Asian Journal of Control, Vol. 18, No. 6, pp. 1–14, November 2016
Published online in Wiley Online Library (wileyonlinelibrary.com) DOI: 10.1002/asjc.1299

EMBEDDED MPC CONTROLLER BASED ON INTERIOR-POINT METHOD WITH CONVERGENCE DEPTH CONTROL

Yi Ding, Zuhua Xu, Jun Zhao, Kexin Wang and Zhijiang Shao

ABSTRACT

To allow the implementation of model predictive control on the chip, we first propose a primal–dual interior point method with convergence depth control to solve the quadratic programming problem of model predictive control. Compared with algorithms based on traditional termination criterion, the proposed method can significantly reduce the computation cost while obtaining an approximate solution of the quadratic programming problem with acceptable optimality and precision. Thereafter, an embedded model predictive controller based on the quadratic programming solver is designed and implemented on a digital signal processor chip and a prototype system is built on a TMDSEVM6678LE digital signal processor chip. The controller is verified on two models by using the hardware in loop frame to mimic real applications. The comparison shows that the whole design is competitive in real-time applications. The typical computation time for quadratic programming problems with 5 decision variables and 110 constraints can be reduced to less than 2 ms on an embedded platform.

Key Words: convergence depth control, digital signal processor, embedded control, interior point method, model predictive control, quadratic programming

I. INTRODUCTION

Model predictive control (MPC) technology has been successfully applied in a variety of industry processes in recent decades [1]. The outstanding advantage of MPC is its ability to handle constraints explicitly by solving a constrained quadratic programming (QP) problem at each control interval. The complexity in QP computation has restricted MPC to process control applications and the algorithm is commonly implemented in a host computer with high computational performance. Hence, there is increased interest for introducing MPC into a wider range of applications outside the process industry, especially in those that require fast response times and in those that must run on resource-constrained, embedded computing platforms with low cost or low power requirements [24,30].

A range of methods has been proposed to solve embedded MPC problems. These methods mostly emphasize QP solving in every control interval, which is the bottleneck of MPC algorithm implementation on embedded platforms. Two approaches to fast QP solution in MPC can be classified. First is the explicit or offline QP solution, which pre-computes the QP solution for all possible problem instances. A method called explicit model predictive control (EMPC) was proposed in [2]; in this method, the online computation is reduced to a simple evaluation of an explicitly defined piecewise affine function. The fatal drawback of EMPC implementation is its memory consumption. As reported in [3], a model helicopter application with six states, two inputs, three outputs, and only input constraints will need 202 KB of memory with only one step prediction. Furthermore, 62 MB of memory is needed when the prediction horizon increases to two; this condition is nearly impossible to achieve for embedded platforms. Several authors have proposed improvements on EMPC in various aspects, such as the use of approximation algorithms [4,5], quick search in piecewise affine function [6,7], and combination of explicit control law and online optimization [8].

The second approach is online computing. The interior-point method and active-set method are two main online algorithms that are widely used in practice. These two methods were first introduced into QP solving for MPC [9], wherein the infeasible-interior-point method and active-set method are studied and the special structures of MPC formulations are exploited. Many studies that focus on tailoring classic active-set method and interior-point method for MPC applications have been published. Rao *et al*. [10] presented a structured interior-point method for the efficient solution of QP problem in MPC. Bartlett *et al*. [11] tailored and applied an object-oriented implementation of a novel dual space, Schur complement algorithm to large MPC applications. Ferreau [7] proposed an online active-set strategy for the fast solution of QP, which exploits solution information of the previous QP under the

Manuscript received April 26, 2015; revised August 24, 2015; accepted January 23, 2016.

Yi Ding, Zuhua Xu (corresponding author, e-mail: xuzh@iipc.zju.edu.cn), Jun Zhao, Kexin Wang, and Zhijiang Shao are with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China.

We acknowledge the support of National High-tech Research and Development Program of China (No.2014AA041802), National Natural Science Foundation of China (No.61273145, 61273146) and NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization (No. U1509209).

assumption that the active set does not change much from one QP to the next. An efficient interior-point methods, tailored to convex multistage problems, can be found in [12], some important algorithmic details such as block-wise Cholesky factorization and rank one modifications are also provided to implement high speed MPC solvers. Besides the active-set method and interior-point method, other online methods include the piecewise smooth Newton method [13], and dual gradient-projection [14], which is simple to implement and the iteration count can be tightly estimated for embedded applications.

Hardware is another important issue in embedded MPC research. As early as 2005, a prototype was designed and implemented on a field programmable gate array (FPGA) chip in [15]. A few subsequent studies have focused on the combination of the characteristics of QP in MPC and hardware features. A custom integrated circuit architecture that is specifically targeted to the MPC problem was designed in [16]. Algorithm parallelization and hardware pipeline were studied for embedded MPC on FPGA in [17] and [18]. The previous designs of embedded MPC were mostly implemented on FPGA because of its high computing speed and parallel processing ability; however, the cost of FPGA is also higher than other hardware platforms. Researchers have recently begun to consider other platforms, such as digital signal processor (DSP) and ARM [19–21]. We choose DSP as our hardware platform because DSP can maintain a relatively low price while obtaining high computational performance.

In practical optimization, designing the starting point and termination criterion of QP algorithm is an important issue and has a significant effect on the performance of the algorithm. A number of studies on fast MPC algorithm have mentioned the choice of the starting point for the QP algorithm. However, few studies have focused on the termination criterion for QP solving in embedded MPC to accelerate the solving process. To date, the traditional termination criteria for optimization problem are based on checking the convergence of iteration point and objective function, which ineffectively reflect the extent of convergence and margin of improvement. Furthermore, this approach usually brings the iterations into a situation wherein the computation cost exceeds the accuracy improvements. For MPC applications, an adequate approximate solution to the optimum is always sufficient and high precision is unnecessary from an engineering viewpoint. We illustrate the convergence process in QP solving in MPC and find that most improvements during the optimization procedure are made within a small part of total computation time. Thus, the convergence depth control (CDC), a strategy to terminate the iterations early and wisely, can be used to save time without degrading performance. In this paper, we introduce the CDC strategy into the interior-point method to present a new algorithm, and the algorithm is implemented on an embedded platform. The algorithm can significantly reduce the computation cost while obtaining an approximate solution of the QP problem with acceptable optimality and precision.

The rest of this paper is organized as follows. Section II discusses a compact MPC form based on the state-space model and presents the formularization of a standard QP problem. Section III shows the interior-point method and CDC and discusses a few details. Section IV presents the implementation details of the MPC controller on DSP. A prototype system is built on a TMDSEVM6678LE DSP evaluation board, and a hardware in loop (HIL) frame is set up to verify the control performance. Section V applies the embedded controller on the DSP to two processes to verify the performance. Section VI concludes.

II. MODEL PREDICTIVE CONTROL

We consider a discrete linear time-invariant system in the following state-space form:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k) \end{aligned} \tag{1}$$

where $y \in \mathbf{R}^l$ is the output variable, $u \in \mathbf{R}^m$ is the input variable, and $x_m \in \mathbf{R}^n$ is the state variables of the system. By introducing a new state vector $x(k) \underline{\underline{\text{def}}} \left[\Delta x_m(k)^T \quad y(k)^T \right]^T$, we can rewrite 1 as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k), \end{aligned} \tag{2}$$

where

$$A = \begin{bmatrix} A_m & 0 \\ C_m A_m & I \end{bmatrix}, B = \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}, C = [0 \quad I].$$

Denote

$$Y_P(k) = \left[y(k+1|k)^T \quad y(k+2|k)^T \quad \cdots \quad y(k+P|k)^T \right]^{\mathrm{T}}$$

$$\Delta U_M(k) = \left[\Delta u(k)^T \quad \Delta u(k+1)^T \quad \cdots \quad \Delta u(k+M-1)^T \right]^T$$

where $Y_P(k)$ is the predicted behaviors at time instant k over the prediction horizon P, and $\Delta U_M(k)$ is the control move vector of the plant over the control horizon M with $\Delta u(k) = u(k) - u(k-1)$. Thereafter, the following multi-step prediction based on state-space model 2 can be derived:

$$Y_P(k) = Fx(k|k) + \Phi \Delta U_M(k) \tag{3}$$

with

$$F=\begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^P \end{bmatrix}, \Phi=\begin{bmatrix} CB & & & & \\ CAB & CB & & & \\ CA^2B & CAB & CB & & \\ \vdots & & & & \\ CA^{P-1}B & CA^{P-2}B & CA^{P-3}B & \cdots & CA^{P-M}B \end{bmatrix}$$

At each control interval, the MPC controller optimizes the future control moves over the control horizon to drive the prediction as closely as possible to a desired trajectory over the prediction horizon, which can be described as follows:

$$\min_{\Delta U_M(k)} J(k)=||R(k)-Y_P(k)||_Q^2+||\Delta U_M(k)||_R^2 \tag{4}$$

$$\text{s.t. } Y_P(k)=Fx(k|k)+\Phi\Delta U_M(k)$$

$$Y_{min}\leq Y_P(k)\leq Y_{max}$$

$$\Delta U_{min}\leq\Delta U_M(k)\leq\Delta U_{max}$$

$$U_{min}\leq U_M(k)\leq U_{max}$$

where $R(k)$ defines a reference trajectory for the outputs; Y_{min} and Y_{max} are the lower and upper output limits; U_{min} and U_{max} are the lower and upper operating limits for U; ΔU_{min} and ΔU_{max} are the lower and upper rate limits for U. Q is the output weighting matrix, and R is the control weighting matrix.

The optimization problem 4 can be formulated into a standard QP as follows:

$$\begin{aligned} &\min_z J=\frac{1}{2}z^TGz+c^Tz \\ &\text{subject to}: \ \Omega z\geq\omega \end{aligned} \tag{5}$$

where $G=\Phi^{\mathrm{T}}Q\Phi+R$, $c=\Phi^{\mathrm{T}}Q^T(Fx(k|k)-R(k))$, $z=\Delta U_M$ and

$$\Omega=\begin{bmatrix} -I \\ I \\ -B \\ B \\ -\Phi \\ \Phi \end{bmatrix}, \omega=\begin{bmatrix} -\Delta U_{max} \\ \Delta U_{min} \\ -U_{max}+U(k-1) \\ U_{min}-U(k-1) \\ -Y_{max}+Fx(k|k) \\ Y_{min}-Fx(k|k) \end{bmatrix}, I=\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, B=\begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \cdots & I \end{bmatrix},$$

where $I\in\mathbf{R}^{n_uM\times n_uM}$, $B\in\mathbf{R}^{n_uM\times n_uM}$, and n_u is the number of control variables.

In the standard QP formulation 5, $G\in\mathbf{R}^{\mathrm{n_{dec}}\times\mathrm{n_{dec}}}$ is a symmetric positive semi-definite matrix. $c\in\mathbf{R}^{\mathrm{n_{dec}}}$, $\Omega\in\mathbf{R}^{\mathrm{m_c}\times\mathrm{n_{dec}}}$, $b\in\mathbf{R}^{\mathrm{m_c}}$. Here, m_c denotes the number of inequality constraints and n_{dec} denotes the number of decision variables.

The QP problem in 5 can be analytically solved if constraints do not exist, and the optimal value is expressed as follows:

$$z^*=-G^{-1}c \tag{6}$$

With regard to input and output constraints, this QP problem can be solved by the method we introduce in the next section.

III. QUADRATIC PROGRAMMING SOLVING FOR MPC

3.1. Primal–dual interior-point algorithm

The QP problem has been extensively studied in the literature. The active-set method and interior-point method are two popular methods used to solve the QP problem. However, the worst-case complexity of the active-set method increases exponentially with the problem size, and the size of the linear system that needs to be solved at each iteration changes depending on the working set. From the perspective of real-time computation and hardware implementation, the interior-point method is applied to solve a QP problem in our work since it has polynomial complexity and maintains a constant predictable structure.

In our work, we apply the primal–dual algorithm and tailor the algorithm for embedded applications. The primal–dual algorithm uses Newton's method to solve a nonlinear system of equations known as the KKT optimality conditions [23], which are illustrated as follows.

For the standard QP problem as 5, the KKT conditions are the following:

$$Gz-\Omega^T\lambda+c=0, \tag{7a}$$

$$\Omega z-s-\omega=0, \tag{7b}$$

$$s_i\lambda_i=0, \ i=1,2,\ldots,m_c, \tag{7c}$$

$$(s,\lambda)\geq 0 \tag{7d}$$

where s is the slack variable and λ is the dual variable.

The KKT conditions can be solved by iteratively solving the following linear equation systems:

$$\begin{bmatrix} G & 0 & -\Omega^T \\ \Omega & -I & 0 \\ 0 & \Lambda & \mathcal{S} \end{bmatrix}\begin{bmatrix} \Delta z \\ \Delta s \\ \Delta\lambda \end{bmatrix}=\begin{bmatrix} -r_d \\ -r_p \\ -\mathcal{S}\Lambda e+\sigma\mu e \end{bmatrix} \tag{8}$$

where

$$r_d=Gz-\Omega^T\lambda+c, \ r_p=\Omega z-s-\omega$$

$$\mathcal{S}=\mathrm{diag}\left(s_1,s_2,\ldots,s_{m_c}\right), \ \Lambda=\mathrm{diag}(\lambda_1,\lambda_2,\ldots,\lambda_{mc}), \ \mathrm{e}=(1,1,\ldots,1)^{\mathrm{T}}.$$

σ is the centering parameter and $\mu=\frac{s^T\lambda}{m_c}$ is the duality measure.

3.2. Convergence depth control

Designing an appropriate termination criterion is important to judge whether an adequate approximate solution

to the optimum is obtained. A less stringent criterion may substantially reduce the solution accuracy, whereas an over stringent one makes the optimization proceed too long with little improvement on the solution.

A traditional termination criterion for interior-point method with a QP problem 5 is as follows:

$$\text{C1. } ||z_{k+1} - z_k|| < \varepsilon, \tag{9a}$$

$$\text{C2. } ||J_{k+1} - J_k|| < \varepsilon, \tag{9b}$$

$$\text{C3. } (\omega - \Omega z) < 0, \tag{9c}$$

where ε is a tolerance value, such as 10^{-8} (a default value in Matlab function quadprog). In this criterion, 9a and 9b are used to check the convergence of the iteration point and objective function, 9c is used to check the constraint violation.

This type of termination criterion is rigid and the only conclusion that can be reached is that the optimization process is "converged" or "not converged," which cannot effectively reflect the extent of convergence and margin of improvement. Thus, this criterion usually brings the iterations into a situation wherein the computation cost exceeds the accuracy improvements.

The behaviour of the interior-point method with traditional termination on two models from the literatures is illustrated in Fig. 1 [15,24]. In both cases, the algorithm is terminated at the expense of approximately 13 iterations, but iterates after the 5th iteration makes little progress compared with final results.

For both problems, most of the objective function and constraint violation improvements in the optimization process take place over a small part of the whole executing time. Thus, CDC [25], which is proposed by our team and has already been applied to problems from CUTE test sets and distillation sequence optimization, can be applied to the interior-point method algorithm to accelerate the solving process. CDC is designed to find a reasonable time to terminate the optimization algorithm by conducting an achievement estimation of the optimization process. With the help of CDC, the optimization algorithm will be aware of the improvement it has achieved and will achieve in the future; the optimization algorithm will also terminate properly and cleverly regardless of whether it converges or fails in the end.

In this strategy, the following indexes are introduced to evaluate the status of the current iterate. First, feasibility and predictive improvements on the objective at iterate z_k can be defined as follows:

$$\delta^k_{feasErr} = \max\{\omega - \Omega z_k\}, \tag{10}$$

$$\delta^k_{objErr} = \left|(Gz_k + c)^T \Delta z_k\right|, \tag{11}$$

Thereafter, the progress of the optimization process at iterate z_k can be defined as follows:

$$\delta^k_{feasChg} = \left|\delta^k_{\text{feasErr}} - \delta^{k-1}_{\text{feasErr}}\right|, \tag{12}$$

$$\delta^k_{objChg} = |J_k - J_{k-1}|, \tag{13}$$

By taking 12 and 13 as the measure of the errors, the convergence depth and degree of progress at iterate z_k can be defined as follows:

$$\theta^k_{convg} = S\left(\max\left\{\delta^k_{feasErr},\ \delta^k_{objErr},\ \mu^k\right\}, \varepsilon_0\right), \tag{14}$$

$$\theta^k_{prog} = S\left(\max\left\{\delta^k_{feasChg},\ \delta^k_{objChg},\ \mu^k\right\}, \varepsilon_0\right), \tag{15}$$

where ε_0 is a given tolerance (10^{-8} in our case), and S is the

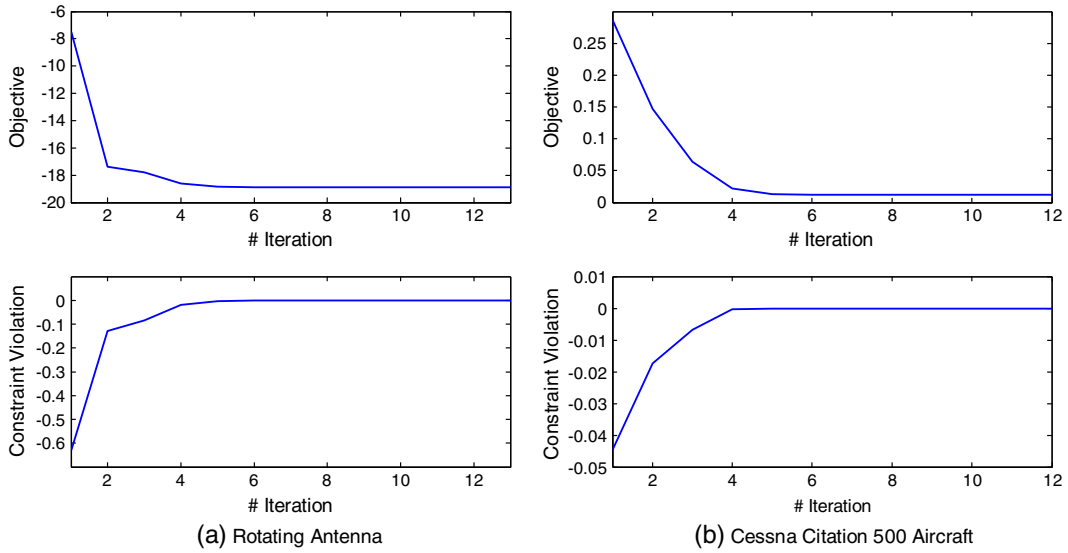


Figure 1. Behavior of interior-point methods with traditional termination criterion.

transformed sigmoid function defined as

$$S(\delta_k, \varepsilon_0) = \frac{\tanh[\xi(\log\delta_k)/(\log\varepsilon_0)]}{\tanh\xi} \tag{16}$$

ξ is a transformation parameter that influences the variation (being steeper or flatter) of $S(\delta_k, \varepsilon_0)$, which helps to obtain the desired properties of the sigmoid function in the interval between ε_0 and $1/\varepsilon_0$ with the opposite order of magnitude. Thus, the function has the value of 1 or -1 at the end of the interval and improves the estimation of the convergence achievement

With this function, the rigid criteria is softened into a flexible criterion ($\xi = 1.5$). This can be observed clearly by comparing with the traditional termination criteria, which can only give the answer of "converged" or "not converged."

Fig. 2 proposes the implementation framework of the CDC.

The optimization procedure is considered to be successfully converged if the convergence depth θ^k_{convg} has reached a predefined threshold θ_0; otherwise, the optimization procedure is considered to have "no improvements" if the degree of progress has reach the threshold θ_1 for more than η times. Thus, an "under-converged result" is put out. As proven in the literature [25], the iteration sequence produced by the interior-point method can be terminated by the CDC. Furthermore, the convergence depth is also proven to indicate the degree of current iterate converging to the optimum. Note that the sigmoid function 16 is implemented by using the piecewise polynomial fitting to reduce computational burden.

According to the optimization theory and CDC mentioned previously, a practical interior-point method algorithm with CDC is given in the appendix.

3.3. Details in QP solving

3.3.1 Warm start

As discussed in [22], a warm start plays an important role in interior-point method solving. We use a similar method to initialize the optimization iterations. Suppose at time $k-1$, the computed manipulate value change is expressed as follows:

$$\Delta\tilde{U} = \left[\Delta\tilde{u}(k)\ \ \Delta\tilde{u}(k+1)\ \ \cdots\ \ \Delta\tilde{u}(k+M-1)\right].$$

We can initialize the decision variable for time k by using the following:

$$z_0 = \left[\Delta\tilde{u}(k+1)\ \ \Delta\tilde{u}(k+2)\ \ \cdots\ \ \Delta\tilde{u}(k+M-1)\ 0\right].$$

Dual variables are initialized in a similar way. We maintain the value of dual variables when iterations end at time $k-1$ as $\tilde{s}$ and $\tilde{\lambda}$. Thereafter, we can initialize the dual variables for time k by using the following:

$$s_0^i = \max\left\{\tilde{s}^i, s^{ini}\right\},\ i = 1\ldots m_c,$$

$$\lambda_0^i = \max\left\{\tilde{\lambda}^i, \lambda^{ini}\right\},\ i = 1\ldots m_c,$$

where s^{ini} and λ^{ini} are the predefined constants. We usually give these variables a small value.

3.3.2 Solving linear equations

The major computational operation in the interior-point method is the solution of the system 8. Thus exploiting the problem's structure and choosing a suitable factorization algorithm are important. An "augmented system" can be derived from 8 through block elimination, and finally the "normal equations" can be obtained as follows:

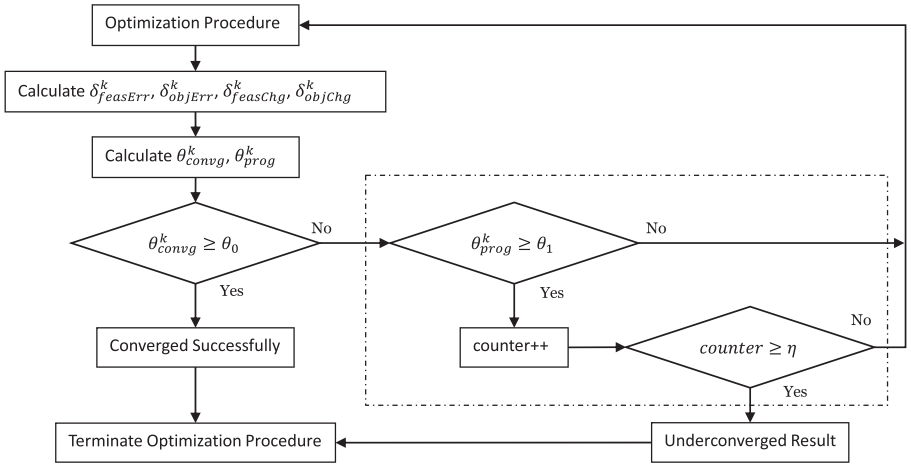


Figure 2. Implementation framework of the CDC.

$$\left(G+\Omega^T\mathcal{S}^{-1}\Lambda\Omega\right)\Delta z=-r_d+\Omega^T\mathcal{S}^{-1}\Lambda\left[-r_P-s+\sigma\mu e\Lambda^{-1}\right]. \quad (17)$$

For the MPC optimization problem, the coefficient matrix in 17 is symmetric positive semi-definite; thus, the linear equations can be solved by Cholesky factorization, which is more efficient than normal LU factorization.

Once Δz is obtained, we can acquire the following:

$$\Delta s=\Omega\Delta z+r_p. \quad (18)$$

$$\Delta\lambda=-\lambda-\mathcal{S}^{-1}(\Lambda\Delta s-\sigma\mu e). \quad (19)$$

After the transformation, the scale of the matrix to be factorized will be $(n_{dec}*n_{dec})$ instead of $((n_{dec}+2m_c)*(n_{dec}+2m_c))$. Given the fact that m_c is usually larger than n_{dec}, the computation time is significantly reduced.

3.3.3 Unconstrained solution check

To accelerate the QP solving speed, the unconstrained solution of the QP problem is first obtained according to 6. If the unconstrained solution meets the constraints, considerable time will be saved by directly producing the unconstrained solution instead of executing iterations. If the solution does not meet the constraints, QP solving iterations is invoked subsequently.

IV. DSP IMPLEMENTATION

4.1. Hardware

A prototype control system is implemented on a TMDSEVM6678LE evaluation module produced by Texas Instruments (TI). The key component of the module is the TI multicore DSP – TMS320C6678. The processor is integrated with eight C66x CorePac DSPs. Each core runs at 1.0 GHz to 1.25 GHz.

In our work, only a single DSP core is used in the implementation of MPC controller, and all memory sections are mapped in a local 512 KB L2 SRAM. In the future, we will study how to use multicore in the implementation of MPC control.

In our design, the IEEE 754 double-precision floating-point format is used for arithmetic because a single-precision floating-point format implementation can cause numerical instability and provide unbounded results. A similar problem is discussed in the literature [26]. The main difference between single precision and double precision is the unit roundoff u, also referred to as machine precision [27,28]. For single precision and double precision, the value is $u=2^{-24}\approx 5.96\times10^{-8}$ and $u=2^{-53}\approx 1.11\times10^{-16}$ respectively. When solving linear equations $Ax=b$ with Cholesky factorization, we obtain the following [28]:

$$||\Delta A||_2\leq 4n(3n+1)u||A||_2, \quad (20)$$

where ΔA denotes a perturbation on A, n denotes the dimension of A, and u is the machine precision.

We also obtain the following bound on the basis of matrix perturbation analysis:

$$\frac{||\Delta x||}{||x||}\leq\frac{\text{cond}(A)}{1-\text{cond}(A)\frac{||\Delta A||}{||A||}}\frac{||\Delta A||}{||A||}, \quad (21)$$

where Δx denotes the errors between the real x and computed solution and cond(A) is the condition number of matrix A. Another bound can be derived by combining 20 and 21:

$$\frac{||\Delta x||}{||x||}\leq\frac{4n(3n+1)u\times\text{cond(A)}}{1-4n(3n+1)u\times\text{cond(A)}}=\frac{1}{\frac{1}{4n(3n+1)u\times\text{cond(A)}}-1}. \quad (22)$$

A higher machine precision corresponds to less errors in the solving process. The condition number of A also plays an important role. When the problem is ill-conditioned, giving accurate solutions will be difficult. Given the inevitable transforms that make matrix ill-conditioned, importing a high precision floating-point format is consequential.

4.2. MPC controller on DSP

A flow chart of the controller is proposed in Fig. 3. The operations of memory allocation and deallocation are managed offline because the structure of interior point method is relatively fixed; thus the computation process is practically accelerated. The initialization of constants and parameters is also completed offline.

In the online part, a function named mpc_online is invoked after the sampling is conducted at every control interval and the function will produce the control action after QP solving is finished. Inside the mpc_online, the states are initially updated directly if the whole state vector can be measured or a state observer is used otherwise. Thereafter, the unconstrained solution check mentioned previously is performed. If the solution does not meet the constraints, the warm start module is executed to initialize the variables and the QP iteration is invoked subsequently. Finally, CDC will check the convergence depth and the degree of progress of the QP iterations and find a proper time to quit iterations according to a few pre-determined threshold values.

The design is developed in ANSI C language under Code Composer Studio™, A TI C6000 complier 7.4.7 is used to generate the target code, and a few code optimization techniques are used to improve performance. Code optimization is a fundamental procedure in embedded

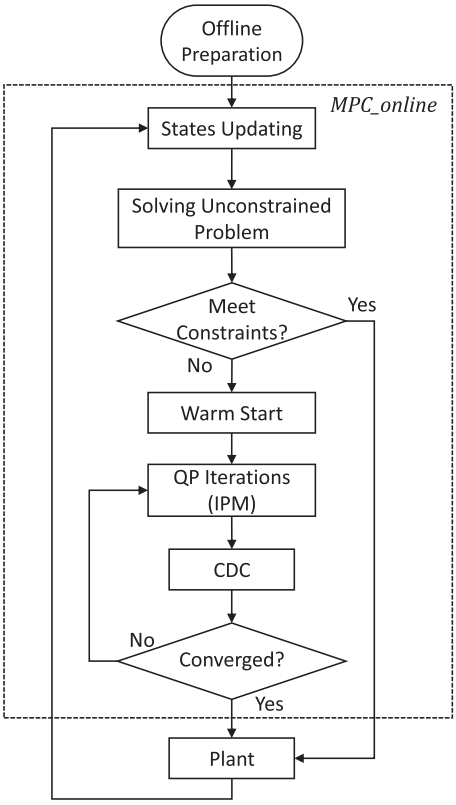


Figure 3. MPC_online framework.

development. In the present study, we use two tricks. First is the use of **MUST_ITERATE** pragma before a for-loop process to tell compiler the minimum iteration numbers of a for-loop, thus helping the complier with unrolling and software pipelining. The other is to qualify pointers in the program with **restrict** keyword to help the complier determine memory dependencies and evoke a parallelization strategy within the processor.

4.3. Hardware in loop implementation

To simulate the linear MPC controller on a nonlinear process, we use the HIL method for nonlinear processes in design. HIL is a technique that benefits researcher and engineer in research and design by placing the hardware (DSP in our case) into the simulation loop to verify the effect of hardware and software implementation. Fig. 4 is a sketch of the HIL in our implementation.

An USB-serial link is used to transport data byte by byte between PC and DSP. We build a simple master slave communication protocol in MPC controller on DSP and Matlab on PC.

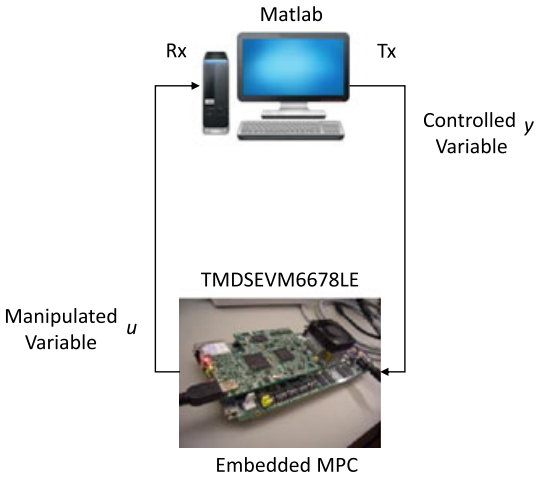


Figure 4. HIL Implementation.

V. CASE STUDY

The high performance of the proposed algorithm and design is illustrated through two examples in this section. For comparison, we use models from other reported embedded MPC applications, and both examples are small-scale models with fast dynamics.

5.1. Example 1

As a first example, we consider the control of a rotating antenna, which is modeled by the following equations [20,24,29]:

$$x(k+1) = \begin{bmatrix} 1 & 0.1 \\ 0 & 0.9 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0.0787 \end{bmatrix} u(k)$$

$$y(k) = [\,1 \quad 0\,]x(k)$$

where u is motor voltage, and y is angle of the antenna.

This model uses the input voltage of the motor (volts) to rotate the antenna. Thus, the antenna always meets a predefined set point (i.e., the antenna points toward a moving object in the plane). The control parameters are set as $P=20$ and $M=3$, with the sample time of 0.1. The constraints on the control increment and control input are brought in as $|\Delta u(k)| \le 1$V and $|u(k)| \le 2$V. Thus, QP problem to be solved at each control interval has $n_{dec}=3$ decision variables and $m_c = 12$ constraints. To investigate the performance of the proposed primal–dual algorithm based on CDC, the model is compared with a primal–dual algorithm based on traditional termination criterion. The tolerance in traditional termination criterion is set as $\varepsilon = 10^{-8}$ (a default value for quadprog in Matlab).

Selecting the threshold for convergence depth θ^k_{convg} and degree of progress θ^k_{prog} in CDC is a tradeoff between control performance and computing cost. The threshold θ_1 for the degree of progress is often set in a relatively high value, such as 0.9, to make sure that the optimization process will continue with an available progress margin. With regard to the value of θ_0, the threshold for convergence depth, an appropriate choice of θ_0 should remarkably reduce the computing cost while maintaining the control performance of the MPC controller. We test the performance of MPC with CDC for different values of convergence depth threshold θ_0, the performance is assessed by the following integral error criteria.

$$\text{IAE}: \int_0^\infty |e(t)|dt$$

$$\text{ITAE}: \int_0^\infty t|e(t)|dt$$

$$\text{ISE}: \int_0^\infty e(t)^2 dt$$

Table I. Integral error criteria for different θ_0 values for example 1.

θ_0	IAE	ITAE	ISE	Average time* (ms)
1.0	81.62	358.90	162.72	0.047
0.9	81.62	358.90	162.72	0.043
0.8	81.62	358.91	162.72	0.0399
0.7	81.64	359.02	162.73	0.0369
0.6	81.71	359.57	162.80	0.0327
0.5	81.89	360.70	162.99	0.0304
0.4	82.09	362.35	163.17	0.0294
0.3	83.37	371.34	164.81	0.0248
0.2	83.93	374.13	166.23	0.0238
0.1	91.07	419.80	181.23	0.0202

*The average time is the average runtime of mpc_online function during the whole simulation (200 sampling period).

The results are demonstrated in Table I and Fig. 5.

When $\theta_0 = 0.4$, the control performance for this model is barely affected, whereas the computation time is remarkably reduced (Table I, Fig. 5). That is, the design of MPC with CDC reached a good balance between control performance and computing cost.

The computation speed and control performance can be evaluated in detail based on the parameters discussed previously. Fig. 6 plots the output, the input, the computation time of mpc_online, and the number of QP iterations. Note that the computation time in Fig. 6 not only includes the QP computation time, but also includes other parts in mpc_online. The computation time and number of QP iterations decrease significantly by bringing in the CDC, whereas the closed-loop control performance is almost the same (Fig. 6).

When analyzing the embedded MPC controller efficiency, we focus on two sides during the whole horizon: the average performance and the worst-case performance, which indicates the maximum execution time to calculate the MPC control moves. The QP computation time and QP iteration count on average and under the worst case as we change the control horizon M from 3 to 10 are shown in Figs 7 and 8. By introducing CDC, the execution time is reduced by more than 1/2 on average, and no less than 1/3 even under the worst cases. With regard to the number of QP iterations, CDC has a significant impact to allow the optimization process to stop iteration early and properly. In practice, the convergence depth threshold θ_0 can be user-determined. Thus, a good balance can be maintained between the performance and the cost.

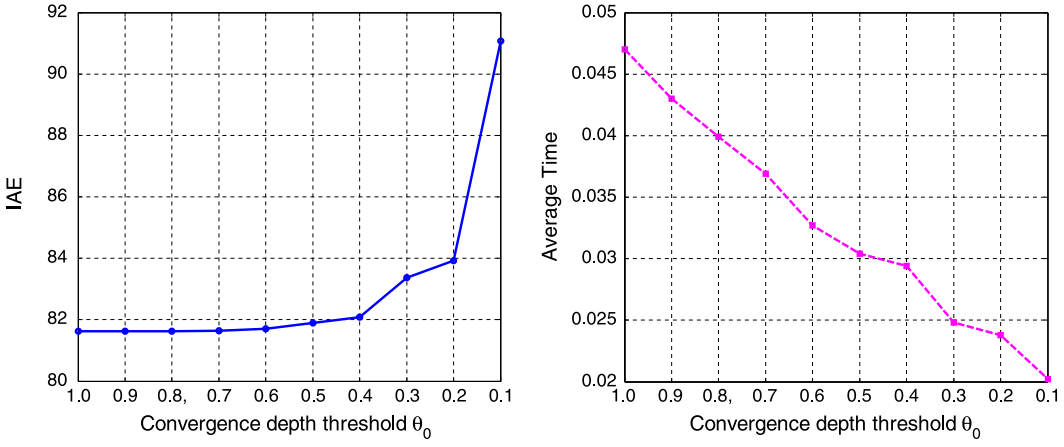


Figure 5. The control performance (IAE) and computing cost (runtime) trade off with different θ_0 values.

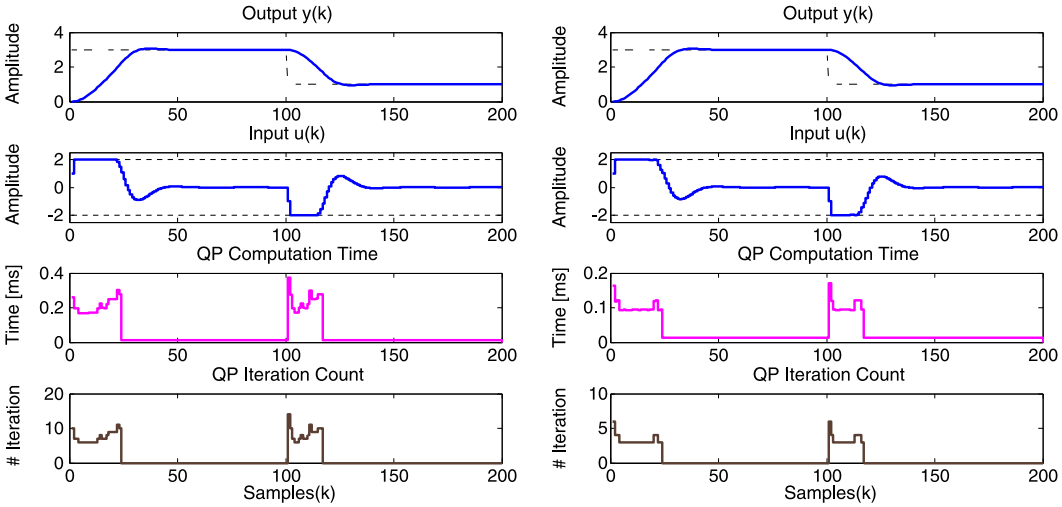


Figure 6. HIL results based on traditional termination criterion (left) and based on CDC (right).

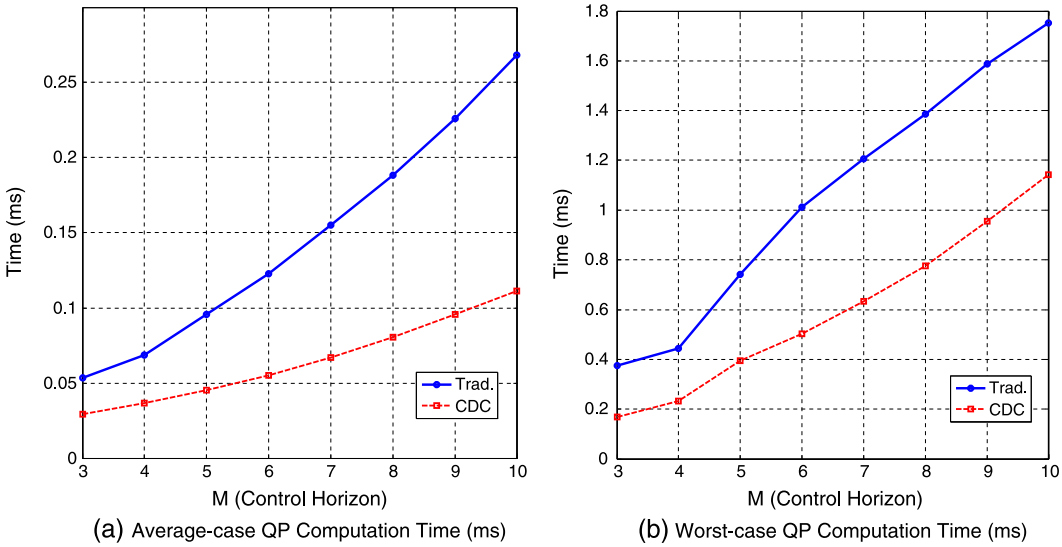


Figure 7. QP computation time on average and under worst case.

5.2. Example 2

In the second example, we consider the inverted pendulum on a moving cart, which required high speed control to keep balance [20]. The following nonlinear equations describe the dynamics of the inverted pendulum:

$$\ddot{y}_m = \frac{\frac{F_m}{m} + l{\theta_m}^2 \sin\theta_m - g\sin(\theta_m \cos\theta_m)}{\frac{M}{m} + \sin^2\theta_m}$$

$$\ddot{\theta}_m = \frac{\frac{-F}{m}\cos\theta_m + \frac{M+m}{mg\sin\theta_m} - l{\theta_m}^2 \sin\theta_m \cos\theta_m}{l\left(\frac{M}{m} + \sin^2\theta_m\right)}$$

where F_m is the force applied to the cart in newtons (N), y_m is the cart position in meters (m), and θ_m is the angle of the pendulum from vertical in radians (rad). The model parameters are listed in Table II.

In this example, controlled variables are y_m and θ_m, and manipulated variable is F_m. The linearized model is discretized at a sampling rate of 20 Hz. Thus, we acquire a discrete linear state-space model as follows:

$$x(k+1) = \begin{bmatrix} 1 & 0.05 & -0.0057 & -0.000094883 \\ 0 & 1 & -0.2308 & -0.0057 \\ 0 & 0 & 1.0593 & 0.051 \\ 0 & 0 & 2.3968 & 1.0593 \end{bmatrix} x(k) + \begin{bmatrix} 0.0028 \\ 0.1106 \\ -0.0091 \\ -0.3674 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x(k)$$

where $u = F_m$ and $y = [\,y_m \quad \theta_m\,]^T$.

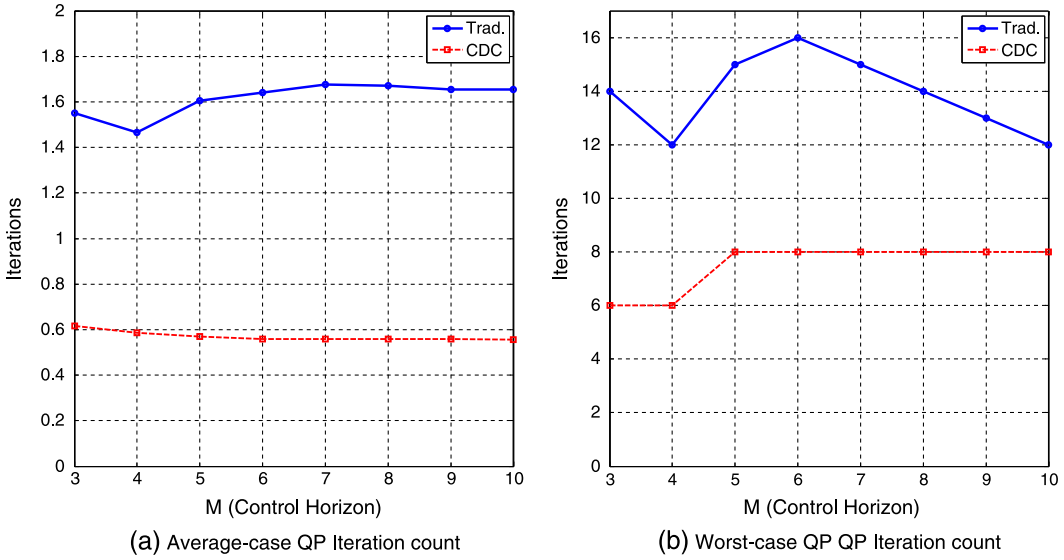


Figure 8. Number of QP iterations on average and under worst case.

Table II. Parameter for Example 2.

Cart mass (M)	=	0.455	kg
Pendulum mass (m)	=	0.21	kg
Distance to the center of mass of pendulum (l)	=	0.305	m
Gravitational constant (g)	=	9.81	m/s^2

Similarly to the literature [20], controller parameters M and P are set as 5 and 25. Note that the weighting matrix Q is set as diag[1.5, 0] for y_m and θ_m pair in the whole prediction horizon. Thus, θ_m is not controlled, but only constrained. The constraints for manipulate variable u is ± 5 N,the constraint for cart position y_m is ± 2 m and the constraint for cart pendulum angle θ_m is ± 45° (± 0.79 rad). Thus, the QP problem to be solved at each control interval has $n_{dec}=5$ decision variables and $m_c=110$ constraints.

The HIL framework introduced previously is used for simulation to observe the control performance. The nonlinear equations are solved in Matlab at every sample interval to simulate a real object, and the linear model is used to implement the embedded MPC controller on DSP. The tolerance in traditional termination criterion is set as $\varepsilon=10^{-8}$ and the convergence depth threshold is set as $\theta_0=0.5$ to maintain the balance between computing speed and control

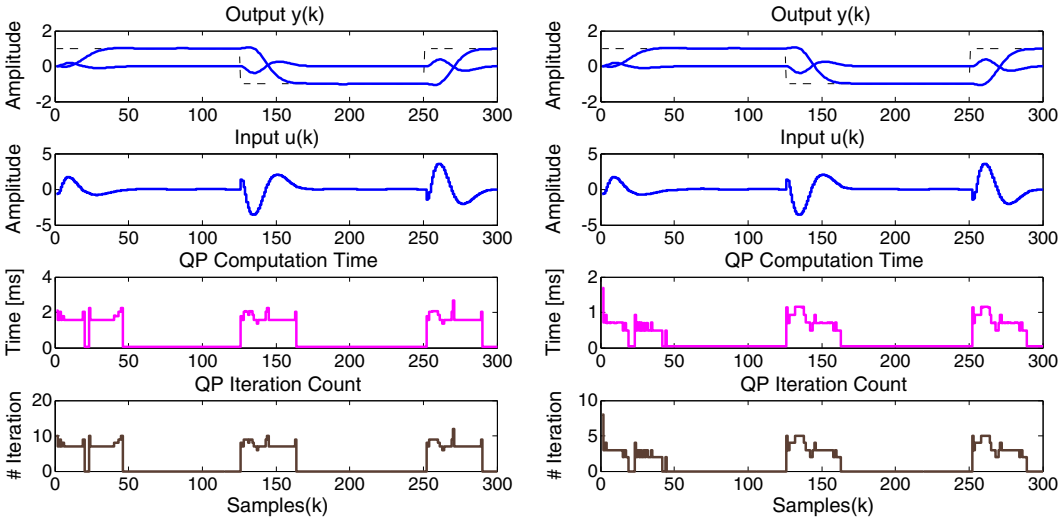


Figure 9. HIL results based on traditional termination criterion (left) and based on CDC (right).

performance. The control performance, computation time of mpc_online, and number of QP iterations based on traditional criterion and CDC are illustrated in Fig. 9. The import of CDC significantly reduced the QP iterations and computation time with little effects on its control performance (Fig. 9).

Based on the above case, the advantage of CDC is illustrated through the comparison with traditional termination criterion on the QP computation time and the QP iteration count both on average and under the worst cases during the whole simulation horizon (Figs 10 and 11).

The algorithm based on CDC outperforms that based on traditional termination criterion in terms of average runtime by more than 50 % on the embedded platform. Even under the worst cases, the computation time is reduced by more than 30 %. A similar observation can be found in the comparison of the number of QP iterations. We can conclude from the performance of CDC in both the rotating antenna model and the inverted pendulum model that CDC can significantly reduce the QP computation cost without degrading the control performance on DSP, which can be beneficial in practical real-time applications. The main

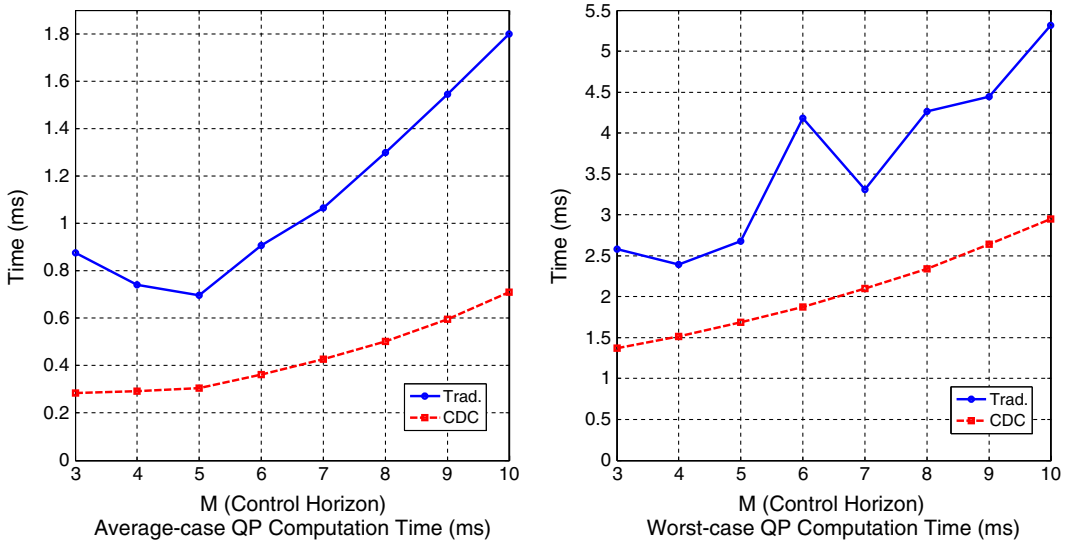


Figure 10. QP computation time on average and under worst case.

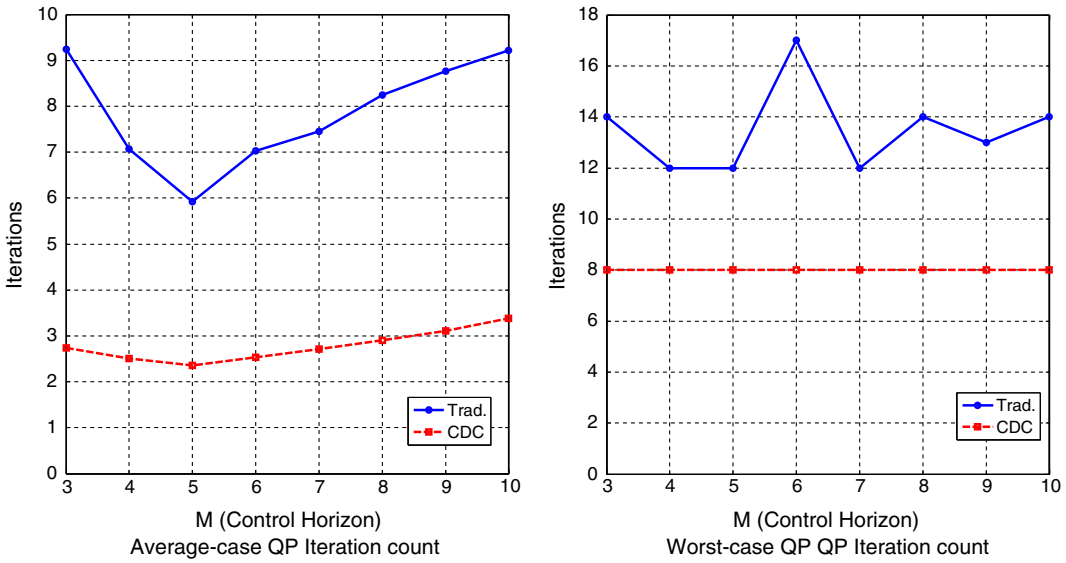


Figure 11. Number of QP iterations on average and under worst case.

advantage of the result is that we create a practical framework for embedded MPC and illustrate its superiority by introducing CDC into the optimization algorithm. Compared with other studies on embedded MPC, our approach offers a more general framework and sheds light on some details in implementation. Note that the MPC computation time needed in both examples are much shorter than the sampling period, which guarantees the controller's performance at corresponding frequency. The success of CDC in our design might open up a new way for others who want to accelerate the optimization algorithm in embedded applications.

VI. CONCLUSION

In embedded MPC applications, such as a motion system, QP solving speed is a key factor for the performance of the controller. This paper introduces the convergence depth control method into the interior-point method to accelerate the QP solving process while maintaining satisfactory control performance. Notably, the CDC method, which is originally designed for large-scale optimization problems in process control, is introduced here for small and fast applications. Besides, a general framework for embedded MPC is designed and a prototype system is implemented on a DSP module. The competence of the design is verified on two models via the HIL method, making an MPC controller promising enough in embedded and real-time applications.

Several extensions can be made based on existing achievements. The performance of the controller on real processes should be further studied. The application of CDC on other algorithms, such as the gradient method, could also be studied.

REFERENCES

1. Qin, S. J. and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Eng. Practice*, Vol. 11, No. 7, pp. 733–764 (2003).
2. Bemporad, A., M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, Vol. 38, No. 1, pp. 3–20 (2002).
3. Johansen, T. A., W. Jackson, R. Schreiber, and P. Tondel, "Hardware synthesis of explicit model predictive controllers," *IEEE Trans. Control Syst. Technol.*, Vol. 15, No. 1, pp. 191–197 (2007).
4. Bemporad, A. and C. Filippi, "Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming," *J. Optim. Theory Appl.*, Vol. 117, No. 1, pp. 9–38 (2003).
5. Summers, S., C. N. Jones, J. Lygeros, and M. Morari, "A multiresolution approximation method for fast explicit model predictive control," *IEEE Trans. Autom. Control*, Vol. 56, No. 11, pp. 2530–2541 (2011).
6. Johansen, T. A. and A. Grancharova, "Approximate explicit constrained linear model predictive control via orthogonal search tree," *IEEE Trans. Autom. Control*, Vol. 48, No. 5, pp. 810–815 (2003).
7. Ferreau, H. J., H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *Int. J. Robust Nonlinear Control*, Vol. 18, No. 8, pp. 816–830 (2008).
8. Zeilinger, M. N., C. N. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization," *IEEE Trans. Autom. Control*, Vol. 56, No. 7, pp. 1524–1534 (2011).
9. Wright, S. J., "Applying new optimization algorithms to model predictive control," *Fifth Int. Conf. Chemical Process Control*, Lake Tahoe, CA (1996).
10. Rao, C. V., S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *J. Optim. Theory Appl.*, Vol. 99, No. 3, pp. 723–757 (1998).
11. Bartlett, R. A., L. T. Biegler, J. Backstrom, and V. Gopal, "Quadratic programming algorithms for large-scale model predictive control," *J. Process Control*, Vol. 12, No. 7, pp. 775–795 (2002).
12. Domahidi, A., A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," *Proc. 51st IEEE Conf. Dec. Control* (No. EPFL-CONF-181938) (2012).
13. Patrinos, P., P. Sopasakis, and H. Sarimveis, "A global piecewise smooth Newton method for fast large-scale model predictive control," *Automatica*, Vol. 47, No. 9, pp. 2016–2022 (2011).
14. Patrinos, P. and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Trans. Autom. Control*, Vol. 59, No. 1, pp. 18–33 (2014).
15. He, M. and K. V. Ling, "Model predictive control on a chip," *IEEE Int. Conf. Contr. Autom.*, Vol. 1, pp. 528–532.
16. Wills, A. G., G. Knagge, and B. Ninness, "Fast linear model predictive control via custom integrated circuit architecture," *IEEE Trans. Control Syst. Technol.*, Vol. 1, No. 20, pp. 59–71 (2012).
17. Jerez, J. L., G. A. Constantinides, E. C. Kerrigan, and K. V. Ling, "Parallel MPC for real-time FPGA-based implementation," *Proc. 18th IFAC World Congr.*, Vol. 18, pp. 1338–1343 (2011).
18. Jerez, J. L., K. V. Ling, G. A. Constantinides, and E. C. Kerrigan, "Model predictive control for deeply pipelined field-programmable gate array implementation: algorithms and circuitry," *IET Control Theory Appl.*, Vol. 6, No. 8, pp. 1029–1041 (2012).

19. Lu, Y., D. Li, Z. Xu, and Y. Xi, "Convergence analysis and digital implementation of a discrete-time neural network for model predictive control," *IEEE Trans. Ind. Electron*, Vol. 61, No. 12, pp. 7035–7045 (2014).
20. Currie, J., A. Prince-Pike, and D. I. Wilson, "Auto-code generation for fast embedded Model Predictive Controllers," *IEEE 19th Int. Conf. Mechatronics and Machine Vision in Practice (M2VIP)*, pp. 116–122 (2012).
21. Patrinos P., A. Guiggiani, and A. Bemporad, "Fixed-point dual gradient projection for embedded model predictive control," *European Control Conf. (ECC)*, pp. 3602–3607 (2013).
22. Wang Y. and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, Vol. 18, No. 2, pp. 267–278 (2010).
23. Nocedal, J. and S. J. Wright, *Numerical Optimization*, Springer Press, London (1999).
24. Vouzis, P. D., M. V. Kothare, L. G. Bleris, and M. G. Arnold, "A system-on-a-chip implementation for embedded real-time model predictive control," *IEEE Trans. Control Syst. Technol.*, Vol. 17, No. 5, pp. 1006–1017 (2009).
25. Chen, W., Z. Shao, K. Wang, X. Chen, and L. T. Biegler, "Convergence depth control for interior point methods," *AICHE J.*, Vol. 56, No. 12, pp. 3146–3161 (2010).
26. Lau, M. S., S. P. Yue, K. V. Ling, and J. M. Maciejowski, "A comparison of interior point and active set methods for FPGA implementation of model predictive control," *Proc. European Control Conf.*, pp. 157–161 (2009).
27. Goldberg, D., "What every computer scientist should know about floating-point arithmetic", *ACM Comput. Surv. (CSUR)*, Vol. 23, No. 1, pp. 5–48 (1991).
28. Higham, N. J., *Accuracy and stability of numerical algorithms*, Siam, Philadelphia (2002).
29. Kothare, M. V., V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, Vol. 32, No. 10, pp. 1361–1379 (1996).
30. Jerez, J. L., P. J. Goulart, S. Richter, G. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Trans. Autom. Control*, Vol. 59, No. 12, pp. 3238–3251 (2014).

VII. APPENDIX

Here we give a practical interior-point method algorithm with CDC, in which the choice of σ is based on a heuristic method from [23].

Algorithm I: Interior Point Method with CDC

Choose $(z_0, s_0, \lambda_0)$with $(s_0, \lambda_0) > 0$

Calculate $\mu = s^T\lambda/m_c$;

for k = 0, 1, 2 …

$\Lambda \leftarrow \mathrm{diag}(\lambda^k)$, $\mathcal{S} \leftarrow \mathrm{diag}(s^k)$;

Set $(z, s, \lambda) = (z_k, s_k, \lambda_k)$Solve 8 for (**Δ**z, **Δ**s, **Δ**λ);

Calculate $\hat{\alpha} = \max\{\alpha \in (0,1) | (s,\lambda) + \alpha(\mathbf{\Delta}s,\ \mathbf{\Delta}\lambda) \geq \mathbf{0}\}$;

Set $(z_{k+1},\ s_{k+1},\ \lambda_{k+1}) = (z_k,\ s_k,\ \lambda_k) + \hat{\alpha}\,(\mathbf{\Delta}z,\ \mathbf{\Delta}s,\ \mathbf{\Delta}\lambda)$

$\mu_{old} \leftarrow \mu$;

Calculate $\mu = \frac{s^T\lambda}{m_c}$;

Set centering parameter to $\sigma = \min\left[\left(\frac{\mu}{\mu_{\mathrm{old}}}\right)^3, 0.99999\right]$

if (CDC converged)

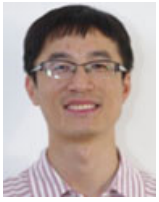
Terminate with $z^* = z_k$

else if (CDC underconverged)

Report error with original problem.

end (for)

Yi Ding received his BS degree in control engineering from Zhejiang University in 2013. He is currently a master student at the college of Control Science and Engineering, Zhejiang University. His research interests include fast model predictive control, embedded model predictive control, and numerical optimization.



Zuhua Xu was born in Hangzhou, China. He received his BS degree from Zhejiang University of Technology in 1999 and his PhD from Zhejiang University in 2004. Now he is Associate Professor at Zhejiang University. His research interests include model predictive control and iterative learning control.

Jun Zhao received a masters degree in the Chemical Engineering Department of Zhejiang University in 1996. He received his PhD in Control Science and Engineering from Zhejiang University in 2000, and is now an associate professor at Zhejiang University. His main research fields include: model predictive control, system identification, chemical process modelling, and applications.

Kexin Wang received her MS degree in 2004 from the School of Computer Science and Technology, Shandong University, China, and PhD in 2008 from the Department of Control Science and Engineering, Zhejiang University, China. Her research interests include the theory and applications of dynamic optimization and nonlinear programming.

Zhijiang Shao received his BS degree in 1992 from the Department of Chemical Engineering, Zhejiang University, China, and PhD in 1997 from the College of Control Science and Engineering, Zhejiang University, China. He was a lecturer at Zhejiang University from 1997 to 1999, and an Associate Professor from 1999 to 2004 and is now a Professor. His research interests include large-scale optimization, real-time optimization, process system optimization, and dynamic optimization.