac{1}{2} p' H p + g' p + \mu \sum_{i \in E} (v_i + w_i) + \mu \sum_{i \in I} t_i, \quad (10a)
\]

\[
\text{s.t.} \quad \nabla b(\tilde{x})' p + w - v = -b, \quad (10b)
\]

\[
\nabla c(\tilde{x})' p \geq -c(\tilde{x}) - t, \quad (10c)
\]

\[
v, w, t \geq 0, \quad (10d)
\]

\[
\|p\| \leq \Delta. \quad (10e)
\]

in which \( E \) is the set of equality constraints and \( I \) is the set of inequality constraints. For simplicity, we use the infinity norm for the constraint (10e), such that it becomes a bound constraint

\[
-\Delta \leq p_i \leq \Delta. \quad (11)
\]

The addition of the slack variables, \( v, w \) and \( t \), can reduce the problem size by omitting the slack variables \( t_i \).

3.1 Sensitivity computation

The main computational challenges in the SQP algorithm are the computation of the objective function, \( \phi(\tilde{x}) \), the derivatives of the objective function, \( \nabla_{\tilde{x}} \phi(\tilde{x}) \), the dynamics, \( b(\tilde{x}) \), and the sensitivities, \( \nabla_{\tilde{x}} b(\tilde{x}) \), \( b(\tilde{x}) \) and \( \phi(\tilde{x}) \) are computed by evaluation of (2) and (3), respectively.

For instance, the gradients of the equality constraints, \( b(\tilde{x}) \), with respect to the decision variable, \( \tilde{x} \), can be written for a prediction horizon \( N = 2 \) as

\[
\nabla_{\tilde{x}} b = \begin{bmatrix}
B_0 & 0 & 0 \\
-I & A_1 & 0 \\
0 & B_1 & 0 \\
0 & 0 & B_2 \\
0 & 0 & -I
\end{bmatrix}, \quad (12)
\]

in which \( A_k \) and \( B_k \) are the sensitivities of the equality constraints, \( b(p) \), with respect to the states and inputs. They are
$A_k = \nabla x_k F_k(x_k, u_k, d_k)$, \hfill (13a)  
$B_k = \nabla u_k F_k(x_k, u_k, d_k)$. \hfill (13b)

### 3.2 Interior point algorithm

We use a structured primal-dual interior point algorithm for the solution of the constrained OCP (10). In this algorithm, we implement a Riccati recursion to compute the Newton iterations in the primal-dual interior point algorithm (Rao et al. (1998); Jørgensen et al. (2004); Boiroux et al. (2012)). This Riccati recursion exploits the structure of the Hessian of the Lagrangian (28) and the gradients of the equality constraints (12). The computation time of this procedure scales linearly with the prediction horizon, whereas the computational time of classical matrix factorization techniques grows as $O(N^3)$. This factorization can be used to compute the optimal variation in the manipulated variables $\Delta u_k$, the optimal change in states variables $\Delta x_{k+1}$, and the Lagrange multipliers, $y_k$ (Jørgensen (2005)). For many OCPs with long control prediction horizon, $N$, Riccati recursion-based interior point solvers are considered as the most computationally efficient method to solve the QP sub-problem (10) (Jørgensen et al. (2012); Frison and Jørgensen (2013); Hagdrup et al. (2017)).

The equality constraints (10b) can be rewritten as

$$\Delta x_{k+1} = A_k' \Delta x_k + B_k' \Delta u_k + \Delta y_k + b_k,$$ \hfill (14)

for $k = 0, 1, \cdots, N$, or equivalently

$$\Delta x_{k+1} = A_k' \Delta x_k + B_k' \Delta \tilde{u}_k + b_k,$$ \hfill (15)

where

$$\tilde{B}_k = [B_k I - I]', \quad \tilde{u}_k = [u_k \; v_k \; w_k]'.$$ \hfill (16)

By using this convention, it is possible to use a Riccati recursion algorithm to solve (10).

### 3.3 Second order correction

The second order correction (SOC) avoids the Maratos effect (Maratos (1978)), i.e. a loss of the superlinear convergence. A similar effect may happen to trust region algorithms, as explained in the work by Yuan (1984). The SOC algorithm is described in Exler and Schittkowski (2007). The objective function for the SOC is

$$\phi_c = \frac{1}{2} p^T H_c p_c + (g + H_p p_c)^T p_c + \mu \sum_{i \in E} (v_{e,i} + w_{e,i}) + \mu \sum_{i \in E} f_{e,i}. \hfill (17)$$

It requires the solution of the following QP for the correction step, $p_c$;

$$\min_{p_c, v_{e,i}, w_{e,i}, f_{e,i}} \phi_c, \quad s.t. \quad \nabla b(\tilde{x}_k)'(p_c + p_c) + w_c - v_c = -b(\tilde{x}_k + p_c), \hfill (18a)$$

$$\nabla c(\tilde{x}_k)'p_c \leq -c(\tilde{x}_k + \tilde{p}_c) - t_c \hfill (18b)$$

$$v_{e,i} + v_{e,i} + w_{e,i} + w_{e,i} t_c + t_c \geq 0 \hfill (18c)$$

$$\Delta \leq p_c + p_c \leq \Delta \hfill (18d)$$

The QP (18) requires the evaluation of the equality constraints, inequality constraints, but does not require an update of the sensitivities.

### 3.4 Computation of the merit function

The $\ell_1$ merit function is

$$\phi_1(x, \mu) = f(x) + \mu \sum |b(x)| + \mu \sum \max(c(x), 0).$$ \hfill (19)

The estimated approximation of the objective function, $q_\mu(p)$, is

$$q_\mu(p) = f_k + \nabla f_k p + \frac{1}{2} p' B_k p + \mu \sum |b(x_k) + \nabla b(\tilde{x}_k)' p| +$$

$$\mu \sum \max(-\nabla c(\tilde{x}_k)' p - d, 0).$$ \hfill (20)

To accept or reject a step $p_k$, we evaluate the ratio

$$\rho = \frac{a_{red}}{p_{red}} = \frac{\phi_1(\tilde{x}_k, \mu) - \phi_1(\tilde{x}_k + p_k, \mu)}{q_\mu(0) - q_\mu(p_k)}.$$ \hfill (21)

The numerator describes the actual reduction (or increase) in the merit function between the initial point $\tilde{x}_k$ and the new iterate $\tilde{x}_k + p_k$. The denominator is the estimated decrease in the merit function. Thus, the ratio (21) compares the actual reduction in the objective function, $a_{red}$, with the reduction predicted by the model, $p_{red}$. It monitors the accuracy of the second order approximation of the NLP in the trust region and ensures the global convergence of the trust-region algorithm. We accept the step $\tilde{x}_k + p_k$ if $\rho > 0$, and reject it otherwise.

### 3.5 Update of the trust region

If the current step is accepted, we want to increase or reduce the size of the trust region depending on $\rho$

$$\Delta_{k+1} = \gamma(\rho) \Delta_k.$$

If the current step is rejected, we would like to shrink the trust region

$$\Delta_{k+1} = \gamma(\rho) \|p\|_{\infty}.$$

If $\rho$ is greater than 0.5, we assume that our model is accurate enough to represent the NLP, and therefore we increase the size of the trust region to allow larger steps. Conversely, if $\rho < 0.5$, it means that the QP is not accurate enough, and we reduce the size of the trust region to avoid rejected steps. To ensure smooth variations on $\Delta$, we use the following function to compute $\gamma(\rho)$

$$\gamma(\rho) = \min \left( \max \left( (2\rho - 1)^3 + 1, 0.25 \right), 2 \right).$$ \hfill (24)

Fig. 1 depicts $\gamma(\rho)$ and compares it with the classical update

$$\begin{cases} 
\gamma = 0.25, & \rho < 0.25, \\
\gamma = 1, & 0.25 \leq \rho \leq 0.75, \\
\gamma = 2, & \rho > 0.75. 
\end{cases}$$ \hfill (25)

### 3.6 Update of penalty parameter

At each iteration, we want to update the $\ell_1$ penalty parameter, $\mu$. If the equality and/or inequality constraints are not satisfied, we want to increase $\mu$ to make sure that the equality and inequality constraints are satisfied whenever possible. On the other hand, the algorithm must ensure that $\mu$ is not too large to avoid taking too small steps. Thus, if the equality and/or inequality constraints are not satisfied, we increase $\mu$ by setting

$$\mu_{k+1} = \min(2\mu_k, \bar{\mu}).$$ \hfill (26)
If the equality and inequality constraints can be satisfied, we use an \(\ell_1\) Powell update of the penalty parameter and we set
\[
\mu_{k+1} = \max\left(\frac{1}{2} (\mu_k + \|y\|_\infty), \frac{1}{2} (\mu_k + \|z\|_\infty), \|y\|_\infty, \|z\|_\infty\right)
\] (27)

3.7 Numerical integration

We use the classical explicit Runge Kutta solver developed by Dormand and Prince (1980) to numerically compute the system state transition (2), the stage cost (3), the sensitivities, as well as the stage cost derivatives. We use the PI stepsize controller developed by Gustafsson (1992) to minimize the computational cost and to ensure that the selected steps meet a tolerance requirement.

3.8 BFGS update

Due to the structure of the OCP (1), the Hessian of the Lagrangian is a block diagonal matrix. It is
\[
\nabla^2_{pp} L = \begin{bmatrix} R_0 & Q_1 M_1 & M_1' R_1 \\ Q_1' M_1 & \ddots & \vdots \\ M_1' R_1 & \ddots & Q_{N-1} M_{N-1} \\ M_1' R_1 & \ddots & \ddots & M_{N-1}' R_{N-1} \\ \vdots & \ddots & \ddots & \ddots \\ P_N \end{bmatrix}. \] (28)

The block-matrices in (28) are symmetric. Each block is updated at each SQP iteration using a modified BFGS update. The modified BFGS update ensures that the Hessian is positive definite at each iteration (Nocedal and Wright (2006)). Since each block is updated with a rank 2 update, the Hessian of the Lagrangian is updated with a rank 2 \((N + 1)\) update. Thus, the performance of the BFGS update is not affected by the length of the prediction horizon (Steinbach (1994)).

3.9 Algorithm

Algorithm 1 summarizes its implementation. The highlighted statements in Algorithm 1 show the most computationally expensive steps.

Algorithm 1 Sequential \(\ell_1\) quadratic programming

Require:
while not converged do
Evaluate the objective function (6), the equality constraints (7b), the inequality constraints, the state sensitivity (13a) and the input sensitivity (13b).
Solve the \(\ell_1\) QP sub-problem (10) to get \(\bar{p}_k\), \(\bar{y}_{k+1}\) and \(\bar{z}_{k+1}\).
Compute \(b(\tilde{x}_k + p_k)\) and \(c(\tilde{x}_k + p_k)\).
Second order correction: Solve (18) and set
\[
p_k \leftarrow p_k + p_c.
\] (29)
Compute the merit function.
Update \(\mu\) using either (26) or (27).
Compute \(\rho\) using (21).
if \(\rho > 0\) then
Accept the current step. Update the solution
\[
\tilde{x}_{k+1} = \tilde{x}_k + p_k,
\] (30)
and the Lagrange multipliers
\[
\tilde{y}_{k+1} \leftarrow \tilde{y}_{k+1}, \quad (31a)
\tilde{z}_{k+1} \leftarrow \tilde{z}_{k+1}. \quad (31b)
\]
Update the size of the trust region using (22).
Update the Hessian of the Lagrangian using a block diagonal BFGS update.
else
Reject the current step.
Reduce the size of the trust region using (23).
end if
Check convergence with respect to the KKT conditions.
end while
return

Fig. 2. States (top) and inputs (bottom) for the Van der Pol problem. The sampling time is 0.05 s, i.e. \(N = 100\).

4. NUMERICAL RESULTS

We consider the Van der Pol oscillator to test our algorithm. The objective function is
scales linearly for the solution of the QP subproblem algorithm in Matlab 2017a. The computational burden increases with the prediction and control length. We implemented a prototype of our trust-region trust-region algorithm for different control and prediction horizons. We implemented a prototype of our trust-region algorithm for different control and prediction horizons. Compared to the Matlab prototype, a C implementation of the SQP algorithm and the complementarity condition norm (8e). We computed the bound constraints, \(||b(x)||\), and the complementarity condition norm (bottom right) for the Van der Pol oscillator.

\[
\phi = \frac{1}{2} \int_0^5 (x_1^2(t) + x_2^2(t) + u^2(t)) \, dt + \frac{\eta}{2} (x_1(5) - x_2(5) + 1)^2. \tag{32}
\]

and the dynamics are described by the following ODEs
\[
\dot{x}_1(t) = x_2(t), \tag{33a}
\]
\[
\dot{x}_2(t) = -x_1(t) + \alpha (1 - x_1^2) x_2 + u(t). \tag{33b}
\]

For the simulations, we set \(\eta = 100\) and \(\alpha = 1\). We set the bound constraints, \(u_{\text{min}}\) and \(u_{\text{max}}\), to \(-1\) and \(1\), respectively. The tolerance requirements are set to \(10^{-6}\). The initial condition is \(x(0) = [1; 0]\).

Fig. 2 depicts the state vector, \(x(t)\), and the input, \(u(t)\). The shape of the solution is similar to the optimal states and inputs in Bock and Plitt (1984).

Fig. 3 shows the objective function, the gradient of the Lagrangian norm (8a), the equality constraints norm (8b), and the complementarity condition norm (8e). We compare between our trust-region algorithm with SOC, our trust-region algorithm without SOC and a previously published local SQP algorithm with line search (Boiroux et al. (2010)). For the Van der Pol problem, the local SQP algorithm converges more slowly than the trust region-based algorithms. Also, adding the SOC makes the convergence even faster for the equality constraints.

Table 1 shows the number of SQP iterations, the total number of model evaluations and the total number of sensitivity evaluations for different problem sizes. As for the multiple shooting algorithm presented in Bock and Plitt (1984), the number of SQP iterations does not increase with the prediction and control length.

![Graphs showing objective function, gradient of the Lagrangian norm, equality constraints norm, and complementarity condition norm for the Van der Pol oscillator.]

Table 1. Number of SQP iterations, model evaluations and sensitivity evaluations for different control and prediction horizon lengths, \(N\).

<table>
<thead>
<tr>
<th>(N)</th>
<th>SQP iterations</th>
<th>Model evaluations</th>
<th>Sensitivity evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>13</td>
<td>8855</td>
<td>7805</td>
</tr>
<tr>
<td>25</td>
<td>27</td>
<td>18277</td>
<td>18235</td>
</tr>
<tr>
<td>30</td>
<td>13</td>
<td>7413</td>
<td>7406</td>
</tr>
<tr>
<td>40</td>
<td>17</td>
<td>10136</td>
<td>10136</td>
</tr>
<tr>
<td>60</td>
<td>27</td>
<td>23520</td>
<td>23520</td>
</tr>
<tr>
<td>80</td>
<td>28</td>
<td>32480</td>
<td>32480</td>
</tr>
<tr>
<td>100</td>
<td>28</td>
<td>40600</td>
<td>40600</td>
</tr>
</tbody>
</table>

Fig. 4 shows the CPU times of an implementation of this trust-region algorithm for different control and prediction horizons. We implemented a prototype of our trust-region algorithm in Matlab 2017a. The computational burden scales linearly for the solution of the QP subproblem and the numerical integration/sensitivity computation. For \(N = 100\), it takes approximately 60% of the total computation time, whereas the interior point algorithm takes approximately 25% of the computation time.

5. CONCLUSION

In this paper, we developed and tested a trust-region algorithm for the solution of OCPs. The algorithm ex-

![Graph showing CPU times for different prediction and control horizons.]

This graph shows the CPU times for different prediction and control horizons.
exploits the sparsity and the structure of the OCP. Second order correction accelerates the convergence, especially of the nonlinear equality constraints. The Riccati recursion becomes particularly computationally efficient when the number of states is small relatively to the prediction horizon. In particular, the complexity scales linearly with the prediction horizon length. Compared to the Matlab prototype, a C implementation of the SQP algorithm and warm-start strategies in closed-loop would significantly decrease the computational time.

REFERENCES


