2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

An Embedded MPC for High-Speed Trajectory Tracking of Piezoelectric Actuators with Implementation on an ARM Microprocessor

Fei Dong, Wenguang Zuo, Hongyang Xie, Xinyu Wang, Qinglei Hu*

Abstract— **This work studies the problem of high-speed trajectory tracking for a piezoelectric actuator (PEA) subject to constraints on both the displacement output and control input via a resource-limited embedded processor. To ensure tracking performance, an extremely high control frequency of 10kHz is recommended. In response, we propose an embedded model predictive control (MPC) algorithm and deploy it on a low-cost STM32F407 ARM microprocessor operating at 168MHz. Specifically, a fast coordinate ascent algorithm is designed to online solve the constrained quadratic programming (QP) problem derived from the MPC, and a delayed Kalman filter (KF) is introduced to compensate for the computation latency. Through hardware-in-the-loop (HIL) simulations, we demonstrate the feasibility and effectiveness of our embedded MPC algorithm.**

I. Introduction

Piezoelectric actuators (PEAs) possess high precision, ultra-fine resolution, and an immediate response [1], making them ideal for use in various high-precision systems such as nano positioning stages [2] and beam steering mirrors [3]. In these applications, PEAs serve the critical function of precisely tracking trajectories. However, due to the extremely short control period, it remains a significant challenge to track a high-speed reference trajectory using a resource-limited embedded platform such as an advanced RISC machine (ARM).

By its simplicity of deployment, the PID control is widely used in embedded applications. In [4], a PI control is implemented onto a digital signal processor (DSP) to reduce the tracking error. The Physik Instrumente also employs the PID control in its E-727 digital piezo controller, as indicated in the user manual [5]. However, the PID control is only suitable for low-order systems, which severely limits its closed-loop bandwidth. To cope with the high-order dynamics of PEA, Kong et al. present a backstepping control with an adaptive neural network [6], Chowdhury et al. propose a feedback linearization method with an extended high-gain observer [7], and Habibullah et al. design a linear quadratic regulator (LQR) with a Kalman filter (KF) [8]. However, none of these methods addresses the constraints on state and input during the control design process.

Model predictive control (MPC) is an effective approach for handling physical constraints and exploiting future reference trajectories. However, it requires solving a complex numerical optimization problem in each sampling period and thus results in a huge computational burden. As a result, most of the literature can only solve the MPC optimization problem by SIMULINK on the high-performance platform. For example, Ref. [9] adopts the Hildreth's quadratic programming (QP) method on a host computer, while Ref. [10] applies the gradient descent method on a workstation but dose not consider the constraints.

The implementation of MPC on resource-limited embedded processors is an area of significant interest, not only in the field of PEA control. Numerous commercial or open-source solvers have been developed, such as `OSQP` [11], `HPIPM` [12], and `acados` [13]. Notably, `acados` includes the first two solvers and offers user-friendly interfaces for MATLAB and Python. Ref. [13] also demonstrates the effectiveness on a dSPACE platform, leveraging the high-performance linear algebra package `BLASFEO`.

In this work, we consider the constraints on both PEA's states and inputs to prevent damage and propose an STM32-based MPC method to regulate it to track a high-speed reference trajectory. To ensure reliable tracking performance, a high control frequency of 10kHz is required. Thus, the constrained MPC optimization problem must be solved within the sampling time of 0.1ms on the resource-limited embedded processor. To this end, we design a coordinate ascent algorithm that can efficiently solve the optimization problem without relying on third-party libraries. Furthermore, we introduce a delayed KF to compensate for the computation latency and deploy the proposed method on a low-cost STM32F407 ARM microprocessor and carry out hardware-in-the-loop (HIL) simulations for performance validation.

The remainder of this work is given as follows. In Section II, we formulate the trajectory tracking problem by introducing the PEA's dynamics and constraints. In Section III, we pose an MPC problem and convert it into a constrained QP

This work was supported in apart by the National Natural Science Foundation of China (62203027 and 62227812); in part by the China Postdoctoral Science Foundation (BX20220369 and 2022M710310); in part by the Tianmushan Laboratory Research Project (TK-2023-B-010 and TK-2023-C-020); and in part by a grant from the National Key Laboratory of Aerospace Flight Dynamics (AFDL), Northwestern Polytechnical University. (*Corresponding author: Qinglei Hu*)

F. Dong is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China, and also with the Tianmushan Laboratory, Hangzhou, 310023, China.

W. Zuo and H. Xie are with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China.

X. Wang is with the School of Automation Science and Electrical Engineering and the Shen Yuan Honors College, Beihang University, Beijing 100191, China.

Q. Hu is with the Tianmushan Laboratory, Hangzhou, 310023, China, and also with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China. huql_buaa@buaa.edu.cn.

problem. Then, we propose an embedded coordinate ascent solution algorithm in Section IV. Moreover, we use HIL simulations to validate its performance and draw conclusions in Sections V and VI, respectively.

II. Problem Formulation

In this work, we require a PEA to track a high-speed reference trajectory while satisfying the constraints on both the displacement output and control input.

A. The SISO transfer model of PEA

As [2], [8], [14], the PEA's dynamics is usually modeled as the following single-input and single-output (SISO) system

$$\frac{Y(z)}{U(z)} = \frac{b_{n-1}z^{n-1} + \cdots + b_1 z + b_0}{z^n + a_{n-1}z^{n-1} \ldots + a_1 z + a_0}, \tag{1}$$

where z denotes the time delay operator, e.g., z^{-k} represents a shift of k sampling periods ahead.

To better capture the dynamical characteristics of the PEA, a 5-th order continuous-time model is identified in [2], which has a resonant frequency greater than 1kHz.

B. The objective of trajectory tracking control

Given a feasible reference trajectory $\{r(k)\}_{k=0}^{\infty}$, the objective of this work is to regulate the displacement output $y(k)$ of (1) to track it asymptotically, i.e., the deviation between $y(k)$ and $r(k)$ is expected to be zero as time goes to infinity.

$$\lim_{k\to\infty} |y(k) - r(k)| = 0. \tag{2}$$

To avoid damaging the PEA, the limits on both the displacement output and control input are taken into account, i.e.,

$$y(k) \in [y_{\min}, y_{\max}] \text{ and } u(k) \in [u_{\min}, u_{\max}] \tag{3}$$

where $y_{\min}$, $y_{\max}$, $u_{\min}$, and $u_{\max}$ are constants. Moreover, the control frequency is required to be 10kHz on a resource-limited embedded ARM microprocessor.

III. Model Predictive Control

To achieve the objective (2) under the constraints (3), the MPC is a natural alternative, which generates control inputs by fully exploiting the PEA dynamics and reference trajectories.

A. The MPC optimization problem

For reducing the computational burden, we first convert the SISO model in (1) into the following state space form as [15]

$$\begin{aligned} \boldsymbol{x}(k+1) &= A\boldsymbol{x}(k) + B(u(k) + w(k)), \\ y(k) &= C\boldsymbol{x}(k) + v(k), \end{aligned} \tag{4}$$

where the state matrix $A \in \mathbb{R}^{n\times n}$, input matrix $B \in \mathbb{R}^{n\times 1}$, and output matrix $C \in \mathbb{R}^{1\times n}$ are given by

$$A = \left[\begin{array}{c:ccc} -a_{n-1} & 1 & & \\ \vdots & & \ddots & \\ -a_1 & & & 1 \\ \hdashline -a_0 & 0 & \cdots & 0 \end{array}\right], \; B = \begin{bmatrix} b_{n-1} \\ \vdots \\ b_1 \\ b_0 \end{bmatrix},$$
$$C = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}.$$

The Gaussian white noises $w(k) \sim \mathcal{N}(0, R_w)$ and $v(k) \sim \mathcal{N}(0, R_v)$ are introduced to model the uncertainties of (4).

Then, we define the following value function

$$J(u(\cdot|k)) = \frac{1}{2}\sum_{i=0}^{N_p} (y(i|k) - r(i|k))^2 + \frac{\rho}{2}\sum_{i=0}^{N_p-1} (\Delta u(i|k))^2,$$

where $u(\cdot|k) = \{u(0|k), \ldots, u(N_p-1|k)\}$ is the input sequence to optimize with the prediction horizon length N_p. Note that we penalize the input increment $\Delta u(i|k) \triangleq u(i|k) - u(i-1|k)$ instead of the input $u(i|k)$ to reduce the tracking error. Thus, the goal is to realize that $|y(i|k) - r(i|k)| = |\Delta u(i|k)| = 0$.

Based on (4), the constrained optimization problem is formulated as follows

$$u^*(\cdot|k) = \underset{u(\cdot|k)}{\operatorname{argmin}} J(u(\cdot|k)) \tag{5a}$$
$$\text{s.t.} \quad \boldsymbol{x}(0|k) = \boldsymbol{x}(k) \tag{5b}$$
$$u(-1|k) = u(k-1), \tag{5c}$$
$$\boldsymbol{x}(i+1|k) = A\boldsymbol{x}(i|k) + Bu(i|k), \tag{5d}$$
$$y(i|k) = C\boldsymbol{x}(i|k), \tag{5e}$$
$$u(i|k) = u(i-1|k) + \Delta u(i|k), \ \forall i < N_c, \tag{5f}$$
$$u(i|k) = u(i-1|k), \ \forall i \geq N_c - 1, \tag{5g}$$
$$y(i|k) \in [y_{\min}, y_{\max}], \tag{5h}$$
$$u(i|k) \in [u_{\min}, u_{\max}], \tag{5i}$$

where (5b) is the state initialization, and (5c) is the input initialization since the cost is on the increment $\Delta u(i|k)$ with the relation in (5f). Moreover, a control horizon with $1 \leq N_c \leq N_p$ is introduced to reduce the dimension of decision variable $u(\cdot|k)$ by (5g).

By solving (5), we obtain the optimal input sequence $u^*(\cdot|k)$. However, only its first component is applied to the PEA to regulate the displacement $y(k)$ to track the reference command $r(k)$ and satisfy the constraints on both the state and input, i.e., $u(k) = u^*(0|k)$. Then, the above process is repeated in the moving horizon manner to improve the tracking performance of reference $\{r(i)\}_{i=k}^{\infty}$.

B. The quadratic programming for solving (5)

To solve (5), we first define the vector variables as follows

$$\begin{aligned} \boldsymbol{y} &= \begin{bmatrix} y(1|k) & \cdots & y(N_p|k) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{N_p\times 1}, \\ \boldsymbol{u} &= \begin{bmatrix} u(1|k) & \cdots & u(N_p|k) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{N_c\times 1}, \\ \boldsymbol{r} &= \begin{bmatrix} r(1|k) & \cdots & r(N_p|k) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{N_p\times 1}, \\ \Delta\boldsymbol{u} &= \begin{bmatrix} \Delta u(0|k) & \cdots & \Delta u(N_c-1|k) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{N_c\times 1}. \end{aligned}$$

Their time indexes are omitted for brevity. Then, the future N_p-step prediction of $y(k)$ is directly given by

$$\boldsymbol{y} = F\boldsymbol{x}(k) - \Phi\boldsymbol{u} \tag{6}$$

where the transition matrices F and Φ are obtained from (7). Moreover, the control input increment $\Delta\boldsymbol{u}$ can be obtained

$$F=\begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix} \in \mathbb{R}^{N_p\times n},\quad \Phi=\left[\begin{array}{ccc:c} CB & & & \\ CAB & CB & & \\ \vdots & \vdots & \ddots & \\ CA^{N_c-1}B & CA^{N_c-2}B & \cdots & CB \\ \hdashline CA^{N_c}B & CA^{N_c-1}B & \cdots & CAB+CB \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & \cdots & CA^{N_p-N_c}B+\cdots+CAB+CB \end{array}\right] \in \mathbb{R}^{N_p\times N_c}. \tag{7}$$

by the decision vector $\boldsymbol{u}$ and the previous input $u(k-1)$ as

$$\Delta\boldsymbol{u}=\Psi\boldsymbol{u}-\Gamma u(k-1) \tag{8}$$

where

$$\Psi=\begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}\in\mathbb{R}^{N_c\times N_c},\ \Gamma=\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}\in\mathbb{R}^{N_c\times 1}.$$

Based on (6) and (8), we rewrite $J(u(\cdot|k))$ in (5a) into that

$$\begin{aligned} J(\boldsymbol{u}) &= \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{r}\|_2^2+\frac{\rho}{2}\|\Delta\boldsymbol{u}\|_2^2 \\ &= \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}H\boldsymbol{u}+\boldsymbol{h}^{\mathrm{T}}\boldsymbol{u}+h \end{aligned} \tag{9}$$

where

$$\begin{aligned} &H=\Phi^{\mathrm{T}}\Phi+\rho\Psi^{\mathrm{T}}\Psi\in\mathbb{R}^{N_c\times N_c}, \\ &\boldsymbol{h}=\Phi^{\mathrm{T}}\left(F\boldsymbol{x}(k)-\boldsymbol{r}\right)-\rho\Psi^{\mathrm{T}}u(k-1)\in\mathbb{R}^{N_c\times 1}, \\ &h=\frac{1}{2}\|F\boldsymbol{x}(k)-\boldsymbol{r}\|_2^2+\frac{\rho}{2}u^2(k-1)\in\mathbb{R}. \end{aligned} \tag{10}$$

Thus, the MPC problem (5) is converted into a quadratic programming (QP) problem as follows

$$\boldsymbol{u}^*=\underset{\boldsymbol{u}}{\operatorname{argmin}}\ \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}H\boldsymbol{u}+\boldsymbol{h}^{\mathrm{T}}\boldsymbol{u},\quad \text{s.t.}\quad W\boldsymbol{u}\le\boldsymbol{b}, \tag{11}$$

where the matrix $W\in\mathbb{R}^{2(N_p+N_c)\times N_c}$ and vector $\boldsymbol{b}\in\mathbb{R}^{2(N_p+N_c)\times 1}$ are given by

$$W=\begin{bmatrix} \Phi \\ -\Phi \\ I \\ -I \end{bmatrix}\quad\text{and}\quad \boldsymbol{b}=\begin{bmatrix} Y_{\max}-F\boldsymbol{x}_k \\ -Y_{\min}+F\boldsymbol{x}_k \\ U_{\max} \\ -U_{\min} \end{bmatrix}$$

with $Y_{\max}=y_{\max}\cdot\mathbf{1}_{N_p}$, $Y_{\min}=y_{\min}\cdot\mathbf{1}_{N_p}$, $U_{\max}=u_{\max}\cdot\mathbf{1}_{N_c}$, and $U_{\min}=u_{\min}\cdot\mathbf{1}_{N_c}$. Note that the scalar term h in (9) is omitted since it has no effect on the optimal solution $\boldsymbol{u}^*$.

IV. STM32-based MPC and its Implementation

Although there are many optimization solvers, e.g. the `quadprog` in MATLAB, it is still hard to solve (11) on a resource-limited STM32 ARM microprocessor in the sampling time of 0.1ms. Thus, an embedded model predictive control approach is proposed as shown in Fig. 1, which includes a coordinate ascent method for fast solving the constrained optimization problem and a delayed KF to compensate for the computation latency.

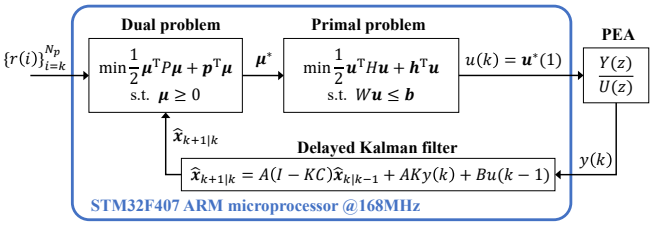

Fig. 1. Embedded MPC approach for PEA.

A. The coordinate ascent algorithm

Instead of using the open source solvers, e.g., `acados` in [13] which relies on the high-performance linear algebra package `BLASFEO`, a coordinate ascent solution algorithm is designed by the following Lagrangian function.

$$L(\boldsymbol{u},\boldsymbol{\mu})=\frac{1}{2}\boldsymbol{u}^{\mathrm{T}}H\boldsymbol{u}+\boldsymbol{h}^{\mathrm{T}}\boldsymbol{u}+\boldsymbol{\mu}^{\mathrm{T}}(W\boldsymbol{u}-\boldsymbol{b}) \tag{12}$$

where $\boldsymbol{\mu}\in\mathbb{R}^{2(N_p+N_c)\times 1}$ is a Lagrange multiplier vector to penalize the inequality constraint of (11).

It follows from $\nabla_{\boldsymbol{u}}L(\boldsymbol{u},\boldsymbol{\mu})=H\boldsymbol{u}+\boldsymbol{h}+W^{\mathrm{T}}\boldsymbol{\mu}=0$ that

$$\boldsymbol{u}=-H^{-1}\left(\boldsymbol{h}+W^{\mathrm{T}}\boldsymbol{\mu}\right). \tag{13}$$

By inserting (13) into (12), we obtain the dual problem of (11) as follows

$$\boldsymbol{\mu}^*\triangleq\underset{\boldsymbol{\mu}}{\operatorname{argmin}}\ \frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}P\boldsymbol{\mu}+\boldsymbol{p}^{\mathrm{T}}\boldsymbol{\mu}, \tag{14a}$$

$$\text{s.t.}\quad \boldsymbol{\mu}\ge 0, \tag{14b}$$

where the weighting matrix $P\in\mathbb{R}^{2(N_p+N_c)\times 2(N_p+N_c)}$ and vector $\boldsymbol{p}\in\mathbb{R}^{2(N_p+N_c)}\times 1$ are given by

$$P=WH^{-1}W^{\mathrm{T}}\ \text{and}\ \boldsymbol{p}=\boldsymbol{b}+WH^{-1}\boldsymbol{h}. \tag{15}$$

Compared with solving the primal problem in (11) directly, it is more efficient to obtain the optimal solution $\boldsymbol{\mu}^*$ to the dual problem in (14) first, since the constraint $\boldsymbol{\mu}\ge 0$ is simple. Moreover, both H and P are symmetric and positive definite by (10) and (15), which implies that the unconstrained function in (14a) is strictly convex along the element of $\boldsymbol{\mu}$. Thus, the coordinate ascent method is adopted to iteratively solve (14) with an explicit form [16, Proposition

Algorithm 1 Coordinate ascent algorithm for solving (11).

1) **Input:** the reference trajectory $\boldsymbol{r}$, current state $\boldsymbol{x}(k)$, and last control input $u(k-1)$.
2) **Output:** the optimal solution $\boldsymbol{u}$.
   - Let $\boldsymbol{\mu} = \mathbf{0}_{2(N_p+N_c)\times 1}$.
   - Compute $\boldsymbol{h}$ and $\boldsymbol{b}$ of the QP problem in (11).
   - **For** $m = 1, \ldots, M$:
     - **For** $i = 1, \ldots, 2(N_p + N_c)$:
       - $\boldsymbol{\mu}_i = \max\left(0, \boldsymbol{\mu}_i - P_{ii}^{-1}\left(\boldsymbol{p}_i + P_{i:}\boldsymbol{\mu}\right)\right)$.
     - **End for**.
   - **End for**.
   - **Return** $\boldsymbol{u} = -H^{-1}\left(\boldsymbol{h} + W^{\mathrm{T}}\boldsymbol{\mu}\right)$.

3.7.1].

Denote the i-th coordinate of the vector $\boldsymbol{\mu}$ by $\boldsymbol{\mu}_i$. Then, the partial derivative of (14a) with respect to $\boldsymbol{\mu}_i$ is directly obtained as follows

$$\boldsymbol{p}_i + P_{ii}\boldsymbol{\mu}_i + \sum_{j\neq i} P_{ij}\boldsymbol{\mu}_j, \tag{16}$$

where $\boldsymbol{p}_i$ is the i-th element of the vector $\boldsymbol{p}$, and P_{ij} represents the element in row i and column j of the matrix P. Thus, the minimum of (14a) along $\boldsymbol{\mu}_i$ is attained at $\widehat{\boldsymbol{\mu}}_i$ when the partial derivative in (16) is zero. That is

$$\begin{aligned}\widehat{\boldsymbol{\mu}}_i &= -\frac{1}{P_{ii}}\left(\boldsymbol{p}_i + \sum_{i\neq j} P_{ij}\boldsymbol{\mu}_j\right)\\ &= \boldsymbol{\mu}_i - P_{ii}^{-1}\left(\boldsymbol{p}_i + P_{i:}\boldsymbol{\mu}\right),\end{aligned} \tag{17}$$

where $P_{i:}$ is the i-th row of the matrix P.

Considering the constraint $\boldsymbol{\mu}_i \geq 0$, the coordinate ascent updating for $\boldsymbol{\mu}_i$ is performed as follows

$$\boldsymbol{\mu}_i = \max\left\{0,\ \widehat{\boldsymbol{\mu}}_i\right\}. \tag{18}$$

Thus, the optimal solution $\boldsymbol{\mu}^*$ can be obtained by recurrently executing the coordinate ascent operation in (18). By the expression in (13), the optimal solution $\boldsymbol{u}^*$ to the primal problem in (11) satisfies that

$$\boldsymbol{u}^* = -H^{-1}\left(\boldsymbol{h} + W^{\mathrm{T}}\boldsymbol{\mu}^*\right). \tag{19}$$

The solving process is summarized in Algorithm (1) with a maximum number M of iterations. The matrices F, Φ, Ψ, Γ, W, H and P are time-invariant for given parameters N_p, N_c, and ρ. Thus, they are computed before the iteration loop in Algorithm (1) to reduce the computation latency.

B. Numerical example in MATLAB

For an illustration, we directly discrete the model in [2] with a sampling time of $\tau = 0.0001$s, and obtain the parameters for the SISO model in (1) as: $b_0 = 0.2536$, $b_1 = 0.3351$, $b_2 = 0.2654$, $b_3 = 0.8734$, $b_4 = 0.9523$, $a_0 = -0.09212$, $a_1 = 0.1754$, $a_2 = 0.5851$, $a_3 = 0.3625$, $a_4 = 0.6599$.

As shown in Fig. 2, we select a reference trajectory as

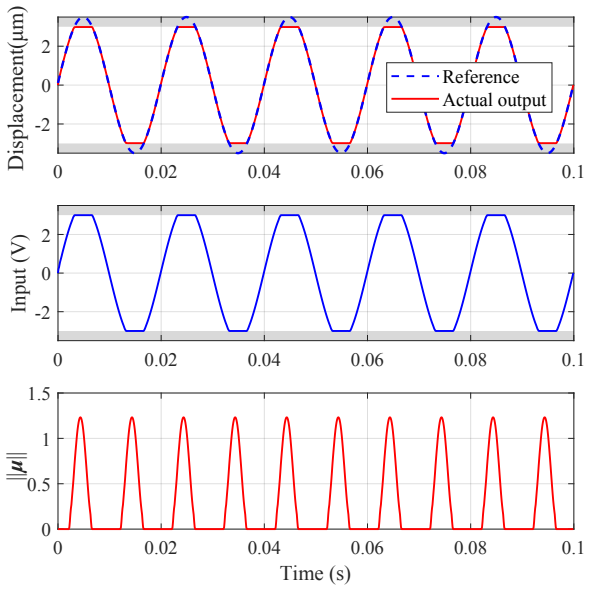

Fig. 2. Tracking result of the coordinate ascent solution algorithm.

$r(k) = 3.5 \cdot \sin(100 \cdot 2\pi \cdot k\tau)$, set $y_{\min} = u_{\min} = -3$ and $y_{\max} = u_{\max} = 3$ for the PEA in (1), and determine $N_p = N_c = 10$ and $\rho = 0.1$ for the MPC in (5). The tracking results in Fig. 2 show the effectiveness of the above Algorithm 1 with $M = 10$ in MATLAB R2021b. Both the displacement output and control input satisfy their constraints. Although the computation latency is as low as 0.14ms on our computer with an Intel Core i7-1270KF @3.61GHz, it still exceeds the sampling time of $\tau = 0.1$ms. For a low-cost STM32F407 @168MHz, the computation latency is expected to be up to 3.0ms by [17, Equation (9)]. Thus, it requires more efforts to tailor the Algorithm (1) for embedded implementation.

By Fig. 2, we observe that the Euclid norm $\|\boldsymbol{\mu}\|$ of the Lagrange multiplier vector changes continuously over time, i.e., $\boldsymbol{\mu}$ also changes continuously in this case. Thus, it is of significant potential to apply the warm-starting strategy to facilitate the convergence of the solution algorithm and thus reduce the computation latency.

C. The Implementation on an STM32 microprocessor

Although the iterations in Algorithm 1 have explicit forms, the dual solution $\boldsymbol{\mu}$ has a dimension of $2(N_p + N_c)$ which is much greater than the dimension N_c of the primal solution $\boldsymbol{u}$. Thus, the dimension of matrix P may be very large when the long horizon lengths are selected. To solve it, an auxiliary vector and auxiliary matrix [16, Section 7.2] are introduced as follows

$$\boldsymbol{v} \triangleq -W^{\mathrm{T}}\boldsymbol{\mu} \text{ and } S \triangleq WH^{-1}. \tag{20}$$

By the definition in (20), the updating in (17) is rewritten as

$$\widehat{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_i - P_{ii}^{-1}\left(\boldsymbol{p}_i - S_i^{\mathrm{T}}\boldsymbol{v}\right), \tag{21}$$

where S_i^{T} denotes the i-th row of the matrix S and W_i stands for the i-th column of matrix W^{T}.

Based on (21), the coordinate ascent operation in (18) is

converted into that

$$\boldsymbol{\mu}_i \leftarrow \boldsymbol{\mu}_i - \delta_i \tag{22}$$

where $\delta_i = \min\left\{\boldsymbol{\mu}_i,\ P_{ii}^{-1}\left(\boldsymbol{p}_i - S_i^{\mathrm{T}}\boldsymbol{v}\right)\right\}$. Moreover, the updating for $\boldsymbol{v}$ is obtained from (20) as follows

$$\boldsymbol{v} \leftarrow \boldsymbol{v} + \delta_i W_i.$$

Compared with the dual variable $\boldsymbol{\mu}$ with a dimension of $2(N_p + N_c)$, the auxiliary vector $\boldsymbol{v}$ has a much lower dimension of N_c, which is the same as that of the primal variable $\boldsymbol{u}$. By Fig. 2, the optimal dual solution $\boldsymbol{\mu}^*$ changes continuously over time. So, the solution $\boldsymbol{\mu}(k-1)$ from the previous time instant $k-1$ is used to speed up the solution algorithm.

Due to the sampling period is as low as 0.1ms, the computation time of Algorithm 2 is not negligible on the embedded platform. Thus, a delayed KF [18] is introduced as shown in Fig. 1 to compensate for the computation time spent on online solving the QP problem (11) as follows

$$\begin{aligned}&\widehat{\boldsymbol{x}}(k+1|k)\\ &\quad = A(I-KC)\widehat{\boldsymbol{x}}(k|k-1) + AKy(k) + Bu(k-1)\end{aligned} \tag{23}$$

where $\widehat{\boldsymbol{x}}(k+1|k)$ is the one time step delayed estimation of the current state $\boldsymbol{x}(k)$, which is used to initialize (5b) in embedded implementation. To further reduce the computation time, we adopt the steady-state filter gain [19]

$$K = \Sigma C^{\mathrm{T}}\left(R_v + C\Sigma C^{\mathrm{T}}\right)^{-1}$$

by solving the following discrete-time algebraic Riccati equation (DARE) of Σ offline

$$\Sigma = A\left(\Sigma - \Sigma C^{\mathrm{T}}\left(R_v + C\Sigma C^{\mathrm{T}}\right)^{-1} C\Sigma\right)A^{\mathrm{T}} + BR_wB^{\mathrm{T}}.$$

The covariance matrices R_w and R_v are given in (4).

To sum up, the embedded version of Algorithm 1 is described in Algorithm 2. Note that the Algorithm 2 needs to be coded in C language. The required constant matrices F, Φ, Ψ, Γ, W, H, H^{-1}, P, and S are pre-computed in MATLAB and then written into a header file for the C program. To further reduce the computational complexity, the sparsity of the matrices A, F, W, and Φ is fully exploited, and only the first element of $\boldsymbol{u}$ is computed by using the first row H_1^{-1} of the matrix H^{-1}.

V. Hardware-in-the-Loop Simulations

By implementing the Algorithm 2 onto a low-cost STM32F407 microprocessor, we carry out the HIL simulations to demonstrate its performance in this section.

A. The settings of HIL simulation

As shown in Fig. 3, the PEA's dynamics in (1) is run on the computer while the proposed algorithms are executed on the STM32F407 @168MHz. They communicate with each other through the COM port. To ensure that the feasible solution can be obtained within the sampling period of 0.1ms, we select $N_p = 6$ and $N_c = 4$ for the MPC and

Algorithm 2 Embedded coordinate ascent algorithm.

1) **Input:** the reference $\boldsymbol{r}$, current output $y(k)$, last control input $u(k-1)$, and last dual solution $\boldsymbol{\mu}(k-1)$.
2) **Output:** the control input $u(k)$ and dual solution $\boldsymbol{\mu}(k)$.
   - Compute $\widehat{\boldsymbol{x}}(k+1|k)$ by (23).
   - Let $\tilde{\boldsymbol{\mu}} = \boldsymbol{\mu}(k-1)$.
   - Compute $\boldsymbol{h}$ and $\boldsymbol{b}$ of the primal problem in (11).
   - Compute $\boldsymbol{v} = -W^{\mathrm{T}}\tilde{\boldsymbol{\mu}}$ in (20).
   - Compute $\boldsymbol{p} = \boldsymbol{b} + S\boldsymbol{h}$ of the dual problem in (14).
   - **For** $i = 1, \ldots, 2(N_p + N_c)$:
     - $\delta_i = \min\left(\tilde{\boldsymbol{\mu}}_i, P_{ii}^{-1}\left(\boldsymbol{p}_i + S_i\boldsymbol{v}\right)\right)$.
     - $\tilde{\boldsymbol{\mu}}_i = \tilde{\boldsymbol{\mu}}_i - \delta_i$.
     - $\boldsymbol{v} = \boldsymbol{v} - \delta_i W_i$.
   - **End for**.
   - **Return** $u(k) = -H_1^{-1}\left(\boldsymbol{h} + W^{\mathrm{T}}\tilde{\boldsymbol{\mu}}\right)$ and $\boldsymbol{\mu}(k) = \tilde{\boldsymbol{\mu}}$.

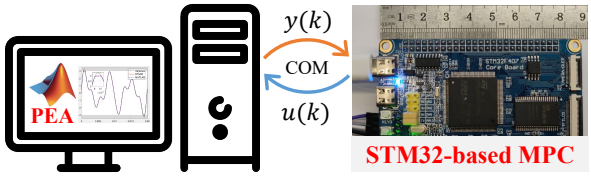

Fig. 3. STM32F407-in-the-loop simulations.

Fig. 4. The total computation latency of the Algorithm 2 on the STM32F407 @168MHz with $N_p = 6$ and $N_c = 4$.

reduce the total computation latency of the Algorithm 2 including the embedded MPC and delayed KF to 0.0664ms ($< \tau \triangleq 0.1$ms), which is measured by a scope. As shown in Fig. 4, we pull the voltage of one pin up at the beginning of Algorithm 2 and then pull it down at the end.

Thus, it is feasible to deploy the proposed Algorithm 2 to achieve the control frequency of 10kHz on such a resource-limited ARM microprocessor.

B. The high-speed trajectory tracking

As Fig. 5, we select a high-speed reference trajectory for the PEA as follows

$$\begin{aligned}r(k) = {}& 2\cdot\sin(150\cdot 2\pi\cdot k\tau) + 0.9\cdot\sin(250\cdot 2\pi\cdot k\tau)\\ &+ 0.5\cdot\sin(350\cdot 2\pi\cdot k\tau) + 0.6\cdot\sin(400\cdot 2\pi\cdot k\tau)\end{aligned}$$

which is consisted of multiple sine signals with the frequencies of 150Hz, 250Hz, 350Hz, and 400Hz. In addition to the settings in IV-B, we set $R_w = 0.01^2$ and $R_v = 0.001^2$ for

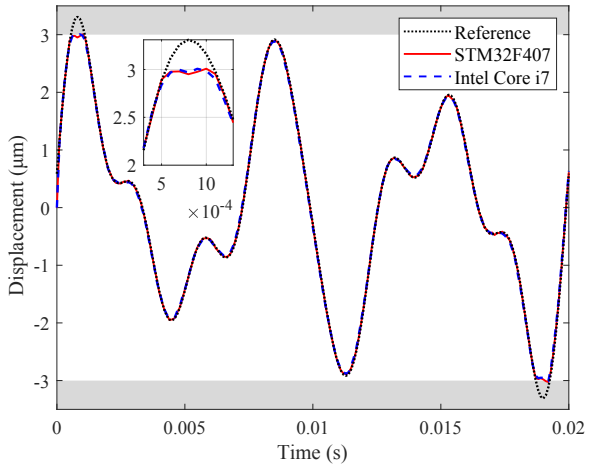

Fig. 5. Tracking trajectories of the MPCs on different platforms.

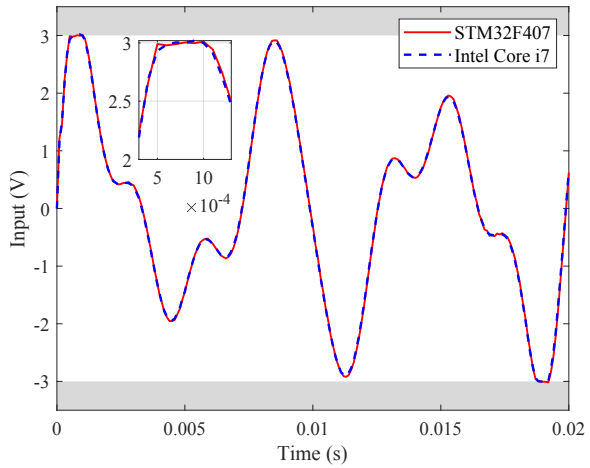

Fig. 6. Control inputs of the MPCs on different platforms.

the Gaussian white noises in (4).

The HIL simulations last 0.02s as shown in Figs. 5 and 6. The STM32-based MPC has the same performance with the Algorithm 2 run in MATLAB on the computer with an Intel Core i7. The both achieve the objective (2) of high-speed trajectory tracking and satisfy the constraints (3) of avoiding damage. Moreover, the root-mean-square-error (RMSE) is 0.0193μm between 0.005s and 0.015s.

VI. Conclusion

In this work, we proposed an STM32-based model predictive control (MPC) approach for achieving high-speed trajectory tracking of a piezoelectric actuator (PEA) while satisfying constraints on both the displacement output and control input. Specifically, we first convert the tracking problem into a constrained quadratic programming problem and then designed an embedded coordinate ascent algorithm to addresses the challenges of online computation. Moreover, we introduced a delayed Kalman filter with a steady-state gain to compensate for the computation latency. To evaluate the performance of our approach, we conduct hardware-in-the-loop simulations on a resource-limited STM32F407 ARM processor. The results demonstrate that our algorithm achieves a control frequency of 10kHz and successfully accomplishes the tracking task. In future work, we plan to extend the proposed algorithm to account for the hysteresis nonlinearity of the PEA.

References

[1] H. Jin, X. Gao, K. Ren, J. Liu, L. Qiao, M. Liu, W. Chen, Y. He, S. Dong, Z. Xu *et al.*, "Review on piezoelectric actuators based on high performance piezoelectric materials," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 2022.

[2] C. Yang, F. Xia, Y. Wang, and K. Youcef-Toumi, "Comprehensive study of charge-based motion control for piezoelectric nanopositioners: Modeling, instrumentation and controller design," *Mechanical Systems and Signal Processing*, vol. 166, pp. 1–14, 2022.

[3] J. Zhong, L. Li, R. Nishida, and T. Shinshi, "Design and evaluation of a PEA-driven fast steering mirror with a permanent magnet preload force mechanism," *Precision Engineering*, vol. 62, pp. 95–105, 2020.

[4] J.-J. Tzen, S.-L. Jeng, and W.-H. Chieng, "Modeling of piezoelectric actuator for compensation and controller design," *Precision Engineering*, vol. 27, no. 1, pp. 70–86, 2003.

[5] Physik Instrumente, "User manual of E-727." [Online]. Available: https://www.physikinstrumente.com/en/

[6] L. Kong, D. Li, J. Zou, and W. He, "Neural networks based learning control for a piezoelectric nanopositioning system," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 6, pp. 2904–2914, 2020.

[7] D. Chowdhury, Y. K. Al-Nadawi, and X. Tan, "Dynamic inversion-based hysteresis compensation using extended high-gain observer," *Automatica*, vol. 135, pp. 1–9, 2022.

[8] H. Habibullah, H. R. Pota, and I. R. Petersen, "A novel control approach for high-precision positioning of a piezoelectric tube scanner," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 325–336, 2017.

[9] Y. Cao, L. Cheng, X. Chen, and J. Peng, "An inversion-based model predictive control with an integral-of-error state variable for piezoelectric actuators," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 3, pp. 895–904, 2013.

[10] S. Xie and J. Ren, "Recurrent-neural-network-based predictive control of piezo actuators for trajectory tracking," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 6, pp. 2885–2896, 2019.

[11] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[12] G. Frison and M. Diehl, "HPIPM: A high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[13] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "Acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.

[14] W. Han, S. Shao, S. Zhang, Z. Tian, and M. Xu, "Design and modeling of decoupled miniature fast steering mirror with ultrahigh precision," *Mechanical Systems and Signal Processing*, vol. 167, p. 108521, 2022.

[15] J. K. Huusom, N. K. Poulsen, S. B. Jørgensen, and J. B. Jørgensen, "Tuning SISO offset-free model predictive control based on ARX models," *Journal of Process Control*, vol. 22, no. 10, pp. 1997–2007, 2012.

[16] D. Bertsekas, *Nonlinear programming*, 3rd ed. Athena Scientific, 2016.

[17] E. N. Hartley, J. L. Jerez, A. Suardi, J. M. Maciejowski, E. C. Kerrigan, and G. A. Constantinides, "Predictive control using an FPGA with application to aircraft control," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1006–1017, 2013.

[18] G. Cimini, D. Bernardini, S. Levijoki, and A. Bemporad, "Embedded model predictive control with certified real-time optimization for synchronous motors," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 893–900, 2020.

[19] J. Yan, X. Yang, Y. Mo, and K. You, "A distributed implementation of steady-state Kalman filter," *IEEE Transactions on Automatic Control*, pp. 1–8, 2022.