Journal of Manufacturing Systems 68 (2023) 695–703

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

# Learning digital emulators for closed architecture machine tool controllers

Akash Tiwari [a], Yuandong Wang [a], Kyle Saleeby [b], A.L. Narasimha Reddy [c], Satish Bukkapatnam [a,*]

[a] Wm Michael Barnes '64 Department of Industrial and System Engineering, Texas A&M University, College Station, TX 77840, USA
[b] Manufacturing Science Division, Oak Ridge National Laboratory, Knoxville, TN 37932, USA
[c] Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77840, USA

ARTICLE INFO

*Keywords:*
Manufacturing cybersecurity
Controllers
Emulator
Genetic algorithm

ABSTRACT

Machine tool controllers (MTCs) are evolving rapidly in response to the growing precision requirements, industry-wide thrusts to integrate cloud applications, external sensors and other data sources, as well to assure cybersecurity. However, most of the MTCs have closed architecture. This severely impedes the innovations to enhance their performance as well as assure their safety and security. Digital emulators, as alternatives to real closed architecture MTCs, are considered essential to assess the performance of the closed MTCs and various innovations therein. We present a machine learning method to create digital emulators that can mimic the dynamics of closed architecture MTCs. The proposed method is based on interrogating the controller using a set of production recipes (e.g., G-codes) and employing the response of the controller to learn a low-dimensional parameterization of the underlying architecture. The low-dimensional base model captures the key aspects of the structure and dynamic connectivity among the various components of an MTC. The parameters of the base model are tuned based on the outputs of an MTC gathered using standard data exchange protocols (e.g. OPC UA, MTConnect). We applied the proposed approach to develop emulators for a SIEMENS controller that mimics X–Y motion of their 2-axis motion control systems. The current implementation of the emulator can track the measured path trajectories up to 0.23 mm accuracy for various geometries tested. Our approach can be used to generalize emulator development for different types of real world MTCs.

## 1. Introduction

Most industrial MTCs have had a closed architecture for several decades [1,2]. The closed architecture is aimed at preserving proprietary information about the controllers, offering a competitive advantage to the original equipment manufacturers. However, it comes at the expense of limited understanding of the underlying behaviors and vulnerabilities, besides impeding possible innovations to enhance the controller. These enhancements are necessitated for two critical reasons. First, manufacturing precision assurance is increasingly pushed from the design stages onto the controller [3]. Innovations in the underlying architecture is at the central towards assuring precision control of machine and process parameters. Second, with worldwide growth in industrial automation and digitalization of manufacturing environments [4,5], MTCs are increasingly integrated with cloud applications, external Internet of Things (IoT) sensors, and other data sources. This trend towards digitalization and smartification introduces significant cybersecurity challenges. MTCs are becoming primary attack targets for cyber-criminals [6]. The consequences of attacks on MTCs can lead to unreliable parts and accidents in a manufacturing plant floor [7]. MTCs need appropriate defense mechanisms which can provide cybersecurity assurance in a manufacturing plant floor.

Additionally, offline testing of MTCs in a production scenario can be very expensive due to safety implications and restrictive down time costs. Therefore inexpensive and non-invasive approaches are highly desirable to assess controller performance and effectiveness of cybersecurity assurance techniques to protect against vulnerabilities [6]. Another example includes cloud based control-by-software paradigms [8–10]. Under this paradigm, the control policy is generated in the cloud and the control signals to actuators of mechanical systems (like autonomous vehicles) are received from geographically distant locations. Such paradigms bring various uncertainties and open novel vulnerabilities for controllers. Again, controller performance assessment (for example, as a function of network and cloud delays) becomes important in this setting. The current "black-box" MTCs, offering minimum access to assess their behaviors under various industrial manufacturing scenarios, hamper the innovations towards, and evaluation of, quality [11], performance [12], safety [13], and cybersecurity [14] assurance.

* Corresponding author.
*E-mail address:* satish@tamu.edu (S. Bukkapatnam).

https://doi.org/10.1016/j.jmsy.2023.05.013

Available online 5 June 2023

**Nomenclature**

| | |
|---|---|
| CLBD | Closed loop Block Diagram |
| DED | Directed Energy Deposition |
| DWM | Dynamic Watermarking |
| GA | Genetic Algorithm |
| IoT | Internet of Things |
| MSE | Mean Square Error |
| MTC | Machine Tool Controller |
| OPC | Open Platform Communications |
| UA | Unified Architecture |

Our paper bridges the gap between the (i) knowledge of the black-box governing the dynamic input–output relations of controllers, and (ii) development and testing of cybersecurity and quality assurance techniques for controllers. We propose a novel machine-learning based approach to developing emulators of MTCs. The emulator is developed by parameterizing the underlying transfer functions of the widespread control structure employed in manufacturing MTCs [1]. The closed architecture of MTCs make it very challenging to decipher the parameter values therefore encouraging the use of data-driven approaches to decipher the parameters that can accurately emulate physical MTCs. The parameterized emulator used in this work serves as a minimal representation of the MTCs which accurately captures the dynamics of the physical MTC. While more complex non-parametric emulators which are optimally tuned can offer superior performance over traditional physics-based models, owing to their flexibility, often times in such cases the underlying structure in relevance to the physical MTC is lost. This makes it challenging to rely on these complex emulator structures in novel situations. In this work, the emulator is modeled such that the MTC structure is preserved and allows use of machine-data to tune the parameters of the emulator for resembling physical MTCs in response to manufacturing instructions.

We note that in recent years there have been increased and often overloaded use of terms such as “digital twins” and “virtualizations” of various entities and processes of a manufacturing system [15,16]. While the emulator we present can be characterized in terms of these, we note that the emulator presented here specifically refers to a software code that captures the dynamics, and hence mimics the salient behaviors of the controllers. We also contrast an emulator from a simulation model in the sense that the emulator aims to mimic an entity, here, an MTC and its dynamics, and a simulator focuses on mimicking the actions and/or activities of the various entities of a system.

The proposed emulator development approach employs the current understanding of MTC architecture to parameterize a “best guess” low-dimensional model of a controller, and uses the response of the machine tool when “probed” with different manufacturing instructions (i.e., G-codes) to learn the model parameters. The method was implemented to create an emulator of a Sinumerik 828D MTC from Siemens that serves an Optomec MTS 500 Hybrid Additive (DED) machine tool. The emulator was able to mimic the motion of the physical MTC to submillimeter precisions for various horizontal (X–Y table) motion trajectories. The proposed approach allows the derivation of emulators whose structure can be directly related to the various elements of physical MTCs, and their parameters can be tuned to match the behaviors of a broad range of industrial MTCs.

The emulator which has been tuned to a certain MTC can then be used for further testing of novel applications under simulated scenarios before their actual deployment on a real machine. Unraveling of a black-box controller improves the understanding of the controllers and therefore help with testing and enhancing the controllers. Also, many of the emerging cybersecurity assurance techniques require knowledge of the transfer functions of the controllers. For example, dynamic watermarking (DWM) [17] helps in detecting attacks to the system. However,

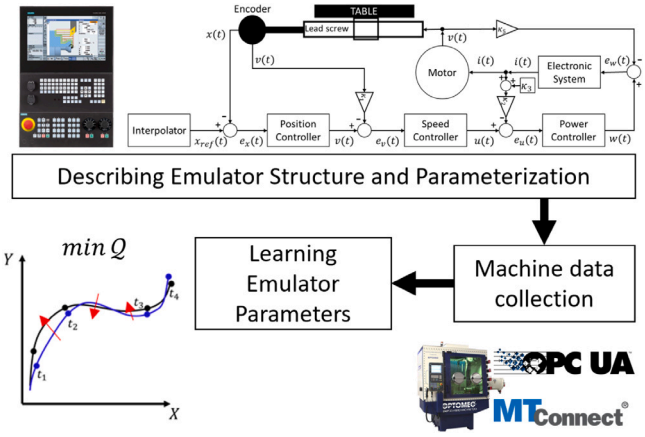


**Fig. 1.** Overview of methodology for development of MTC emulator.

this requires the state-space equation of the dynamic system which may not be readily available. Knowledge of the black box can also help in devising effective defense mechanisms and conduct what-if analysis. The key contribution of this work is methodological in nature which proposes the development of a physics-driven parametric model for an MTC emulator. This emulator type is an alternate to other parametric and non-parametric models [18]. Non-parametric models although can offer superior performance, they fail to generalize well due to lack of resemblance to underlying MTC structures. Such non-parametric emulators are also data-intensive which may be restrictive in manufacturing systems due to limitations imposed by MTC communication protocols. On the other hand, our choice of a physics-driven parametric model for the emulator are a minimal representation which retains the underlying MTC structure by introducing the knowledge of dynamics [13,19] which allows us to estimate and tune the impedance characteristics to required performance criteria for the MTC. Emulators with physics-driven parametric models also have the ability to generalize to novel physical MTCs. Such tuning approaches are not straightforward in other types of parametric and non-parametric emulator models.

The remainder of the paper is organized as follows. Section 2 discusses development of an emulator. In Section 3, we validate the emulator using different contour designs for 2-axis processes such as 3D printing and milling. Section 4 concludes the paper.

## 2. Methodology

In our approach to developing an emulator for an MTC, we limit the functionalities of the emulator to accurately model a 2-axis motion control system. This choice is informed by the ubiquity of these 2-axis motion systems in real world manufacturing systems, and process-agnostic nature of these motion controllers [20]. In developing an emulator, we follow three key steps in our methodology which is outlined in Fig. 1. First, we define an appropriate structural basis for our emulator via a parameterization step. Here, we identify a minimal set of emulator parameters which can capture the salient behaviors of real world MTCs. Next, we probe a real world closed architecture controller using specific set of G-Codes. As the controller executes the G-code, we collect the machine data, specifically, the key way points on actual motion trajectory, including the positions and velocities, employing OPC UA interface. Last, the parameter values are learned by formulating and solving an optimization problem. The emulator parameters learned from the machine data would accurately generate 2-axis motion trajectories resembling that of MTCs. The remainder of the section discusses the methodology in detail.

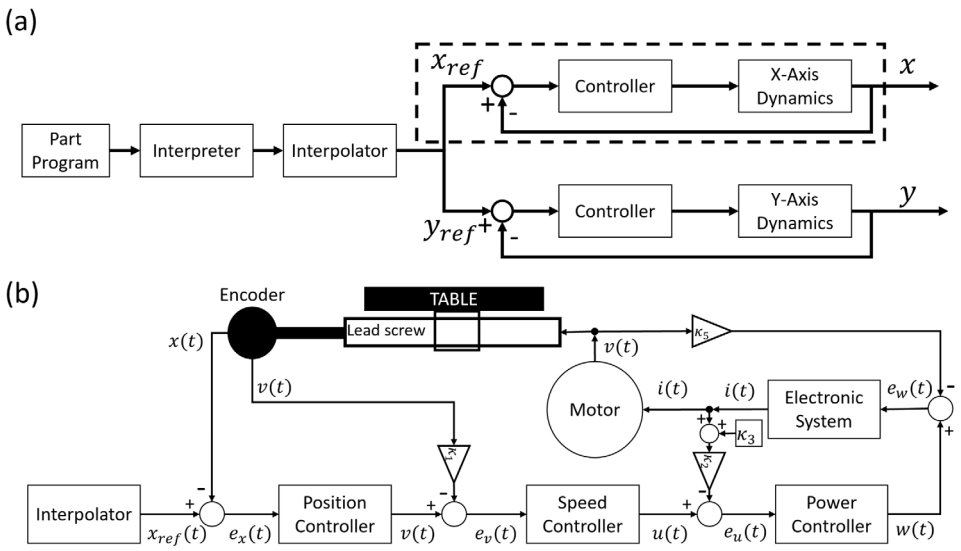


**Fig. 2.** (a) Schematic diagram for 2-axis motion control system (adapted from [21]) (b) X-axis closed loop block diagram with input and output signals of various controllers.

### 2.1. Emulator structure and parameterization

The structural basis of choice for the development of an emulator for a 2-axis X–Y motion control system is illustrated in Fig. 2(a). Such a representation captures the salient behaviors of controllers in a variety of automated manufacturing machine tools, including conventional CNC machining, 3D printing, forming processes. Although the scope of our emulator in this work is limited to 2-axis it can be extended to additional axes.

In automated manufacturing machine tools, a part program (G-code) is provided as input to the MTC (see Fig. 2(a)). An interpreter module of the controller processes the G-codes, typically, it uses an interpolator to generate the reference coordinates and feed rate $(x_{ref}(t), y_{ref}(t), v^x_{ref}(t), v^y_{ref}(t), t)$, which altogether define the motion trajectory. The corresponding reference coordinates are then passed to the position control system of the X and Y axes [21]. Each axis comprises cascaded control loops. A Multioscillatory (MOSC) controller is employed for position control [22].

Fig. 2(b) illustrates the closed loop block diagram (CLBD) with flow of signals between the components of the control system of the X-axis. The CLBD for the Y-axis is similar with a difference in the transfer function of the position controller as described later in this section. The transfer functions governing the relation between input and output signals of the various controllers and sub-systems of the CLBD are described in the remainder of this section. The error between the reference coordinate $x_{ref}$ obtained from the interpolator and initial coordinate from feedback $x(t-\delta t)$ is calculated as $e_x(t) = x_{ref}(t) - x(t-\delta t)$ which is sent to the position controller. The position controller determines the velocity signal $v(t)$. We denote $E_x(s) = \mathcal{L}\{e_x(t)\}$, where $\mathcal{L}$ denotes the Laplace transform. The transfer function relating the output from the position controller $V(s) = \mathcal{L}\{v(t)\}$ and the input position error $E_x(s)$ is given by

$$V(s) = p_{24}\left(p_{23} + \sum_{\substack{i=\{1,\dots,4\}\\ j=\{19,\dots,22\}}} \frac{p_i p_j s}{s^2 + 2p_i p_{(i+4)} s + p_i^2}\right) E_x(s) \tag{1}$$

where parameters $\{p_1, \dots, p_8\}$ and $\{p_{19}, \dots, p_{22}\}$ are coefficients of the transfer function and the gain applied to output from each of the four oscillatory elements in the X-axis, respectively.

The error $e_v(t) = v(t) - \kappa_1 v(t-\delta t)$ between reference velocity signal from the position controller and initial velocity from feedback enters the speed controller. $\kappa_1$ is the gain applied to the feedback velocity signal. The speed controller determines the control signal $u(t)$ for the electronics. Let $U(s) = \mathcal{L}\{u(t)\}$ and $E_v(s) = \mathcal{L}\{e_v(t)\}$, then the transfer function for the speed controller is given by

$$U(s) = p_{25}\left(p_{13} + p_{14}\frac{1}{s}\right) E_v(s) \tag{2}$$

where $p_{13}$ and $p_{14}$ are the proportional and integral parameters for a PI controller within the speed controller. $p_{25}$ is the gain applied to the output from the PI controller. The error in control signal to the electronics, $e_u(t) = u(t) - \kappa_2(i(t-\delta t) + \kappa_3)$, enters the power controller which sends control signal $w(t)$ to the electronics within the system. $\kappa_2$ and $\kappa_3$ are the gain applied to the feedback current and the additive disturbance to the feedback current, respectively. let $E_u(s) = \mathcal{L}\{e_u(t)\}$ and $W(s) = \mathcal{L}\{w(t)\}$. The transfer function for the power controller is given by

$$W(s) = \frac{p_{26}}{(\kappa_4 s + e^{-s\delta t})}\left(p_{15} + p_{16}\frac{1}{s}\right) E_u(s) \tag{3}$$

where $p_{15}$ and $p_{16}$ are the proportional and integral parameters for a PI controller within the power controller. Gain $p_{26}$ is applied to the output from the PI controller. $\kappa_4$ models the delay within the power controller. The parameters $\{p_{13}, \dots, p_{16}\}$ for the PI controller block within the speed and power controller are determined using the modulus optimum method (Kessler's method) and symmetrical optimum method (Naslin's polynomial method, Kessler's method) [23] and adjusted using Genetic Algorithm. The identification errors used for this method are given by the parameters $\{p_{28}, \dots, p_{34}\}$.

The electronic system determines the current signal $i(t)$ for input into the motor based on the error $e_w(t) = w(t) - \kappa_5 v(t-\delta t)$ where $\kappa_5$ is the gain applied to velocity feedback. Let $E_w(s) = \mathcal{L}\{e_w(t)\}$, $I(s) = \mathcal{L}\{i(t)\}$ and $V(s) = \mathcal{L}\{v(t)\}$, then the transfer function for the electronic system and electromechanical conversion at the motor are given by

$$I(s) = \frac{\kappa_6}{s + \kappa_7 e^{-s\delta t}} E_w(s) \tag{4}$$

$$V(s) = \frac{\kappa_8}{s + \kappa_9 e^{-s\delta t}} I(s) \tag{5}$$

where $\kappa_6$ and $\kappa_8$ are factors for abstracting the model of physical and electronic system. $\kappa_7$ and $\kappa_9$ are the gain applied to the feedback of current and velocity within the electronic system and motor, respectively. Finally, the output from the motors goes through an integrator block which gives the position $X(s) = \frac{1}{s}V(s)$, where $X(s) = \mathcal{L}\{x(t)\}$. $\kappa_1, \dots, \kappa_9$ in Eq. (1)–(5) are chosen to be exogenous parameters of the model.

As mentioned earlier, The Y-axis has a similar CLBD (as illustrated in Fig. 2(b)) with differences in the position controller. Specifically, the position controller (MOSC) uses two oscillatory elements with

**Table 1**
Emulator parameters.

| Parameter | Variable | Count |
|---|---|---|
| MOSC controller transfer function (Axis X) | $\{p_1, \ldots, p_8\}$ | 8 |
| MOSC controller transfer function (Axis Y) | $\{p_9, \ldots, p_{12}\}$ | 4 |
| PID block parameters | $\{p_{13}, \ldots, p_{16}\}$ | 4 |
| MOSC controller gains (Axis X) | $\{p_{19}, \ldots, p_{22}\}$ | 4 |
| MOSC controller gains (Axis Y) | $\{p_{17}, p_{18}\}$ | 2 |
| Control loop gains | $\{p_{23}, \ldots, p_{26}\}$ | 4 |
| Feedback disturbance | $\{p_{27}\}$ | 1 |
| Identification errors | $\{p_{28}, \ldots, p_{34}\}$ | 7 |
| | **Total** | **34** |

transfer functions $\frac{p_9 s}{s^2+2p_{11}p_9 s+p_9^2}$ and $\frac{p_{10} s}{s^2+2p_{12}p_{10} s+p_{10}^2}$, respectively. The gain applied to output from each oscillatory element in Y-axis is $p_{17}$ and $p_{18}$, respectively. The parameters of the PI controller in speed and power controller of the X-axis are the same as for the Y-axis.

The controller and the control loops for X and Y axes altogether comprise 34 parameters represented by the vector $\mathbf{p} = \{p_1, \ldots, p_{34}\}$. The parameters are summarized in Table 1. The parameters of this general structure are part of transfer functions of the internal control structure widely found in physical MTCs and can be tuned appropriately to resemble an MTC.

### 2.2. Interrogating closed architecture MTCs and data acquisition

As noted, a physical MTC takes the G-code (part program) as input. We represent G-code with $\mathcal{G}$. The G-code contains information about the successive segments to traverse in the X–Y plane, whether the traversing segment is a linear or a curved (circular) path, and the feed rate for each segment. The interpolator of the MTC specifies the reference trajectory given by the tuple $(x_{ref}(t), y_{ref}(t), v^x_{ref}(t), v^y_{ref}(t), t)$ to be tracked. The reference trajectory specifies the table motion in terms of the X and Y axes coordinate and feed rate at time $t$. The actual trajectory of the machine however is denoted by the tuple $(x_0(t), y_0(t), v^x_0(t), v^y_0(t), t)$.

G-code must be chosen appropriately, when interrogating the closed architecture MTCs and collecting machine data for learning the emulators. Selection criterion for G-code must be to learn various subtleties of the performance of the motion system. For example, although a constant feed rate may be specified in the G-code for a straight path, the actual feed rates $v^x_0(t)$ and $v^y_0(t)$ vary along the straight path due to the acceleration and deceleration at the start and end of the straight line segments which can be observed from Machine data obtained from OPC UA. G-codes on which the emulator is to be trained for effectively capturing physical MTC performance and responses must have a variety of trajectory features to accurately learn the subtle performance characteristics of a closed architecture MTC. These trajectory features include but are not limited to straight paths, sharp corners, curved paths and changes in feed rate. The accuracy of the emulator increases with the set of trajectory features included.

Due to limitations on temporal and positional accuracy of the MTC, there exists a deviation between $(x_{ref}(t), y_{ref}(t), v^x_{ref}(t), v^y_{ref}(t), t)$ and $(x_0(t), y_0(t), v^x_0(t), v^y_0(t), t)$. The pattern of these deviations constitute the signature of a closed controller. The emulator of an MTC is expected to mimic these signatures as closely as possible. Likewise, for an emulator, there is a deviation between the actual trajectory (output) of the emulator denoted by $(x_E(t), y_E(t), v^x_E(t), v^y_E(t), t)$ and $(x_0(t), y_0(t), v^x_0(t), v^y_0(t), t)$ when the same G-code is passed as input to the emulator. The goal is to learn the emulator such that the deviation in the latter case is minimized (see Fig. 4).

To emulate an MTC, we minimize the deviation between the actual trajectory of the MTC $(x_0(t), y_0(t), v^x_0(t), v^y_0(t), t)$ and the emulator's actual trajectory $(x_E(t), y_E(t), v^x_E(t), v^y_E(t), t)$ for a given G-code. Fig. 3 illustrates the positional deviation $(\Delta X, \Delta Y)$ which is defined as the difference between the emulator trajectory and the measured MTC trajectory at

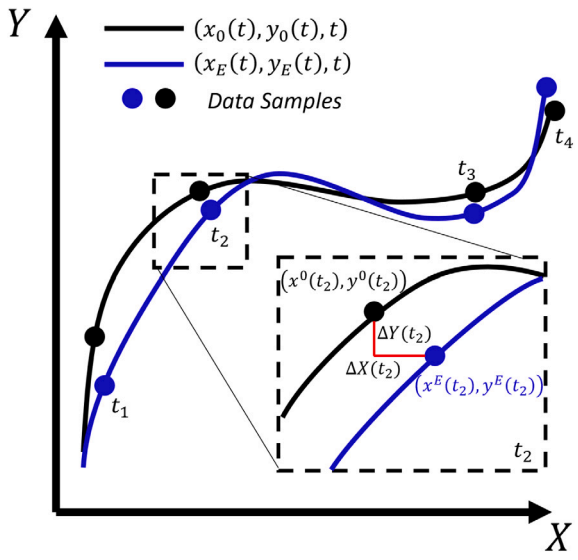


**Fig. 3.** Deviation in position between the emulator trajectory (in blue) and the MTC trajectory (in black). The blue and black circles respectively indicate the times ($T = \{t_1, t_2, t_3, t_4\}$) at which data is sampled from both the trajectories. The inset plot magnifies the trajectories to geometrically illustrate the position deviation (in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

a certain time along each axes. For example, the position deviation along X-axis is given by $\Delta X = (x_E(t) - x_0(t))$. The actual trajectory $(x_0(t), y_0(t), v^x_0, v^y_0, t)$ information is obtained from the MTC using standard data exchange protocols like OPC UA and MTConnect [24]. Machine data from OPC is collected at a maximum sampling rate of 10 Hz. For example, in Fig. 3, $t_{i+1} - t_i \geq 0.1s \; \forall i = \{1, 2, 3\}$. Although the maximum sampling rate is set to 10 Hz, machine data points are not sampled at uniform intervals. Instead the machine data points are sampled only at certain waypoints when the trajectories undergo a major "change of course".

Likewise, $(x_E(t), y_E(t), v^x_E(t), v^y_E(t), t)$ is obtained from the output of the emulator. Due to the limitation in the sampling rate, the information about location and feed rate are obtained at discrete time points. We use $T = \{t_1, \ldots, t_N\}$ to denote the set of timestamps when the location and feed rate information was sampled. We use $N$ to denote the total number of samples collected, i.e., the cardinality of the set $T$.

An emulator having minimum deviation indicates that it is able to accurately emulate the dynamics of the MTC. In other words, such an emulator accurately learns the position, velocity, acceleration and jerk profiles [18] which describe the performance characteristics of the MTC and therefore their behavior. The emulator of an MTC is therefore the structural basis described in Fig. 2 with the corresponding value of emulator parameters listed in Table 1 which accurately emulate the characteristic performance of the MTC. The following subsection describes the setup for learning the emulator parameters $\mathbf{p}$.

### 2.3. Learning emulator parameters

Developing an emulator for an MTC can be formulated as an inverse problem. In the previous section we established the parameters of the emulator and defined the governing dynamic equations for the emulator of a 2-axis motion system. The inverse problem refers to inferring actual value of the model parameters based on measurements from a real physical system [25].

We describe a mapping $\mathcal{M} : \{\mathcal{P} \longrightarrow \mathcal{E}\}$. $\mathcal{P} \in \mathbb{R}^{+34}$ is the vector space of parameters listed in Table 1. $\mathcal{E}$ is the space of emulator trajectory $(x_E(t), y_E(t), v^x_E(t), v^y_E(t), t)$. The inverse problem is the identification of the vector of parameters $\mathbf{p} \in \mathcal{P}$ such that the deviation between

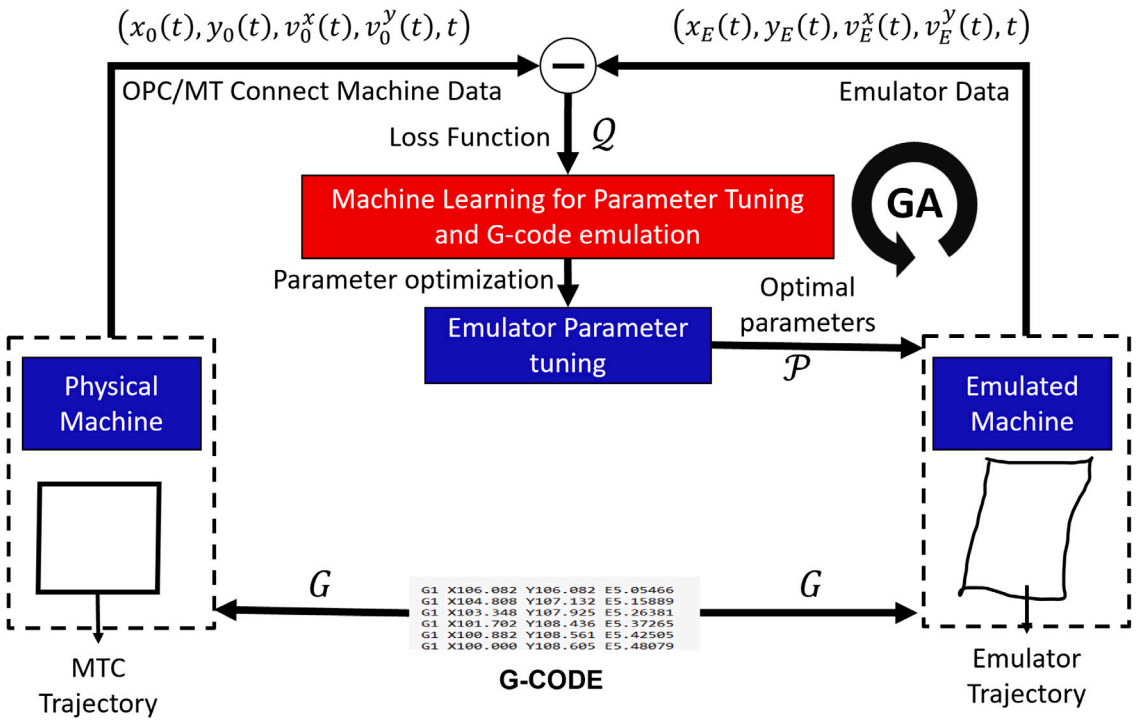


**Fig. 4.** Schematic for learning emulator parameters.

$(x_E(t), y_E(t), v_E^x(t), v_E^y(t), t) \in \mathcal{E}$ and $(x_0(t), y_0(t), v_0^x(t), v_0^y(t), t)$ is minimum. Learning the parameter values $\mathbf{p} \in \mathcal{P}$ can be formulated as the following optimization problem.

*Objective Function:*

$$\min \mathcal{Q} = \text{MSE(x)} + \text{MSE(y)} + \lambda(\text{MSE}(\text{v}^\text{x}) + \text{MSE}(\text{v}^\text{y}))$$

*subject to:* $p_i > 0 \forall i = \{1, \ldots, 34\}$

where $\text{MSE(x)} = \sum_{t \in T}(x_0(t) - x_E(t))^2/N$ is the Mean Square Error (MSE) between the MTC trajectory and the emulator trajectory along the X-axis as described in Section 2.2. Likewise, the MSE for position along Y-axis MSE($y$) and feed rate, MSE($v_x$) and MSE($v_y$), are defined. $\lambda$ is introduced for regularization of the feed rate. The minimization of the above objective function $\mathcal{Q}$ subject to the linear constraints is performed using Genetic Algorithm (GA) [26].

In GA, an initial population of size $pop_{size}$ from a feasible space of parameters $\mathbf{p}_j \in \mathcal{P} : j \in \{1, \ldots, pop_{size}\}$ is randomly initialized. The fitness function used is the objective $\mathcal{Q}$. In each generation of GA, individuals of the population are scored based on $\mathcal{Q}$. A proportion of individuals of current generation ($e_{prop} \in (0, 1)$) are identified as elite which are chosen to be present in the next generation. The remaining members of the population undergo selection followed by mutation or crossover operations to replace remaining individuals of the current generation to form the next generation. GA stops when maximum number of generations ($gen_{max}$) is reached.

Our methodology for training the emulator is illustrated in Fig. 4. The emulator is trained using set of G-codes $\mathcal{G} = \{G_1, \ldots, G_M\}$. Each G-code $G_m \in \mathcal{G}$ is passed as input to the MTC and the emulator. Data collected from the MTC is used to calculate loss function $\mathcal{Q}$. In the following section, we validate our methodology for developing an emulator for a SIEMENS MTC using a set of G-code designs $\mathcal{G} = \{G_1, G_2, G_3\}$.

## 3. Implementation details and results

Our approach to develop an emulator is validated on a Sinumerik 828D MTC from Siemens. This controller operates the Optomec MTS 500 Hybrid Additive (DED) machine tool illustrated in Fig. 5(a). The

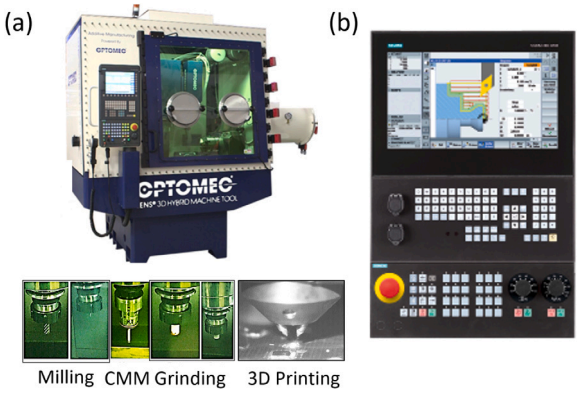


**Fig. 5.** Hybrid Machine Tool (a) OPTOMEC MTS 500 Hybrid Machine Tool with available tools (b) Sinumerik 828D Machine Tool Controller.

hybrid machine tool is capable of performing both additive and subtractive manufacturing processes. The machine tool is also equipped with an automatic tool change carousel which houses milling, grinding, polishing and Coordinate measuring tools, all of which are operated by the MTC. The Sinumerik 828D MTC (Fig. 5(b)) serves as an interface between operators and the machine tool for handling tool change and executing manufacturing processes. The MTC monitors and controls the coordination of the multiple axes and process variables to accurately perform the desired tooling and manufacturing operations. In this section, we develop the emulator which emulates the behavior of 2-axis motion system for contouring operations observed for example in milling and 3D printing operations.

Complex contouring paths can be broken into simpler paths comprised of straight and curved paths. The interpolator described in Section 2 (see Fig. 2) achieves this simplification and develops appropriate reference trajectories $(x_{ref}(t), y_{ref}(t), v_{ref}^x(t), v_{ref}^y(t))$ for the emulator to track. Therefore, to learn an emulator which can accurately execute complex contours, such emulators must be capable of executing

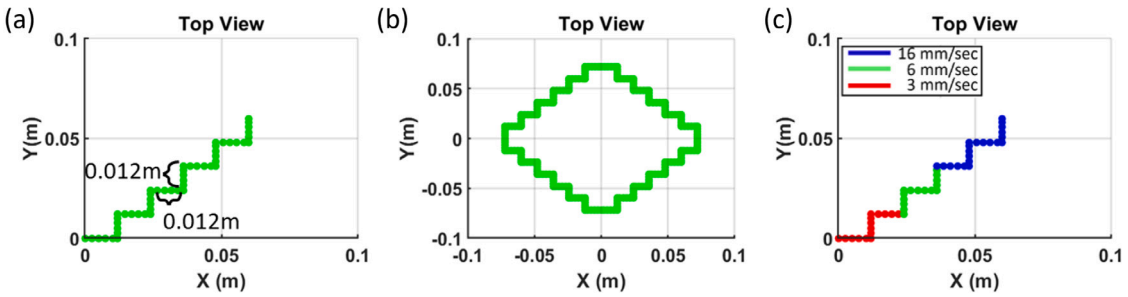


**Fig. 6.** Contour designs (a) Staircase pattern with uniform speed ($\mathcal{G}_1$) (b) Staircase diamond pattern with uniforms speed ($\mathcal{G}_2$) (c) Staircase pattern with non-uniform speed ($\mathcal{G}_3$).

simpler paths. For this reason we choose G-codes to train an emulator which can execute four motion actions - (i) **T1**: Travel along straight paths with minimum deviation. (ii) **T2**: Execute perpendicular corners (iii) **T3**: Traveling in opposite directions (iv) **T4**: Changing feed rate. Three design contours illustrated in Fig. 6 and described by G-codes $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ are chosen to learn the emulator parameters. Fig. 6(a) (representing $\mathcal{G}_1$) is a staircase pattern with straight segments which are 0.012 m in length and have a constant feed rate of 6 mm/sec. After each straight segment there is a perpendicular turn which makes this design appropriate for learning **T1** and **T2**. Fig. 6(b) (representing $\mathcal{G}_2$) is again a stair case pattern but mirrored along two perpendicular axis to resemble a diamond. Again, the feed rate is 6 mm/sec. This design contour can be used to learn **T1-T3**. Fig. 6(c) (representing $\mathcal{G}_3$) is the same stair case pattern as in Fig. 6(a) but has three segments with different speeds. The three speeds chosen are 3 mm/sec, 6 mm/sec and 16 mm/sec. This design contour can be used to learn **T1**,**T2** and **T4**.

The dynamic equations for the emulator's structural basis for the 2-axis X–Y motion control system described in Section 2.1 is implemented using MATLAB Simulink 2022a [22]. The emulator takes G-codes ($\mathcal{G}$) as input and emulates the behavior of an MTC to provide output trajectory $(x_E(t), y_E(t), v_E^x(t), v_E^y(t), t)$. Machine data $(x_0(t), y_0(t), v_0^x(t), v_0^y(t), t)$ is collected from the Sinumerik 828D Controller using OPC UA data exchange protocol.

G-codes for the three contour designs $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ are used as inputs to MATLAB Simulink and MTC. The emulator parameters $\mathbf{p}$ are learned for two cases (i) $\lambda = 0$ (ii) $\lambda = 0.8$ for each of the three designs $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$. The case of $\lambda = 0$ represents no feed rate regularization over the objective function $\mathcal{Q}$, while the case of $\lambda = 0.8$ corresponds to when feed rate is considered in the optimization. The choice of $\lambda = 0.8$ was chosen arbitrarily to have MSE($v^x$)+MSE($v^y$) comparable but lower than MSE(x)+MSE(y). In the former case, deviation is minimized only with respect to position while in the latter case deviation is minimized with respect to both position and feed rate.

### 3.1. Optimization results and discussion

As described in Section 2, the structure of the MTC has been modeled essentially as a cascade of transfer functions that translate to a set of linear differential equations with delays (from Eqs. (4) and (5)). The data obtained from interrogating the MTCs using G-codes can be used to learn the parameters of this MTC model. Multiple alternative machine learning methods have been considered to model such delay differential equations in the manufacturing literature (e.g., [19,27,28]). In this work, we employ genetic algorithms, as they can be extended to not just estimate the model parameters but also allow the adjustment of the model structure.

GA optimizer routine in MATLAB 2022a is used for optimization. The optimization routine is performed on a computer with Windows 10 Enterprise 64-bit operating system, Intel Core i7-10700T processor with 8 cores, clock speed 2.00 GHz, and memory of 16 GB RAM. In the GA optimizer, Stochastic uniform Selection is the selection strategy employed. Uniform weighted average of parents are chosen as the crossover function with a crossover fraction of 0.8. The function used for mutation is the adaptive feasible mutation. The population size and generation size are set to $pop_{size} = 50$ and $gen_{max} = 100$, respectively. Elite count is set to $e_{prop} = 0.05$ proportion of the population size.

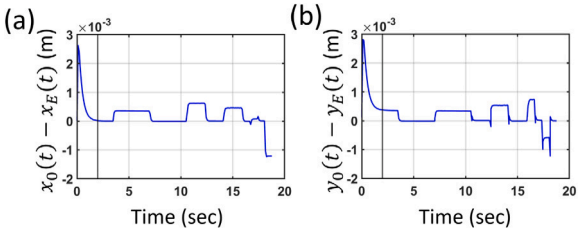


**Fig. 7.** Transience in emulator position based on difference between MTC and emulator data for (a) X axis (b) Y axis. The vertical line indicates $t = 2$ s.

**Table 2**
Position MSE $\mathcal{Q}(x, y)$ for training design (diagonal entries) and testing design (off-diagonal entries) without velocity regularization ($\lambda = 0$). The minimum $\mathcal{Q}(x, y)$ is underlined.

| Design | $\mathbf{p}_1^*$ | $\mathbf{p}_2^*$ | $\mathbf{p}_3^*$ |
|---|---|---|---|
| $\mathcal{G}_1$ | (0.25 mm)$^2$ | (0.24 mm)$^2$ | (0.25 mm)$^2$ |
| $\mathcal{G}_2$ | (1.70 mm)$^2$ | (1.60 mm)$^2$ | (1.70 mm)$^2$ |
| $\mathcal{G}_3$ | (0.52 mm)$^2$ | (0.52 mm)$^2$ | (0.51 mm)$^2$ |

**Table 3**
Position MSE $\mathcal{Q}(x, y)$ for training design (diagonal entries) and testing design (off-diagonal entries) with velocity regularization ($\lambda = 0.8$). The minimum $\mathcal{Q}(x, y)$ is underlined.

| Design | $\mathbf{p}_1^*$ | $\mathbf{p}_2^*$ | $\mathbf{p}_3^*$ |
|---|---|---|---|
| $\mathcal{G}_1$ | (0.53 mm)$^2$ | (0.38 mm)$^2$ | (0.23 mm)$^2$ |
| $\mathcal{G}_2$ | (1.30 mm)$^2$ | (1.40 mm)$^2$ | (1.60 mm)$^2$ |
| $\mathcal{G}_3$ | (0.80 mm)$^2$ | (0.65 mm)$^2$ | (0.53 mm)$^2$ |

The optimal parameters $\mathbf{p}_m^* \; \forall \; m \in \{1, 2, 3\}$ are obtained by minimization of fitness function $\mathcal{Q}$. However, we compare the performance of the parameter based on position MSE which we denote by $\mathcal{Q}(x, y)$ = MSE(x)+MSE(y) when steady state is achieved. Fig. 7 illustrates the transience in the emulator position for the X and Y axes for $\mathcal{G}_3$. We make a conservative choice of $t = 2$ s, after which the emulator achieves a steady state for the three designs $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$. $\mathcal{Q}(x, y)$ is therefore calculated for $t \geq 2$ s for all designs. The average training time on design files $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ are 182 min, 874 min and 193 min respectively. The majority of the training lead time is contributed by the Simulink model simulation. Corresponding $\mathcal{Q}_m^*(x, y) \; \forall \; m \in \{1, 2, 3\}$ are presented in Table 2 for $\lambda = 0$ and Table 3 for $\lambda = 0.8$ on training and testing designs. The diagonal entries of Tables 2 and 3 indicate position MSE $\mathcal{Q}_m^*(x, y)$. The off diagonal entries represent position MSE $\mathcal{Q}(x, y)$ obtained when the optimal parameters $\mathbf{p}_m^*$ are tested on design file $\mathcal{G}_n : m \neq n$ to illustrate the performance of optimal parameters on unseen design contours (test designs).

The optimal parameters $\mathbf{p}^*$ learned from training designs show improved performance on simpler test designs ($\mathbf{p}_m^*$ tested on $\mathcal{G}_n : m \neq n$)

for $\mathcal{G}_1$ and $\mathcal{G}_2$. For example, based on the off-diagonal entries we observe at least 6.25% improvement when testing on design $\mathcal{G}_2$.

Minimum position deviation is observed for the case when $\mathbf{p}_3^*$ is applied to $\mathcal{G}_1$ for case of $\lambda$=0.8. This is likely because the parameter $\mathbf{p}_3^*$ was tested on a simpler design. The emulator learned using $\mathcal{G}_3$ was trained on trajectory features **T1**, **T2** and **T4**, which is a superset of the trajectory features of $\mathcal{G}_1$ (**T1**, **T2**).

The emulator performance for design contour $\mathcal{G}_3$ is visualized in Fig. 8 with optimal parameters $\mathbf{p}_3^*$ obtained for $\lambda = 0.8$. Fig. 8(a) illustrates the time series plot for position and velocity along the X and Y axes. The position MSE $\mathcal{Q}(x, y)$ observed in this case is $(0.53\ \text{mm})^2$ (corresponding to ($\mathcal{G}_3$, $\mathbf{p}_3^*$) in Table 3). Minimum disagreement is observed between the coordinates of emulator and MTC (1.73% contribution to $\mathcal{Q}$) while larger amount of disagreement is observed for velocities (98.26% contribution to $\mathcal{Q}$). The disagreement in velocities $v_E^x(t)$ and $v_E^y(t)$ (with $v_0^x(t)$ and $v_0^y(t)$) are due to slow attainment of reference velocities ($v_{ref}^x(t), v_{ref}^x(t)$). This can be observed as steep edges of the segments with non-zero velocities for the emulator velocity. Furthermore, oscillations are observed in the inset plot within X and Y axes contour plot for position in Fig. 8(b). These oscillations in contour are likely to arise again due to oscillations in $v_E^x(t)$. Additionally, we observe overshoots at corners (trajectory features **T2**) as illustrated in the inset plots of Fig. 8(b) which contribute to larger $\mathcal{Q}$.

We note that highest position error is observed for $\mathcal{G}_2$ and the lowest error in position is observed for $\mathcal{G}_1$. This can likely be attributed to the larger number of **T2** features in $\mathcal{G}_2$ and the simplicity of $\mathcal{G}_1$, respectively. As we note from the inset plot of Fig. 8(b), there exist overshoots at the corners. The error in overshoots at **T2** features add up to larger $\mathcal{Q}(x, y)$ for $\mathcal{G}_2$.

It was expected for the case of velocity regularization ($\lambda \neq 0$) — learning both position and velocity profiles results in an improved emulator. However, the counterintuitive result of deterioration of performance in Table 3 can be explained based on the requirement of a larger sample size to learn higher order profiles such as velocity. Furthermore, by mitigating sampling rate limitations on the OPC UA protocol (10 Hz) emulator performance can likely be improved for complex designs and effectively minimizing deviation between higher order profiles (e.g.: velocity, acceleration, jerk). Appropriate upsampling approaches can be used to obtain larger number of samples $N$ from MTC trajectory.

The results in Tables 2 and 3 points towards using composite designs. Composite designs can have a combination of features in $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ along with multiple replications of trajectory features (**T1-T4**). Such composite designs can likely result in an improved training data set.

In the following section, we identify the parameters that are most likely to be responsible for such variations in the trajectory. Such parameters are likely to have significant effect on the emulator performance. Therefore, these parameters can carefully tuned to further improve the emulator performance.

### 3.2. Emulator parameter analysis

The optimal parameters $\mathbf{p}^*$ identified from GA optimization in Section 3.1 are further analyzed to identify the extent of effect the individual parameters $\{p_i, i \in \{1, \dots, 34\}\}$ have on fitness function value $\mathcal{Q}$. The sensitivity of $\mathcal{Q}$ towards the parameters $\{p_i, i \in \{1, \dots, 34\}\}$ is called as the importance of the parameters. This quantification of importance can be used for two purposes. First, to understand impact of individual parameters $\{p_i, i \in \{1, \dots, 34\}\}$ on emulator performance. Second, to explain the emulator performance in Section 3.1 based on which elements of the CLBD (Fig. 2) may be suboptimally tuned and tune them carefully for improving emulator performance.

To determine parameter importance, a predictive model is built using individuals from each generation of GA optimization. Specifically,

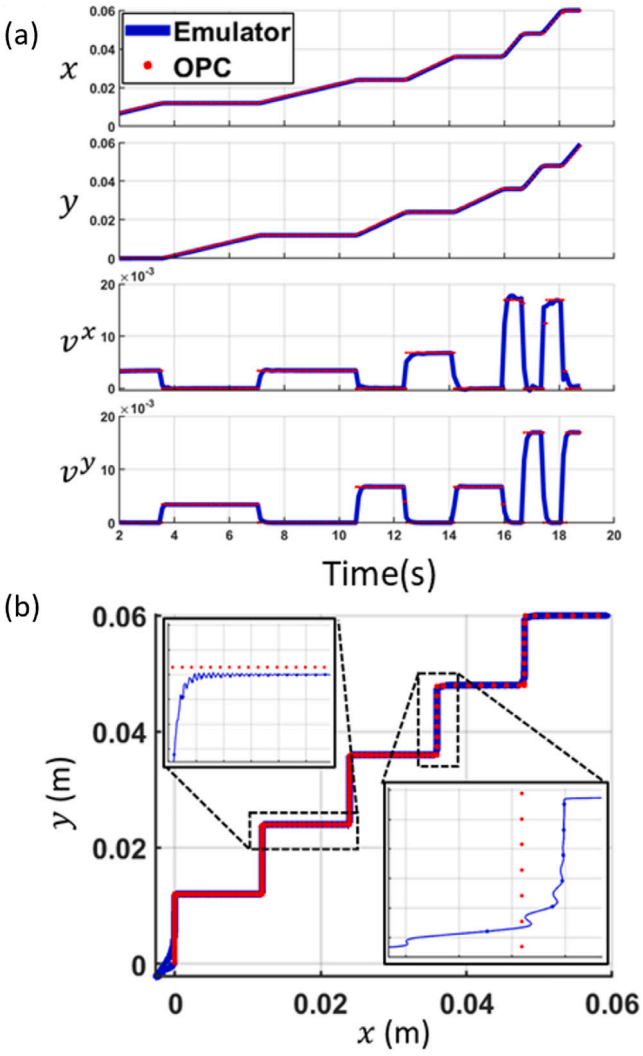


**Fig. 8.** Visualization of emulator performance (during steady state) with optimal parameter $\mathbf{p}^*$ on $\mathcal{G}_3$ (a) Emulator trajectory $(x_E(t), y_E(t), v_E^x(t), v_E^y(t), t)$ and MTC trajectory $(x_0(t), y_0(t), v_0^x(t), v_0^y(t), t)$ for design $\mathcal{G}_3$. MTC data is obtained using OPC UA. (b) X and Y axes contour plot for emulator (in blue) and MTC (in red). The inset plots show a comparison of emulator and OPC data for horizontal and vertical paths. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

we build a regression tree ensemble [29] because of the likely nonlinear relation between the predictors and output. The predictors used here are the parameters $\{p_i, i \in \{1, \dots, 34\}\}$ and output of the regression is fitness function value $\mathcal{Q}$. The observations used in developing the regression tree ensemble include predictors corresponding to individuals from all generations of GA optimization and corresponding fitness function value $\mathcal{Q}$ as outputs. $S$ Decision trees are used as the weak learners and the regression ensemble are fit using Least Squares Boosting. The weak learners are represented by $\mathcal{T}_s, s \in \{1, \dots, S\}$ The importance $\mathcal{I}_i^2$ of parameter $p_i$ is calculated as the importance of $p_i$ in tree $\mathcal{T}_s$ — $\mathcal{H}_i^2(\mathcal{T}_s)$ — averaged over all $S$ trees and is given by

$$\begin{aligned} \mathcal{H}_i^2(\mathcal{T}) &= \sum_{k=1}^{K-1} \hat{r}_k^2 \mathrm{I}(v(k) = i) \\ \mathcal{I}_i^2 &= \frac{1}{S} \sum_{s=1}^{S} \mathcal{H}_i^2(\mathcal{T}_s) \end{aligned} \tag{6}$$

where $\mathrm{I}(v(k) = i)$ is an indicator of whether splitting in node $k$ of tree $\mathcal{T}$ used parameter $p_i$ and $\hat{r}_k^2$ is the squared error risk on node $k$.

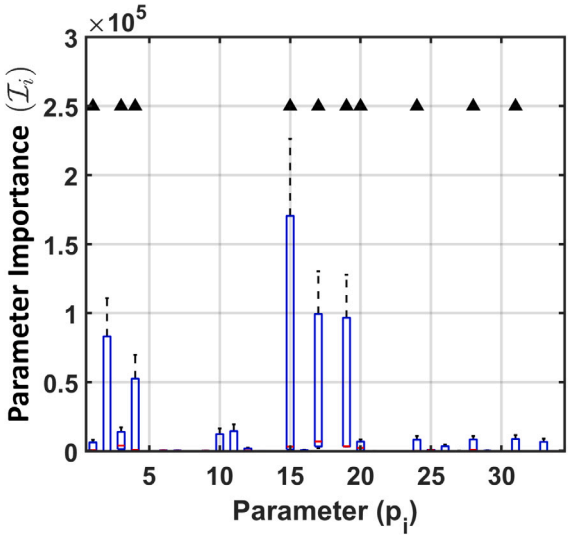


**Fig. 9.** Importance of parameters $p_i, i \in \{1, \ldots, 34\}$ using Regression tree ensembles. Black triangles identify the top 10 important parameters.

Three Regression tree ensembles are developed from individuals of generations from GA optimizer for the three design files $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$. Parameter importance from Eq. (6) is obtained for each of the three ensembles. Fig. 9 illustrates the parameter importance as box plot (developed from the ensembles for $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$). The top 10 important parameters (identified by black triangles in Fig. 9) identified based on descending order of average importance are $\{p_{17}, p_{19}, p_{15}, p_4, p_3, p_1, p_{24}, p_{20}, p_{28}, p_{31}\}$.

Based on parameters listed in Table 1, 5 of the top 10 important parameters correspond to position controller of the X-axis ($\{p_1, p_3, p_4, p_{19}, p_{20}\}$) and one Y-axis position controller parameter ($p_{17}$). The remaining parameters include gain ($p_{24}$), PI parameters ($p_{15}$) and identification errors ($\{p_{28}, p_{31}, \}$) common to both axes. The majority of top 10 important parameters correspond to X-axis of position controller. Also, we have noted in Section 3.1, large values of fitness function $Q$ is contributed by the large oscillations in X-axis velocity $v_E^x(t)$ (see Fig. 8(b)). Therefore, careful tuning of the X-axis position controller can help improve the emulator performance. Furthermore, one can choose to tune a subset of the parameters $\{p_1, \ldots, p_{34}\}$, and this subset can be chosen based on criteria of parameter importance. This can allow faster and improved convergence of GA optimizer. Training is further continued with only the top ten important parameters $\{p_{17}, p_{19}, p_{15}, p_4, p_3, p_1, p_{24}, p_{20}, p_{28}, p_{31}\}$ and leaving the remaining parameters unchanged. The GA optimizer settings are unchanged except for setting $gen_{max} = 25$. The MSE $Q(x, y)$ with velocity regularization is further improved by 36% and 38% for design files $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. However, in $\mathcal{G}_3$ the MSE $Q(x, y)$ deteriorates by 71%. The deterioration in $\mathcal{G}_3$ is likely due to sub-optimal solutions and warrants further investigation by allowing the GA optimizer to run longer. This improvement in $Q(x, y)$ for $\mathcal{G}_1$ and $\mathcal{G}_2$, achieved by adjusting only a subset of **p**, validates the importance of the identified subset for the learning process.

## 4. Conclusions and future work

We have developed a machine learning approach to develop emulators for closed architecture MTCs. The approach is intended to be generalized to emulate any MTC by tuning the parameters to the MTC of choice. We validated our approach on a Siemens Sinumerik 828D controller that is integrated with a hybrid additive (DED) machine tool.

Our emulator employs a parameterized base model that captures the structure of an MTC. Different feedback loops and cascading control modules are included in the form of various transfer functions as part of the structure. The basic structure parameterization can be adjusted to resemble an MTC.

It is important for the emulator to learn from the right design files and for sufficient duration. This ensures the emulator has learned to accurately emulate the different paths of an MTC. For this purpose we trained the emulator on multiple design files. We use Genetic Algorithm to learn the optimal emulator parameters. Variability is observed between the optimal parameters learned by the emulator for the different design files. Our emulator is accurate up to 0.23 mm. The emulator performance can further be improved by using composite G-code designs which include a combination of complex trajectory features. Several replications of the trajectory features within the design can ensure that the emulators are trained to high accuracy.

We intend to use the emulator for testing novel cybersecurity assurance techniques for MTCs. Approaches like Dynamic Watermarking which require knowledge of system can be tested. The future work also includes extension to an active learning framework. The emulator will receive G-codes based on where poor performance is observed. This can be achieved by developing a random G-code generator inclusive of straight paths (of varying length and angles) and arc motions. Developing an emulator which captures all aspects of an MTC, although accurate can be restrictive and intractable, therefore we restrict the level of detail of our MTC emulator for 2-axis motion control. However, our methodology for developing an emulator can be extended beyond 2-axis motion control to control rotational axes and various other manufacturing process parameters.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

Authors Akash Tiwari and Satish Bukkapatnam would like to thank TAMU X-grants and Rockwell Professorship in supporting this research.

## References

[1] Pritschow G, Altintas Y, Jovane F, Koren Y, Mitsuishi M, Takata S, et al. Open controller architecture–past, present and future. CIRP Ann 2001;50(2):463–70.
[2] Adam A, Yusof Y, Latif K, Kadir AZA, Sam T-H, Ahmad MI. Development of a novel open platform controller machining module for 3-axis CNC milling machine. In: Proceedings of the 12th national technical seminar on unmanned system technology 2020. Springer; 2022, p. 839–51.
[3] Cheng C, Bukkapatnam ST, Raff LM, Komanduri R. Towards control of carbon nanotube synthesis process using prediction-based fast Monte Carlo simulations. J Manuf Syst 2012;31(4):438–43.
[4] Zhong Y, Wang Z, Yalamanchili AV, Yadav A, Srivatsa BR, Saripalli S, et al. Image-based flight control of unmanned aerial vehicles (UAVs) for material handling in custom manufacturing. J Manuf Syst 2020;56:615–21.
[5] Tootooni MS, Liu C, Roberson D, Donovan R, Rao PK, Kong ZJ, et al. Online non-contact surface finish measurement in machining using graph theory-based image analysis. J Manuf Syst 2016;41:266–76.
[6] Mahesh P, Tiwari A, Jin C, Kumar PR, Reddy AN, Bukkapatanam ST, et al. A survey of cybersecurity of digital manufacturing. Proc IEEE 2020;109(4):495–516.
[7] Belikovetsky S, Yampolskiy M, Toh J, Gatlin J, Elovici Y. dr0wned– Cyber-Physical Attack with additive manufacturing. In: 11th USENIX workshop on offensive technologies. 2017.
[8] Liu L, Yao Y. Development of a CNC interpretation service with good performance and variable functionality. Int J Comput Integr Manuf 2022;35(7):1–18.
[9] Liu C, Le Roux L, Körner C, Tabaste O, Lacan F, Bigot S. Digital twin-enabled collaborative data management for metal additive manufacturing systems. J Manuf Syst 2022;62:857–74.
[10] Xu W, Cui J, Li L, Yao B, Tian S, Zhou Z. Digital twin-based industrial cloud robotics: Framework, control approach and implementation. J Manuf Syst 2021;58:196–209.

[11] Iquebal AS, Wang Z, Ko W-H, Wang Z, Kumar P, Srinivasa A, et al. Towards realizing cybermanufacturing kiosks: Quality assurance challenges and opportunities. Procedia Manuf 2018;26:1296–306.
[12] Wang K, Dave P, Hanchate A, Sagapuram D, Natarajan G, Bukkapatnam ST. Implementing an open-source sensor data ingestion, fusion, and analysis capabilities for smart manufacturing. Manuf Lett 2022;33:893–901.
[13] Weistroffer V, Keith F, Bisiaux A, Andriot C, Lasnier A. Using physics-based digital twins and extended reality for the safety and ergonomics evaluation of cobotic workstations. Front Virtual Real 2022;3:781830.
[14] Tiwari A, Reddy ALN, Bukkapatnam STS. Cybersecurity assurance in the emerging manufacturing-as-a-service (MaaS) paradigm: A lesson from the video streaming industry. Smart Sustain Manuf Syst 2020;4(3):324–9.
[15] Qi Q, Tao F, Hu T, Anwer N, Liu A, Wei Y, et al. Enabling technologies and tools for digital twin. J Manuf Syst 2021;58:3–21.
[16] Leng J, Wang D, Shen W, Li X, Liu Q, Chen X. Digital twins-based smart manufacturing system design in Industry 4.0: A review. J Manuf Syst 2021;60:119–37.
[17] Satchidanandan B, Kumar PR. Dynamic watermarking: Active defense of networked cyber–physical systems. Proc IEEE 2016;105(2):219–40.
[18] Endo M, Sencer B. Accurate prediction of machining cycle times by data-driven modelling of NC system's interpolation dynamics. CIRP Ann 2022;71(1):405–8.
[19] Bukkapatnam S, Clark B. Dynamic modeling and monitoring of contour crafting—An extrusion-based layered manufacturing process. J Manuf Sci Eng 2007;129(1):135–42.
[20] Hardt DE. Modeling and control of manufacturing processes: Getting more involved. J Dyn Syst Meas Control 1993;115(2B):291–300.
[21] Huo F, Poo A-N. Precision contouring control of machine tools. Int J Adv Manuf Technol 2013;64(1):319–33.
[22] Ufnalski B. Two-axis CNC RC cutting machine. MATLAB Cent File Exch 2022.
[23] Voda A, Landau I. A method for the auto-calibration of PID controllers. Automatica 1995;31(1):41–53.
[24] Liu C, Vengayil H, Lu Y, Xu X. A cyber-physical machine tools platform using OPC UA and MTConnect. J Manuf Syst 2019;51:61–74.
[25] Tarantola A. Inverse problem theory and methods for model parameter estimation. SIAM; 2005.
[26] Golberg DE. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley; 1989.
[27] Rao P, Bukkapatnam S, Beyca O, Kong ZJ, Komanduri R. Real-time identification of incipient surface morphology variations in ultraprecision machining process. J Manuf Sci Eng 2014;136(2):021008.
[28] Cheng C, Wang Z, Hung W, Bukkapatnam ST, Komanduri R. Ultra-precision machining process dynamics and surface quality monitoring. Procedia Manuf 2015;1:607–18.
[29] Hastie T, Tibshirani R, Friedman JH, Friedman JH. The elements of statistical learning: data mining, inference, and prediction, vol. 2. Springer; 2009.